Representation and Learning of Context-Dependent Choice Functions

Karlson Pfannschmidt

July 8, 2024

PADERBORN UNIVERSITY

Department of Computer Science
Warburger Straße 100
33098 Paderborn

Dissertation

In partial fulfillment of the requirements for the academic degree of
Doctor rerum naturalium (Dr. rer. nat.)

Representation and Learning of Context-Dependent Choice Functions

Karlson Pfannschmidt

1st Reviewer Prof. Dr. Eyke Hüllermeier
Institute of Informatics
Ludwig Maximilian University of Munich

2nd Reviewer Prof. Dr. Heike Trautmann
Department of Computer Science
Paderborn University

Supervisor Prof. Dr. Eyke Hüllermeier

July 8, 2024

Representation and Learning of Context-Dependent Choice Functions

Authorship
This dissertation was authored by Karlson Pfannschmidt. Minimal use of thesauri and language models was made for spell-checking and minor wording refinements; however, the content remains entirely original and independently created.

Colophon
This document was typeset with the help of KOMA-Script and LaTeX using the kaobook class.

Abstract

This thesis addresses the learning of preferences, particularly *choices*, within the machine learning framework. It is positioned at the intersection of machine learning, behavioral economics, and psychology. Preference learning, a growing subfield of machine learning, has many vital applications, such as information retrieval, recommender systems, and, recently, aligning language models and agents with human preferences.

We consider the choice setting, where a decision maker is presented with a set of objects and expresses a choice by selecting the subset of objects they prefer. Given observations of sets of objects and the chosen objects, the overall goal is to learn the function that maps from sets of objects to subsets that can generalize to unseen sets of objects. Many models make the simplifying assumption that decision makers evaluate the utility of each object independently. However, this conflicts with experimental findings from psychology and behavioral economics, which indicate that the *context* in which choices are made significantly influences the choices.

Therefore, we set out to study how to capture the influence of the surrounding objects on a choice. We propose to generalize the notion of utility in two complementary ways. The first methodology explicitly aims to incorporate the set-based context into the assessment of the utilities. We propose two decomposition approaches to make the learning of such a generalized utility function feasible. The first approach, First Evaluate Then Aggregate (FETA), decomposes the generalized utility into an aggregation of lower-order sub-utilities. The second approach, First Aggregate Then Evaluate (FATE), first computes a representation of the set of objects and then evaluates the utility of each object within the context of this representation. For both utility decompositions, new choices can be predicted by selecting objects whose utility exceeds a certain threshold.

The second methodology resulted from the question of what would happen if we moved from a one-dimensional to a multi-dimensional utility function. This extension naturally leads to a choice function for subset selections by identifying objects on the Pareto-front of a given set, eliminating the need to specify or tune a threshold.

We analyze mathematical properties for each of the models we develop, such as expressive power and identifiability. We further derive suitable differentiable loss functions and design appropriate neural network architectures. Next, we extensively compare these neural networks against state-of-the-art approaches across a variety of datasets to assess their effectiveness in different settings. Our results demonstrate that our approaches perform very favorably in learning tasks with a high degree of context-dependence.

Zusammenfassung

Diese Dissertation behandelt das Lernen von Präferenzen, und dabei insbesondere von *Auswahlen*, im Rahmen des maschinellen Lernens. Sie ist an der Schnittstelle von maschinellem Lernen, Verhaltensökonomie und Psychologie angesiedelt. Präferenzlernen, ein wachsendes Teilgebiet des maschinellen Lernens, hat viele wichtige Anwendungen, wie beispielsweise im Bereich der Information Retrieval, Empfehlungssysteme und jüngst bei der Ausrichtung von Sprachmodellen und Agenten an menschlichen Präferenzen.

Wer betrachten das Auswahl-Setting, bei dem einer Person oder einer abstrakten Auswahlfunktion eine Menge von Objekten präsentiert wird und eine Auswahl durch die Selektion einer Teilmenge dieser Objekte getroffen wird. Ziel ist es, aus Beobachtungen von Objektmengen und den ausgewählten Objekten eine Funktion zu lernen, die von Objektmengen auf Teilmengen abbildet und auf bisher ungesehene Objektmengen verallgemeinern kann. Viele Modelle treffen die vereinfachende Annahme, dass der Nutzen jedes Objekts unabhängig voneinander bewertet wird. Dies steht jedoch im Widerspruch zu experimentellen Erkenntnissen aus der Psychologie und Verhaltensökonomie, die darauf hinweisen, dass der *Kontext*, in dem Entscheidungen getroffen werden, die Auswahlen erheblich beeinflusst.

Daher haben wir untersucht, wie der Einfluss der umgebenden Objekte auf die Auswahl erfasst werden kann. Wir schlagen vor, den Begriff des Nutzens auf zwei komplementäre Weisen zu generalisieren. Die erste Methodik zielt ausdrücklich darauf ab, den kontextbasierten Nutzen in die Bewertung der Nutzwerte einzubeziehen. Um das Lernen einer solchen generalisierten Nutzenfunktion machbar zu machen, schlagen wir zwei Zerlegungsansätze vor. Der erste Ansatz, FETA, zerlegt den generalisierten Nutzen in eine Aggregation von niedrigeren Teilnutzwerten. Der zweite Ansatz, FATE, berechnet zunächst eine Repräsentation der Objektmenge und bewertet dann den Nutzen jedes Objekts im Kontext dieser Repräsentation. In beiden Ansätzen können neue Auswahlen getroffen werden, indem Objekte ausgewählt werden, deren Nutzen einen bestimmten Schwellenwert überschreitet.

Die zweite Methodik ergab sich aus der Frage, was passieren würde, wenn wir von einer eindimensionalen zu einer mehrdimensionalen Nutzenfunktion übergehen. Diese Erweiterung führt natürlicherweise zu einer Auswahlfunktion für Teilmengen, indem Objekte auf der Pareto-Front einer gegebenen Menge identifiziert werden, wodurch die Notwendigkeit entfällt, einen Schwellenwert festzulegen oder zu optimieren.

Für jedes der entwickelten Modelle analysieren wir zunächst theoretische Eigenschaften wie die Ausdrucksstärke und die Identifizierbarkeit. Außerdem leiten wir geeignete differenzierbare Verlustfunktionen her und entwerfen passende neuronale Netzwerkarchitekturen. Wir vergleichen diese neuronalen Netzwerke umfangreich mit dem Stand der Technik mittels einer Vielzahl von Datensätzen, um ihre Effektivität in unterschiedlichen Settings zu bewerten. Unsere Ergebnisse zeigen, dass unsere Ansätze bei Lernaufgaben mit einem hohen Grad an Kontextabhängigkeit sehr gut abschneiden.

Acknowledgments

First and foremost, I extend my deepest gratitude to my Ph.D. supervisor, Prof. Eyke Hüllermeier, who not only introduced me to the field of preference learning but also inspired my research trajectory. His patient guidance and our many years of fruitful discussions have been invaluable to my development as a researcher. In particular, I appreciate his sharp mind combined with an excellent sense for promising research questions, which have shaped my research over the years.

I am immensely grateful for the inspiring conversations and collaborations with my fellow Ph.D. students, starting on the O4 floor of Paderborn University, later moving to the Zukunftsmeile 2 building. The camaraderie and intellectual exchange with the wonderful people in these groups, as well as the memorable research retreats with the Munich team, have enriched my academic journey. I will fondly remember our social activities, ranging from calm archery, bowling, and running to exciting laser tag. These experiences provided wonderful opportunities to get to know you all better.

I would also like to thank all my co-authors. The unique synergy that emerges when great minds collaborate is still special to me. Your contributions and insights have been instrumental in advancing my work. I am confident that our exchange of ideas has been mutually beneficial, driving progress in our respective research areas.

A special thanks goes to my diligent proofreaders, Vitalik Melnikov and Alexander Tornede. Your meticulous feedback, often provided within much too tight deadlines, has been invaluable. I am grateful for the time you invested in improving my work. Vitalik has been my longtime office mate throughout the years. I always appreciate his astute questions that quickly get to the core of an issue. I look forward to our continued collaboration in our new joint endeavor, hoping it will be even more fruitful.

I would also like to thank Elisabeth Lengeling, our dear secretary, whose consistent support and dedication have fostered a warm and friendly atmosphere within the group. Her proactive approach to problem-solving was invaluable; whenever an issue arose, she would swiftly work to resolve it without hesitation. Elisabeth's efforts have not gone unnoticed and are deeply appreciated.

Finally, I want to express my deepest gratitude to my family: to my parents for their ongoing support over the years and to my wife, Julia, for her unwavering encouragement and patience throughout the long writing process. Julia, you have been my anchor. Your support has been indispensable, and I am profoundly grateful to have you by my side.

Contents

List of Figures

List of Tables

1. Introduction

Imagine the following situation: Your trusty headphones recently broke, and you are looking for a replacement. You are not familiar with the ins and outs of headphones and what to look for and consult with a large language model (LLM) for a set of useful criteria to compare headphones. The language model generates two lists of criteria and asks you which of those you prefer. You pick the first list, feeling it aligns better with your needs. You head over to your favorite search engine and begin searching. A few results appear, and one in particular catches your eye. You click on it. Upon visiting the site, the headphones displayed do not quite meet your expectations, but an advertisement for another pair of headphones grabs your attention. You click on the ad and are directed to an online store. After a bit more browsing, you decide not only to purchase the headphones but also a case that was recommended at the checkout.

Figure 1.1.: An illustration of a person who is thinking about new headphones.

You might not realize that through these actions, you have generated a wealth of *preference data*. These data are invaluable for training and improving machine learning models designed to tackle various tasks. When you indicated your preference for one set of criteria over another while consulting the LLM, you provided *pairwise preference* data, which can be used to better align future models with user intentions. Furthermore, your interaction with the search engine–specifically the click on your *chosen* link–is a piece of information for refining the algorithms that power information retrieval. The advertisement placement that caught your eye is the result of preference-based bandit algorithms designed to maximize the likelihood of a click. Finally, the additional product recommendation at the checkout is a direct application of *recommender systems*. These systems are ubiquitous in modern e-commerce and help connect customers with products they are likely to buy.

As we can see, learning from preference data is of central importance in the field of machine learning, as it spans several relevant applications. What all of these applications have in common is that we are observing *preferences* of individuals in a certain *context*. Most of these preferences are implicit, as we observe how users interact with a system and assume that certain actions indicate a preference for some alternatives over others.

Now, certainly, researchers in machine learning have not invented the field of preference modeling. The study of human preferences goes back to the 18th century when researchers began considering the valuation of alternatives in the context of gambling [Mon13; Ber38]. In the 19th century, Fechner [Fec60] pioneered psychophysical research. In particular, the concept of "just noticeable difference" played a crucial role in shaping the foundations of preference models by integrating psychological insights into economic decision-making. The next revolution happened in the 20th century, when Pareto [Par06], Von Neumann and Morgenstern [VM47], and Luce [Luc59] developed robust axiomatic foundations for theories of utility based on human preferences. Random utility models (RUMs) like the multinomial logit (MNL) model were developed in in the first half of the 20th century and then established in

economic theory by McFadden [McF74]. These allowed practitioners to infer utility functions from preference data, facilitating the explanation of behavior and predicting future preferences.

However, in a number of empirical studies it was observed that in some situations, individuals deviated from what was deemed "rational" behavior [Coo58; Chi60; Kra67; BDM63; TE69; HPP82; HP83; RSS87; Sim89]. What do we mean by rational here? In general, it is posited that a rational decision maker acts according to a utility function. Alternatives are judged according to this utility function, and the best one is chosen based on the principle of *utility maximization*. The combined axioms underlying the aforementioned axiomatic approaches imply the existence of a utility function. It was, therefore, common to say that if decision makers act in a way that violates one or more axioms, their behavior is "irrational". Systematic experimentation revealed that the preference *context* has a noticeable effect on the preferences.

context effects

Several distinct *context effects* were identified that reliably caused such deviations in individuals. As an example, consider a situation where a customer is choosing between three different laptops, priced at 1000 €, 1500 € and 2000 €. We often observe that in such a situation, the customer is more likely to choose the middle option because it is seen as a compromise between the other two. This is known as the *compromise effect* and is a well-documented context effect in the literature [Sim89]. In Section 2.3, we will discuss several other context effects that have been observed in the literature. These context effects demonstrate that human choice behavior may not simply be represented by a straightforward utility function that only judges each alternative individually. Rather, utilities need to become *context-dependent* to account for such context effects.

compromise effect

Motivated by these observations, researchers devised utility models that could account for some of the context effects (see Section 2.4). However, these models are often limited in their expressive power and are not able to capture all context effects. This includes context effects that are not yet known or are difficult to model explicitly. In this thesis, we aim to develop models that can capture a wide range of context effects and are able to adapt to new context effects.

While the main focus of this thesis is on modeling human preferences with arbitrary context effects, there is a good argument to be made that these models can also be applied to other settings where preferences are expressed. It is fruitful to generalize this concept to the inference of abstract preference functions. One example is to observe and learn from the preferences expressed by a hard-to-compute optimization algorithm (in the typical sense of computational complexity). In such cases, the optimal selection is highly contextual, and it is difficult to capture these context effects explicitly. We, therefore, require models that can flexibly adapt to any context effects present in the data. Generally speaking, we can treat this as a problem to infer an unknown algorithm that can only be observed by its preferences. Modeling human preferences is then simply a special case where each individual decides on the preferred alternatives based on some underlying decision algorithm.

preference learning

In the machine learning literature, the problem of learning from preferences is often referred to as *preference learning* [FH10]. The goal of preference learning is to infer a model that can predict the preferences of a decision maker based on observed data. The subfield has already seen a surge in interest in recent years,

as it is a central component in many applications, such as information retrieval, recommender systems, and online marketing [HS24]. However, the most influential applications of preference learning are in the field of *reinforcement learning*, where it was used to train the reward function of an agent with the preferences of a human supervisor [Chr+17]. This method was later used to align the large language model GPT-3 with human preferences, resulting in InstructGPT, now known as ChatGPT [Ouy+22]. Learning from implicit behavior will be increasingly important as we are confronted with larger and more complex datasets without the ability to label all of them manually.

In the preference learning literature, numerous approaches in various learning settings have been proposed to model preferences. Each with their own advantages and tradeoffs [FH10; HS24]. Specific learning settings are distinguished based on which features are available, the modality of preference feedback (e. g., rankings, choices or ordinal scores), and the type of learning paradigm (e. g., supervised, unsupervised, reinforcement learning, etc.). An explicit or implicit assumption many approaches make is that each alternative can, in some way, be assigned a *utility*, and the observed preferences result from a preference process operating based on these utilities. As we will see later in Chapter 4, this causes the majority of these approaches to be *context-independent*, i. e., the other alternatives present in the context do not play a role in the utility assessment. In typical web contexts, users are often confronted with several options, making it plausible that context effects play a role and need to be accounted for.

utility

The main goal of this thesis is to address the aforementioned problems in the context of supervised preference learning. We investigate whether it is possible to efficiently learn general, *context-dependent* preference models. Specifically, we will focus on settings where the preferences are expressed as *choices*. A choice is a preference, where a decision maker is confronted with a set of objects from which they select the most preferred object or subset of objects.

Learning from subset choices as a generalization of singleton choices is—in the context of choice modeling as a whole—a recent endeavor [BKT18]. Subsets allow us to cover more complex preferences, where the optimal decision is to select a set of objects rather than a single one. Take, for instance, the choice of a portfolio of investment opportunities. Depending on the specific investment goals, a diversified set of investments might be preferable to a single one to maximize a certain reward-to-risk ratio. Subset preferences may also indicate indifference with respect to the order of objects, which can occur if the objects have certain properties that can be traded off against each other. However, modeling such preferences is not straightforward, as the number of possible subsets grows exponentially with the number of objects. This is why one goal of this thesis is to develop models that can efficiently predict subset choices.

In the following Section 1.1, we will first detail the research questions that guided the research presented in this thesis and the specific objectives that were derived. Then, we highlight the main contributions of this thesis in Section 1.2 and discuss the potential impact on the field of preference learning. Given that the intended audience of this thesis encompasses experts in machine learning and preference learning, as well as researchers from relevant fields utilizing preference models, such as economics, psychology, marketing research, and operations research, the thesis includes several foundational chapters. These

chapters are designed to provide a comprehensive background for our diverse readership. Therefore, the thesis provides several preliminary chapters to provide the necessary background to this diverse readership. We present dependencies between chapters and the suggested sequence for reading the thesis in Section 1.3. Finally, it is clear that good research is done in conjunction with multiple co-authors, which is why in Section 1.4, it is specified which parts of each chapter were contributed by co-authors.

1.1. Research Questions and Objectives

Given that the main goal of this thesis is to research efficient *context-dependent* models for preferences, we derive the main *research questions* from this goal. These guided the research that was performed over the course of the thesis. For each research question, we also detail *objectives*. These are certain tasks or experiments that need to be performed to answer the research question at hand. In the main chapters of this thesis, we may further specify research questions for each investigation. However, here, we only consider the overarching themes.

Research Questions (RQs)

RQ1: Which approaches exist in preference learning, and of those, which have a mechanism to account for context effects?

RQ2: What are different efficient and novel mechanisms to capture the context in a choice setting?

RQ3: What is the expressive power of these models, and which other interesting theoretical properties can be shown?

RQ4: How can these models be inferred from data, and how do they compare to existing models?

Elaboration and Objectives We will now explore each research question in greater detail and present the planned objectives for the thesis. As with all research endeavors, gaining a thorough understanding of the existing literature is important. With RQ1, we therefore want to know which approaches exist in the domain of preference modeling and learning. This should cover both the historical context and state-of-the-art developments. It is especially important to review the literature in different fields that employ preference modeling since modern preference learning is a fast-growing research area. This lets us properly contextualize new developments and establish bridges between different fields. The scope of this literature analysis is to provide a thorough overview of the main themes in each research area. However, a systematic literature review, which would require the exhaustive analysis and aggregation of the entire literature, is outside the scope of this thesis. Therefore, for RQ1, the objectives are straightforward:

RQ1

1. Perform a literature review of classic utility theory, important axiomatic approaches, and choice models.
2. Perform a literature review of the problems and approaches tackled in preference learning.

As for RQ2, the aim is to develop novel ways in which context-dependent choices can be modeled. In addition, we want to be able to learn from both singleton and subset choices, which would make these approaches more generally applicable. Besides these primary goals, the models should also fulfill secondary requirements. Firstly, the developed models should be computationally efficient so that they can be utilized in practice. Secondly, the models should be designed with end-to-end training in mind. With this, they can be used as part of the larger model, allowing, for instance, the use of different input transformations (e. g., convolutions for image processing). The main objectives are: RQ2

1. Formally define the framework of context-dependent choice.
2. Develop principled approaches capable of learning from context-dependent, and both singleton and subset choices.

For each approach that we develop, we also want to know about its theoretic properties (see RQ3). A property that is often analyzed in the context of preferences is the *expressive power* of a model. We want to know which kind of preferences it can represent, giving us insight into the flexibility/inductive bias of the model. We may want to investigate different theoretical properties depending on the specific models we develop. Therefore, we specify our objectives suitably broadly to be able to adapt these on the fly. For RQ3 these are: RQ3 expressive power

1. For each developed approach, analyze the expressive power.
2. Analyze interesting properties of loss functions that are used to fit these models to data.

Throughout this thesis, we will maintain a clear separation of the (abstract) general models and their specific *implementation*. Here, the term "implementation" encompasses not only the implementation in a particular programming language but also the *instantiation* of a model using a parametric model, such as neural networks. This allows us to analyze the fundamental properties of the general model without having to impose any additional properties that are inherent to a given instantiation. Regarding RQ4, our aim is to find out how each model can be inferred from a given dataset. We will propose a suitable instantiation for each model in pursuit of this objective. Furthermore, a critical aspect of this thesis involves conducting an experimental comparison between these models and established models documented in the literature. Such comparisons are crucial to identify advantages and trade-offs associated with each approach. Joining all these requirements, we formulate the following objectives for RQ4: RQ4

1. Instantiate the approaches developed for RQ2 using appropriate machine learning models, loss functions, and inference methods.
2. Develop rigorous experimental protocols to ensure a comprehensive and fair comparison. This includes selecting a diverse array of datasets (both real-world and synthetic), baseline approaches, evaluation measures, and auxiliary experiments.
3. Implement all proposed approaches and conduct empirical evaluations, adhering to the highest standards of experimental design in machine learning algorithm comparison.
4. Execute the planned experiments and engage in a detailed analysis and discussion of the findings.

In conclusion, the primary goal of this research is to establish the concept of context-dependent (subset) choice as a new branch within the broader field of preference learning. We also want to bootstrap this branch with an array of approaches accompanied by both theoretical and empirical results. In the following section, we will detail the main contributions that have emerged from this research. We will also touch on the potential impact this work can have on the field of preference learning and preference modeling as a whole. Additionally, we will touch on the broader implications of this work, specifically its potential to influence and reshape the landscape of preference learning and preference modeling.

1.2. Contributions and Potential Impact

In this section, we will now give an overview of the major contributions that were achieved over the course of this thesis. To this end, we will also reference the corresponding chapters and sections for easy access. For general advice on the thesis outline and reading order, please refer to Section 1.3. Then, we will analyze the broader potential impact of this thesis within the field of preference learning, preference modeling, and applications.

1.2.1. Main Contributions

Novel Models for Context-Dependent Choice The main contribution of this thesis is the development of *three* complementary approaches that can model context-dependent choice. Each approach can, in some way, account for context effects present in the data. Furthermore, each method can model *subset choices* without the need to restrict the space of admissible choice sets or to optimize likelihood over a space of choice sets with an exponential size.

generalized utility: Section 5.1

We derive the first two approaches from a *generalized utility framework*. Central to this framework is a generalized utility function, that maps from an object and the entire task context (meaning all other objects available during the choice) to a real-valued utility. To make learning such a function feasible, we propose two decompositions of this generalized utility function, FETA and FATE. The former decomposes the utility function into an average of lower-order utility functions. In particular, the pairwise decomposition is of interest since it balances computational overhead and expressive power. Our FATE decomposition instead first computes a representation of the task context. Then, each object is evaluated in the context of this representative.

FETA: Section 5.3.1
FATE: Section 5.3.2

thresholding: Section 5.2.1

In this generalized utility function framework, we propose to represent subset choices by employing *thresholding*, a successful technique employed in multi-label classification. This approach offers two key advantages: it can naturally produce subsets of any size, and it does so with computational efficiency, as it eliminates the need for optimization over subsets.

In Chapter 6, we introduce our third approach, which uses a methodology orthogonal to FETA and FATE. Instead of modeling preferences using a one-dimensional but contextual utility function, it represents each object in a multi-dimensional utility space. Subset choices then emerge naturally as the Pareto set in this space. We call functions that map objects into such a

utility space *Pareto-embeddings*. Furthermore, we also introduce a pairwise Pareto-embedding approach that is able to adapt the embedding based on the task context, similar to pairwise FETA.

Pareto-embeddings

To summarize, the resulting three approaches and their inductive biases complement each other. FETA is advantageous if the observed choices arise from the valuation of the objects in smaller contexts, such as a human that compares pairs of objects and then chooses the object that performs well in all comparisons. FATE supposes that the complete set of available objects can be condensed into an efficient representation such that it is enough to judge the utility of an object in the context of this set representation. This can be beneficial in situations where the choices express preferences relative to properties of the whole set, e. g., choosing a set of politicians that best reflects the overall views of all politicians. Finally, with the Pareto-embedding approach, we are able to learn a low-dimensional utility representation of the objects. This is particularly well-suited for practical situations where objects are judged based on several criteria that may be traded off against each other. For all three approaches, we analyze mathematical properties and find that they achieve excellent empirical results (more on this below).

Theoretical Results Concerning the New Models We analyze the ability of the models to represent different choice functions. We find that the FATE decomposition is able to represent *any* choice function. This flexibility of FATE comes at the cost of *identifiability*, which we also prove.

Theory FATE: Section 5.3.2.1

For the pairwise FETA decomposition, we show that there exist choice functions that cannot be represented. This indicates the existence of a certain pairwise inductive bias. In addition, we show that the pairwise FETA decomposition is identifiable up to the choice of the 0th order utility. We investigate the inductive bias further by empirically comparing the performance of the pairwise FETA decomposition to the more general FATE decomposition in terms of data efficiency. We find that the pairwise FETA decomposition needs less data to learn a particular choice function than the FATE decomposition.

Theory FETA: Section 5.3.1.2

We propose instantiations of the models using neural networks. To this end, we define *convex* target losses that should be minimized and surrogate losses that can be optimized using a gradient descent procedure.

For the Pareto-embeddings, we prove that not all choice functions can be represented. As a follow-up question, we are interested in what kind of choice functions can be represented and if there is a defining property of such choice functions. We define the *Pareto-property* on the level of choice functions that ensures the existence of a certain partial order relation. We then prove that any choice function that has the Pareto-property can be represented by Pareto embeddings. This means we have identified a class of choice functions that arise naturally as a generalization of the maximum utility principle, and we know how they can be represented.

Pareto-property: Definition 6.2.4

Similarly to the other models, we propose an instantiation of the model and a differentiable surrogate loss function. We prove that a surrogate loss of 0 guarantees that a Pareto-embedding has been identified. Furthermore, we show that it suffices to minimize the loss on pairs of objects, which is a desirable property for optimization in practice.

Implementation and Evaluation of the New Models Our proposed models can be instantiated using a wide variety of parametric models. We use *neural networks* (called FETA-Net and FATE-Net) in conjunction with suitable surrogate losses. This choice comes with several important advantages. Firstly, the models can be trained in an end-to-end fashion for a variety of structured inputs. Numeric input data as well as text, images, etc., can be utilized by passing inputs through specialized neural network layers. Secondly, neural networks are powerful nonlinear models that can represent arbitrarily complex target functions. This is useful when we want to learn complicated context effects or highly nonlinear mappings of the inputs to choices.

We then test our models in comprehensive empirical evaluations. Here, the main goal is to evaluate the models and not the idiosyncrasies of specific neural network architectures. Therefore, we make sure to use neural network architectures as straightforward as possible. The remaining choices of hyperparameters, such as number of units, layers, etc. are optimized in a data-driven fashion.

Evaluation FETA-Net/FATE-Net: Section 5.5

We empirically test the generalized utility approaches extensively against a suite of representative approaches from the literature. As datasets, we use both real-world datasets and (semi-)synthetic datasets. We observe that both our approaches are competitive with existing models on the former. On the latter set of datasets, we see a considerable improvement in performance for the context-dependent models, with our methods beating state-of-the-art approaches.

Evaluation Pareto: Section 6.3

Similarly, we test our Pareto-embedding approach against FETA-Net and FATE-Net and a set of competitive baselines. The datasets here are a benchmark suite of hard test functions commonly employed in the field of multi-objective optimization to test optimization algorithms. We observe that both the Pareto-embedding and the pairwise Pareto-embedding approach outperform the competition on this suite. This demonstrates that more specialized models with certain inductive biases, such as the family of Pareto-embedding models, can significantly outperform more general models, such as FETA-Net and FATE-Net, on certain problems in practice.

1.2.2. Potential Impact

subset choices

In the field of preference learning, modeling of choices and specifically *subset choices* is a recent development [BKT18]. Existing approaches for subset choice, such as the one by Benson et al. [BKT18], lacked an efficient mechanism for predicting subsets. Our approaches, however, bridge this gap and are able to predict subsets within polynomial or even linear time. This is done while still taking the interactions of the objects in the given task into account. This advancement in computational efficiency sets the stage for wider adoption in practical applications such as e-commerce or market research.

context-dependence

Similarly, the dimension of *context-dependence* has remained largely unexplored in the field of preference learning until now. Existing models in the prevailing literature on context effects are limited in their capability to model context effects, thereby limiting their applicability. As we will see in Chapter 4, a common assumption in the preference learning literature is the independent evaluation of utilities for each object. In contrast, the context-dependent

models introduced in this thesis are strictly more expressive and enable the modeling of non-standard human choice behavior or more complex choice functions.

1.3. Thesis Outline and Reading Order

While the main focus of this thesis is contributions to the field of preference learning, the insights gained here extend beyond this subfield of machine learning. As such, the thesis is written with a diverse readership in mind, and additional preliminary chapters have been included to provide the necessary context for understanding the main chapters.

The chapters and dependencies between them are outlined in Figure 1.2. Three preliminary chapters are designed to provide context for readers of different backgrounds.

Readers with a Background in Machine Learning For readers with a background in machine learning, we give an overview of seminal results concerning utility theory in Chapter 2. This includes *ordinal utility theory* as introduced by Pareto [Par06], *cardinal utility theory* made viable by Von Neumann and Morgenstern [VM47] and Luce's *choice axioms* [Luc59]. In the chapter, we also highlight the shortcomings of these axiomatic approaches, leading to the *context effects* that were identified in a series of experiments. Readers with a background in machine learning may be unfamiliar with the literature on classical *preference modeling* we give an overview in Section 2.4. This includes both context-independent and context-dependent models.

ordinal utility theory: Section 2.2.1.1
cardinal utility theory: Section 2.2.1.2
choice axioms: Section 2.2.2
context effects: Section 2.3
preference modeling: Section 2.4

Readers with a Background in Economics, Psychology and Related Fields For this group of readers, we include a preliminary chapter about important *concepts from machine learning*. We start with the underpinnings of *generalization* and how models are *validated* to estimate their generalization performance. In the experimental sections of this thesis, we employ extensive *optimization of hyperparameters* to ensure a fair comparison between all evaluated approaches. In Section 3.2, we therefore explain why this is a problem and how it can be solved efficiently. Lastly, the models developed over the course of this thesis and also many of the comparison approaches are instantiated using *neural networks*. Since neural networks are a vast field, we give an overview of the necessary background in Section 3.3. The focus being on how a suitable *architecture* and a *task-dependent loss* can be combined to flexibly tackle a wide variety of learning problems.

concepts from machine learning: Chapter 3
generalization: Section 3.1
validation: Section 3.1.2
neural networks: Section 3.3

Readers Unfamiliar with Preference Learning *Preference learning* is a machine learning subfield focused on learning from preference data. Its many applications–from implicit feedback models in e-commerce to state-of-the-art direct preference optimization (DPO) [Raf+23] used to train the current generation of LLMs–make it a highly relevant subfield in machine learning. Chapter 4 begins by introducing the core learning problems and how they differ with respect to the type of preference data considered and the target function. Next, we define evaluation measures that are used to judge the

preference learning: Chapter 4

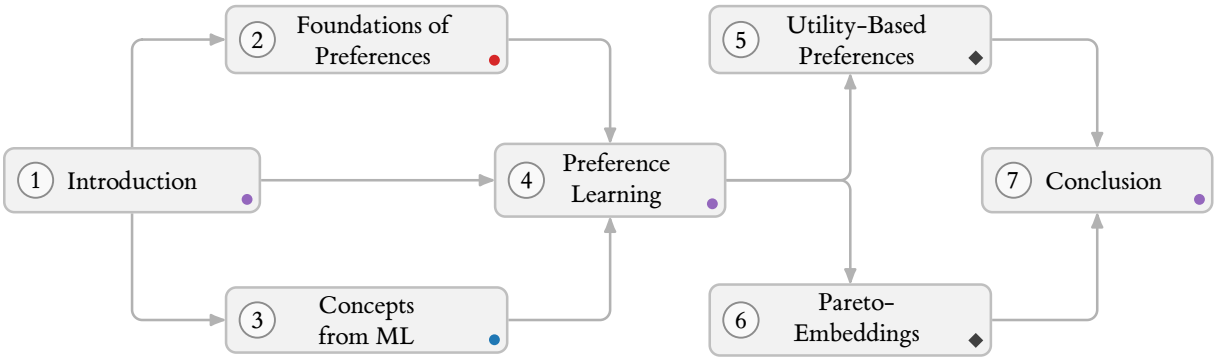


Figure 1.2.: Overview of the thesis chapters and their dependencies. The main content is contained within the chapters marked with a diamond. The blue and red circles indicate preliminary chapters recommended for readers unfamiliar with machine learning or preferences. Chapters marked by a purple circle are recommended for all readers.

solution approaches: Section 4.3

quality of different learners, focusing on those that are relevant to this thesis. We also highlight the most important *solution approaches* in Section 4.3, and how they broadly can be categorized.

Chapters Containing the Main Contributions In Chapter 5, we introduce the notion of a *generalized utility function* and detail how it can be used to model a variety of choice functions. As it is infeasible to learn this utility function directly, we propose two methods on how it can be decomposed into aggregations of tractable utility functions. The first decomposition, FETA, evaluates the utility of each object in all subcontexts, and these sub-utilities are then averaged to arrive at the overall utility of an object for a given choice task. Our second decomposition, FATE, first produces a representation of the task context, and then a utility of each object is computed in the context of this representation. We propose suitable neural network instantiations of these models in Section 5.4. The setup and results of the extensive *empirical evaluation* of these models are presented in Section 5.5.

generalized utility: Section 5.1

FETA: Section 5.3.1

FATE: Section 5.3.2

We develop a conceptually orthogonal approach in Chapter 6. This new concept we call *Pareto-embedding* and define it in Section 6.2. The idea is to map objects into a new space such that the Pareto-optimal sets of objects correspond to the chosen subsets. In Section 6.2.1, we propose a procedure for learning such embeddings from data, including a surrogate loss and neural network architectures. We then analyze theoretical properties in Section 6.2.2. We compare the Pareto-embedding approaches to other baseline models in a thorough experimental evaluation. The experimental setup and the results are described in Section 6.3.

Pareto-embedding: Section 6.2

1.4. Co-Author Contribution Statement

Academic research thrives on collaboration, which is why I want to use this section to acknowledge the contributions of my co-authors to our joint publications. I am profoundly grateful for the expertise and support of my co-authors, who have been pivotal in achieving the comprehensive results presented in this thesis. In the following, the specific contributions are listed per chapter.

Chapter 5: Utility-Based Preferences

- Pritha Gupta helped in the implementation of the experimental evaluation and in the subsequent reporting thereof in paper drafts and in the final publication.
- Dr. Björn Haddenhorst collaborated with me on the theoretical results of this chapter and contributed to the writing of the final publication.
- Prof. Eyke Hüllermeier contributed on a conceptual level and helped revise paper drafts and the final publication.

Chapter 6: Subset Choices using Pareto-Embeddings

- Prof. Eyke Hüllermeier contributed on a conceptual level and helped revise paper drafts.

1.5. Other Publications

Throughout my time as a PhD student, I collaborated on a number of publications that are not showcased as part of this thesis. These works, while tangential to the core focus of my dissertation, significantly contributed to broadening my research perspective and enhancing my understanding of the field. They encompass a variety of topics that, although not directly related to the thesis, are relevant to the broader discourse in machine learning. The insights gained from these collaborations have informed my approach and methodology, providing a richer context to the research presented herein. Therefore, I include a detailed list of these publications, along with brief summaries, in Appendix B for interested readers seeking to explore other avenues of research I pursued during my doctoral studies.

Preliminaries

2. Foundations of Preferences

Imagine the following situation: We are in a small town where almost everyone knows everyone else. A hatmaker is situated right in the center of town and sells extravagant hats. Our protagonist sees two particularly flamboyant hats in the display window, a blue and a red one. They like flamboyant hats because it allows them to show off. Since they slightly prefer the color red, they decide to buy it.

Figure 2.1.: A person has to make a decision between two hats.

What we have observed here is a *preference*. Our protagonist preferred the red-colored hat over the blue-colored hat. In addition, they also liked flamboyant hats in general, so we can assume that they implicitly prefer them over all other basic hats. It would be useful if we could assign a number to each hat, representing the utility it provides to our protagonist. In our case we know that the utility of the red hat has to be greater than the utility of the blue hat. In addition, we can posit that the utility of both hats is high overall since they are preferred over most other hats. We do not however know how such a mapping to utilities could be computed, which assumptions would need to hold and whether such a mapping is unique.

Utility theory has developed over the last centuries to answer these and more related questions. Several axiomatic approaches exist that impose different sets of assumptions on the preferences which can then be used to prove that a utility function that is consistent with the preferences does indeed exist. These theories were developed and proved especially during the 20th century and were instrumental in shaping modern economic analysis [Hea+92]. They provided a rigorous mathematical foundation for understanding consumer choice, market behavior, and the allocation of resources. Furthermore, these developments helped bridge the gap between abstract economic theory and real-world economic practices, enhancing the predictive power and practical relevance of economic models.

[Hea+92]: Heap et al. (1992), *The Theory of Choice: A Critical Guide*

Now, coming back to the situation from Figure 2.1. Assume instead that there is another red hat in the shop's display that looks very similar to the existing one. Our protagonist, if their goal is to show off and to be the only one wearing this particular hat, may now switch their preference to the blue one instead. Even though they prefer red hats in general, the fact that it is no longer unique changed their opinion. This is what is known in the literature as a *context effect*. The addition of a third object to the choice context had an effect on the preference of the original two objects. It should be clear that in this situation, it is no longer possible to assign a global utility to each object since that would imply the same ordering of the two hats, regardless of the context. Even though this is a fictional situation, these context effects can also be observed in practice and are at odds with the requirements imposed by the formal axioms.

context effect

In this chapter, we want to shine a light on the history of utility theory and its axiomatization, as well as the context effects that were observed. As the main focus of the thesis lies on choice modeling in the context of machine learning, we assume that the typical reader is not that familiar with the classical

Chapter overview

foundations of preferences and utility. In Section 2.2, we will therefore give an overview of the typical *axiomatic approaches* and their consequences. We will also see the shortcomings of these axioms and why they sometimes fail in practice in Section 2.3, giving the reader an overview of the most common *context effects* that were identified. Finally, in Section 2.4, we introduce how utility theory is commonly applied in practice and present so-called *random utility models*.

2.1. Introduction to Notation

Before we introduce the history of preferences and the axioms of utility theory, it is important to introduce the notation that will be used throughout this and subsequent chapters. This section is designed to be self-contained, serving as a reference to revisit as needed.

The setting we will consider in this thesis is very general. As is apparent, we want to be able to represent all these different settings formally. We will start by restricting ourselves to one particular problem. For a given problem, we will consider a *reference set* of *objects* or *alternatives*, which we will denote by X. In our motivating example shown in Figure 2.1, X could refer to all hats in existence. Each of the objects are hats that have certain attributes, such as color, extravagance, etc. As is typical, an machine learning, we will call these attributes *features*. Thus, each object can be regarded as a vector $\boldsymbol{x} = (x_1, \dots, x_d)$, and we will let X be a subset of $\mathbb{R}^d$. Note that we use the term *object* here to refer to the items that are being compared. In the literature, these are also called *alternatives* or *items*. The person or process that is expressing a preference over these objects is typically called the *decision maker*.

reference set

In some application contexts, the typical assumption is that each of the attributes contributes monotonically to the overall value of the object. Since it is neither necessary nor desirable, we will not make this assumption here. This is particularly relevant in machine learning, where it is usually not a priori known whether the impact of the features is monotonic.

While X, being a subset of $\mathbb{R}^d$, can be a set of infinite size, we will usually assume, for simplicity, that X is finite. Certainly, a decision maker will not be confronted with the full set X, each time they have to express a preference. Rather, we will consider subsets $Q \subseteq X$ to be *decision tasks* (task for short) for which the decision maker has to provide their decision[1]. With *decision* we mean the generic expression of a preference, which can, for example, be a choice, a ranking or a utility value.

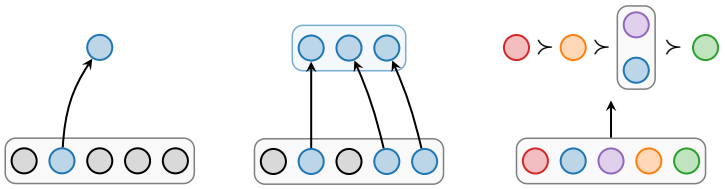
1: We can easily recover the setting where the decision maker is always confronted with all objects by setting $\mathcal{Q} := \{X\}$.

decision tasks

The non-empty set, containing all of these tasks, we denote by $\mathcal{Q} \subseteq 2^X \setminus \{\emptyset\}$. Consequently, if a family of subsets $\mathcal{Q}$ satisfies $\emptyset \notin \mathcal{Q} \neq \emptyset$, then we call it a *decision task space* . Depending on the specific preferences expressed by the decision maker, we may choose to specialize these notions. For example, if the observed preferences are choices, we will call $\mathcal{Q}$ the *choice task space*.

decision task space

Intuitively, the task space defines the kind of situations the decision maker is confronted with and impacts the difficulty of the decision problem. For example, if the task space is the set of all objects X, it means one must always express a preference over all objects. On the other hand, if the task space is the set of all possible subsets of size 2, it means one only has to compare pairs of



(a) Singleton choice (b) Subset choice (c) (Partial) ranking

Figure 2.2.: Common modes of expressing preferences.

objects. In practice, the task space can be controlled by the system designer to achieve a good trade-off between the expressiveness and complexity of a decision.

Now that we have defined the objects we want to consider and subsets thereof, we still need to specify the decisions the decision maker is able to express. If the decision maker chooses one or more of the objects while discarding the rest, we call this a *choice*. Abstractly speaking, a *choice function (for $\mathbb{Q}$)* $c\colon \mathbb{Q} \to \mathcal{C}$ maps choice tasks $Q \in \mathbb{Q}$ to choices $c(Q) \subseteq Q$, such that $c(Q) \neq \emptyset$. In case that $|c(Q)| = 1$, we will call c a *singleton choice function* (see Figure 2.2a). To be able to distinguish the general case, a choice function mapping to sets of arbitrary length is called a *subset choice function* (see Figure 2.2b). The corresponding set $\mathcal{C} \coloneqq \bigcup_{Q \in \mathbb{Q}} 2^Q \setminus \{\emptyset\}$ of choices for any $Q \in \mathbb{Q}$ is called the *choice space*.

choice function

A slightly different way to express *preferences*, as the name suggests, is to let the decision maker order the objects in a given choice task Q. We will use the notation $\boldsymbol{x} \succ \boldsymbol{x}'$ to denote that the object $\boldsymbol{x}$ is *preferred over* object $\boldsymbol{x}'$. If the decision maker is indifferent between two objects, we will write $\boldsymbol{x} \sim \boldsymbol{x}'$.

$\boldsymbol{x} \succ \boldsymbol{x}'$ ("$\boldsymbol{x}$ is preferred over $\boldsymbol{x}'$")

In many settings, we are interested in finding a *total order* of all objects. These orders are encoded using permutations $\pi \in \mathcal{S}_M$, where M is the number of objects in Q and $\mathcal{S}_M$ is the symmetric group of order M. Thus, $\pi(k)$ is the rank of the k-th object, while the inverse $\pi^{-1}(\ell)$ returns the index of the object that is in the ℓ-th position. To illustrate, if we have the ranking $\pi = (2, 1, 3)$, then the first object is ranked second, the second object is ranked first, and the third object is ranked third. With $\pi^{-1} = (2, 1, 3)$, we express that on rank one is the second object, on rank two is the first object, and on rank three is the third object.

For a given permutation π on a task Q, we therefore know that for all pairs $1 \leq i < j \leq |Q|$, we have the preference relation $\boldsymbol{x}_{\pi^{-1}(i)} \succ \boldsymbol{x}_{\pi^{-1}(j)}$. It is common to call π itself a *ranking*, since it returns the ranks for given objects indexes, while π^{-1} is called an *ordering* because it can be used to order the objects according to their preferences. Analogously to the choice setting, we can define a *ranking function* $\rho\colon \mathbb{Q} \to \mathcal{S}$, which maps from (indexed) sets of objects to rankings, and where $\mathcal{S} \coloneqq \bigcup_{M \in [|\mathcal{X}|]} \mathcal{S}_M$.

ranking

ordering

ranking function

An alternative way to represent rankings is using a set of pairwise preferences $R \subset \mathcal{X}^2$. Compared to a permutation π, this is advantageous since it allows one to specify partial orders as well. That is why we require that the relation encoded by R is irreflexive, asymmetric and transitive. One such partial order is depicted in Figure 2.2c, where we can see that the decision maker was *indifferent* between the blue and the purple object, while the remaining objects are in a total order.

Probabilistic preferences Until now, we only looked at choices and rankings as constant, deterministic statements. That means if we give a decision maker a set of objects, it will always choose the same subset or order them in the same way. In practice, it is useful to assume that preferences are random, i. e., different outcomes can occur with varying probability. One motivation for this is when modeling the behavior of a population of decision makers.

Consider the example shown in Figure 2.3, where 50 % of the given population prefer chocolate, 30 % prefer vanilla and another 20 % prefer strawberry ice cream. When randomly sampling a person from this population, they will tell us their preference. While each person may have a deterministic preference, our observation is now a random variable. Even in the case of a single decision maker it can be worthwhile to model their preferences as a probability distribution over possible outcomes. There are a few reasons why the preference may appear random for an observer, such as unobserved external factors that can cause preferences to change depending on the context. One such factor is time: preferences may not remain constant and could evolve over extended periods. Another example is the influence of location on preferences. For instance, a person might favor more extravagant clothing in a fashion-forward city like Milan but opt for more conservative attire in a traditional town. This probabilistic model does not lose generality; the deterministic case is simply a special case where the probability of one outcome is set to 100 %, with all others at zero.

Formally, these probability distributions $p(\cdot \mid Q)$ will be defined for every task $Q \in \mathbb{Q}$. In the case of choice, for a given task Q, we will have a probability $0 \leq p(C \mid Q) \leq 1$ for each subset $C \subseteq Q$ to be chosen. The usual singleton choice setting can be obtained by additionally requiring that $p(C \mid Q) = 0$, if $|C| \neq 1$. For the probabilities to be valid, we require that $\sum_{C \in \mathcal{C}} p(C \mid Q) = 1$ for all $Q \in \mathbb{Q}$. That means all of the probabilities for all possible choice sets of a task Q have to sum up to 100 %.

It is useful to extend $p(\cdot \mid Q)$ to $\mathcal{C}$ by setting $p(C \mid Q) \coloneqq 0$ for all $C \in \mathcal{C} \setminus 2^Q$. We will do so for the remainder of this thesis.

As for the ranking setting, let $p(\pi \mid Q)$ be the probability that ranking π is observed for a given task $Q \in \mathbb{Q}$. Similar to the choice setting, we require the probabilities to obey $\sum_{\pi \in \mathcal{S}_{|Q|}} p(\pi \mid Q) = 1$ for all $Q \in \mathbb{Q}$. If we are interested in how frequently an object $\boldsymbol{x}$ is preferred over another object $\boldsymbol{x}'$, we will also denote the *marginal probability* of this preference by

$$p(\boldsymbol{x}_i \succ \boldsymbol{x}_j \mid Q) = \sum_{\pi \in \mathcal{S}_{|Q|}} p(\pi \mid Q) [\![\pi(i) < \pi(j)]\!]$$

where $[\![\cdot]\!]$ is the Iversion bracket

$$[\![P]\!] \coloneqq \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

that we will use throughout the thesis.

Equipped with the basic language for representing both deterministic and probabilistic preferences, we can now dive deeper into the background material and see what additional structure can be imposed on the preferences or their probabilities. In the next section, we will first look at a brief history of preference modeling, and then present important axioms for utility and choice theory.

2.2. Classical Preference Axioms

Probability theory was developed over the course of the 16th and 17th centuries by mathematicians such as Cardano, Fermat, Pascal, and Huygens [Huy57]. The calculation of an expected value, especially in the context of gambling, was well understood [Hal90]. The definition of value was not under scrutiny since it appeared to be clear in this context, with no room for subjectivity. It was only in 1713 that this view was challenged by the *St. Petersburg paradox* created by Nicolas Bernoulli in a correspondence with Montmort [Mon13]. The subject of the paradox is a one-player game where a fair coin is repeatedly tossed. The game begins with a price of 2. If the coin shows tails, the price is doubled, and the game continues. If it shows heads, the game ends with the player winning the current bet. The question is how much a rational player should be willing to pay to participate in this game. If we calculate the expected value of this game, we get

St. Petersburg paradox

$$\sum_{i=1}^{\infty} 2^i \frac{1}{2^i} = \sum_{i=1}^{\infty} 1 = \infty,$$

which would imply that, if given the opportunity, every rational person should play this game and pay any price. Daniel Bernoulli [Ber38] argues that most informed people would not pay more than 20 to play this game because the utility of the player's wealth is not linear. He proposes to use logarithmic *utility function*, which results in larger and larger payouts diminishing in utility. It turns out that when optimizing the expected utility instead of the expected value, the optimal amount is indeed finite *and* does depend on the current wealth of the player.

What this paradox demonstrates is that one needs to distinguish the raw value of something from the utility it actually has. The utility of an object is a subjective measure, which makes it difficult even to define. In the 19th century, formal axiomatizations were developed to support theories of utility and choice. With *utility theory*, the aim is to infer a utility function from observed behavior, such that the behavior can be explained by the maximization of this utility function and future behavior can be predicted. *Choice theory* is concerned specifically with modeling preferences in the form of choices. In Section 2.2.1 we will introduce *ordinal theory* as introduced by Pareto [Par06] in the year 1906, and *cardinal theory* which was developed by Von Neumann and Morgenstern [VM47] in 1947. In the context of choice, Luce [Luc59] published his well-known axiomatization for choices in the year 1959, which we will discuss in Section 2.2.2.

2.2.1. Axioms of Utility Theory

The main goal of utility theory is to explain the behavior exhibited by a decision maker by positing that they are assigning a utility to each possible decision outcome and select a decision based on what course of action maximizes their utility [VM47; Deb54]. The goal is often, given observed behavior, to find the underlying utility function.

Utility theory is used in many fields nowadays, most notably in economics. Besides its fundamental role in economics, it is instrumental in social choice

theory [Arr51], decision theory, game theory [VM47] and artificial intelligence [RN20]. However, it has been shown in psychological experiments that basic utility theory is limited in explaining all of human behavior [All53; Sim57; Ell61; KT79]. Allais [All53] found that participants in his experiments showed inconsistent behavior when faced with choices involving uncertainty, leading to the so-called Allais paradox. Ellsberg [Ell61] found similar inconsistencies in his experiments, where participants were asked to make choices involving unknown probabilities. Subsequently, theories such as prospect theory [KT79] have been developed to provide a more accurate representation of human behavior, particularly under conditions of uncertainty and risk. Over time, many more violations of the axioms of utility theory have been identified, which is why we will discuss these in more detail in Section 2.3.

The advent of artificial intelligence (AI) and machine learning has given rise to a new class of agents, which leverage the concept of expected utility maximization (EUM) to solve a variety of problems [RN20]. Unlike their human counterparts, these AI agents are engineered to exhibit fewer biases and idiosyncrasies. This is due to their ability to process information and make decisions based on algorithms and data. However, it is important to note that these agents are not immune to biases. In particular, AI agents can inherit and even amplify biases present in their underlying training data [BHN23].

Utility theory can broadly be divided into *ordinal* (see Section 2.2.1.1) and *cardinal* utility theory (see Section 2.2.1.2). While the former focuses on certain outcomes and only the order thereof, the latter admits the study of probabilistic outcomes and considers objects on an *interval scale*. Although cardinal utility may appear to be the more "powerful" of the aforementioned theories, it needs to be mentioned that it requires much stronger assumptions. We will see a similar theme emerge when we contrast these with the axioms of choice theory in Section 2.2.2.

2.2.1.1. Ordinal Utility

In ordinal utility theory, the point of departure is that the preferences are expressed on an ordinal scale. That means we can only make statements like "I prefer chocolate ice cream over strawberry" or "I am indifferent between mango ice cream and lemon ice cream", but the magnitude of the preference cannot be expressed. One would typically denote $\boldsymbol{x}$ *is preferred over* $\boldsymbol{y}$ by $x \succeq y$ (with $\succ$ indicating strict preference) and one *is indifferent between* $\boldsymbol{x}$ and $\boldsymbol{y}$ by $x \sim y$. It stands to reason that this restriction loses a lot of information and one would expect it to be too restrictive for an underlying theory of economics. To the contrary, Pareto [Par06] demonstrated that a rich economic theory could be developed using purely ordinal assumptions. It subsequently became the prevalent theory in economics in the early 20th century [Mos18; LS65].

The theory is usually introduced in terms of a consumer, which expresses preferences on so-called commodity bundles. These are represented by vectors $\boldsymbol{x} = (x_1, \dots, x_d)$ of which each entry stands for the quantity of one commodity. Since the general concepts of ordinal utility theory are more general, we can drop the specific application and assume that a decision maker has to decide between different objects $\boldsymbol{x}$ of which each vector component is an attribute to

be maximized. We are then able to define a natural dominance relation on the vector space.

We write that $\boldsymbol{x} > \boldsymbol{y}$ if $x_i \geq y_i$ for all $i \in [d]$ and $x_i > y_i$ for at least one $i \in [d]$. See Figure 2.4 for an example in two-dimensional space. Here, it obviously holds that $\boldsymbol{y}_3 > \boldsymbol{x}$, but also $\boldsymbol{y}_1 > \boldsymbol{x}$ and $\boldsymbol{y}_2 > \boldsymbol{x}$, since both have at least one coordinate which is strictly larger than the same coordinate of $\boldsymbol{x}$.

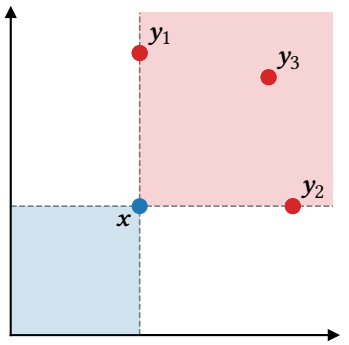


Figure 2.4.: Vectors admit a natural dominance relation $\boldsymbol{x} < \boldsymbol{y}$.

With that, we can define a preference relation on the set of objects as follows.

Definition 2.2.1 ([LS65]) *A relation R on a set $\mathcal{X} \subseteq \mathbb{R}^d$ is a preference relation if the following axioms hold for any $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$:*

(i) If $\boldsymbol{x}R\boldsymbol{y}$ and $\boldsymbol{y}R\boldsymbol{z}$, then $\boldsymbol{x}R\boldsymbol{z}$. (Transitivity)
(ii) $\boldsymbol{x}R\boldsymbol{y}$ or (inclusive) $\boldsymbol{y}R\boldsymbol{x}$. (Connectivity)
(iii) If $\boldsymbol{x} > \boldsymbol{y}$, then $\boldsymbol{x}R\boldsymbol{y}$ and not $\boldsymbol{y}R\boldsymbol{x}$. (Nonsatiety)
(iv) If $\boldsymbol{x}R\boldsymbol{y}$ and $\boldsymbol{y}R\boldsymbol{z}$, then there exists $\alpha \in [0, 1]$ such that $[\alpha\boldsymbol{x}+(1-\alpha)\boldsymbol{z}] \sim \boldsymbol{y}$. (Continuity)

Here $\boldsymbol{x} \sim \boldsymbol{y}$ denotes the case where $\boldsymbol{x}R\boldsymbol{y}$ and $\boldsymbol{y}R\boldsymbol{x}$, indicating *indifference*. As mentioned before, property (i) is enforcing the transitivity of preferences, which prevents cycles. Property (ii) ensures that each object is in a relation with every other object. As of now, there is no connection of the preference relation with the dominance relation in the d-dimensional space, which is why property (iii) establishes that $\boldsymbol{x} > \boldsymbol{y}$ corresponds to a strict preference in R. Finally, the continuity property (iv) is required to prove the existence of a utility function.

With these assumptions, it is then possible to show that a utility function u exists.

Theorem 2.2.2 (Theorem 1 by Luce and Suppes [LS65]) *Let R be a preference relation on $\mathcal{X} \subseteq \mathbb{R}^d$. Then there exists a utility function $u : \mathcal{X} \to \mathbb{R}$ such that for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$:*

$$\mathrm{u}(\boldsymbol{x}) \geq \mathrm{u}(\boldsymbol{y}) \text{ if and only if } \boldsymbol{x}R\boldsymbol{y}$$

An immediate consequence is that $\mathrm{u}(\boldsymbol{x}) = \mathrm{u}(\boldsymbol{y})$ if, and only if $\boldsymbol{x} \sim \boldsymbol{y}$. One may ask whether this utility function is, in fact, unique, and it can be shown that u is unique up to increasing monotone transformations [LS65, Theorem 2].

This result still requires that $\mathcal{X}$ consists of d-dimensional vectors. Luce and Suppes [LS65, Theorem 3] shows that a similar result can be obtained for arbitrary infinite sets $\mathcal{X}$ with a preference relation. A necessary and sufficient condition is that there exists a countable subset of $\mathcal{X}$, which is order-dense and there cannot be a pair of objects in the subset that are indifferent with respect to the preference relation (either $\boldsymbol{x} \succ \boldsymbol{x}'$ or $\boldsymbol{x} \prec \boldsymbol{x}'$ has to hold). Additionally, it is often advantageous that the resulting utility function is continuous. Here Debreu [Deb54] showed that a continuous utility function exists if the order topology resulting from the preference relation is separable (also see [LS65, Section 2.2]).

We can see that with a relatively small set of assumptions, requiring only an ordinal graduation of preferences, the existence of a utility function can be

guaranteed. In the next section, we will see what happens when we assume that preferences are expressed on a cardinal scale.

2.2.1.2. Cardinal Utility

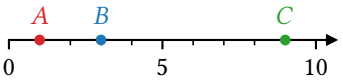


Figure 2.5.: Three objects on a utility scale.

Contrasted with ordinal utility, where we assume that only the order of the objects is important, cardinal utility posits that the magnitude of the differences in utility is important as well. See, for example, the utility scale in Figure 2.5 where $u(A) = 1$, $u(B) = 3$ and $u(C) = 9$. In ordinal utility, we would only be able to deduce that $A \prec B \prec C$. If we treat u as a cardinal utility function, then we can state that the difference in utility between A and B is 2, which is much smaller than the difference in utility between B and C, which is 6.

$$\arg\max_{x \in \mathcal{X}} \sum_{s} u(s)\, p(\text{Outcome}(x) = s \mid x)$$

[Ben89]: Bentham (1789), *An Introduction to the Principles of Morals and Legislation*

Cardinal utility theory goes back to Bernoulli 1738 [Ber38], who first stated the expected utility hypothesis, i. e., that in order to decide in the presence of uncertainty, one should weigh the utility of each outcome by the probability of it occurring and then picking the object with the highest weighted average. Of course, this presumes that an individual can accurately assign an absolute utility value to every object. This led to the problem of *measurability* of utility, which challenged economists and psychologists up until the 20th century [Mos18]. Bentham [Ben89] in his influential work, was certain that measuring utility was possible. Researchers in the 19th century unsuccessfully tried to come up with different ways to solve the measurability problem [Sti50]. In addition, several researchers argued that utility scales are not interpersonally comparable [Rob32; Arr51; Hau95], which made it difficult for utilitarians to reason about collective utilities. This led to the rise of ordinal utility theory in the early 20th century, spearheaded by the theory developed by Pareto [Par06]. While it was very successful, uncertainty in the outcome of a choice was not modeled. It was only in 1947, when Von Neumann and Morgenstern proposed their axiomatization, that the existence of a cardinal utility function could be proven [VM47]. Since the axioms were influential, we will now recall them here.

2: also called *lotteries*

Von Neumann and Morgenstern distinguish between *pure* objects and *mixtures*[2] of objects. Assume we have a set of objects $\mathcal{X}$. Then if we take two objects $x, y \in \mathcal{X}$, then we can form a mixture of these by specifying that x happens with probability α and y occurs with probability $1 - \alpha$. We will denote this using the operation $h(x, \alpha, y)$ or $x\alpha y$ for short. Mixtures of objects can be nested arbitrarily, e. g., $(x\alpha y)\beta z$ would be a valid way to specify a mixture of three objects $x, y, z \in \mathcal{X}$.

We will also assume that there exists a relation R on these objects, where xRy denotes that x is preferred over y or the decision maker is indifferent. It is common to define additional notation to express indifference and strict preference [LS65, p. 260]:

$$xIy \text{ if and only if } xRy \text{ and } yRx$$
$$xPy \text{ if and only if } xRy \text{ and not } yRx$$

Definition 2.2.3 (Von Neumann and Morgenstern [See 3.6.1 in VM47; LS65]) *A triple $\mathcal{A} = (\mathcal{X}, R, h)$ is a Von Neumann and Morgenstern system of utility if for every $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z} \in \mathcal{X}$ and every $\alpha, \beta \in (0,1)$ the following axioms hold:*

(i) R *is a weak ordering of* $\mathcal{X}$
(ii) $\boldsymbol{x}\alpha\boldsymbol{y} \in \mathcal{X}$
(iii) $\boldsymbol{x}\alpha\boldsymbol{y} = \boldsymbol{y}(1-\alpha)\boldsymbol{x}$
(iv) $(\boldsymbol{x}\alpha\boldsymbol{y})\beta\boldsymbol{y} = \boldsymbol{x}\alpha\beta\boldsymbol{y}$
(v) if $\boldsymbol{x}I\boldsymbol{y}$*, then* $\boldsymbol{x}\alpha\boldsymbol{z}I\boldsymbol{y}\alpha\boldsymbol{z}$
(vi) if $\boldsymbol{x}P\boldsymbol{y}$*, then* $\boldsymbol{x}P\boldsymbol{x}\alpha\boldsymbol{y}$ *and* $\boldsymbol{x}\alpha\boldsymbol{y}P\boldsymbol{y}$
(vii) if $\boldsymbol{x}P\boldsymbol{y}$ *and* $\boldsymbol{y}P\boldsymbol{z}$*, then there exist* $\gamma, \delta \in (0,1)$ *such that* $\boldsymbol{y}P\boldsymbol{x}\gamma\boldsymbol{z}$ *and* $\boldsymbol{x}\delta\boldsymbol{z}P\boldsymbol{y}$.

A relation is said to be a *weak order relation*, if it satisfies transitivity, i. e., if $\boldsymbol{x}R\boldsymbol{y}$ and $\boldsymbol{y}R\boldsymbol{z}$, then $\boldsymbol{x}R\boldsymbol{z}$ also has to hold and if it always holds that $\boldsymbol{x}R\boldsymbol{y}$ or $\boldsymbol{y}R\boldsymbol{x}$. Axiom (ii) ensures that $\mathcal{X}$ is closed under the mixture operation h. Thus, if we know that two objects $\boldsymbol{x}$ and $\boldsymbol{y}$ are in $\mathcal{X}$, then any mixture of them also has to be in $\mathcal{X}$.[3] Axioms (iii) and (iv) specify the basic properties of our mixture operation, how the order of the mixture objects can be swapped, and how nested mixtures can be flattened. Axiom (v) states that if one is indifferent between two objects $\boldsymbol{x}$ and $\boldsymbol{y}$, then one also has to be indifferent if a third object $\boldsymbol{z}$ is mixed into both with equal proportion. In a similar vein, axiom (vi) assumes that if $\boldsymbol{x}$ is strictly preferred to $\boldsymbol{y}$, then $\boldsymbol{x}$ is always strictly preferred over a mixture of both, while any mixture is always strictly preferred over $\boldsymbol{y}$. Finally, axiom (vii) is a continuity condition, assuming that if $\boldsymbol{y}$ is between $\boldsymbol{x}$ and $\boldsymbol{z}$ with respect to strict preference, then you can find a mixture of $\boldsymbol{x}$ and $\boldsymbol{z}$ which is strictly preferred to $\boldsymbol{y}$ and also a mixture to which $\boldsymbol{y}$ is preferred.

3: If we assume that there exist at least two objects $\boldsymbol{x}$, $\boldsymbol{y} \in \mathcal{X}$ with $\boldsymbol{x}P\boldsymbol{y}$, then (ii) implies that $\mathcal{X}$ is infinite [LS65].

Equipped with these axioms, we are now able to state the main result of Von Neumann and Morgenstern, the existence of a utility function u.

Theorem 2.2.4 (Von Neumann and Morgenstern [See A.2 in VM47]) *Let $\mathcal{A} = (\mathcal{X}, R, h)$ be a Von Neumann and Morgenstern utility system. Then there exists a function $\mathrm{u} : \mathcal{X} \to \mathbb{R}$ such that for every $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathcal{X}$ and α in $(0,1)$*

(i) $\boldsymbol{x}R\boldsymbol{y}$ *if and only if* $\mathrm{u}(\boldsymbol{x}) \geq \mathrm{u}(\boldsymbol{y})$.
(ii) $\mathrm{u}(\boldsymbol{x}\alpha\boldsymbol{y}) = \alpha\mathrm{u}(\boldsymbol{x}) + (1-\alpha)\mathrm{u}(\boldsymbol{y})$.

The function u *is unique up to positive linear transformations.*

Consequence (i) states that the utilities returned by the function u can be used to order the objects based on preference. The second property (ii), on the other hand, allows us to express the utility of any mixture as a mixture of utilities of pure objects. The literature has explored several alternative axiomatizations, either to simplify the axioms or to generalize the system of utility. A prominent example of the former is the Herstein and Milnor system of utility [HM53], which simplifies the existence proof of a utility function.

As we have seen, with careful assumptions on the preferences expressed on mixtures of objects, we are able to show the existence of a cardinal utility function. With ordinal utility, it was only possible to arrive at a utility function which is unique up to monotone transformations. That is why the absolute utility values and differences thereof were meaningless. Intuitively speaking, what distinguishes cardinal utility from ordinal utility is the ability to interpolate

between objects. This ensures that the magnitude of the utility function is meaningful.

One could argue that the assumptions underpinning Von Neumann and Morgenstern's utility system are quite technical and more of a normative nature, and thus are removed from actual human behavior. At the same time, only the outcomes of each object are assumed to be probabilistic, while the preferences themselves are still assumed to be deterministic. That is the reason why in 1959 Luce proposed his axioms of choice [Luc59].

2.2.2. Axioms of Choice

Before we take a closer look at the axiomatization of choice, first note that we are now moving from preferences, which are always defined on pairs of objects, to choices on subsets of objects. For example, for a given set $\{a, b, c\}$, we could observe a person to choose a. It is possible to extract some preference information from choices[4], for example here we could state that $a \succ b$ and $a \succ c$, but the preference between b and c is not visible. Luce's main point of departure is the assumption that human choice itself should be treated as a probabilistic process [Luc59]. So, we will consider the probability that a particular object x will be chosen from a set Q. These probabilities are allowed to be 0 or 1, so the special case of certain choices is covered as well. We will also use the notation $p_U(A \mid Q)$ to denote the probability that the chosen object is part of the set $A \subseteq Q$. With this, we want to distinguish it from the probability that subset A itself is chosen, which we do not need in this context.

4: These are studied under the name *revealed preferences*.

First, recall the following probability axioms [Luc59]:

1. For $Q \subset Q', 0 \leq p_U(Q \mid Q') \leq 1$.
2. $p_U(Q \mid Q) = 1$.
3. If $R, S \subset Q$ and $R \cap S = \emptyset$, then $p_U(R \cup S \mid Q) = p_U(R \mid Q) + p_U(S \mid Q)$, which implies
$$p_U(S \mid Q) = \sum_{x \in S} p(x \mid Q)$$

These axioms ensure that for each $Q \in \mathbb{Q}$ the function $p(\cdot \mid Q)$ is a valid probability distribution. What these do *not* do is impose a global structure across all of these probability distributions. In the worst case, there could be a completely different distribution for each set $Q \in \mathbb{Q}$. If two sets have many objects in common, it is natural to assume that their choice behavior should be similar. It is also natural to posit that the choice probabilities observed on smaller sets should in some way carry over to larger (super)sets.

It is for this reason that Luce introduces the following choice axiom:

Axiom 2.2.5 ([Axiom 1 in Luc59]) *Let T be any finite subset of $\mathcal{X}$ such that, for every subset $S \subseteq T$, p_S is defined. Then the following properties hold:*

(i) If $p(\boldsymbol{x} \mid \{\boldsymbol{x}, \boldsymbol{y}\}) \notin \{0, 1\}$ for all $\boldsymbol{x}, \boldsymbol{y} \in T$, then for $R \subseteq S \subseteq T$

$$p_{\mathsf{U}}(R \mid T) = p_{\mathsf{U}}(R \mid S)\, p_{\mathsf{U}}(S \mid T)$$

(ii) If $p(\boldsymbol{x} \mid \{\boldsymbol{x}, \boldsymbol{y}\}) = 0$ for some $\boldsymbol{x}, \boldsymbol{y} \in T$, then for every $S \subseteq T$

$$p_{\mathsf{U}}(S \mid T) = p_{\mathsf{U}}(S \setminus \{x\} \mid T \setminus \{\boldsymbol{x}\})$$

The axiom has the effect of coupling the choice probability distributions, which so far could have been completely independent. Property (ii) states that as soon as an object $\boldsymbol{x}$ has probability 0 in a pairwise choice in the presence of object $\boldsymbol{y}$, then it can never be chosen in any superset of $\{\boldsymbol{x}, \boldsymbol{y}\}$. In other words, it takes a local, pairwise probability and spreads it to supersets. Another consequence is that the object $\boldsymbol{x}$ could safely be removed from these same supersets since the choices will not be impacted. Property (i) imposes that a choice probability distribution p_T defined on a set T can be decomposed into choice probability distributions on its subsets $R \subseteq S \subseteq T$.

Consequences of Luce's Choice Axiom The choice axiom implies a variety of properties, which are important for several reasons. On the one hand, these properties are usually *observable*, meaning that they can be experimentally verified, which may not be possible for a specific model directly. On the other hand, we will see that most of the properties can be sorted by how strongly they constrain the choice probabilities. This allows one to categorize models based on where they can be inserted into this hierarchy. The following will be an overview of the most important consequences of Luce's choice axiom.

The first important property is *independence of irrelevant alternatives (IIA)*:

independence of irrelevant alternatives

Lemma 2.2.6 (Independence of irrelevant alternatives [Luc59]) *If $p(\boldsymbol{x} \mid \{\boldsymbol{x}, \boldsymbol{y}\}) \notin \{0, 1\}$ for all $\boldsymbol{x}, \boldsymbol{y} \in T \in \mathbb{Q}$, then Axiom 2.2.5 implies that for any $S \subseteq T$ such that $\boldsymbol{x}, \boldsymbol{y} \in S$,*

$$\frac{p(\boldsymbol{x} \mid \{\boldsymbol{x}, \boldsymbol{y}\})}{p(\boldsymbol{y} \mid \{\boldsymbol{x}, \boldsymbol{y}\})} = \frac{p(\boldsymbol{x} \mid S)}{p(\boldsymbol{y} \mid S)}$$

This lemma has important consequences for the possible types of choice models. Immediately, we can show that given any set $Q \in \mathbb{Q}$ and an object $\boldsymbol{x} \in Q$, we can determine its probability of being chosen using only the pairwise probabilities [Luc59, Theorem 1]:

$$p(\boldsymbol{x} \mid Q) = \left(\sum_{\boldsymbol{y} \in Q} \frac{p(\boldsymbol{y} \mid \{\boldsymbol{x}, \boldsymbol{y}\})}{p(\boldsymbol{x} \mid \{\boldsymbol{x}, \boldsymbol{y}\})} \right)^{-1}$$

This certainly allows one to efficiently specify a choice model since it is only necessary to specify the pairwise probabilities. It also allows one to easily test whether observed choice data obeys Axiom 2.2.5 by comparing the choice frequencies observed on pairs with those observed on larger sets of objects.

A slightly weaker property of a similar flavor is called *regularity* , which is a

regularity condition

condition on choice probabilities which asserts that the choice probabilities of items cannot increase if you add objects to an existing set. Formally, for all $x \in A \subseteq B$ with $A, B \in \mathbb{Q}$ it holds that $p(x \mid A) \geq p(x \mid B)$. The condition only weakly restricts the choice probabilities and early experimental data were consistent with the property [BDM63]. It only became clear that the regularity condition is not universally true for human choice data, when researchers discovered the *attraction effect*.

attraction effect: Page 30

A key property which is often considered in the context of preferences is *transitivity*. The motivation is clear: a violation of transitivity would imply that the preferences can exhibit cycles, which would make any further analysis difficult. It is easy to see that the choice probabilities constrained by Axiom 2.2.5 have to be deterministically transitive, i. e., assume Axiom 2.2.5 holds for a set $Q = \{a, b, c\}$ with $p(a \mid \{a, b\}) = 1$ and $p(b \mid \{b, c\}) = 1$, then also $p(a \mid \{a, c\}) = 1$. To prove this, we can make use of property (ii) of Axiom 2.2.5 in conjunction with $p(b \mid \{a, b\}) = p(c \mid \{b, c\}) = 0$, which allows us to state $p(a \mid \{a, b, c\}) = p(a \mid \{a, c\}) = p(a \mid \{a, b\}) = 1$.

deterministic transitivity

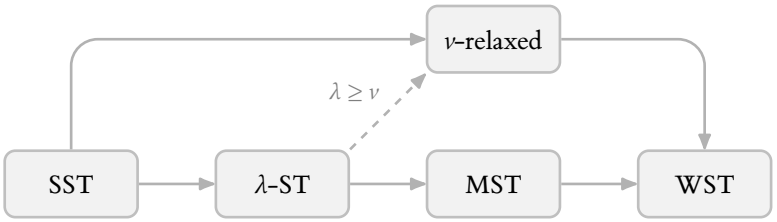


Figure 2.6.: Hierarchy of transitivity definitions. Each arrow denotes the direction of implication. Adapted from [HHK20].

When generalizing the transitivity to probabilistic preferences, it is important to note that it is no longer a clear, unique concept. That is why, over the years, a multitude of different definitions were proposed, which are usually ordered with respect to their "strength" by which they constrain the choice probabilities (see Figure 2.6 for an overview).

The earliest definitions for stochastic transitivity were proposed by Davidson and Marschak [DM58]. They propose *weak stochastic transitivity (WST)*, which simply assumes that if $p(a \mid \{a, b\}) \geq \frac{1}{2}$ and $p(b \mid \{b, c\}) \geq \frac{1}{2}$, then also $p(a \mid \{a, c\}) \geq \frac{1}{2}$. Strong stochastic transitivity (SST) further requires that $p(a \mid \{a, c\} \geq \max(p(a \mid \{a, b\}), p(b \mid \{b, c\}))$. Luce [Luc59] shows that if Axiom 2.2.5 holds for a set of objects $\mathcal{X}$, then the probabilities have to satisfy SST. Moderate stochastic transitivity (MST) is similar to SST, but uses the minimum instead of the maximum. When interpolating between SST and MST, we obtain λ-ST, which has been proposed by Basu [Bas84] in the context of fuzzy preference relations. We can also interpolate between SST and WST [YJ11], which we refer to as ν-relaxed stochastic transitivity here. Interestingly, it turns out that it cannot be neatly placed within the existing implication chain.

weak stochastic transitivity

strong stochastic transitivity

Arguably, the most impactful consequence of Axiom 2.2.5 is the existence of a utility function, which can be used to determine all of the probabilities.

existence of a utility function

Theorem 2.2.7 (Theorem 3 by Luce [Luc59]) *Let $\mathcal{X}$ be finite and $p(x \mid \{x, y\}) \in (0, 1)$ for all $x, y \in \mathcal{X}$. Assume that Axiom 2.2.5 holds for all $Q \subseteq \mathcal{X}$. Then there exists a utility function $u : \mathcal{X} \to \mathbb{R}_{\geq 0}$ such that for every $Q \subseteq \mathcal{X}$ and every $x \in Q$:*

$$p(x \mid Q) = \frac{u(x)}{\sum_{y \in Q} u(y)}$$

This utility function is unique up to multiplication by a positive constant.

This is the basic form of many subsequent *random utility models* like for instance, the multinomial logit, which we will discuss in more detail in Section 2.4. Furthermore, since $u(x) = k\, p(x \mid \{x, y\})\, p(y \mid \{x, y\})$ for any $x, y \in \mathcal{X}$ and $k \in \mathbb{R}_{\geq 0}$, one could estimate these utilities by only observing pairwise comparisons.

That the existence of a utility function follows from Axiom 2.2.5 is remarkable, since earlier work in economics by Von Neumann and Morgenstern [VM47] required restrictive axioms (see Section 2.2.1). It also does not require the assumption of a parametric model like Thurstone's models [Thu27], which presuppose that utilities follow a normal distribution. A caveat of Theorem 2.2.7 is that it requires that all probabilities to be in $(0, 1)$, i. e., it does not handle the case where choices are deterministic. In practice, that would mean that we can only find separate utility functions for subsets of $\mathcal{X}$ for which this assumption holds. Luce ameliorates this problem by proving [Luc59, Theorem 4] that if all objects $x, y \in \mathcal{X}$ are *finitely connected*, i. e., there exists a sequence of imperfect pairwise comparisons between them, and if strong stochastic transitivity holds, then there exists one utility function that can compute the probabilities of all subsets which contain only imperfect probabilities.

While Luce's choice axiom was a seminal step in the theoretical treatment of choices and spawned a plethora of subsequent literature, cracks started to appear in the theory when human experimental data appeared to disagree with what the application of Axiom 2.2.5 predicted. In Section 2.3 we are going to look at *context effects* which were identified, i. e., certain situations in which choice probability ratios deviate across different subsets.

An alternative axiomatization Given that Luce's choice axiom is so restrictive, we may wonder what an alternative axiomatization of choice could look like and what suitable requirements are for it to still be a sensible model of reality. Masatlioglu and Nakajima [MN07] propose the axiom of choice by elimination (ACE), which allows for a much larger variety of possible choice behaviors:[5]

5: The axiom is only defined for deterministic choices.

Axiom 2.2.8 ([MN07]) *Let $\mathcal{X}$ be a finite set of objects and $c : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ be a choice function. For any $Q \in 2^{\mathcal{X}}$ there exists an object $x \in c(Q)$ such that if $x \in Q' \subset Q$ then $x \in c(Q')$.*

In other words, this says that for any choice task, there needs to be at least one element in the choice set that is chosen in all of the subsets of the choice task. This is in contrast to Luce's choice axiom, which would require this to be true for all chosen objects.

The authors argue that ACE is the minimum requirement for rational choice, while still preventing *choice cycles*. An example of a choice function that exhibits a choice cycle could be $c(\{a,b,c\}) = \{a,b,c\}$, $c(\{a,b\}) = \{a\}$, $c(\{b,c\}) = \{b\}$ and $c(\{a,c\}) = \{c\}$. Here, we are not able to find an element of $\{a,b,c\}$ that is chosen in all subsets. Therefore, ACE is violated.

2.3. Context Effects

The development of axiomatic approaches in modeling human preferences and choices was a significant step forward. Theoretically, these approaches should enable us to model all facets of human preferences, especially in decision-making contexts. The natural question that arises is whether these models are an accurate representation of human choice behavior. Any axiomatic system depends on a set of assumptions, and it is these assumptions that must be tested against real-world data to verify their validity.

Researchers have, therefore, conducted several experiments to examine the behavior of individuals in decision-making contexts. Section 2.3.1 will present an overview of these experiments, highlighting the specific properties of the axioms that were found to be inconsistent with observed behavior. While identifying these properties is useful to pinpoint the limitations of our current models, it does not provide a comprehensive understanding of the underlying psychological mechanisms that drive these deviations.

As a consequence, researchers have shifted their focus to identifying specific situations in which the preferences of individuals can reliably be influenced by the context of the preference. The mechanisms that were identified in this way are called *context effects*. The investigation of these effects offers a more nuanced understanding of how environmental and situational factors shape human preferences. In section Section 2.3.2, we will explore these context effects and present the most well-known ones.

context effects

2.3.1. Deviations from Axioms

Many of the experimental results in this regard came from the field of experimental psychology, where researchers were interested in how humans estimate the difference in various stimuli (e. g., weight, brightness, loudness, duration, etc.).

Coombs [Coo58] performed an experiment where subjects were repeatedly shown sets of four gray color patches and asked to rank them based on how well they represented their ideal image of "gray". For the analysis, they tabulated the pairwise frequencies that one patch was preferred over another. Analyzing the data, they found that WST was satisfied, i. e., for any triple of patches A, B and C with $p(A \succ B) \geq 0.5$ and $p(B \succ C) \geq 0.5$, it was also the case that $p(A \succ C) \geq 0.5$. What they found, however, was that SST was violated for some of the triples.

Chipman [Chi60] conducted experiments with army volunteers that were asked to compare matchboxes. The matchsticks were split in half and then the boxes were filled with varying proportions of heads and tails. In the main

experiment, the participants were asked to bet on either a matchbox with a known proportion or a matchbox with an unknown proportion, but for which a small sample was drawn. The results were tabulated and statistical tests were used to check which properties of the axioms were violated. Chipman demonstrated that the properties SST, linearity [Luc58] and IIA can be confidently rejected.

In 1967, Krantz tested the simple scalability axiom using color triad tests [Kra67]. Subjects were shown three monochromatic lights x, y and z and given the task to determine which color difference was smaller, that of x and z or y and z. Based on the collected data they test Luce's choice model and simply scalability axiom in general. With the results, they are able to reject both confidently.

Tversky and Edward Russo [TE69] show that simple scalability, SST, substitutability and IIA are equivalent. That means the above experiments often tested for the same underlying property. Roughly speaking, what all these properties have in common is that preferences observed in these experiments are in some way dependent on other objects present in the preference context. Recently, Benson et al. [BKT16] devised a suite of statistical tests able to detect deviations from IIA. Each test assumes as the null hypothesis that IIA holds, and deviations can then be detected by defining different random variables on the preference data.

statistical testing for IIA violation

To gain some insights into what specific contexts cause these deviations, we will now take a look at *context effects*.

2.3.2. Identification of Context Effects

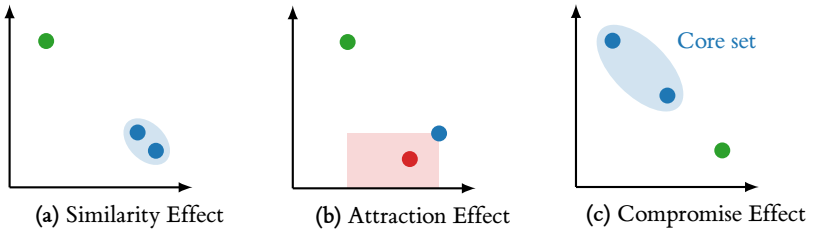


Figure 2.7.: An overview of the most commonly identified context effects in the literature.

In an effort to categorize, *how* humans are deviating from behavior expected by following the axioms, a variety of context effects have been identified. Refer to Figure 2.7 for an overview. The similarity effect occurs when there are several similar items in a preference task. Then it happens that the probability mass is distributed amongst them without impacting other dissimilar items. In the attraction effect we see an increase in probability mass of an item, if we introduce another item it dominates. Finally, the compromise effect arises when we add an extreme item to a preference context, and the central item becomes more favorable. We will now go over each effect in greater detail, presenting the corresponding studies that were performed.

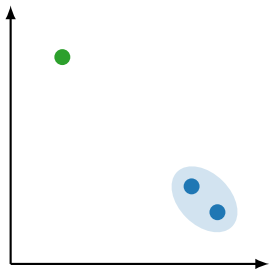


Figure 2.8.: Similarity effect: In the blue region, the objects are very similar and, therefore, only share probability weight amongst each other. Typical choice models would predict that a new blue object should also steal probability weight of green.

The Similarity Effect In an instructive example provided by Debreu [Deb60], the use of Luce's choice axiom results in a clear deviation from expected choice probabilities. Consider a scenario where an individual is presented with the option to choose between a recording of a composition by Debussy and two recordings of the same Beethoven composition. Assuming the individual

does not prefer one recording over any other, the axiom would predict a choice probability of 1/3 for each recording. However, this prediction is counterintuitive. Intuitively, one would anticipate that the two Beethoven recordings, being identical, would share their probability mass and sum up to a probability equal to that of the single Debussy recording. As such, we would expect the probability of the Debussy recording to be close to 1/2 and the probability of each Beethoven recording to be close to 1/4.

The empirical verification of this hypothesis was conducted by Becker et al. [BDM63]. In their study, 62 students were asked to select wagers from sets comprising either two or three options.[6] In the three-option sets, one wager was either an exact duplicate or a closely similar wager, albeit with permuted vector entries. The findings corroborated the similarity effect, demonstrating that similar items indeed tend to share their probability mass, as reflected in the participants' choices.

6: Here, a wager refers to a vector assigning rewards across four equally probable events.

Tversky and Edward Russo [TE69] further investigate the similarity effect, by letting 161 prisoners compare the size of rectangles or lenses. The goal was to investigate whether a more similar shape allows the participants to assess the size difference more accurately, which would affirm the similarity hypothesis. The subsequent analysis revealed that choice probabilities were significantly influenced by the similarity to the reference object, leading to the rejection of the independence axiom.

attraction effect

The Attraction Effect was introduced by Huber et al. [HPP82] and Huber and Puto [HP83], who noted that adding an asymmetrically dominated object can cause the choice probability of the now dominating object to increase. This, in turn, violates both the regularity condition and the similarity effect. In Figure 2.9, we can see an example situation that depicts the attraction effect. The blue object is *asymmetrically* dominating the red object, i. e., the green object does *not* dominate the red object.

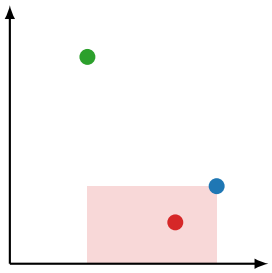


Figure 2.9.: Attraction Effect: The addition of an asymmetrically dominated object in the red region causes the blue object to increase its probability share.

Huber et al. [HPP82] demonstrate the attraction effect empirically by letting 153 students choose from two or three objects within six product categories. Each object was specified by two preference features. In the sets containing three objects, one object is asymmetrically dominated by another object, called the *target*. The study compared the choice probabilities between sets with and without a decoy. The underlying hypothesis posited that the presence of a decoy would increase the choice probability of the target. This was confirmed in the subsequent experiments. The extent of the probability increase varied depending on the decoy's placement, with the largest average increase being 13 % (absolute difference).

Huber and Puto [HP83] extend their investigation to assess the impact of what they term "slightly inferior" decoys. These decoys are not strictly dominated, but they offer a bad trade-off of the preference features. After analyzing the results, they observe two distinct effects: firstly, a *global attraction effect*, wherein there is a redistribution of probability weight towards a new object; secondly, a *local substitution effect* that happens after the global attraction effect. This latter effect is characterized by the new object drawing probability weight primarily from its nearest competitors in the preference space. Notably, the intensity of this local attraction effect depends on the precise location of the new object within the preference landscape.

global attraction effect

local substitution effect

Ratneshwar et al. [RSS87] argue that the attraction effect may be attributed to an insufficient elaboration on the implications of the object features. They hypothesize that a comprehensive understanding of the features, coupled with familiarity with the product category, should prevent the attraction effect. To validate this hypothesis, the authors performed four experiments with participants. The findings revealed that while clear descriptions of features substantially reduced the attraction effect, they failed to eradicate it entirely. Contrary to expectations, familiarity with the product category did not significantly diminish the attraction effect.

Simonson [Sim89] posit that the attraction effect may partly stem from study participants anticipating that their decisions are being evaluated by others. They propose that a (near-)dominance relation serves as a compelling rationale for justifying a decision. This hypothesis was examined through a series of empirical studies. The results show that the attraction effect increases when participants are informed that they will need to justify their choices. Conversely, in scenarios where participants were assured of the confidentiality of their choices, the attraction effect persisted, albeit in a diminished capacity. The authors conjecture that, in these instances, a dominance relation may function as a decisional tiebreaker. Simonson [Sim89] note that the slightly inferior decoys, as utilized in the previous studies by Huber and Puto [HP83], tended to transform one of the objects into a perceived compromise option. This observation led them to introduce the concept of the *compromise effect*.

compromise effect

The Compromise Effect The compromise effect, as proposed by Simonson [Sim89], states that an object may gain probability weight if new object(s) are added to the set of objects such that the original object becomes a compromise option. The authors hypothesize that (1) an object will gain probability share if it becomes a compromise option and (2) the anticipation of being judged should strengthen the effect. These hypotheses are subsequently confirmed in three studies with up to 372 students, i. e., compromise options were significantly more likely to be selected by the participants, and even more so when they were told that they were going to have to justify their choice at a later point. In questionnaires after the experiments, the participants rated compromise options less likely to be criticized, but not easier to justify.

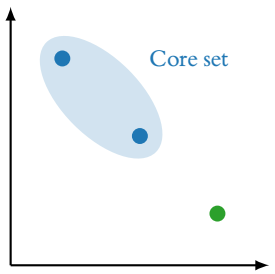


Figure 2.10.: Compromise effect: A new object (here green) is added to a core set (blue). The central object becomes the compromise option.

Simonson and Tversky [ST92] then argue that there are two underlying principles that can explain the attraction and the compromise effect, as well as making new testable predictions. These two principles they call *tradeoff contrast* and *extremeness aversion*. The former posits that an individual, when given a choice between two non-dominated objects, will assess the tradeoffs of the features. The presence of additional objects during the decision-making process can influence this assessment, for instance, by presenting an option with inferior tradeoffs, thereby rendering one of the original options more appealing. Extremeness aversion, as the name suggests, is a general tendency to avoid losses. In the context of preferences, this leads to a tendency to favor more moderate, compromise objects over those perceived as being extreme. The authors also propose an asymmetric version of extremeness aversion, which they call *polarization*, where extremes are only avoided for certain features. To illustrate this concept, they refer to the commonly observed quality-price tradeoff in consumer behavior. Empirical evidence from their studies indicates

tradeoff contrast

extremeness aversion

polarization

that participants often avoided low-quality/low-price objects. However, this behavior was not observed for high-quality/high-price objects.

other context effects

Other Context Effects Research in the domains of experimental psychology and marketing has found various additional context effects observable in experimental settings. Iyengar and Lepper [IL00] demonstrate that too many objects available during a choice lead to decision paralysis and overall less satisfaction (a phenomenon also known as *choice overload*). However, contrary to the common belief that context effects vanish in the presence of overload, the authors do not find this effect in their experiments, and Hawkins et al. [Haw+12] and Wedell et al. [WHV22] corroborate this finding in their experiments. Hawkins et al. [Haw+12] demonstrate in their experiments that the number of mistakes made by participants was independent of the number of objects in the choice task. The exception to this rule was when the choice had to be made in a very short time frame. Pettibone [Pet12] and Cataldo and Cohen [CC19] corroborate this finding that context effects are less pronounced with less deliberation time.

choice overload

Furthermore, several studies have demonstrated that the external context in which a choice is presented influences the resulting decision. Tversky and Kahneman [TK81] find that the way information is presented has an impact on the preferences observed, a concept called *framing*. In the realm of medical policy, Johnson and Goldstein [JG03] explore the influence of default options on decision-making, revealing a significant increase in the willingness of participants to become organ donors when this is presented as the default choice. Similarly, Kahneman et al. [KKT90] show in experiments that individuals ascribe a greater value to objects that are in their possession (called the *endowment effect*). While these studies provide insights into the broader spectrum of context effects, it is important to note that for the remainder of the thesis, we assume that the complete preference context is expressed by the preference task, i. e., the set of objects available.

framing

endowment effect

Overall, we can see that the presence of context effects necessitates that models trained on preference data need to incorporate these, either explicitly or implicitly. The inherent challenge of explicitly accounting for context effects lies in the prerequisite of having a priori knowledge concerning which specific context effects are present within the given dataset. The models presented over the course of this thesis are able to implicitly take the object context into account, thereby enabling the representation of previously unexplored context effects. In addition, as we transition from modeling human choices to learning arbitrary choice functions, the choices are the result of an abstract choice function rather than human decision making biases and heuristics. Consequently, prespecifying the correct context effects becomes difficult and should be learned in a data-driven way.

2.4. Preference Modeling

In Section 2.2, we examined various axiomatic systems that guarantee the existence of a utility function. What remains to be addressed is a systematic method through which one can deduce such a utility function from empirical

preference data. Consequently, this section is dedicated to an exploration of so-called RUMs. These models specify a probability distribution over individual utilities and can be inferred from observational data employing the method of maximum likelihood estimation–bridging the gap between theory and practice.

Early models of utility had their roots in binary outcome models. The first models of that kind that operated on binary outcomes were so-called *probit models* [Thu27]. These go back to work by Fechner [Fec60], but became popular due to research conducted by Gaddum [Gad33] and Bliss [Bli34]. Probit models are able to model binary outcomes by estimating the outcome probability based on the cumulative distribution function of the standard normal distribution.[7]

probit models

7: The term *probit* is a portmanteau of "probability" and "unit".

In the year 1951, Berkson [Ber51] argued for the use of the logistic function instead, causing a controversy at that time. The logistic function was derived by Verhulst [Ver38] for the study of populations. Berkson also proposed the term "logit" as an analogy for the widely used probit. The logistic curve is however very similar to the cumulative distribution function of the normal distribution, and the initial debate on which model should be preferred went away with the advent of computers that helped with the calculation. In the context of biological assays Gurland et al. [GLD60] needed an analysis method that supports more than two outcomes and were the first to use what is now called a multinomial logit model. Multinomial logit first by Gurland et al. [GLD60] then later generalized by various authors [BW67; Boc69; REB71; McF76; Sto69; The69; The70].

The initial adoption of the logit model in the domain of choice modeling was pioneered by McFadden [McF74]. He developed a principled theoretic framework for probabilistic choice, wherein the logit model emerged naturally, a contribution for which he was subsequently awarded the Nobel Memorial Prize in Economic Sciences in 2000 [McF01]. Models that follow this probabilistic framework are nowadays known as RUMs. We will take a closer look at these models in Section 2.4.1. An interesting extension of RUMs involves assuming the existence of multiply utility functions, which we will detail in Section 2.4.2.

2.4.1. Random Utility Models

The foundational derivation of RUMs by McFadden is encapsulated within his seminal book chapter dedicated to this subject [McF74]. We will follow the derivation outlined in the chapter here but adapt the language and notation to be consistent with the remainder of the thesis. An important point to make is that McFadden posits that each individual has a deterministic choice function. The actual randomness arises because we sample an individual randomly from a population. We then assume that there exists an underlying utility function on the population level and individual deviations can simply be regarded as random noise. The framework by McFadden allows one to also take attributes of the individuals into account. Let $S \subseteq \mathbb{R}^\ell$ be the set of individuals, where each $\boldsymbol{s} \in S$ is represented by a vector of attributes.

[McF74]: McFadden (1974), "Conditional Logit Analysis of Qualitative Choice Behavior"

Formally, the random utility function of an individual s can then be decomposed as follows:

$$U(s, x) = u(s, x) + \varepsilon(s, x) \tag{2.1}$$

Here u represents the deterministic, population-level utility component, while ε is the stochastic component that models the deviations observed for each individual. It is assumed that each individual makes choices by choosing the object x that maximizes $U(s, x)$. It can be shown [McF74, Lemma 1] that if we assume ε to be independent and identically distributed (i. i. d.) with a Weibull distribution, then we obtain the following well-known expression of the choice probability:

$$p(x \mid s, Q) = \frac{e^{u(s,x)}}{\sum_{x'} e^{u(s,x')}} \tag{2.2}$$

MNL model

This form is widely known as the *MNL model*. The same form can be derived by positing three axioms: IIA, strictly positive probabilities and irrelevance of an alternative set. The latter axiom requires that the log-odds ratio $\log(p_{xx'} / p_{x'x})$ can be represented by a simple difference:

IIA: Lemma 2.2.6

$$\log\left(\frac{p_{xx'}}{p_{x'x}}\right) = u(s, x) - u(s, x')$$

where $p_{xx'}$ denotes the probability that x is chosen from the task containing only x and x'. In case ε is assumed to be distributed according to a multivariate Gaussian distribution, we get the *multinomial probit model* [McF81].

multinomial probit model

As previously demonstrated, these axioms imply that the context effects introduced in Section 2.3 cannot be modeled. McFadden therefore cautions the use of the aforementioned model in situations where the axioms are not satisfied. In particular, the objects available should be clearly distinct.

In case the utility function $u(s, x)$ is a linear function with weight vector θ and we observe each choice of an individual just once, McFadden calls the resulting model the *conditional logit model*. The parameters of this model can be derived using maximum likelihood estimation.

conditional logit model

2.4.1.1. Variants and Generalizations

The MNL model assumes that the noise ε is independently distributed. A principled way to generalize the MNL model is to instead assume a multivariate extreme value distribution for ε. The resulting models are called generalized extreme value (GEV) models [McF81; Tra09]. For an in-depth exploration of GEV models, we direct the discerning reader to the comprehensive treatise by Train [Tra09], which shall serve as the foundational reference throughout this section.

[Tra09]: Train (2009), *Discrete Choice Methods with Simulation*

One representative of this class of models is the *nested logit (NL) model*. Recall that in Section 2.3, we introduced the *similarity effect*, where very similar objects share a common probability weight, and adding a new similar object will not steal probability weight from dissimilar objects. The default MNL model is not able to account for this context effect since it will receive a proportional share of probability weight from each other object (see (2.2)). In the nested

nested logit (NL) model

logit (NL) model, we partition all objects into so-called *nests*. We then posit that IIA holds within nests but not between objects of different nests.

Let $I(\mathcal{X})$ be a mapping of the set of objects to natural numbers for the purpose of indexing. We assume that $I(\mathcal{X}) = \bigsqcup_{i=1}^{K} B_i$, i. e., the objects are divided into disjoint sets. For each individual $s \in S$ let $\varepsilon_s = (\varepsilon_{s1}, \dots, \varepsilon_{s|\mathcal{X}|})$ be a vector of noise terms (also treated as unobserved utility deviations). Then a NL model is defined to have the following cumulative distribution:

$$\exp\left(-\sum_{k=1}^{K}\left(\sum_{i\in B_k} e^{-\frac{\varepsilon_{si}}{\lambda_k}}\right)^{\lambda_k}\right).$$

Each nest B_k has a parameter λ_k that indicates the independence of the objects within the nest. That means $\lambda_k = 1$ specifies that there is no correlation at all between the objects of B_k, and if this holds for all nests, the model simplifies to the usual MNL. From that, we can calculate the choice probability of the NL model as follows:

$$p(\boldsymbol{x} \mid \boldsymbol{s}, Q) = \frac{e^{u(\boldsymbol{s},\boldsymbol{x})/\lambda_k}\left(\sum_{j\in B_k} e^{u(\boldsymbol{s},\boldsymbol{x}_j)/\lambda_k}\right)^{\lambda_k-1}}{\sum_{k=1}^{K}\left(\sum_{j\in B_k} e^{u(\boldsymbol{s},\boldsymbol{x}_j)/\lambda_k}\right)^{\lambda_k-1}}.$$

One interesting consequence is that the random utility of an object $\boldsymbol{x}_i \in B_k$ for individual $\boldsymbol{s}$ decomposes additively:

$$U(\boldsymbol{s}, \boldsymbol{x}, B_k) = u_{B_k}(\boldsymbol{s}) + u(\boldsymbol{s}, \boldsymbol{x}) + \varepsilon(\boldsymbol{s}, \boldsymbol{x}),$$

where $u_{B_k}(\boldsymbol{s})$ is a constant utility assigned to each nest.

NL models can be nested more than one layer deep [McF77; BL18] to account for a hierarchical dependence structure. Benson et al. [BKT16] propose an efficient algorithm that can learn the tree structure using hypothesis testing. Their approach, however, requires the availability of pairwise comparisons in the preference data. There are also variants where objects can belong to multiple nests at the same time [Vov97; Bie98; BB99]. The *generalized nested logit (GNL)* by Wen and Koppelman [WK01] generalizes the latter models by allowing each object to be fractionally allocated to each nest. To this end allocation parameters $\alpha_{ik} \in [0, 1]$ for each object $\boldsymbol{x}_i$ and each nest B_k are introduced with $\sum_{k=1}^{K} \alpha_{ik}$ for all i. For the complete choice probability $p(\boldsymbol{x} \mid \boldsymbol{s}, Q)$ of an object $\boldsymbol{x}_i$, we refer to the publication by Wen and Koppelman [WK01]. It is, however interesting to note that the probability decomposes as follows:

generalized nested logit (GNL)

$$p(\boldsymbol{x} \mid \boldsymbol{s}, Q) = \sum_{k=1}^{K} p(\boldsymbol{x} \mid \boldsymbol{s}, Q, B_k)\, p(B_k \mid \boldsymbol{s}, Q),$$

where $p(B_k \mid \boldsymbol{s}, Q)$ is the probability that a nest B_k is chosen and $p(\boldsymbol{x} \mid \boldsymbol{s}, Q, B_k)$ denotes the probability that object $\boldsymbol{x}$ is chosen if we know that nest B_k was chosen. In other words, it is a weighted average of the probability that an object is being chosen within each nest.

A very general variant of the MNL model is the *mixed logit (ML) model*. As the name suggests, it is defined as the *mixture* of several logit models. This mixture can be discrete, with a set of parameters for each mixture model, or continuous where suitable prior distributions are specified for the parameters of an infinite

mixed logit (ML) model

amount of mixture models. These mixtures can very flexibly represent a variety of RUMs and McFadden [McF01] show that indeed the mixed logit (ML) models are universal, meaning that they are able to approximate any RUM.

While the aforementioned models can model some context effects, such as the similarity effect, their capabilities are constrained by several notable limitations. NL models require the explicit specification of nests, information that may not be available in practice. Conversely, ML models necessitate the specification of suitable priors for the weights of each feature. In addition, the interpretation of the resulting mixed model is difficult. The main limitation, however, is that it is assumed that the decision makers have an underlying utility function for the objects, and individual deviations are modeled as independent or correlated noise.

An alternative approach to arrive at a different utility-based framework is to assume the existence of multiple utility functions. These are then suitably aggregated to arrive at the overall utility of an object. This versatile framework will be considered in the following section.

2.4.2. Multiple Utility Functions

As we have seen in the preceding section, RUMs exhibit inherent limitations in modeling context effects. A major focus of this thesis is *context-dependent models*, models which are able to capture a wide range of context effects. In this section, we will explore a notable approach from the existing literature that accomplishes this to some extent. It does so by aggregating multiple *context-independent* utility functions into a possibly context-dependent utility. A very general framework in this domain was proposed by Ambrus and Rozen [AR15], which encompasses many multi-utility models previously suggested in the literature. These utility functions are frequently interpreted through a psychological lens. For instance, Kalai et al. [KRS02] describe these as "rationales", utilized to express preferences. In the context of research on addiction, Bernheim and Rangel [BR04] discuss various "systems" that interact within one individual. The now prevalent name of so-called "selves" was introduced by Fudenberg and Levine [FL06] and then used in the framework proposed by Ambrus and Rozen [AR15].

[AR15]: Ambrus et al. (2015), "Rationalising Choice with Multi-Self Models"

The *multiple selves* (or multi-self for short) framework introduced by Ambrus and Rozen [AR15] unifies several (singleton) choice models where a choice is made by aggregating multiple utilities produced by so-called selves. It is assumed that there is a *finite* set of objects $X \in \mathfrak{X}$, where $\mathfrak{X}$ is the class of all conceivable sets of objects. Then one observes choice tasks $Q \in 2^X$ and corresponding choices $c(Q)$, where $c\colon 2^X \to X$ is called a *collective* choice function. The framework also allows each self to have a certain *type* $t \in \mathcal{T}$. This generalization allows one to have aggregation functions that treat selves differently based on their type. A *self* is therefore defined as a tuple (u, t), consisting of a utility function $u\colon X \to \mathbb{R}$ and a type $t \in \mathcal{T}$. The set of available selves is denoted by $\mathcal{S}$.

self

Finally, the aggregated (collective) utility is computed for a given object $x \in X$ in the context of the choice task Q, the set of objects X, the set of selves $\mathcal{S}$ and the set of types $\mathcal{T}$:

$$f\colon X \times 2^X \times 2^{\mathcal{S}} \times \mathfrak{X} \times \mathcal{T} \to \mathbb{R}$$

In this general framework, the set $\mathcal{X}$ is an input because it would allow someone to aggregate based on which objects are not present in a given context. For most models in the literature, it is sufficient to drop the set of objects $\mathcal{X}$ and the set of types $\mathcal{T}$ from the signature.

loss aversion model

As an example, a well-known model covered by this framework is the *loss aversion model* proposed by Tversky and Kahneman [TK91].

$$f(x, Q, S) = \sum_{(u,t)\in S} m(u(x)) + \sum_{(u,t)\in S} \ell(u(x) - r(\{u(x')\}_{x'\in Q}))$$

where roughly speaking, an aggregated utility of an object is computed, and then a loss-aversion term is added. The loss-aversion term is the difference between the utility of the object and a reference utility r. A simple instantiation of this model has been introduced by Orhun [Orh09], which let m be a linear function, $\ell(a) := a$ if $a \geq 0$ and $\ell(a) := \lambda a$ otherwise. For the reference function r, they choose a weighted average. To see how we are able to model context-dependent utilities, consider Example 2.4.1.

The multi-self framework is, therefore, able to model context-dependent utility and hence also choice functions. It is remarkable that the utility functions employed by the selves are context-independent. Only by suitably aggregating the utilities evaluated for a given choice task do we arrive at a collective utility that is context-dependent. We shall revisit this concept in Chapter 6, where we introduce the concept of *Pareto-embeddings* that also use context-independent utility dimensions to achieve a context-dependent choice. The main difference will be that these utility dimensions will not be aggregated, as is the case for the multi-self framework.

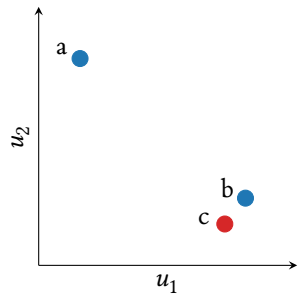


Figure 2.11.: Visualization of the points a, b and c along axes u_1 and u_2.

Example 2.4.1 (Context-dependence of the loss aversion model) Let $\mathcal{X} = \{a, b, c\}$ and assume that we evaluated their utility according to two utility functions u_1 and u_2 as depicted in Figure 2.11.
The exact values are given in the following table:

	a	b	c	r_{ab}	r_{abc}
u_1	1	7	6	4	4.7
u_2	10	3	2	6.5	5.0

We now use a simple variant of the model by Orhun [Orh09] to compute the aggregated utilities for the objects. We let $m := \mathrm{id}$ (the identity function) and r be the arithmetic mean. The loss-aversion parameter λ is set to 5. We will only have one type t, and therefore, our complete set of selves is $S = \{(u_1, t), (u_2, t)\}$.
Therefore, we get

$$\begin{aligned} f(a, \{a, b\}, S) &= 1 + 10 + \ell(1 - 4) + \ell(10 - 6.5) = 14.5 - \lambda \cdot 3 \\ &= 14.5 - 15 = \mathbf{-0.5} \\ f(b, \{a, b\}, S) &= 7 + 3 + \ell(7 - 4) + \ell(3 - 6.5) = 13 - \lambda \cdot 3.5 \\ &= 13 - 17.5 = -4.5 \\ f(a, \{a, b, c\}, S) &= 1 + 10 + \ell(1 - 4.7) + \ell(10 - 5) = 16 - \lambda \cdot 3.7 \\ &= 16 - 18.5 = -2.5 \\ f(b, \{a, b, c\}, S) &= 7 + 3 + \ell(7 - 4.7) + \ell(3 - 5) = 12.3 - \lambda \cdot 2 \\ &= 12.3 - 10 = \mathbf{2.3} \end{aligned}$$

As we can see, the introduction of object c caused the preference between objects a and b to flip. On the set $\{a, b\}$, object a was preferred, while for the set $\{a, b, c\}$, object b achieved the higher utility.

3. Concepts from Machine Learning

While the majority of the literature on preference and choice modeling has focused on normative and descriptive approaches, the availability of large-scale preference data facilitated the application of more data-centric methods from the field of machine learning. It is clear that the world of modern e-commerce has a variety of settings where users express preference information. Think of a search engine where a user inputs a search term, receives a list of relevant results, and finally clicks on one or more of the links. The click can be treated as implicit preference information in the form of a choice, while the complete list of results is the set of alternatives. Another area where choices appear frequently is implicit feedback in the context of recommendations. Streaming services preselect a set of shows, movies, songs, and other content for the user, who then chooses which to play. The objective is to identify and recommend content with a high probability of being chosen by the user.

A big difference in philosophy between classical preference modeling and machine learning is the choice of models. In classical preference modeling, it is important that the models used are behaviorally plausible, i. e., it can be used to explain observed choices, and that the necessary assumptions/axioms are consistent with human behavior. In machine learning, however, the main goal is to maximize predictive performance, which means we are looking for a model that works well when applied to new data. These approaches certainly are not mutually exclusive. A good classical preference model should also be able to generalize to new data, while there are also some machine learning models that explicitly incorporate a behaviorally plausible inductive bias [ROS20].

We will now take a closer look at the main concepts of machine learning. As before, we will have a space $\mathcal{X}$, but this time called the *instance space*. Each $\boldsymbol{x} \in \mathcal{X}$ is called an *instance* and could for example be an image, if the setting is image classification or a textual document, in the setting of natural language processing. Often, the instance space is assumed to be a d-dimensional real-valued space, such that $\mathcal{X} \subseteq \mathbb{R}^d$ and $\boldsymbol{x} = (x_1, x_2, \dots, x_d)$ for each $\boldsymbol{x} \in \mathcal{X}$, but this is not necessarily always the case.

instance space

Usually, we will also have a corresponding *output space* $\mathcal{Y}$, which provides the values or labels we actually care about. Take, for instance, image classification, where an appropriate output space could be the set of all category labels in the English language. Thus, if we are given a picture depicting a cat, we would be able to assign to it the label "cat". Depending on what we choose for $\mathcal{Y}$, we usually end up with different machine learning settings. If $\mathcal{Y}$ is a continuous space, we call the problem a *regression* problem, while if $\mathcal{Y}$ is a discrete space, we call it a *classification* problem. Here an important special case is binary classification, where $\mathcal{Y} = \{-1, +1\}$ or $\{0, 1\}$. Since the values of the output space, in a sense, "tell" the learner what to predict, the setting overall is called *supervised learning*.

output space

Certainly, there are many other settings in machine learning, such as unsupervised learning, where the learner does not have access to an explicit output space and has to find patterns in the data on its own. Other settings include

reinforcement learning, where the learner interacts with an environment and receives rewards for its actions, or semi-supervised learning, where only a small portion of the data is labeled. This thesis is concerned with supervised learning, which is why we will focus on this setting for the remainder of this chapter.

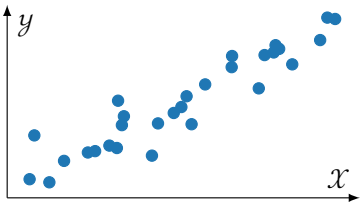


Figure 3.1.: A simple regression problem in one dimension.

We will then assume that both spaces are related by a function $f\colon \mathcal{X} \to \mathcal{Y}$. This is the function we generally want to model using machine learning. Consequently, it is also called the *target function*. Since data in practice is rarely noise-free, the usual assumption is that $f(\boldsymbol{x})$ is corrupted by noise to arrive at $y \in \mathcal{Y}$. As an example, one commonly describes the relation between $\boldsymbol{x}$ and $\boldsymbol{y}$ using

$$Y = f(X) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)\,,$$

where the random variable ϵ is independent and identically distributed noise following a normal distribution (see Figure 3.1 for an example). Since the noise does not depend on the value of the input instances $\boldsymbol{x} \in \mathcal{X}$, it is called *homoscedastic noise*. Correspondingly, if we were to assume that the noise can vary based on the current inputs, then we call it *heteroscedastic noise*. In general, we will treat both the input instances $\boldsymbol{x}$ and the outputs y to be random variables with the corresponding joint probability distribution $P(\boldsymbol{x}, y)$.

homoscedastic noise

heteroscedastic noise

One could say the main goal of machine learning is finding the target function f. However, this goal presents several challenges. Firstly, in most cases, the class of functions to which f belongs is unknown, making the search problem ill-defined. Secondly, we do not yet have a notion of how "close" a guess for f is to the ground truth target function. To address the former problem, we will assume that there exists a *model space* $\mathcal{H}$ containing models $h\colon \mathcal{X} \to \mathcal{Y}$.[1] Thus, instead of considering the class of all possible functions, we restrict our search to models $h \in \mathcal{H}$, which usually is some clearly defined class of functions.

model space

1: This model space is often referred to as the "hypothesis space".

As for the latter problem, we will define a function $L\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ called a *loss function*, which measures the mismatch between an observed output y and the prediction of a model $\hat{y} = h(\boldsymbol{x})$. Roughly speaking, our goal will be to select a model for which the average loss on a given dataset is minimal. Simple examples of loss functions include the 0/1 loss

loss function

$$L(y, \hat{y}) = [\![y \neq \hat{y}]\!]$$

which is applicable to classification problems, or the squared loss

$$L(y, \hat{y}) = (y - \hat{y})^2$$

which can be used in regression. As an example, consider the linear regression problem in Figure 3.2. The shaded squares depict the deviations of the observed points from the model line. The area of each of these squares is exactly the squared loss.

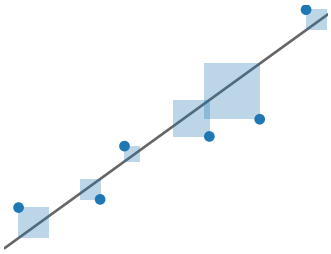


Figure 3.2.: The mean squared error penalizes the squared deviations of the observed outputs and the model outputs.

So far, this loss function only measures the error for one pair $(y, h(\boldsymbol{x}))$, but we would like to quantify how well a given model h works in general. This is why we consider the expected loss

$$\mathcal{R}(h) = \int L\big(y, h(\boldsymbol{x})\big)\, dP(\boldsymbol{x}, y)\,,$$

also called the *risk*. It is natural now to seek models which minimize the risk

risk

$$h^* \in \arg\min_{h \in \mathcal{H}} \mathcal{R}(h) \, .$$

This is sensible, but in most cases, the joint distribution $P(\boldsymbol{x}, y)$ is not known. Thus, we require one last step before we can apply machine learning in practice.

While we do not have access to the joint probability distribution, we will assume that there is training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \in (\mathcal{X} \times \mathcal{Y})^N$ consisting of a set of N instances $(\boldsymbol{x}_i, y_i)$. With that, we can approximate the risk using the average loss

$$\mathcal{R}_{\text{emp}}(h) = \frac{1}{N} \sum_{i=1}^{N} L\big(y_i, h(\boldsymbol{x}_i)\big) \tag{3.1}$$

obtained on the training instances $\mathcal{D}$, called the *empirical risk*. The corresponding *empirical risk minimizer* can be defined analogously to h^* as

empirical risk

$$\hat{h} \in \arg\min_{h \in \mathcal{H}} \mathcal{R}_{\text{emp}}(h) \, .$$

The process is in general known as empirical risk minimization (ERM). However, approximating the risk using finite data does not come without its own problems. As an example, assume that we let $\mathcal{Y} = \mathbb{R}$ and $\mathcal{H}$ be the set of all polynomials. Then for any finite $\mathcal{D}$ we can find a polynomial h, which simply interpolates all points $\boldsymbol{x}_i$.

empirical risk minimization

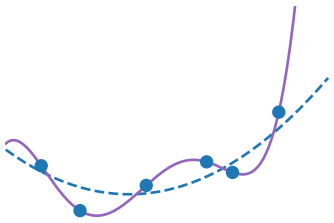


Figure 3.3.: A polynomial of degree 5 (purple) fits all training examples perfectly.

See Figure 3.3 for an example of this phenomenon. Here, we have simulated a dataset of 6 points in one dimension. The target function is shown in blue dashed lines, and the 6 points had noise applied to their output values. We let $\mathcal{H}$ be the set of all polynomials up to the degree 5 and compute the empirical risk minimizer $\hat{h}$ (shown in purple). As is apparent, the risk minimizer achieves a loss of 0 on the training examples, but is exhibiting large deviations from the target functions as soon as we move away from the given points. Therefore, the model $\hat{h}$ is not able to *generalize* well to points not part of the training set. This phenomenon is called *overfitting* and demonstrates a key challenge in the field of machine learning. In Section 3.1, we will take a closer look how this problem can be detected and tackled.

overfitting

3.1. Generalization

We have seen that by approximating the risk with the empirical risk and selecting the corresponding empirical risk minimizer $\hat{x}$, we may overfit the training data. The problem is a mismatch of the in-sample performance $\mathcal{R}_{\text{in}} = \mathcal{R}_{\text{emp}}$ and the out-of-sample performance $\mathcal{R}_{\text{out}} = \mathcal{R}$. This raises several questions. First, how the difference $|\mathcal{R}_{\text{out}} - \mathcal{R}_{\text{in}}|$ can be characterized, both theoretically and practically. Second, if learning is at all possible, and if yes, under which conditions.

3.1.1. Theoretical Considerations

There are a few general bounds one can prove for the difference between in-sample risk and out-of-sample risk. For an i.i.d. set of data $\mathcal{D}$ and a fixed model $h \in \mathcal{H}$, we can apply the Hoeffding inequality to arrive at the bound

$$P(|\mathcal{R}_{\text{out}} - \mathcal{R}_{\text{in}}| > \epsilon) \leq 2\exp(-2\epsilon^2|\mathcal{D}|)\,,$$

which states that as the size of our dataset $\mathcal{D}$ increases, the in-sample estimate of the risk approaches the out-of-sample risk very quickly. This bound, however, ignores the complete learning process used to select an appropriate model h. Since a model arrived at using ERM and $\mathcal{R}_{\text{in}}$ is minimized, it will probably underestimate the true $\mathcal{R}_{\text{out}}$.

VC inequality

In the case of binary classification, i.e., $\mathcal{Y} = \{-1, +1\}$, an important result is the *Vapnik–Chervonenkis (VC) inequality*:

Theorem 3.1.1 (VC inequality [VC71]) *Let $\mathcal{X}$ be an instance space, $\mathcal{Y} = \{-1, +1\}$ an output space and $\mathcal{H}$ a model space with $h : \mathcal{X} \to \mathcal{Y}$ for $h \in \mathcal{H}$. Let $\mathcal{D}$ be a dataset of size N and $m_{\mathcal{H}} : \mathbb{N} \to \mathbb{N}$ be the growth function of $\mathcal{H}$. Then for all $\epsilon > 0$:*

$$P\left(\sup_{h\in\mathcal{H}} |\mathcal{R}_{\text{out}}(h) - \mathcal{R}_{\text{in}}(h)| > \epsilon\right) \leq 4m_{\mathcal{H}}(2N)\exp\left(-\frac{1}{8}\epsilon^2 N\right)$$

growth function

The *growth function* $m_{\mathcal{H}}$, roughly speaking, specifies the effective number of distinct hypotheses that a hypothesis space $\mathcal{H}$ generates on a dataset $\mathcal{D}$. We will not go into the exact details of its definition here and refer the interested reader to the excellent overview by Abu-Mostafa et al. [AML12, Chapter 2]. It is easy to see that the bound decays exponentially in the dataset size N. Therefore, if there exists an $N \in \mathbb{N}$ such that for all $k \geq N$ we have that $m_{\mathcal{H}}(k) < 2^k$, we know that that the bound will eventually go to 0. For a given hypothesis space $\mathcal{H}$, we call the largest N for which it holds that $m_{\mathcal{H}}(N) = 2^N$ its *VC dimension*. If $m_{\mathcal{H}}(N) = 2^N$ for all $N \in \mathbb{N}$, the VC dimension is infinite, and the VC inequality can no longer guarantee that the in-sample risk converges to the out-of-sample risk. The VC inequality can be reordered to arrive at a direct bound of the out-of-sample risk.

VC dimension

Theorem 3.1.2 (VC generalization bound [AML12]) *Let $\mathcal{X}$ be an instance space, $\mathcal{Y} = \{-1, +1\}$ an output space and $\mathcal{H}$ a model space with $h : \mathcal{X} \to \mathcal{Y}$ for $h \in \mathcal{H}$. Let $\mathcal{D}$ be a dataset of size N and $m_{\mathcal{H}} : \mathbb{N} \to \mathbb{N}$ be the growth function of $\mathcal{H}$. Then for all $\delta > 0$:*

$$\mathcal{R}_{\text{out}}(\hat{h}) \leq \mathcal{R}_{\text{in}}(\hat{h}) + \sqrt{\frac{8}{N}\log\frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

From this very general result, we can deduce that learning is indeed possible, if we consider hypothesis spaces with a finite VC dimension.

Another aspect we can gauge with these generalization bounds is the *complexity* of a hypothesis space. In the case of binary classification, if the VC dimension is higher, we need a higher number of training instances N to constrain the out-of-sample performance $\mathcal{R}_{\text{out}}$. The same is true in other settings as well.

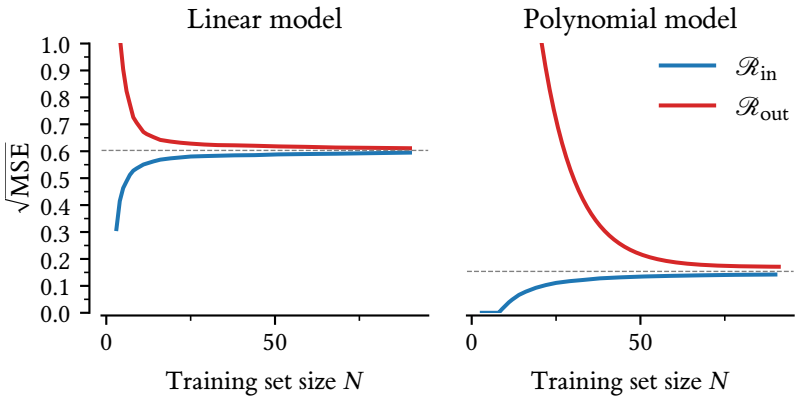


Figure 3.4.: Learning curves for a linear and a polynomial model of degree 7.

learning curves

To visualize this, consider Figure 3.4, where we plot the *learning curves* for a simple linear model and a more complex polynomial model (of degree 7). We train each model on a training dataset of size N and do so for all N shown. Then, the in-sample performance $\mathcal{R}_{\text{in}}$ is calculated on the training data, while the out-of-sample performance $\mathcal{R}_{\text{out}}$ is computed on a separate holdout dataset. Our loss function is the root-mean-square error $\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{h}(\boldsymbol{x}_i) - y_i\right)^2}$. We can observe that the $\mathcal{R}_{\text{in}}$ is always underestimating $\mathcal{R}_{\text{out}}$, but both converge to a common limit. The linear model needs fewer training instances than the more complex polynomial model to get close to its limit. The polynomial model, however, is ultimately able to achieve a lower $\mathcal{R}_{\text{out}}$. The limiting performance is also called the *bias* with respect to the target function. A more complicated model usually is more flexible and is able to adapt to more target functions, hence a lower bias. The deviation of $\mathcal{R}_{\text{out}}$ from the limit value is called the *variance*, showing how much a hypothesis class is able to adapt to small variations in the data, causing large errors.

bias

variance

In practice, we will not be able to compute the out-of-sample performance $\mathcal{R}_{\text{out}}$ and the generalization bounds are too loose to be useful. Therefore, we need an alternative way to approximate $\mathcal{R}_{\text{out}}$.

3.1.2. Validation

The idea of validation is remarkably simple: first split the dataset $\mathcal{D}$ into a training set $\mathcal{D}_{\text{train}}$ (size $N - M$) and validation set $\mathcal{D}_{\text{val}}$ (size M)[2]. Then, train the model only on $\mathcal{D}_{\text{train}}$, which we will call h^- to signify that it was trained on a reduced set of data. We can then compute the *validation risk*

2: Usually, we will split the dataset into $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{test}}$, since the validation set is used for *model selection* or *hyperparameter optimization*.

$$\mathcal{R}_{\text{val}}(h^-) = \frac{1}{M}\sum_{\boldsymbol{x}_i \in \mathcal{D}_{\text{val}}} L\left(h^-(\boldsymbol{x}_i), y_i\right),$$

which is an unbiased estimate of the out-of-sample risk $\mathcal{R}_{\text{out}}$, since the data was not used to train the model h^-. Formally, this means that $\mathbb{E}_{\mathcal{D}_{\text{val}}}[\mathcal{R}_{\text{val}}(h^-)] = \mathcal{R}_{\text{out}}(h^-)$. This estimate is also quite accurate. In the case of binary classification, we can show that the following bound holds with high probability:

$$\mathcal{R}_{\text{out}}(h^-) \leq \mathcal{R}_{\text{val}}(h^-) + \mathcal{O}\left(\frac{1}{\sqrt{M}}\right).$$

We have to take into account that the larger we make $\mathcal{D}_{\text{val}}$, the fewer training instances $\mathcal{D}_{\text{train}}$ we have, resulting in a higher $\mathcal{R}_{\text{out}}$. Thus, by increasing M, we reduce the error of our estimate but increase the overall risk, making the choice of the right M a trade-off. It is usual in practice to train a model on the complete dataset $\mathcal{D}$ after performing the validation to estimate the risk. This will result (on average) in a reduction of the out-of-sample risk and somewhat counteracts the disadvantage of using a validation set.

3.1.2.1. Cross-Validation

A problem with validation so far is that if M is small, then the variance of our estimate $\mathcal{R}_{\text{val}}$ is high. One idea to reduce this variance is to repeat the split of our dataset $\mathcal{D}$ several times using different sets for training and validation and subsequently averaging the risk estimates for more stable results. There are several ways to do this, the most common approach of which is called *K-fold cross-validation*, where we partition the dataset into $K := \lceil N/M \rceil$ sets $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \dots, \mathcal{D}^{(K)}$. Let $h_1, h_2, \dots, h_K$ be the corresponding models obtained by training on the remaining data, i.e., h_i is obtained by training on $\mathcal{D}^{(1)} \cup \dots \cup \mathcal{D}^{(i-1)} \cup \mathcal{D}^{(i+1)} \cup \dots \mathcal{D}^{(K)}$ and validated on $\mathcal{D}^{(i)}$. The cross-validation risk estimate is then computed as follows:

K-fold cross-validation

$$\mathcal{R}_{\text{val}} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|\mathcal{D}^{(k)}|} \sum_{(\boldsymbol{x},y)\in\mathcal{D}^{(k)}} L\big(h_k(\boldsymbol{x}), y\big) \,.$$

Note how this estimate is no longer computed for only one hypothesis but for a whole set of hypotheses. Since the validation sets are not overlapping, $\mathcal{R}_{\text{val}}$ is an unbiased estimator for the out-of-sample risk on datasets of size $N - M$. As we are interested in estimating the out-of-sample risk for our given dataset $\mathcal{D}$, it is natural to choose a K that is as large as possible. If $K = N$, then the resulting method is called *leave-one-out cross-validation*. Zhang and Yang [ZY15], based on theoretical and empirical results, find that leave-one-out cross-validation has the lowest bias and variance for stable modeling procedures. While this provides us with a good estimate of $\mathcal{R}_{\text{out}}$, it can be computationally intensive to compute N hypotheses on larger datasets. In practice, one would select K so that a good trade-off between estimation accuracy and computational overhead is achieved.

leave-one-out cross-validation

The stability of cross-validation estimates can be further reduced by repeatedly shuffling the data and repeating the same cross-validation method. In the limit, this methodology will converge to *leave-p-out cross-validation*, where we average the validation loss of all $\binom{N}{p}$ possible splits of the data. It is clear that for even slightly larger p, the exhaustive evaluation of all splits becomes infeasible. For example, computing the full leave-5-out cross-validation for a dataset of 100 instances would require the evaluation of 75 287 520 splits. Therefore, the idea of *Monte Carlo cross-validation* is to repeatedly sample a random set of p instances (without replacement) to be used as a validation set. The number of repetitions is usually fixed beforehand as well. The advantage over K-fold cross-validation is that the size of the validation set and the number of repetitions can be controlled independently. This is also why this method was chosen as the main validation method in the experimental section of this thesis.

leave-p-out cross-validation

Monte Carlo cross-validation

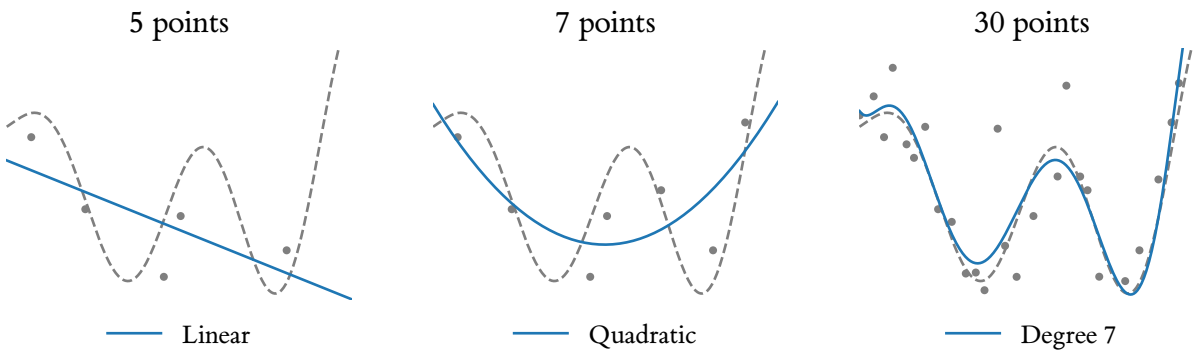


Figure 3.5.: Choosing the best polynomial degree for a given dataset based on leave-one-out cross-validation. The target function is shown as a dashed gray line.

3.1.2.2. Uses of Validation

Other than simply having an estimate of the out-of-sample performance of a model h, an important use of validation is to find the correct model class $\mathcal{H}$ for the given dataset. This problem is called *model selection*. Another application concerns the control of the complexity of hypotheses inside a model class. Models typically have various parameters that impact complexity. In linear regression, for instance, one typically penalizes the L_2-norm of the model coefficients, a very important concept called *regularization*. A parameter is introduced, which allows one to trade off between model fit to the training data and a low norm of the parameters. In general, these parameters[3], which influence the complexity of the model class, should not be optimized on the training set $\mathcal{D}_{\text{train}}$ by minimizing $\mathcal{R}_{\text{in}}$. As we have seen before, a more complex model class $\mathcal{H}$ can fit a given set of data more easily than a more simple, constrained model class. Thus, minimizing hyperparameters on $\mathcal{R}_{\text{in}}$ will result in the choice of the most flexible model class, likely causing overfitting to occur. The task to find good values for hyperparameters is called *hyperparameter optimization (HPO)*.

model selection

regularization

3: Usually called *hyperparameters* to distinguish them from the model parameters.

hyperparameter optimization

Since $\mathcal{R}_{\text{val}}$ is an unbiased estimator for $\mathcal{R}_{\text{out}}$ (albeit for a smaller dataset size), it is natural to use it to solve model selection and HPO tasks. Of course, by minimizing $\mathcal{R}_{\text{val}}$, we again introduce a bias similar to what we did when doing ERM, i. e., on average we underestimate the true $\mathcal{R}_{\text{out}}$. Therefore, one should be aware of the fact that optimizing over many different hypothesis classes and parameters can again introduce the risk of overfitting.

As an example, consider polynomials of a certain degree d, and let $\mathcal{H}_d$ denote the corresponding hypothesis class. In Figure 3.5, we sample random points from a one-dimensional target function (shown as a dashed gray line) plus independent Gaussian noise, varying the number of points available for training. Then, we employ leave-one-out cross-validation to select the best degree d for each of the three datasets. We can see that with increasing number of points, the complexity of the chosen model class increases as well, moving from a linear model for 5 points ($\mathcal{H}_1$) to a polynomial model of degree 7 ($\mathcal{H}_7$) for 30 points. Note that due to variance, this order is not always monotone. Especially with few data points, it can happen that they randomly arrange into specific patterns favoring higher-order polynomials. The example in Figure 3.5 shows one realization where none of these problematic cases happened, but one needs

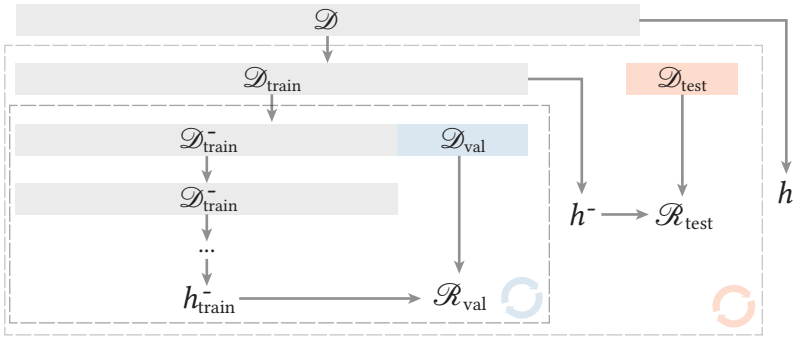


Figure 3.6.: Overview of the nested validation process.

to be mindful when considering smaller datasets and additional regularization may be beneficial.

As we can see, validation is an important part of the machine learning process, and crucial to ensure that the model we use in the end generalizes to unseen data. In Section 3.2, we will go into more detail on how we can optimize the hyperparameters of a model. First, we will take a small detour to discuss *nested validation*, which allows us to both employ validation to optimize parameters and also get an unbiased estimate of $\mathcal{R}_{\text{out}}$.

3.1.2.3. Nested Validation

As we have seen, we can use validation to select a suitable model class for the data at hand or to optimize the hyperparameters of our model. In doing so, we also biased the resulting $\mathcal{R}_{\text{val}}$ since we used it for the optimization. We may still want to have an unbiased estimate of $\mathcal{R}_{\text{out}}$ for our final model to make an informed decision on whether to use the model or not. Furthermore, we may want to separate the HPO task from the model selection task, i. e., optimizing the hyperparameters of each model separately and then use fresh validation data to decide on the final model to use. The natural idea of *nested validation* is, therefore, to partition the dataset $\mathcal{D}$ into the sets $\mathcal{D}_{\text{train}}$, $\mathcal{D}^{(1)}_{\text{val}}$, $\mathcal{D}^{(2)}_{\text{val}}$, ..., $\mathcal{D}_{\text{test}}$. Here $\mathcal{D}_{\text{test}}$ is called the *test set* to distinguish it from the validation sets, and is used to compute the final unbiased estimate of $\mathcal{R}_{\text{out}}$. As before, $\mathcal{D}_{\text{train}}$ is used to train a model using *ERM*. The validation sets $\mathcal{D}^{(1)}_{\text{val}}$, ... can then be used for HPO and model selection. In most cases, only one or two validation sets are necessary.

nested validation

The general process is shown in Figure 3.6. We start with the full dataset $\mathcal{D}$ and split it into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ (orange cycle). This is often called the *outer* validation split. Then, $\mathcal{D}_{\text{train}}$ is further subdivided into $\mathcal{D}^{-}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ in what is called the *inner* validation split. With $\mathcal{D}^{-}_{\text{train}}$ we can then train a model h^{-}_{train}. Alternatively, we can nest more validations (depicted by the three dots), if required. In any case, h^{-}_{train} is then used to produce predictions on $\mathcal{D}_{\text{val}}$ to compute $\mathcal{R}_{\text{val}}$. If the split is repeated, for example, when using K-fold cross-validation, then we average the validation performances obtained on each split: $\mathcal{R}_{\text{val}} = \frac{1}{K}\sum_{i=1}^{K} \mathcal{R}^{(i)}_{\text{val}}$. This estimate can then be used for HPO and/or model selection. In that case, the blue cycle is repeated several times, once for each hypothesis space for model selection or once for each iteration of the HPO. The information gained by running the inner validation cycle is then used when training the model h^{-} on $\mathcal{D}_{\text{train}}$. That is, we use the best parameters θ^* we found during HPO and/or select the best hypothesis space $\mathcal{H}^*$ here. Then

this model predicts on $\mathcal{D}_{\text{test}}$ and we compute our unbiased estimate of the risk, which we denote $\mathcal{R}_{\text{test}}$ here. The orange cycle symbol signifies that this outer validation fold can be repeated as well. Note that this entails performing the complete HPO/model selection again on each split. Finally, we train a model h on the complete dataset. Since we do not have a global estimate for θ^* and/or $\mathcal{H}^*$, we either need to pick randomly from one of the splits, or rerun the optimization process on $\mathcal{D}$. The latter will, on average, result in a model with lower $\mathcal{R}_{\text{out}}$, but is computationally more costly.

This process can, in principle, be nested as deep as data is available. However, it is clear that this will decrease the amount of training data available to actually fit a model. Recall that if we decrease the size of the dataset, we will, on average, favor simpler models, which is why excessive nested validation is prone to underfitting the available data. Another consideration is the computational overhead grows exponentially in the number of splits. For example, if 10-fold cross-validation is used for a split into 4 datasets $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}^{(1)}$, $\mathcal{D}_{\text{val}}^{(2)}$ and $\mathcal{D}_{\text{test}}$, then this would require the evaluation of at least $10^3 = 1000$ many splits. This number grows further if HPO is employed in any of the folds since it needs to train models several times until a good parameter configuration is found.

In practice, one has to decide how many splits to do at each level, how many instances to split off and which validation method to employ. This decision is informed by the desired running time, precision of error bars, quality of the final model and more considerations.

3.2. Hyperparameter Optimization

Nowadays, in machine learning research, it is important to do hyperparameter optimization (HPO) in a principled way, not only to eke out the last bit of performance of a model, but more importantly, to facilitate a fair comparison of models [Bis+23; Egg+21]. Bouthillier and Varoquaux [BV20] surveyed the HPO methodology of 786 papers submitted to the NeurIPS2019 and ICLR2020 conferences. They found that 93.7 % of those respondents who do optimize the hyperparameters of their approaches use simple methods like manual tuning (45.7 %), grid search (40.6 %) or random search (7.4 %). More worryingly, 22.6 % of researchers did not optimize the hyperparameters of their baselines at all, which casts doubt on the fairness of such experiments. Consequently, we will go into a bit more detail here on how HPO and specifically *Bayesian optimization* can be done in a principled way. The excellent book by Garnett [Gar23] served as a reference throughout this section.

[Gar23]: Garnett (2023), *Bayesian Optimization*

Given a parameter space Θ and a parametrizable hypothesis space $\mathcal{H}_\theta$, we can write the HPO problem as

$$\arg\min_{\theta\in\Theta} \mathcal{R}_{\text{val}}(h_\theta) \quad \text{where } h_\theta \in \arg\min_{h\in\mathcal{H}_\theta} \mathcal{R}_{\text{train}}(h)\,.$$

The important part here is the optimization over Θ, which is why it makes sense to reframe this problem as a more general function optimization problem:

$$\theta^* \in \arg\min_{\theta\in\Theta} f(\theta)\,, \tag{3.2}$$

where $f\colon \Theta \to \mathbb{R}$ is a real-valued objective function that we want to minimize. Think of it as a black box function, where the machine learning process and computing the validation loss is hidden from us. We can only query points θ and get the corresponding function value $f(\theta)$.

In the field of function optimization, there are various methods to solve this problem, usually with different trade-offs in terms of sample efficiency, computational cost, and robustness. The correct method depends on the properties of the function f. In our case, we first assume that no gradient or higher-order derivative information is available. Often, hyperparameters of a machine learning model are discrete, e. g., the number of layers and units in a neural network, which is why a gradient does not exist. However, even in the cases where we restrict ourselves to real-valued parameters, the gradient is often costly to evaluate, requiring the backpropagation of the validation loss through the complete validation procedure [MDA15]. Secondly, we assume that f is costly to evaluate, motivating the use of sample-efficient optimization methods, and ruling out the exhaustive computation of high-dimensional parameter grids. Finally, we do not restrict the objective function in any way to be "easy" to optimize, e. g., we do not assume that f is convex, smooth, or even continuous. This setting is usually called *black box optimization* since nothing is known about f and we can only use it to query points and to get a corresponding value.

black box optimization

The problem is a *sequential decision problem*, where the optimizer has to repeatedly decide on a point to evaluate using f, updating its internal model of f while doing so. Of course, the optimizer does not know the output of the function f in advance, and therefore needs to account for this uncertainty. A particularly successful approach is *Bayesian optimization*, which learns a surrogate model of f able to express the uncertainty regarding possible function values at previously unseen points. As we shall see, this allows it to employ efficient search strategies to arrive at a global optimum quickly.

sequential decision problem

Bayesian optimization

3.2.1. Bayesian Decision Theory

The hyperparameter optimizer starts with a dataset $\mathcal{D}$ of points that have already been evaluated. Another input is the optimization space Θ, usually assumed to be a subset of $\mathbb{R}^d$. The main decision the optimizer has to make is choosing a point $\boldsymbol{\theta} \in \Theta$ to evaluate. As is usual in decision theory, we will define a utility function $\alpha\colon \Theta \to \mathbb{R}$ which assigns a degree of usefulness to a given point of the optimization space. We adapt this function based on the data $\mathcal{D}$ we already have, which we denote by $\alpha(\theta; \mathcal{D})$. At time step $t \in \mathbb{N}$, the optimizer will then solve the following optimization problem

$$\boldsymbol{\theta}_t \in \arg\max_{\boldsymbol{\theta} \in \Theta} \alpha(\boldsymbol{\theta}; \mathcal{D}_t) \tag{3.3}$$

to determine the next point $\boldsymbol{\theta}_t$ to try. It then evaluates $y_t \coloneqq f(\boldsymbol{\theta}_t)$. The tuple (θ_t, y_t) is then appended to the dataset to get $\mathcal{D}_{t+1} \coloneqq \mathcal{D}_t \cup \{(\boldsymbol{\theta}_t, y_t)\}$. Since the responsibility of function α is to help in the "acquisition" of useful points during the optimization process, it is widely called an *acquisition function*. The reader may notice that the optimization problem (3.3) is very similar to the actual optimization problem (3.2). The difference is that we assume f to be costly and/or time-consuming to evaluate. α is usually cheap to compute, and

acquisition function

as long as the optimization problem (3.3) can be solved orders of magnitude faster than evaluating f, this is beneficial.

Many acquisition functions in the literature can be constructed using *Bayesian decision theory*, which is the application of Bayesian modeling to decision theory in order to facilitate making decisions under uncertainty. The first question we have to answer is, what do we even care about when performing HPO? Usually, the optimization process is run for a fixed number of iterations T, after which we end up with a point $\hat{\boldsymbol{\theta}}$. The points $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_T$ and their results $y_1, \dots, y_T$ are not important for the HPO task, since only the final recommendation is used to parametrize our machine learning model. Formally, this is expressed by what is called the *simple regret*

simple regret

$$r_T = \min_{\boldsymbol{\theta} \in \Theta} \mu_{\mathcal{D}_T}(\boldsymbol{\theta}) - f^* \,, \tag{3.4}$$

where f^* is the ground-truth minimum of f and $\mu_{\mathcal{D}_T}$ is the surrogate model of the optimizer at time step T. This is opposed to settings where we want the whole sequence $y_1, y_2, \dots$ to be as low as possible. This can be a useful measure in applications where the intermediate rewards are important, such as in online advertising, clinical trials, or reinforcement learning. More precisely, in these settings, we want to minimize the *cumulative regret*

cumulative regret

$$R_T = \sum_{t=1}^{T} (y_t - f^*) = \sum_{t=1}^{T} y_t - T f^* \,.$$

The goal then is to define an optimizer, which achieves a *sublinear* cumulative regret, i.e., $\lim_{t \to \infty} R_T / T = 0$.

In the following, we will go into more detail on what suitable surrogate models are for Bayesian optimization. Finally, we will consider typical acquisition functions used in practice in Section 3.2.3.

3.2.2. Surrogate Models

In order to use Bayesian optimization for HPO, we need a *surrogate model*. As the term suggests, it will act as a proxy for the ground-truth function f. There are a few properties a model should have to be a viable surrogate model. Most importantly, it should be much less costly to evaluate than f. Otherwise, one could work directly with f. Secondly, it has to be possible to compute some measure of the *predictive uncertainty*, meaning that the model should be able to express how certain it is about the true function value at a given point.

Consider, for example, the situation shown in Figure 3.7 where we evaluated our (noise-free) target function f at two different points. It is clear that our surrogate model is certain with respect to the predicted function value at exactly those two points while being more uncertain everywhere else. A good way to think about the surrogate model is to treat it as a model space $\mathcal{H}_{\mathcal{D}}$ which roughly contains all models consistent with the data. In Figure 3.7 we see 10 randomly chosen models $h \in \mathcal{H}_{\mathcal{D}}$. The shaded region depicts the 95 % region of the surrogate model, and we can see that it covers its own uncertainty nicely. This uncertainty is called *epistemic uncertainty* [Hor96; KD09; HW21], because the underlying target function in this case is noise-free, and the only

epistemic uncertainty

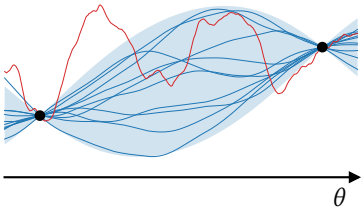


Figure 3.7.: A good surrogate model is able to express its epistemic uncertainty. The further we move away from data points, the more the variance of consistent models increases. The red curve shows a function consistent with the data, but violating the smoothness assumption.

"randomness" is due to not knowing the precise model. If we observe more and more data, i. e., $|\mathcal{D}| \to \infty$, we expect this uncertainty to go to 0.

The ability to inter- and extrapolate well is, of course, dependent on the ability of the surrogate model to model the smoothness and continuity of the target function well. In Figure 3.7, we can also see the red curve that goes through both points of the dataset. The difference is that it has a much lower smoothness and is not well covered by the blue uncertainty region. Deciding on the correct smoothness to use for the data at hand is again a model selection problem, albeit on a meta-level.

There is a variety of different models which have been proposed as surrogate models in the literature. Gaussian processes are a natural choice since they are, by virtue of being a probabilistic model, able to express their epistemic uncertainty [RW06; Gar23]. In addition, they have other useful properties, such as being able to model different kinds of smooth functions, and being able to incorporate prior knowledge about the function by choosing a suitable covariance function. They are also a non-parametric model, which means that given enough data, they can model any function to an arbitrary precision. This is why we will go into more of their details later on.

Bergstra et al. [Ber+11] propose to model $p(\boldsymbol{\theta} \mid y)$ using a simple (Parzen) density estimation approach. They define

$$p(\boldsymbol{\theta} \mid y) = \begin{cases} \ell(\boldsymbol{\theta}) & \text{if } y \leq y^* \\ g(\boldsymbol{\theta}) & \text{if } y > y^* \end{cases}$$

where y^* is a threshold on the function values observed so far, $\ell(\boldsymbol{\theta})$ is the density estimator for all points whose output is smaller than the threshold, and $g(\boldsymbol{\theta})$ of those larger than the threshold. The ratio $\ell(\boldsymbol{\theta})/g(\boldsymbol{\theta})$ can then be used as an acquisition function. The threshold y^* is commonly set to a percentile (e. g., 15 %) of the observed data [Ber+11; Tia+21]. There are a few drawbacks to this approach. Since it depends on density estimation, it suffers from the curse of dimensionality and numerical instability [Tia+21].

Random forests

Random forests are a general model class proposed by Breiman [Bre01], which are used for both regression and classification. A random forest is an ensemble model consisting of multiple decision trees. In each node of a decision tree, a variable is selected, and then the data at that subtree is split based on this variable. At leaf nodes, the empirical mean of the output values is stored for use in prediction purposes. As is, random forests are not able to express their uncertainty at a point. Hutter et al. [Hut+14] introduce a modification where the empirical variance is stored in the leaf nodes as well. Since a forest consists of multiple decision trees, the means and standard deviations have to be aggregated. This aggregation can be achieved by treating a prediction at a point as a mixture of Gaussians, with each Gaussian corresponding to a particular leaf node of the decision tree used for that point. One advantage of random forests as surrogate models is that they can easily deal with dependencies in the parameter space. For example, let $\theta_1 \in \{1, 2\}$ be the number of layers in a neural network and $\theta_2, \theta_3 \in \mathbb{N}$ be the number of hidden units in the first and the second layer. Then it is clear that θ_3 is only applicable if $\theta_1 = 2$. Even with this modification, random forests are not able to extrapolate the uncertainty well to regions with few data points [Gar23].

The success of neural networks in other domains has sparked interest in using them for Bayesian optimization as well. Especially in optimization problems with a high number of parameters, the ability of neural networks to often learn good feature representations is desirable. A major problem with this is that neural networks do not have a direct mechanism to signal their epistemic uncertainty [Hus+21]. Early work by MacKay [Mac92b; Mac92a] defined *Bayesian neural networks* by imposing Gaussian prior distributions on the parameters of the network. In their PhD thesis, Neal [Nea96] describes a principled way to define priors over network parameters, such that they produce useful priors over functions as the number of hidden units of the neural network goes to infinity. The difficulty lies in computing the posterior of the Bayesian neural network, which is intractable. Approximating it requires costly Monte Carlo methods [Spr+16], variational methods [Blu+15] or ensemble approximations [GG16]. A disadvantage of using neural networks for Bayesian optimization is that neural networks as a model class are typically "data hungry". Often, in Bayesian optimization, the goal is to converge quickly with as few iterations as possible. It is, therefore, rarely the case that there is enough data to use a Bayesian neural network.

Bayesian neural networks

[Nea96]: Neal (1996), *Bayesian Learning for Neural Networks*

In case there is a good amount of data available, neural networks are often able to model high-dimensional data well. Instead of going for a Bayesian neural network, it is possible to use neural networks only as feature extractors for a Gaussian process [Cal+16; Wil+16].

3.2.2.1. Gaussian processes

The model class most often used as a surrogate model for Bayesian optimization are Gaussian processes. It is a probability distribution over functions $f\colon \Theta \to \mathbb{R}$, and therefore naturally able to represent epistemic uncertainty. Formally, we define the Gaussian process as follows:

$$p(f) = \mathcal{GP}(f; \mu, K)$$

where $\mu\colon \Theta \to \mathbb{R}$ is a *mean function* and $K\colon \Theta \times \Theta \to \mathbb{R}$ is a *covariance function*. The latter is also often called *kernel function*. While it is intuitively clear what the mean function is doing, it may be less clear what the covariance function is accomplishing. Roughly speaking, the covariance function is a measure of similarity. Consider, for instance, Figure 3.8, where we can see the behavior of a typical covariance function. The covariance function K is maximal when the points $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are equal. If we increase the distance between them, the covariance quickly decays to 0.

mean function
covariance function
kernel function

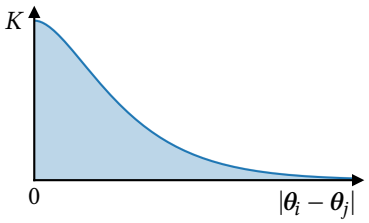


Figure 3.8.: A covariance function determines how "similar" nearby points are.

This is a very simplistic view, however, since the covariance function of a Gaussian process (GP) encodes many important properties of the function space, e. g., smoothness, continuity, differentiability, periodicity and stationarity, to name a few. The choice of the correct covariance function is, therefore, the most important step when using GPs. Nevertheless, when applied as a surrogate model in a Bayesian optimization setup, one can typically use off-the-shelf covariance functions like the Matérn covariance function [Mat60], as a safe default choice.

A simple one-dimensional GP is shown in Figure 3.9. There are 5 observations of our target function f, which we know to be noise-free. As a GP is a

probability distribution over functions, it is common to depict it by plotting functions sampled from it (see the top row of Figure 3.9). Another common method for visualizing GPs is to plot the mean process $\mu_{\mathcal{D}}$ and some quantile of the standard deviation (see the bottom row of Figure 3.9). This is possible, since for each single point $\boldsymbol{\theta} \in \Theta$, the marginal distribution of the GP is simply a univariate Gaussian distribution $\mathcal{N}(\mu(\boldsymbol{\theta}), K(\boldsymbol{\theta}, \boldsymbol{\theta}))$. We depict the mean process μ as the central blue line while we visualize the 99.9 % credible interval of the resulting Gaussian in each point as the shaded blue region. We see that the uncertainty predicted by the GP at the observed points is 0, as we expect. The predicted uncertainty is then increasing smoothly the further we move away from the observations.

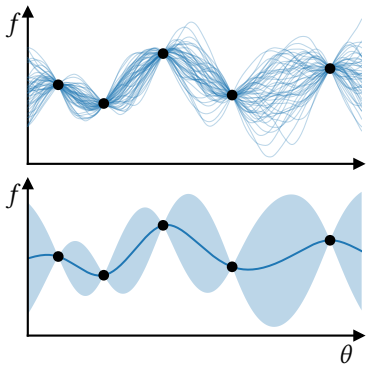


Figure 3.9.: A GP fitted to 5 observations. *Top row*: 50 functions sampled from the GP posterior. *Bottom row*: Mean and 99.9 % process of the GP.

[RW06]: Rasmussen et al. (2006), *Gaussian Processes for Machine Learning*

In order to use GPs in a Bayesian optimization setting, we need to specify a suitable prior GP, i. e., a prior mean function μ and a prior covariance function K. Since the mean function only has an effect on regions far away from data points, and one wants to avoid biasing the model, one typically adopts a constant mean function. The prior covariance function, as mentioned above, is usually set to a Matérn covariance function, unless more prior knowledge is available. Covariance functions can be combined in various ways, which allows a practitioner to accurately represent their assumptions about possible functions. Since this is usually not employed for Bayesian optimization, we will not go into details here and refer the interested reader to the book by Rasmussen and Williams [RW06].

Equipped with the prior GP $p(f)$, we can use it to specify the *posterior* distribution over functions $p(f \mid \mathcal{D})$. Before doing so, it is common to express the prior process on a set of arbitrary points $\boldsymbol{\Theta}$ with corresponding observations $\boldsymbol{\phi}$. The resulting distribution we denote by $p(\boldsymbol{\phi} \mid \boldsymbol{\Theta})$. With that, the posterior distribution is defined by:

$$p(f \mid \mathcal{D}) = \int p(f \mid \boldsymbol{\Theta}, \boldsymbol{\phi})\, p(\boldsymbol{\phi} \mid \mathcal{D})\, d\boldsymbol{\phi} \ .$$

Here $p(\boldsymbol{\phi} \mid \mathcal{D}) \propto p(\boldsymbol{\phi} \mid \boldsymbol{\Theta})\, p(\boldsymbol{y} \mid \boldsymbol{\Theta}, \boldsymbol{\phi})$ is the posterior distribution of the function values. The term $p(\boldsymbol{y} \mid \boldsymbol{\Theta}, \boldsymbol{\phi})$ is a simple Gaussian likelihood. It is also common to denote the posterior GP by $\mathcal{GP}(f; \mu_{\mathcal{D}}, K_{\mathcal{D}})$ with

$$\begin{aligned} \mu_{\mathcal{D}}(\boldsymbol{\theta}) &:= \mu(\boldsymbol{\theta}) + K(\boldsymbol{\theta}, \boldsymbol{\Theta}) K(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} (\boldsymbol{\phi} - \boldsymbol{\mu}), \text{ and} \\ K_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\theta}') &:= K(\boldsymbol{\theta}, \boldsymbol{\theta}') - K(\boldsymbol{\theta}, \boldsymbol{\Theta}) K(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} K(\boldsymbol{\Theta}, \boldsymbol{\theta}') \end{aligned}$$

In practice, one avoids computing the matrix inverse for numerical reasons and instead a Cholesky decomposition is employed. Furthermore, we did not yet mention the hyperparameters of the prior covariance function, which need to be inferred or marginalized out. We will not go into further detail here, but in practice, one typically uses *maximum a posteriori* inference or "exact" inference using Markov chain Monte Carlo (MCMC) methods [Gar23].

With that, we have a complete probabilistic model of the function space, which we can use to make predictions and compute the uncertainty of these predictions. What remains is to define a function that tells us which points are most useful to evaluate to achieve our overall goal of minimizing the simple regret in as few iterations as possible.

3.2.3. Bayesian Optimization using Acquisition Functions

Being able to model the uncertainty of the performance with respect to the hyperparameters is one important component of employing Bayesian optimization. The next question is how new points should be selected in such a way that one converges quickly, and robustly, to a global optimum. We will do so by defining a special function which maps points in the space to real-valued scores. Since the repeated optimization of this function is used to "acquire" new points, it is called an *acquisition function*. Formally, it has the signature $\alpha : \Theta \to \mathbb{R}$. With new information, the acquisition function should change as well, which is why one usually incorporates the current dataset $\mathcal{D}$ as context such that $\alpha(\cdot; \mathcal{D})$ denotes an acquisition function adapted to data $\mathcal{D}$. One could rightfully argue that the acquisition function should also take the optimization budget into account, i.e., the number of iterations we plan to run our hyperparameter optimization for. Doing so, one arrives at Bellman's *principle of optimality* [Bel55], which states that a decision in a certain situation should be taken as if future decisions are taken optimally. In practice, implementing the computation of optimal policies becomes intractable already for small optimization budgets. This is the reason why *one-step lookahead* acquisition functions have become the default in off-the-shelf optimization libraries.

acquisition function

one-step lookahead

In the following, we will go into more detail on how acquisition functions are usually designed, and what popular implementations are. A principled and widely used methodology is to derive acquisition functions based on *decision theory*. Here, one specifies a utility function $u(\mathcal{D})$, which roughly computes the value of the information contained within the given dataset $\mathcal{D}$ to inform some decision. In the case of optimization, this could, for example, be the value of the estimated optimum if one were to terminate immediately. The corresponding acquisition function is then computed as the *one-step marginal gain*:

decision theory

one-step marginal gain

$$\alpha(\boldsymbol{\theta}; \mathcal{D}) = E\big[u\left(\mathcal{D} \cup \{(\boldsymbol{\theta}, y)\}\right) \mid \boldsymbol{\theta}, \mathcal{D}\big] - u(\mathcal{D})$$

Different utility functions result in qualitatively different acquisition functions.

A particularly simple utility function is the point maximizing the expected value of the surrogate model:

$$u(\mathcal{D}) = \max_{\theta \in \Theta_{\mathcal{D}}} \mu_{\mathcal{D}}(\theta)$$

where $\Theta_{\mathcal{D}}$ is the set of points observed so far.[4] In the case of noiseless observations, the one-step marginal gain simplifies to the following expression:

4: Using the complete parameter space Θ here is a completely valid choice and yields the *knowledge gradient* acquisition function.

$$\alpha_{\mathrm{EI}}(\boldsymbol{\theta}; \mathcal{D}) = \int \max\big(y - \max\{y \in Y_{\mathcal{D}}\}, 0\big)\, p(y \mid \boldsymbol{\theta}, \mathcal{D})\, dy$$

It is called the *expected improvement* acquisition function (see Figure 3.10 for a visualization). The potential improvement a point can yield is influenced by the expected performance at that point and also the uncertainty. That way, it achieves a tradeoff between exploitation and exploration.

expected improvement

Another important family of acquisition functions is based on information-theoretic concepts. Our goal in the end is to find the optimal point in parameter space $\boldsymbol{\theta}^*$. The location of this point is uncertain, which is why we can treat

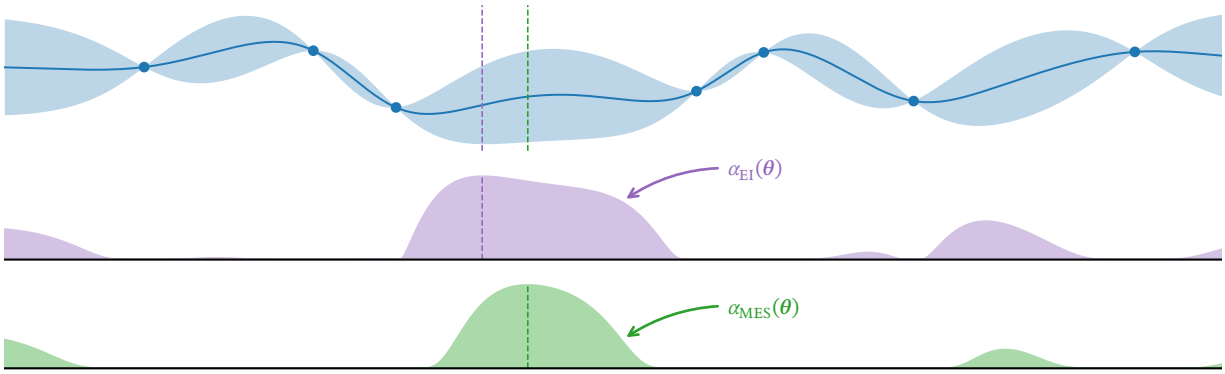


Figure 3.10.: Behavior of different acquisition functions. First row: Gaussian process after 7 observations. Second row: Expected improvement. Last row: Min-value entropy search.

mutual information

θ^* as a random variable whose density concentrates around the true optimum with more and more observations. It is then natural to ask which points should be sampled such that this uncertainty is reduced as quickly as possible. What we are looking for is maximizing the *mutual information* $I(y; \theta^* \mid \theta, \mathcal{D})$ of the value y at our query point θ with the location of the global optimum θ^*. This leads to an acquisition function also called predictive entropy search [HHG14]. The function is difficult to evaluate, which is why, in practice, approximate solutions are used to compute the mutual information.

max-value entropy search

An alternative approach is to instead compute the mutual information $I(y; y^* \mid \theta, \mathcal{D})$ of the value y at our query point θ with the *value* y^* of the global optimum θ^*. This results in the *max-value entropy search* acquisition function α_{MES}. Both expected improvement and max-value entropy search are compared in Figure 3.10. The top row shows 7 observations of an unknown one-dimensional function and a Gaussian process as a surrogate model for the optimization. As is apparent, expected improvement favors points closer to existing observations, since these points are more likely to yield an immediate improvement. Max-value entropy search instead prefers points farther away from existing points where it can reduce the uncertainty in the estimate of the optimum value y^* the most.

Due to its robustness and good performance, we will use max-value entropy search as the acquisition function in our experiments. In the next section, we will give an overview of the main model class that is used to instantiate the models developed in this thesis, namely neural networks.

3.3. Neural Networks and Inductive Biases

[GBC16]: Goodfellow et al. (2016), *Deep Learning*

The pervasive influence of neural networks in the field of machine learning is well-documented [LBH15]. Their adaptability across various domains, coupled with their robust performance and the comprehensive ecosystem of supporting libraries, has established them as the preferred solution for addressing novel learning challenges [Pas+19]. We will make use of various neural networks over the course of this thesis, which is why we will explain the important notions here. The excellent book by Goodfellow et al. [GBC16] will be used as a reference throughout this section.

3.3.1. From Linear Models to Deep Learning

Neural networks as a model class are highly composite models. It is, therefore, useful to first look at the simplest constituent, the linear model. As before, assume that we have a supervised learning problem where $\mathcal{X}$ is the instance space and $\mathcal{Y} \subseteq \mathbb{R}$ the corresponding output space. A linear model $h(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x}$ is parametrized by a weight vector $\boldsymbol{w}$, which is multiplied with an instance $\boldsymbol{x}$ to produce an output. Augmenting the model by including a bias term b is common and results in an affine model of the form $h(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b$. Although linear models are very useful in many applications, the expressive power is very limited.

A common way to allow linear models to learn nonlinear functions is to transform the instances using a nonlinear function $\phi(\boldsymbol{x})$ such that $h(\boldsymbol{x}) = \boldsymbol{w}^\mathsf{T}\phi(\boldsymbol{x}) + b$. Historically, the function ϕ was constructed manually by domain experts. It is natural to ask whether ϕ can instead be learned from data as well. This is the approach taken by neural networks, where *hidden layers* are introduced to act as *feature extractors* for subsequent layers. These hidden layers can be, barring practical limitations, made arbitrarily wide or stacked arbitrarily deep to allow representing highly nonlinear functions [HSW89; Lu+17].

hidden layers

A hidden layer itself consists of one or more *hidden units*, each being an affine function to which an activation function $g : \mathbb{R} \to \mathbb{R}$ is applied. Using linear algebra, we can represent a hidden layer succinctly by

hidden units

$$\boldsymbol{h} = g(W^\mathsf{T}\boldsymbol{x} + \boldsymbol{c}) .$$

That g is nonlinear is important because stacking arbitrary amounts of affine functions is equivalent to a single affine transformation. Note that g itself does not have parameters and is not (usually) adapted to the data at hand.

There are a plethora of activations a practitioner can use, each with their own properties. The most commonly used activation function is the *ReLU* (see Figure 3.11 for a visualization), defined by the simple function $g(x) = \max\{x, 0\}$ [GBB11]. Although it appears that not much is gained by introducing a "kink" into a linear function, it turns out that composing several ReLUs allows us to model nonlinear functions. Consider the function shown in Figure 3.12. It is the result of a small neural network consisting of 5 hidden units, each of which uses the ReLU activation function. As we can see, the resulting function is more complicated than its constituent parts.

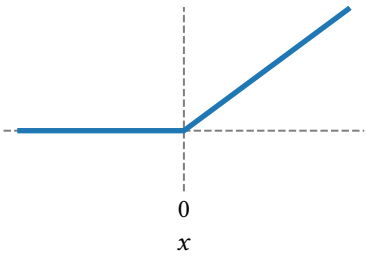


Figure 3.11.: The rectified linear activation function commonly used in neural networks.

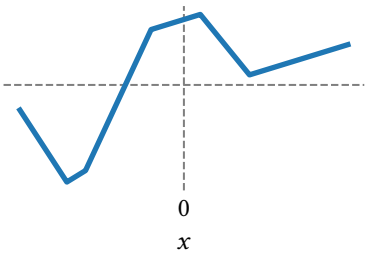


Figure 3.12.: A simple neural network consisting of 5 hidden units using ReLU activation.

Therefore, we can see how neural networks are able to model complex functions by decomposing them into simpler parts. In the next section, we will discuss how neural networks are usually trained, which may seem like a daunting task at first, given their compositional nature.

3.3.2. Training Neural Networks

Now that we have introduced neural networks, we may wonder how such a model can be trained. For linear models, it is possible to derive the empirical loss minimizer in a closed form. This is, in general, not possible for neural networks. Additionally, the loss landscape of neural networks is nonconvex, which makes them hard to optimize. It is for this reason, that *stochastic gradient*

empirical loss minimizer: Page 43

(stochastic) gradient descent

descent (SGD) is used to optimize the parameters of neural networks. Let $\mathcal{J} : W \to \mathbb{R}$ be a function mapping from parameter vectors to loss values, representing our objective to be minimized. The idea of gradient descent is then to update weights iteratively using the update formula

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla \mathcal{J}(\boldsymbol{w}_t) \,,$$

where η is a constant called the *learning rate*. In order to evaluate $\nabla \mathcal{J}(\boldsymbol{w})$, we need to compute the contribution of each weight to the final loss value. Since neural networks are composite models, which can be several layers deep, it is not immediately clear how we can compute the appropriate partial derivative with respect to each weight. Thankfully, it is possible to propagate the loss value using the chain rule of derivatives back through the network in a step-by-step fashion.

We begin by stating the chain rule for the scalar case. Assume that $x \in \mathbb{R}$, $y = g(x)$ and $z = f(g(x))$, then the chain rule is

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \,.$$

As an illustrative example, suppose we have a simple model of the form $\hat{y} = \sigma(wx + b)$ with $\sigma(x) = \log(1 + \exp(x))$ being the activation function, and the squared error being the loss function. Thus, if our true observation is x, then $\mathcal{J}(w) = (\hat{y} - y)^2$ and our goal could be to calculate $\frac{d\mathcal{J}(w)}{dw}$. There are several functions for which we need to use the chain rule. The complete chain will look as follows:

$$\frac{d\mathcal{J}(w)}{dw} = \frac{d\mathcal{J}(w)}{d\hat{y}}\frac{d\hat{y}}{dz}\frac{dz}{dw} \tag{3.5}$$

with $z = wx + b$ here. We can then determine all derivatives separately as follows:

$$\frac{d\mathcal{J}(w)}{d\hat{y}} = 2(\hat{y} - y) \qquad \frac{d\hat{y}}{dz} = \frac{1}{1 + \exp(-z)} \qquad \frac{dz}{dw} = x \,.$$

Inserting into (3.5), we get

$$\begin{aligned} \frac{d\mathcal{J}(w)}{dw} &= \frac{2x(\hat{y} - y)}{1 + \exp(-z)} \\ &= \frac{2x(\sigma(wx + b) - y)}{1 + \exp(-wx - b)} \,. \end{aligned} \tag{3.6}$$

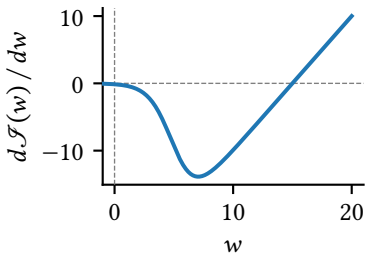


Figure 3.13.: Behavior of the partial derivative of the example for $x = 1$, $y = 10$ and $b = -5$.

In Figure 3.13 we plot (3.6) for a given pair $(x, y) = (1, 10)$ and b fixed to -5. We can see that the derivative is negative for $w < 15$ and positive for $w > 15$. A gradient descent algorithm would, therefore, increase the weight w in the former case, while decreasing it in the latter. Note how the magnitude of the derivative decreases towards $w = 0$ and close to the optimum $w = 15$. The initial value of w dictates the speed of convergence towards the optimum, which is one reason why initialization is an important topic in deep learning [NBS22].

We have seen that we can calculate partial derivatives for each weight in a model using the chain rule. Doing this computation directly can lead to the repeated computation of many terms, which is very inefficient. Approaches that solve this computation problem more efficiently are subsumed under

the term *automatic differentiation* [Bay+18]. The widely used *backpropagation* algorithm in deep learning is a special case of what is called *reverse accumulation* [Gri12]. For most neural network architectures and loss functions encountered in practice, backpropagation incurs a computation cost of $\mathcal{O}(n)$, where n is the number of nodes in the computation graph of the neural network.

automatic differentiation
backpropagation

So far, we have talked about gradient descent and how to compute the gradient in the first place. In practice, the datasets usually are too large that the complete gradient can be computed efficiently. SGD solves this issue by randomly sampling subsets of instances called *minibatches* in each iteration and computing an estimate of the gradient by averaging the gradients on these subsets. Formally we sample a batch $\{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \dots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\} \subseteq \mathcal{D}$ and then compute the gradient estimate

$$\hat{g} = \frac{1}{m} \nabla_{\boldsymbol{w}} \sum_{i=1}^{m} L\left(h_{\boldsymbol{w}}(\boldsymbol{x}^{(i)}), \boldsymbol{y}^{(i)}\right) .$$

With that estimate, we can perform the parameter update as before: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \hat{g}$. An important improvement of SGD, which accelerates the convergence speed, was the introduction of *momentum* [RHW86], where past gradients are incorporated as a velocity vector $\boldsymbol{m}$ into the gradient computation. We introduce a new parameter $\alpha \in [0, 1)$ and update this velocity vector as follows

momentum

$$\boldsymbol{m}_{t+1} = \alpha \boldsymbol{m}_t - \eta \hat{g}$$

and change the update formula of SGD to $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{m}_{t+1}$. A variant is *Nesterov momentum* [Nes83], where the velocity vector is first applied to the weight vector $\boldsymbol{w}$, then the gradient $\hat{g}$ is computed using the updated weight vector and only then the velocity vector and the weight vectors are fully updated. This change improves the convergence speed and stability of the optimization process [Nes83].

A commonly used optimizer is called adaptive moment estimation (Adam), which combines the ideas of several previously proposed approaches [KB15]. Similar to momentum, it tracks a moving average of the gradients $\boldsymbol{m}_t$ (first-moment estimate). In addition, it also tracks the squares of the gradient using a moving average $\boldsymbol{v}_t$ (second raw moment estimate). Since these estimates are initialized to zero, they are biased during the initial iterations, which is why the authors employ bias correction to arrive at $\hat{\boldsymbol{m}}_t$ and $\hat{\boldsymbol{v}}_t$. The full update formulas are defined as follows:

Adam optimizer

$$\begin{aligned}
\boldsymbol{m}_{t+1} &= \beta_1 \boldsymbol{m}_t + (1 - \beta_1)\hat{g} & \boldsymbol{v}_{t+1} &= \beta_2 \boldsymbol{v}_t + (1 - \beta_2)\hat{g}^2 \\
\hat{\boldsymbol{m}}_{t+1} &= \frac{\boldsymbol{m}_{t+1}}{1 - \beta_1^{t+1}} & \hat{\boldsymbol{v}}_{t+1} &= \frac{\boldsymbol{v}_{t+1}}{1 - \beta_2^{t+1}} \\
\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}_{t+1}} + \epsilon} \hat{\boldsymbol{m}}_{t+1} .
\end{aligned}$$

Dividing the learning rate η by the square root of $\hat{\boldsymbol{v}}_{t+1}$ effectively implements different learning rates for each parameter in $\boldsymbol{w}$. A small constant ϵ is added to increase numerical stability.

Table 3.1.: Common learning tasks and their task-dependent losses.

Learning task	Typical domain	Example activation	Example loss
Regression	$\mathbb{R}$	$s = r$	MSE: $(s-y)^2$
Binary classification	$[0,1]$	$s = \frac{1}{1+\exp(r)}$	binary cross-entropy: $y \log s + (1-y)\log(1-s)$
Multi-label classification	$[0,1]^k$	$s_i = \frac{1}{1+\exp(r_i)}$	binary cross-entropy: $y_i \log s_i + (1-y_i)\log(1-s_i)$
Multi-class classification	Δ^{k-1}	$s_i = \frac{\exp(r_i)}{\sum_{j=1}^{k}\exp(r_j)}$	categorical cross-entropy: $-\log s_y$
Ranking	$\mathcal{S}_k$	$s_i = \exp(r_i)$	Plackett-Luce loss: $\sum_{i=1}^{k-1} \log\left(\sum_{j=i}^{k} \exp(s_{y^{-1}(j)})\right) - s_{y^{-1}(i)}$

3.3.3. Task-Dependent Losses

Neural networks are a flexible model class because they can be applied to a variety of settings with minimal changes to the overall architecture. We will make use of this property in this thesis as well, where we apply neural networks in the realm of preference learning. The main methodology is to let the last layer of the neural network output one or more real-valued scores $r \in \mathbb{R}$ for a given instance. An activation function then transforms these scores into a range suitable for the task, e. g., in the case of binary classification, we could output a value s in the range $[0,1]$. Finally, a loss function L is defined, which relates the transformed scores to the observed targets in the data.

In Table 3.1, common learning tasks and their corresponding domains are listed. In the case of ranking, y represents a permutation with y^{-1} being its inverse, i. e., $y^{-1}(i)$ maps the index i of an object to the corresponding rank. As is apparent, we can solve many different settings just by slightly varying the activation function of the last layer and the loss function. What this simplified view glosses over is that the instances $\boldsymbol{x} \in \mathcal{X}$ can also change depending on the setting.

As an example, consider the task of learning from rankings. If we are given an instance vector $\boldsymbol{x}$ for which we have to predict a ranking of a set of labels, then the setting is called *label ranking* [VG10]. If, instead the instance consists of several objects, each represented by a feature vector, and the task is to rank the objects, then this problem is called *object ranking* [KKA10]. While this necessitates a change of the neural network architecture, the same task-dependent losses on the outputs can be used. The only restriction is that in case of labels, one assumes that the set of labels remains fixed, which allows one to give each label a different weight in the loss function. This is useful in settings where there is an imbalance of the different labels. If the instances instead are a set of objects, potentially different for each instance, this weighting is not possible.

Since this methodology is so flexible, it will underpin some of the approaches developed in this thesis. Note that making a neural network compatible with a task is often only the first step. Considerable time and resources are invested into the development of new neural network architectures, which encode suitable inductive biases [TC21; MFL20; Ker+22]. Before we go into more detail with respect to common neural network architectures, we will take a detour into what general regularization methods exist to limit the complexity of neural networks.

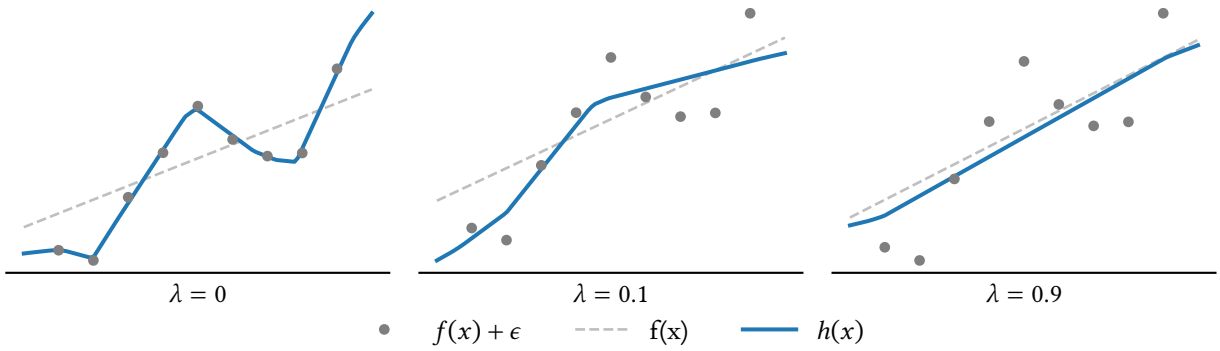


Figure 3.14.: Effect of L_2 parameter norm regularization on the fit of an overparametrized (3 hidden layers with 50 units each) neural network.

3.3.4. Regularization of Neural Networks

Neural networks are a very flexible model class, and the number of parameters quickly grows with the number of hidden layers and units. As we have seen in Section 3.1, the complexity of the model should be appropriate for the amount and information content of the data at hand. There exist several regularization methods—some general, some only applicable to neural networks— which can be utilized.

The most commonly used method is to add a penalty term for the complexity of the model to the loss function. Recall that $\mathcal{R}_{\text{emp}}(h)$ denoted the empirical risk of a model $h \in \mathcal{H}$. We can then define the *augmented risk* as follows:

empirical risk: Equation 3.1, Page 43

augmented risk

$$\mathcal{R}_{\text{aug}}(h) = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, h(\boldsymbol{x}_i)\big) + \lambda\Omega(h)\,.$$

The function Ω should be large when the complexity of the model h is high and low in the opposite case. Minimizing the augmented loss will, therefore, favor models with both a low empirical loss and a low complexity. The parameter $\lambda \in \mathbb{R}^+$ controls the trade-off between these two and is usually optimized on a validation set.

validation: Section 3.1.2, Page 45

In the case of neural networks, penalties on the norm of the weights $\boldsymbol{w}$ are the main method to penalize complexity. If the Euclidean norm is used, the method is called L_2 *regularization*, with $\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_2^2$ being the penalty term used to augment the loss function. In Figure 3.14, we can see the effect L_2 regularization has on the model complexity of a neural network. The network used to generate the plots consists of 3 hidden layers with 50 units each. With 5251 parameters in total, the network is massively overparametrized for the given task of learning a regression function for a dataset of 9 points. In case of $\lambda = 0$ (no regularization), the model happily interpolates all given points and the out-of-sample performance would, therefore, be bad. Increasing λ to 0.1, the model no longer interpolates the points, and the model fit has improved slightly, but it is still influenced too much by the noise of the points. Finally, with $\lambda = 0.9$, the resulting model is almost linear and almost matches the ground-truth model, shown as a dashed gray line. We can see that L_2 regularization is a powerful method to restrict the complexity of neural networks, even if the number of parameters far exceeds the amount of data we have.

L_2 regularization

The Euclidean norm is not the only norm that can be used to penalize the

magnitudes of the weights. If a sparse weight vector $\boldsymbol{w}$ is sought, L_1 regularization is commonly applied. Sparsity can be useful if one wants to compress the model or for the purposes of feature selection. Due to the shape of the L_2 ball unimportant weights will receive weights close to zero, but usually not exactly 0.[5] With L_1 regularization, the shape of the L_1 ball admits solutions to the augmented loss with zero weights.

5: This can only happen in situations, where the optimal weight vector (without penalty) contains zero entries.

dropout

Another important regularization method, specific to neural networks, is called *dropout*. During training, dropout will, with probability p, mask the output of each hidden unit. This effectively samples a subnetwork of the neural network, which is then used to make a prediction. Implicitly, we are defining an ensemble of an exponential number of neural networks, but it is not necessary to exhaustively calculate all of them, and the performance of the ensemble can be approximated well by a few random samples. During inference time, it is possible to forego masking and use the complete network instead. This only requires scaling the outgoing weights of each hidden unit by the probability p, and hence is known as the *weight scaling inference rule* [Hin+12]. Srivastava et al. [Sri+14] compare dropout to L_2 regularization and a few additional regularizers on the MNIST dataset and find that it leads to the smallest test classification error.

dataset augmentation

In certain domains *dataset augmentation* can be a useful regularization method. The idea is that small transformations to the instances will not change their label but help to fill the instance space. Thus, one inflates the original dataset by adding new instances where such transformations have been applied. A typical example of this is image classification, where applying rotations, flipping, blurring, zooming, etc., should not change the overall image. One should only select transformations that do not affect the labels. For example, if the task is to classify bicycles by their color, then a transformation that converts an image to grayscale or rotates the hue angle is not suitable. For the now well-known AlexNet architecture, the authors used data augmentation techniques to improve the classification error of their model and report a reduction of 1 % [KSH12]. Taylor and Nitschke [TN18] compare a small set of common transformations and find that cropping yields the highest increase in test accuracy of 13.82 %. Nowadays, off-the-shelf libraries implement a wide variety of different augmentation methods [Bus+20], making it easy for researchers and practitioners to include data augmentation in their pipeline.

The most impactful way to regularize neural networks is to encode inductive biases directly into the network architecture itself. This is why we will now explore key examples of neural network architectures designed for different types of data.

3.3.5. Network Architectures & Inductive Biases

Classical, fully-connected feed-forward networks, as the name suggests, are networks organized into several layers, each consisting of a certain number of hidden units. The "fully-connected" refers to the property that the outputs of all hidden units in layer $i-1$ are connected to all units of layer i. Among the units of one layer, there are no connections. In these kinds of networks, the only hyperparameters are the number of layers to use and the number of hidden units in each layer. Early work on the representative power of such networks resulted

in the *universal approximation theorem* [HSW89; Cyb89], which states that if you have at least one hidden layer with enough hidden units and a nonlinear activation function, then any one-dimensional Borel-measurable function can be approximated. The impact of these types of results is limited since they will not inform us, whether it is possible to reach such solutions by standard training methods and how many hidden units are necessary. Subsequent work has shown that in the worst case, an exponential number of hidden units is required to achieve universal approximation [Bar93].

universal approximation theorem

While this gives fully-connected layers flexibility in allocating the weights of each connection in order to fit the data well, it is wasteful to have connections quadratic in the number of units. Specialized architectures for common tasks have been developed, which are more economical with the number of weights. In the following, we will go over the most common families of architectures and take a look at their properties.

3.3.5.1. Convolutional Neural Networks

A common way to encode domain-specific properties into the neural network architecture is to restrict the connections between hidden units to a suitable subset. When dealing with image data, it is clear that whether something is a cat should not depend on the *absolute* position in the image. Rather, the *local* pattern is important for the recognition/classification. Convolutional neural networks (CNNs), proposed by Lecun [Lec89], enforce this by the use of local pattern detectors called *kernels*. The kernel is applied to the image by an operation called *convolution*, the output of which is a *feature map* (see Figure 3.15 for an illustration). The formula for the two-dimensional convolution operation is defined as

$$\boldsymbol{h}_{ij} = \sum_{m=1}^{M} \sum_{n=1}^{N} \boldsymbol{x}_{i+m,j+n} \cdot K_{m,n} \,,$$

where $\boldsymbol{x}$ is the input image, K is the kernel of size $M \times N$, and $\boldsymbol{h}$ is the resulting feature map [GBC16]. Padding with zeros is often used to ensure that the output has the same size as the input.

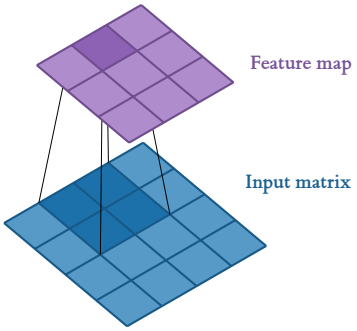


Figure 3.15.: The convolution operation of a neural network. A 2×2 kernel is moved across a 2D input to produce a feature map.

[Lec89]: Lecun (1989), "Generalization and Network Design Strategies"

The convolution operation has the previously mentioned property that moving a pattern to a different position on the image will move the output of the kernel to a different position on the feature map without changing the output itself. This property is called *translation-equivariance* [GBC16]. It is an important property that allows the network to learn local patterns, which are invariant to translations of the input, as is the case in image recognition tasks.

Note that the kernel weights are shared across the whole image, which considerably reduces the number of parameters. This is in contrast to fully-connected layers, where each hidden unit has its own set of weights. The number of parameters in a convolutional layer is, therefore, proportional to the kernel size and the number of kernels used. Similar to a fully-connected neural network, we can stack multiple convolutional layers on top of each other to increase the representational power of the network, using nonlinear activation functions after each convolution operation.

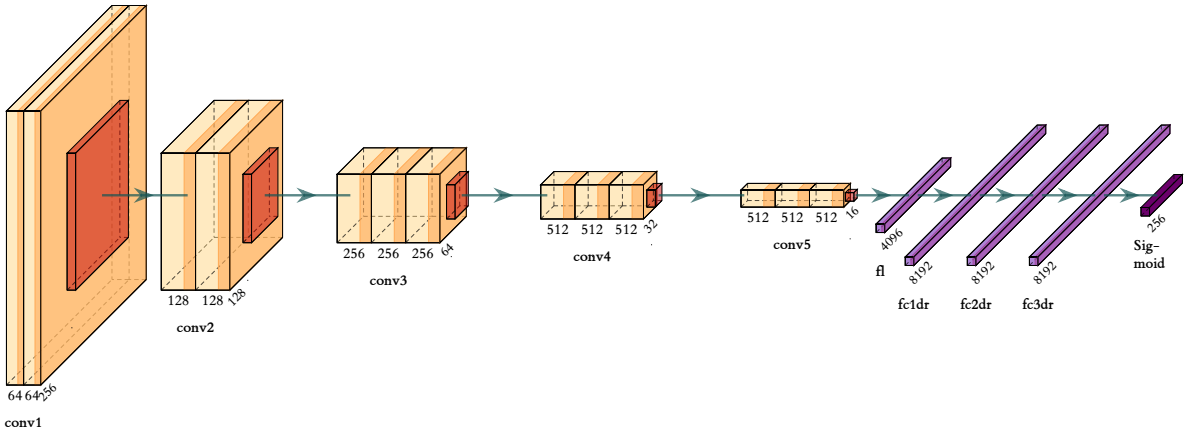


Figure 3.17.: The VGG-16 CNN architecture [SZ15]. Orange boxes depict convolutional layers, red boxes are max-pooling layers, and purple boxes are fully-connected layers.

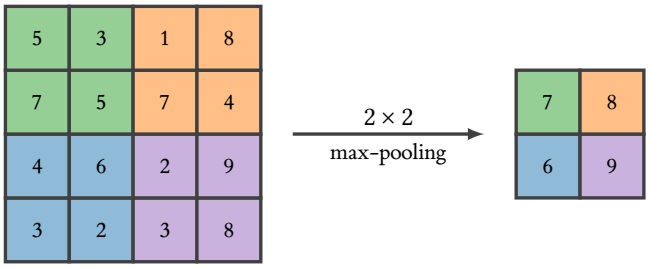


Figure 3.16.: The pooling operation aggregates the outputs of neighboring units (here 2×2 max-pooling). Here, a stride of 2 is used, which reduces the size of the output by half.

In addition, it is useful to insert *pooling* layers between convolutional layers. With pooling, the outputs of the kernels in each region are further aggregated using, e. g., the max-operation [ZC88] (see Figure 3.16 for an illustration). Formally, the $M \times N$ max-pooling operation is defined as

$$\text{maxpool}(\boldsymbol{x})_{i,j} = \max_{m \in [M], n \in [N]} \boldsymbol{x}_{i \cdot s_i + m - 1, j \cdot s_j + n - 1} \,,$$

where s_i and s_j are step sizes in the i and j direction, respectively. These step sizes are usually called *strides* and are used to reduce the size of the output. As an example, if we use 2×2 max-pooling with a stride of 2, the size of the output is halved, as is the case in Figure 3.16. This mechanism also allows one to support input data of varying sizes by dynamically computing the required striding to arrive at a fixed representation size.

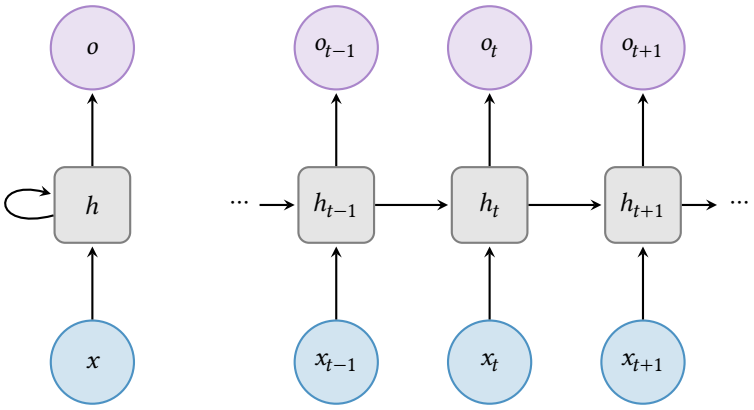
Note that the convolution operation is equivariant, but not *invariant* to translations. The latter is a stronger property and requires that the output of the function f does not change at all when g is applied to the input (see Definition 3.3.1 for the formal definition). With the pooling operation, the network can learn to be invariant to small translations of the input. The bigger the pooling region, the more invariant the network will be to translations.

For a more concrete example, that shows how convolutional layers and pooling layers can be combined to form a useable neural network architecture, consider the example architecture shown in Figure 3.17. It shows the CNN called VGG-16 proposed by Simonyan and Zisserman [SZ15] for the task of image classification. It achieved state-of-the-art results in 2015 and is a prototypical example of how modern convolutional networks are structured. It employs

5 convolutional blocks, 3 fully-connected layers and one sigmoid layer to produce the final classification scores. Each convolutional block consists of 2 to 3 convolutional layers using 3×3 kernels. A 2×2 max-pooling layer with a stride of 2 is used to halve the size of the output. The VGG architectures were an innovation at their time since they used small 3×3 kernels, while other architectures were using larger kernel sizes. This reduced the number of parameters and thus enabled the authors to go for a deeper overall network.

More recently, new techniques like skip connections [He+16] have been employed to further improve the performance of CNNs. Since these are out of the scope of this thesis, we will not go into further detail here. The main takeaway here is that designing the correct neural network architecture requires one to consider which symmetries exist in the problem domain and how these can be encoded. CNNs achieve translation-equivariance and some -invariance through the use of convolution and pooling layers.

3.3.5.2. Recurrent Neural Networks



(a) Recursive (b) Unrolled

Figure 3.18.: A recurrent neural network receives a representation of the previous state as additional input. Adapted from Figure 10.2 in [GBC16].

The architectures considered for now are *feedforward* neural networks, i.e., the output of a layer never is fed into an earlier layer. With recurrent neural networks (RNNs), this restriction is lifted, and circles in the architecture design are allowed. These networks are typically applied to sequential data, e.g., natural language, since the length of each input is not predetermined and the precise order of tokens is less important than the tokens themselves in the surrounding context.

A typical RNN acting on a sequence is shown in Figure 3.18a. It reads an input x, processes it to produce hidden state h and the hidden state is used to produce output o. Note how the hidden state is used as an additional input during the computation of the hidden state. This is the key feature of RNNs and allows them to model sequences of arbitrary length. It is easier to understand the computation of an RNN when it is *unrolled*, e.g., as shown in Figure 3.18b. Here, the network at time step t is applied to input x_t, receives the hidden state h_{t-1}, and produces the output o_t. The network itself is the same at each time step since the weights are shared across all time steps. This idea that the same neural network is applied several times during one forward pass is therefore referred to as *weight sharing*, and we will see it again later when we will develop

weight sharing

neural network architectures operating on sets of objects.

In staying with the theme of inductive biases, RNNs are designed to encode the assumption that the input is sequential, and more importance is given to inputs that are closer in time. As for the representational power, it can be shown that RNNs are able to approximate arbitrary Turing machines [CS21]. In practice, however, RNNs have problems learning long-term dependencies between input tokens. One reason is that the gradients propagated back through the unrolled computation graph can explode or vanish the longer the sequence [PMB13]. Another problem is that the influence of long-term dependencies shrinks exponentially, and thus are dwarfed by short-term interactions. Several methods have been proposed to solve these issues, the most successful of which were *gated* RNN architectures like the long short-term memory (LSTM) architecture proposed by Hochreiter and Schmidhuber [HS97]. It uses two computation paths, one for long-term memories and one for short-term memories. The network learns which information to keep and which to discard by use of *gates* that control the flow of information. They avoid the vanishing/exploding gradient problem by letting only linear operations update the states, which prevents the multiplicative compounding of the weights.

3.3.5.3. Encoding Sets using Neural Networks

Later in the thesis, one of the main challenges will be to deal with sets of objects as input. In a set, the order of the constituents does not matter, e. g., if we have an arbitrary set $\{a, b, c\}$, then any permutation π will result in the same set: $\pi(\{a, b, c\}) = \{a, b, c\}$. When working with neural networks, the input has to be provided as a feature vector, which is sensitive to permutations of the input. As an example, consider the task of learning a real-valued set-function $f\colon 2^{\mathbb{R}^d} \to \mathbb{R}$. A simple input encoding would be to concatenate all d-dimensional vectors into one big vector such that the input space of the neural network becomes $\mathbb{R}^{nd}$. In a typical fully-connected neural network, each vector consisting of d features would be connected to different weights. Thus, without further restrictions on the weights, changing the order of the inputs will result in a different output. One could train the network using randomly shuffled sets of objects to try to get the network to learn that the order is not important. The learned function will, however, not be perfect, and the network will need a factorial amount of data augmentations to learn it. We will now look at restricted network architectures, which can encode this property directly.

Definition 3.3.1 *A function $f\colon \mathcal{X} \to \mathcal{Y}$ is called invariant with respect to operation $g\colon \mathcal{X} \to \mathcal{X}$, if*

$$f(g(x)) = f(x) \quad \forall x \in \mathcal{X}\,.$$

Charles et al. [Cha+17] propose the PointNet neural network architecture for the problem of point cloud classification and segmentation. They pass each point through a deep neural network and utilize a max-pooling layer to achieve *permutation-invariance* (see Definition 3.3.1). Zaheer et al. [Zah+17] tackle this problem by showing that any function $f\colon 2^{\mathcal{X}} \to \mathcal{Y}$, receiving a set as the input, is permutation-invariant, if and only if it can be decomposed into the following form:

[Zah+17]: Zaheer et al. (2017), "Deep Sets"

$$f(X) = \rho\left(\sum_{x \in X} \phi(x)\right), \tag{3.7}$$

where ρ and ϕ appropriately constructed functions. Using the argument that neural networks are universal approximators, the authors then propose to

replace ρ and ϕ by neural networks. The key insight here is that the network corresponding to ϕ is operating on each input $x \in X$ independently, implicitly being equivalent to a network defined on a vector of length nd, but using the exact same weights for each vector of length d. The permutation-invariance is ensured by summing the representations. Since the sum is a commutative operation, the order of the inputs is not important.

Definition 3.3.2 *A function $f\colon \mathcal{X} \to \mathcal{X}$ is called equivariant with respect to operation $g\colon \mathcal{X} \to \mathcal{X}$, if*

$$f(g(x)) = g(f(x)) \quad \forall x \in \mathcal{X}\,.$$

The authors also propose a weight-sharing scheme useful for learning *permutation-equivariant* functions (see Definition 3.3.2):

$$f(\boldsymbol{x}) = \sigma\left(w_1 \boldsymbol{I}\boldsymbol{x} + w_2 \left(\mathbf{1}\mathbf{1}^\top\right)\boldsymbol{x}\right),$$

where σ is a nonlinearity, $w_1, w_2 \in \mathbb{R}$ are the parameters, $\mathbf{1} \in \mathbb{R}^d$ is a column vector of 1s and $\boldsymbol{I} \in \mathbb{R}^{d\times d}$ is the identity matrix. The term $\boldsymbol{I}\boldsymbol{x}$ simply reproduces the input and multiplies it by the weight λ, while $\left(\mathbf{1}\mathbf{1}^\top\right)\boldsymbol{x}$ sums the inputs and multiplies the sums by γ. The latter sum can be replaced by any other commutative operation on the inputs, which is why the authors also propose a variation that uses max-pooling instead. The scheme can also be used when the inputs $\boldsymbol{x}$ are matrices (i. e., each object in the set is represented by a feature vector).

symmetric group S_n

Note that the permutation operation on n elements defines the *symmetric group* S_n. We can generalize the concepts of invariance and equivariance by allowing arbitrary group operations g (see Definition 3.3.1 and Definition 3.3.2). Ravanbakhsh et al. [RSP17] introduce two weight-sharing schemes (one dense and one sparse), which guarantee the equivariance properties as defined by a given group operation, while still being sensitive to other (desirable) group operations. Zhou et al. [ZKF21] go one step further and apply meta-learning to learn the appropriate equivariance scheme for a collection of learning tasks.

[RSP17]: Ravanbakhsh et al. (2017), "Equivariance Through Parameter-Sharing"

3.3.5.4. Graph Neural Networks

We can see that encoding sets using neural networks by enforcing certain symmetries is one way to view the problem. We get a complementary perspective when looking at *graph neural networks (GNNs)*. These are neural networks that operate on graph data to solve a variety of tasks. Before we go into more detail, it is interesting to note that many learning tasks can be represented as graph learning problems. The task of learning from two-dimensional images can be converted to a graph learning problem by letting each pixel be a node and connecting each to all of its neighboring pixels via an edge. More relevant to this thesis, a set can be represented by a complete graph of all of the objects in the set.

The literature on learning from graph data is very heterogeneous with respect to the tasks considered. One may wish to predict labels for the *nodes* of a given graph, e. g., given a graph of user profiles of a social network, try to predict the political affiliation for each user. As the reader may have guessed already, it is possible to target *edge* attributes as well. One important application is *link prediction*, where the goal is to predict how likely it is that two users, or more general entities, are connected [ZC18]. Finally, we may want to predict the properties of the graph as a whole, with the goal of determining if a particular molecule can achieve a desired effect [And03; Yan+22]. For instance, in drug

development, we might predict if a new compound can effectively inhibit a specific protein.

There are several types of GNNs considered in the literature. Wu et al. [Wu+21] distinguish between four main categories of GNNs:

[Wu+21]: Wu et al. (2021), "A Comprehensive Survey on Graph Neural Networks"

- Recurrent GNNs: Messages are exchanged between neighboring nodes until the node features stabilize, and an equilibrium is reached.
- Convolutional GNNs: Node representations are formed by pooling the features of each node and its neighbors, analogous to convolutional operations (see Section 3.3.5.1).
- Graph autoencoders (unsupervised): An encoder embeds the graph into a latent space, while a decoder is trained to reconstruct the original graph from the latent representation. This is useful to learn a lower-dimensional representation of the graph or to solve the task of graph generation.
- Spatial-temporal GNNs: These are used to model dynamic graphs, where the structure of the graph changes over time. For example, consider the traffic flow in a city, where the nodes are intersections, and the edges are roads. Spatial-temporal GNNs can incorporate both spatial information (e. g., the layout of the roads, intersections, etc.) and temporal information (e. g., traffic flow over time, real-time incidents, etc.).

A general formal framework for GNNs was introduced by Xu et al. [Xu+19]. As is common in computer science, a graph is denoted by $G = (V, E)$ with V being the set of nodes and E the set of edges. We assume that each node $v \in V$ is represented by a feature vector $X_v \in \mathbb{R}^d$. The majority of GNNs compute a node representation h_v by repeatedly taking the (current) representation of itself and its neighbors and aggregating these in a suitable fashion. The more iterations are performed, the more information can spread across the graph. Let $h_v^{(k)}$ be the representation of node v after k iterations. This representation is computed as follows:

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(h_u^{(k-1)} \mid u \in \mathcal{N}(v)\right) \tag{3.8}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right) \tag{3.9}$$

Here AGGREGATE is a learnable function aggregating the node representations of all neighbors. COMBINE receives the previous node representation $h_v^{(k-1)}$ and the result of the aggregation $a_v^{(k)}$ to produce the node representation of iteration k. Depending on the architecture of these two functions, we obtain different models known in the literature. A prototypical example is the GraphSAGE architecture proposed by Hamilton et al. [HYL17], where the aggregation is done using max-pooling:

$$a_v^{(k)} = \text{MAX}\left\{\text{ReLU}\left(W \cdot h_u^{(k-1)}\right) \;\middle|\; \forall u \in \mathcal{N}(v)\right\}.$$

Here, W is a matrix of learnable weights. The astute reader may notice that this closely resembles the architectures employed for encoding sets, with the difference being that the sets considered here are the node neighborhoods in the graph.

An important question concerns the representational power of GNNs. Since the data we consider are graphs, we are interested in the effect of the "structure"

of the graph on the prediction target. In graph theory, questions regarding structure ultimately boil down to the question of *graph isomorphism*, i. e., a graph G is isomorphic to a graph H, if there exists a bijection of the nodes such that all edges are preserved (see the margin note for the exact definition).

graph isomorphism

Xu et al. [Xu+19] show that any GNN is at most as powerful as the Weisfeiler-Leman (WL) graph isomorphism test. We will not go into its details here, but roughly speaking, it works by repeatedly collecting node labels in a neighborhood around a node and then mapping them to a new label with a suitable hash function. In order for GNNs to reach the same discriminative power, Xu et al. show that it is necessary to use injective functions for the operations AGGREGATE and COMBINE. They propose the graph isomorphism network (GIN), which does satisfy all necessary conditions. In addition, they investigate popular architectures like GraphSAGE and GCN, and show that these fail the injectivity criterion and are therefore not maximally expressive. Despite this, Xu et al. [Xu+19] show that models with mean aggregations still work well on node classification tasks because although they are not able to distinguish exact graph structures, they are able to capture the distribution of elements in a neighborhood.

Definition 3.3.3 *A graph G is isomorphic to a graph H, if there exists a bijection $\varphi : V(G) \to V(H)$ such that*

$$(u, v) \in E(G) \Leftrightarrow (\varphi(u), \varphi(v)) \in E(H)$$

3.3.5.5. Common Themes in Neural Architectures

In this section, we have seen how we can represent complex functions by composing several simple, nonlinear functions into a (neural) network. By restricting the connections, we are able to encode different inductive biases. In particular, we have seen that invariance and/or equivariance with respect to certain operations is a common thread underlying the design of many architectures. CNNs employ convolutions and pooling to be equivariant with respect to translations. Neural networks for encoding sets compute representations for the constituents of a set and aggregate them using pooling. GNNs extend this idea to graphs, where the neighborhoods around each node are treated as sets. By repeatedly computing new node representations and pooling, signals can propagate across the graph to solve complex tasks. These concepts will reappear in the design of models suitable for learning choice and ranking functions over the course of this thesis.

This section aimed to provide the reader with a high-level overview of common neural network architectures. In practice, a given neural network architecture has additional hyperparameters that control the network's representational power and must be set for good performance. These hyperparameters include, for example, the number of layers and hidden units in each layer. More units and layers generally increase the representational power but also make it more prone to overfitting. For each problem, a good trade-off of width (number of units) and depth (number of layers) must be found. While there are some studies that investigate optimal scaling [TL19], these parameters are commonly set using HPO techniques (see Section 3.2).

In this chapter, we covered the basics of machine learning, including how to ensure that a model generalizes well to unseen data, how to optimize a model's hyperparameters and typical neural network architectures. As this thesis concerns the subfield of preference learning, we will now turn our attention to the problem of learning from preference data in the next chapter.

4. Preference Learning

In the preceding chapters, we introduced classical notions of utility theory and choice modeling (see Chapter 2) and provided an overview of machine learning concepts (Chapter 3). At the intersection of these domains lies *Preference learning*, a sub-field of machine learning that bridges these two realms. Where we have seen a focus on axiomatic approaches in the classical choice theory, in preference learning (and in machine learning in general), we are much more interested in inferring models from data and the subsequent predictive performance of our models on unseen data.

Why is the study of preferences important in machine learning? There are several compelling reasons why preference feedback is often preferred when dealing with human feedback. On the one hand, it is much easier for humans to order alternatives relatively than to assign an absolute score [Phe+15]. On the other hand, in applications, we often only receive indirect feedback. Consider a user who clicks on one of the entries a search engine returns. This can easily be modeled by a preference for this entry over all of the others, but we do not explicitly ask the user to assign scores to each entry.

Another example that motivates preference learning is settings where information may be censored or partially observed. An illustrative setting involves algorithms solving tasks where the runtime is the main target to be predicted. Algorithms may run a very long time on certain input tasks, which is why a timeout may be imposed. While we are not able to assign a precise runtime to instances where a timeout happened, we can still record a preference by recognizing that instances without timeouts are preferred over those where a timeout occurred. Thus, here the preference is used as a way to "coarsen" information, instead of recording human preferences.

Research into preference learning is driven by applications such as recommender systems and information retrieval (IR), just to name a few. Certain settings within this field have attracted more attention and study than others, reflecting their importance and prevalence. However, this focus can sometimes lead to confusion in terminology. Take, for instance, the setting of *learning to rank*, which specifically refers to applying preference learning in the context of IR. Given its widespread application in today's web, the literature on this subject is vast, making it a focal point in the field. It might be tempting to equate the entirety of preference learning with this single application. However, in this chapter, we will see that preference learning extends far beyond learning to rank, encompassing various problems that demand distinct solutions.

Fürnkranz and Hüllermeier [FH10] provide a comprehensive overview of the then-emerging field of preference learning. They differentiate between various settings based on the specific level at which preferences are expressed. For example, when we observe an instance paired with a corresponding ranking of a set of global labels, it fits into the *label ranking* setting. Conversely, if we observe a collection of objects—each characterized by a feature vector—along with a ranking of these objects, we refer to this as the *object ranking* setting. Thus, it becomes apparent that by modifying the inputs and targets, we can

[FH10]: Fürnkranz et al. (2010), *Preference Learning*

define a diverse array of settings within preference learning, each necessitating a unique methodology.

As most of the thesis is about choice modeling of objects, we argue that this setting should be positioned under the umbrella of preference learning as well. This is because a choice represents a preference for one (or more) option(s) over others. Indeed, a large body of classical research looked at what is called "revealed preferences", i. e., preferences that we infer by looking at choices. This is why we, contrary to Fürnkranz and Hüllermeier [FH10], will include it in the general framework of preference learning in the following sections.

Consequently, we have structured this chapter to reflect this perspective. We will begin with an overview of the various settings encountered in preference learning in Section 4.1. As preference learning is a wide field, we will focus on settings that are more relevant to this thesis. We will then introduce evaluation metrics that are commonly used for the different settings in Section 4.2. Finally, we explore general solution approaches that have been proposed in the literature in Section 4.3. In this section, we will also investigate specific algorithms that will be used later in the thesis.

4.1. Preference Learning Settings

Comparable to the broader field of machine learning, preference learning can be classified into fundamental learning paradigms, including supervised, unsupervised, online, and reinforcement learning [HS24]. Each of these paradigms can be further differentiated into specific learning settings. Given that this thesis primarily focuses on the supervised learning paradigm, this section will predominantly address supervised learning settings within the context of preference learning. Nonetheless, a concise overview of other settings is provided in Section 4.1.5.

We are going to use the nomenclature introduced by Fürnkranz and Hüllermeier [FH10, pp. 1–17] and supplement it with our own. Similar to the supervised learning setting, we will encounter *instances*, that are usually represented by a vector $\boldsymbol{x}$ in an instance space $\mathcal{X}$.

instance: Page 41

In some settings, such as object ranking and object choice, we instead have instances that are sets of *objects*. Then, each object is represented by a vector $\boldsymbol{x}$, and $\mathcal{X}$ instead is called the *object space*. The instance space $\mathbb{Q}$ in these settings is then a subset of $2^{\mathcal{X}}$ and, for the rest of the thesis, will be called the *task space*. Then, an element $Q \in \mathbb{Q}$ is called a task or more specific to the setting a *choice task* or *ranking task*.

In the usual classification setting, we observe instances and an assignment of one or more relevant *labels*. Therefore, we also call a finite and discrete set of values $\mathcal{Y} = \{y_1, y_2, \ldots, y_M\}$ that can be ordered labels.

label: Page 41

With these concepts, we can generate the common settings in preference learning (see Table 4.1). In case we observe instances to which a ranking over labels is assigned, we call this setting *label ranking*. It is a setting similar to *multi-label classification*, but instead of a binary decision on whether a label is relevant, we rank the labels. A representative example application could be

Table 4.1.: Comparison of Different Preference Learning Settings

Setting	Input Data	Target/Output Data	Performance Measures
Label Ranking	Instance vector	Preferences on labels	Ranking/Correlation
Object Ranking	Set of object vectors	Preferences on objects	Ranking/Correlation
Object Choice	Set of object vectors	Choice of preferred object(s)	Binary
Instance Ranking	Instance vector	Ordered class label	Retrieval-based

an online shop that wants to predict a ranking of their products that best fit a given customer. We present the setting in detail in Section 4.1.1.

In label ranking, we typically do not observe feature information. When this is the case, we treat it as the *object ranking* setting. Instead of one instance vector, we assume that the instance is a set of object vectors, with the task of ranking them by preference. Here an example application is the ranking of products that have features (for example movies that have tags assigned to varying degrees). We introduce this setting in Section 4.1.3. A related setting is *object choice*, more commonly known as *choice modeling*. It arises when we replace the ranking of objects by a choice, i. e., the input is a set of objects, and the output is a subset of these. As example application consider a user typing in a query into a search engine and then clicking on a subset of the results. This subset could be treated as preferred over the non-chosen links. Even though we already introduce the classical version of this setting in Chapter 2, we will focus on the more recent works in the realm of preference learning in Section 4.1.4.

Furthermore, we have *instance ranking* where, similar to multi-class classification, we have an instance, and the target is a class label. In addition however, we assume that all class labels are ordered. So far, this setting is exactly that of *ordinal classification*. However, in instance ranking, we are not interested in learning a classifier that outputs a class label but a ranking function that can order a given set of instances. As an example application, think of the typical 5-star ratings you can find online. There we are often interested in a fine-grained sorting of the items rather than assigning coarse ordinal classes. We are going to look at this setting in more detail in Section 4.1.2.

Finally, to show the breadth of preference learning as a field and its importance in contemporary machine learning, we will give a short overview of other important learning settings within the field. This includes unsupervised settings such as clustering of preference data. In online learning, the preference-based bandit setting is important in practice because eliciting preferences from users is usually less demanding than absolute numeric values. A related problem is the guiding of reinforcement learning processes via preference feedback. An important application of this is the alignment of LLMs, which is usually done by reinforcement learning from human feedback (RLHF) [Zie+19; Ouy+22] or more recently DPO [Raf+23].

4.1.1. Label Ranking

In label ranking, we have an instance space $\mathcal{X}$, and we call the elements $\boldsymbol{x} \in \mathcal{X}$ of this space *instances*. In addition, there is a finite label space $\mathcal{Y} = \{y_1, \dots, y_M\}$. These spaces are related by a ranking function $f\colon \mathcal{X} \to \mathcal{S}_M$ that assigns to each

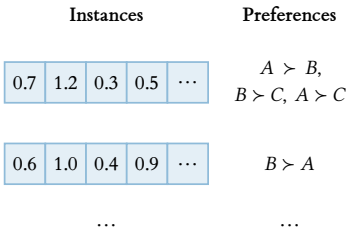


Figure 4.1.: The label ranking setting. Each training instance is assigned a set of pairwise preferences.

instance $\boldsymbol{x}$ a ranking $\pi_{\boldsymbol{x}} := f(\boldsymbol{x})$ such that $\pi_1 \succ \pi_2 \succ \cdots \succ \pi_M$. We will also use the notation $y_i \succ_{\boldsymbol{x}} y_j$ to denote that label y_i is preferred over label y_j in the context of instance $\boldsymbol{x}$ with f and π being implicit.

The learner has access to a dataset $\mathcal{D} = \{(\boldsymbol{x}_n, \mathcal{S}_n)\}_{n=1}^N$ where $\mathcal{S}_n \subset \mathcal{Y}^2$ is a set of pairwise comparisons (see Figure 4.1 for an illustration). A pair (y_i, y_j) is in $\mathcal{S}_n$ if $y_i \succ_{\boldsymbol{x}} y_j$. Note that we do not assume these preferences are complete. The goal is to learn a ranking function $\hat{f} : \mathcal{X} \to \mathcal{S}_k$ that approximates the ground truth ranking function f.

Label Ranking Algorithms In the context of label ranking, numerous methods have been developed. A taxonomy of these methods is presented in Figure 4.2. This taxonomy represents a curated selection from the literature, not an exhaustive list. It organizes the methods into four main categories, inspired by existing surveys [VG10; Zho+14; Der21]. It is worth noting that some methods may fit into more than one category.

reduction techniques

Reduction techniques, as the name suggests, reduce the problem of solving the label ranking problem to existing problems in machine learning. This is advantageous since it allows us to reuse existing algorithms. A natural way to simplify the problem is to decompose it on the level of labels or pairs of labels. The former leads to labelwise and the latter to pairwise decomposition methods.

An early pairwise method was proposed by Har-Peled et al. [HRZ02], who map the d-dimensional training instances into a (sparse) $N \times d$-dimensional space, in which they encode the pairwise relation of two labels. The resulting transformed training instances can then be used to train any binary classifier. Fürnkranz and Hüllermeier [FH03] instead propose to learn a quadratic number of classifiers, one for each ordered pair of labels. At prediction time, all classifiers are queried, and labels are ranked based on the number of times they were preferred over another label. There is, however, in general not a unique mapping from pairwise preferences to complete ranking [Hül+08], so other methods have been investigated in the literature. Vogel and Clémençon [VC20] consider a setting where only the top-ranked label is known for each instance. They reduce the problem to a pairwise classification problem and show that for sensible assumptions on the noise, the ground truth ranking can be recovered with high probability.

In *labelwise* decompositions [CHH13; DMP15] the idea is to predict the *rank* for each label using a separate classifier. Each of the classifiers is, therefore, solving a multi-class classification problem. An optimal assignment problem then has to be solved to predict the ranking for a new instance, which can be done in time cubic of the number of labels. Destercke et al. [DMP15] extend the labelwise decomposition to be able to predict partial orders.

Furthermore, there are additional reduction techniques that do not neatly fit into the pairwise vs labelwise distinction. Dekel et al. [DSM03] consider a setting where each instance is associated with a preference graph instead of a ranking. They propose a framework in which the preference graph is decomposed using a decomposition procedure that produces a set of subgraphs. Each subgraph induces a corresponding loss function. The authors then use a boosting algorithm to solve the resulting learning problems. Gurrieri et al. [GFS14] explore three more reduction techniques based on coding: nominal

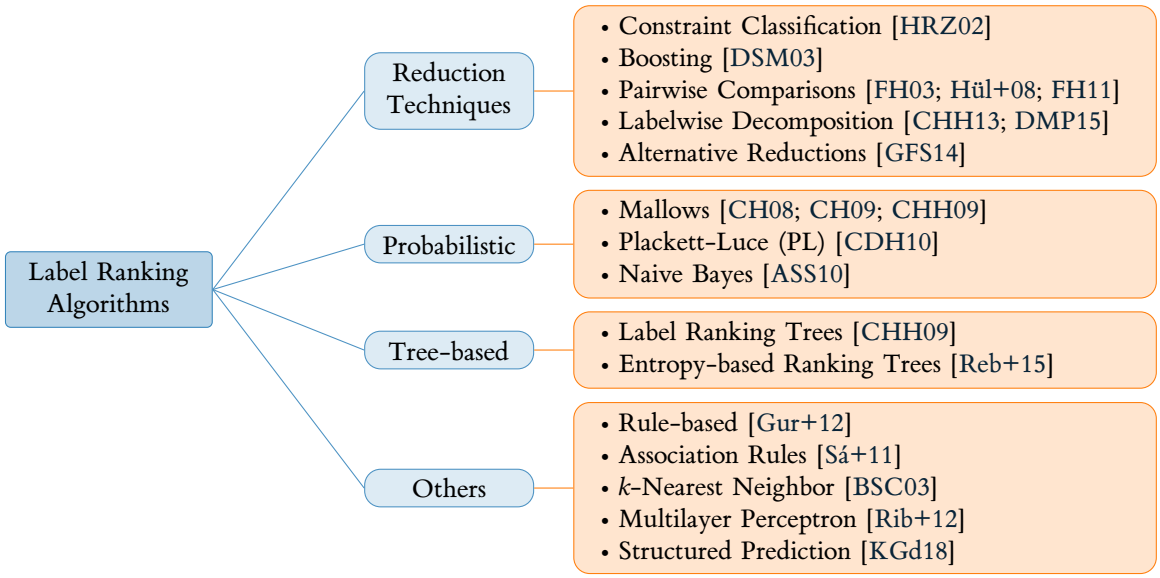


Figure 4.2.: Taxonomy of Label Ranking Algorithms.

coding, dummy coding, and classifier chains. These reductions are able to take correlations between labels into account.

probabilistic techniques

An important cluster of approaches are *probabilistic techniques*. A recurring theme here is that a ranking model like the Mallows [Mal57] or PL [Pla75] is used to handle ranking feedback directly. Cheng and Hüllermeier [CH08] propose an *instance-based* learning method based on the Mallows model. For a given instance, they find the k-nearest neighbors in the training instances. Then, they use maximum likelihood estimation (MLE) on the (partial) rankings of this subset to find the parameters of the Mallows model, in particular, the central ranking that can be used for prediction. Solving for these parameters can be done brute-force [CH08], which is computationally costly, or approximate [CH09; CHH09] based on expectation-maximization.

Cheng et al. [CDH10] also introduce a similar instance-based learning algorithm based on the PL model. MLE of the parameters is done using a minorization and maximization algorithm. A ranking can be predicted by simply sorting the learned parameters of the PL distribution. This prediction is provably optimal for commonly used loss functions. In experiments, the authors demonstrate that the instance-based learning using the Mallows model performs slightly better with complete rankings, but the PL based learners pull ahead as soon as rankings are incomplete. Aiguzhinov et al. [ASS10] adapt the probabilistic naive Bayes classifier to the setting of label ranking by using the Spearman correlation coefficient as a similarity measure. At prediction time, it is necessary to iterate over all possible rankings to evaluate their probability, which is computationally intensive.

tree-based techniques

Tree-based techniques adapt decision tree learning algorithms for use in a label ranking setting. Cheng et al. [CHH09] propose to construct decision trees based on the usual recursive partitioning. As splitting criterion, they fit a Mallows model and use within-leaf variances as a goodness-of-split measure. Prediction is then straightforward since it suffices to propagate an instance to a leaf node, where the central ranking of the Mallows model can be used as a prediction. Rebelo de Sá et al. [Reb+15] introduce a label ranking variant

of the ranking tree algorithm that uses Spearman's correlation coefficient to calculate a weighted mean correlation for each subset in a split. They also propose an entropy-based splitting criterion that uses ranking entropy as a measure of uncertainty, which is to be minimized for a split.

Other techniques

Finally, we will introduce a few *other techniques* that did not neatly fit into the previous categories. Sá et al. [Sá+11] and Gurrieri et al. [Gur+12] apply rule induction to label ranking. Rule learning approaches are particularly interesting from an interpretability perspective since they produce human-readable decision criteria that can be easily understood. To this end, the authors of the former paper define similarity-based support and confidence functions that can be used in the standard rule learning algorithm APRIORI. The similarity is defined in terms of full rankings using Kendall's τ or Spearman's correlation coefficient. Gurrieri et al. instead decompose the full rankings into pairs of labels. They use a dominance-based rough set approach as a classifier that produces classification rules for the pairs. A voting strategy is used to decide on a ranking during prediction time. Ribeiro et al. [Rib+12] propose a simple neural network approach where the output layer returns the scores for the K classes. The authors investigate six loss functions empirically and find that a position-dependent loss combined with an ordering-based loss performs best. Finally, Korba et al. [KGd18] approach the label ranking problem from the angle of structured prediction. They propose several embeddings that can be used in a surrogate least squares framework. That means rankings are mapped into an embedding space in which the Euclidean distance can be used as a loss function.

multi-label ranking

A Variant of the Setting *Multi-label ranking* is a generalization of label ranking [BFH06; Der21], where the learner first has to predict the correct set of labels for a given instance, and only then it is tasked to rank the labels. This is a problem often encountered in practice, where not all labels are applicable to every instance, or if one wants to separate relevance from preference. As an example, consider movie recommendations where each movie is a label and each user an instance. Every user has only seen a subset of the movies, which is their label set, and then we can ask them to rank these movies by preference. To tackle this problem, it is usually decomposed into the subtasks multi-label classification and label ranking. There are, however, methods like *calibrated label ranking* [BFH06] that solve both tasks simultaneously.

4.1.2. Instance Ranking

[Wer22]: Werner (2022), "A Review on Instance Ranking Problems in Statistical Learning"

Werner [Wer22] recently composed a review article on the topic of instance ranking, which we use as the main reference for this section. Recall that in label ranking, there is an external set of labels that are ranked in the context of an instance. Conversely, in object ranking, an instance consists of several objects (represented by feature vectors) that need to be ranked. In the case of object choice, we implicitly separate them into two classes: "chosen" and "not chosen". In *instance ranking*, we are interested in the ranking of the instances themselves. There are several sub-settings within instance ranking that we can distinguish based on the specific ranking task that is to be solved and the type of label space $\mathcal{Y}$ that is observed. Different combinations of ranking task and label space lead to different settings, which we will introduce in the following.

Firstly, we have an external set $\mathcal{Y}$, analogous to the label ranking setting. Each instance $\boldsymbol{x} \in \mathcal{X}$ is assigned an element y from the set $\mathcal{Y}$. This set $\mathcal{Y}$ can be discrete or continuous, but it is important that the set $\mathcal{Y}$ is ordered. In case $\mathcal{Y}$ is binary, i. e., $\mathcal{Y} = \{-1, 1\}$, we call the resulting problem a *bipartite instance ranking problem*. Likewise, in the general discrete case, if $|\mathcal{Y}| = M$, then we call it a *M-partite instance ranking problem*. Lastly, we say the problem is *continuous*, if $\mathcal{Y}$ is a continuous set.

Secondly, there are different ranking tasks that one could be interested in. In the *hard instance ranking* setting, the goal is to rank *all* instances correctly. If the goal is to find and rank only the top-M instances, we call this the *localized instance ranking* setting. Lastly, in the *weak instance ranking* setting, the goal is to detect the top-M instances without the need to rank them.

We assume that a dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ is given. The goal is to learn a ranking function $\hat{f} : 2^{\mathcal{X}} \to \mathcal{S}$, that for a given set of instances outputs a ranking of these instances. It is important to note that we are not necessarily interested in approximating the mapping $\mathcal{X} \to \mathcal{Y}$; rather, our aim is to determine the relative ordering of the instances accurately (within the accuracy constraints imposed by the ranking task).

Before we continue, it is important to distinguish instance ranking from several other similar settings. In classification/regression, we also assign a label/value y to a given instance $\boldsymbol{x}$. The main difference here is that we do not assume that the set $\mathcal{Y}$ has a particular order. This brings us to the setting of *ordinal regression*, where we indeed assume that the set $\mathcal{Y}$ is ordered. The defining difference between ordinal regression and instance ranking is that in instance ranking, it suffices to order a given set of instances, while in ordinal regression, we need to predict a label for each instance, which often requires additional discretization [Wer22]. Finally, instance ranking is also very similar to object ranking. If we make the assumption that all objects can be ordered according to some partial order, then this reduces to an M-partite instance ranking problem.

ordinal regression

Instance Ranking Algorithms The literature on the instance ranking problem is vast due to its wide applicability in IR and recommender systems. Figure 4.3 provides an overview of the techniques that have been utilized to tackle this problem. The taxonomy groups the approaches based on the general modeling approach that is used, in line with the categorization proposed by Werner [Wer22].

The first major category is approaches using support vector machines (SVMs). The seminal paper on this topic was written by Joachims [Joa02], who proposes the *RankingSVM* algorithm. They note that their optimization problem is equivalent to that of a classification SVM where pairwise differences of instance vectors are used as instances. Several improvements to RankingSVM have been investigated in the literature [Cao+06; Qin+07; JJS11]. Of note is the relational RankingSVM algorithm designed for relational ranking data introduced by Qin et al. [Qin+08], which not only takes into account the overall ranking of the instances but also their relation.

SVM approaches

MPRank is a variant that tackles the continuous variant of the instance ranking problem [CMR07b]. The authors impose a penalty on the pairwise differences of the scores such that they are close to the observed difference. In a follow-up

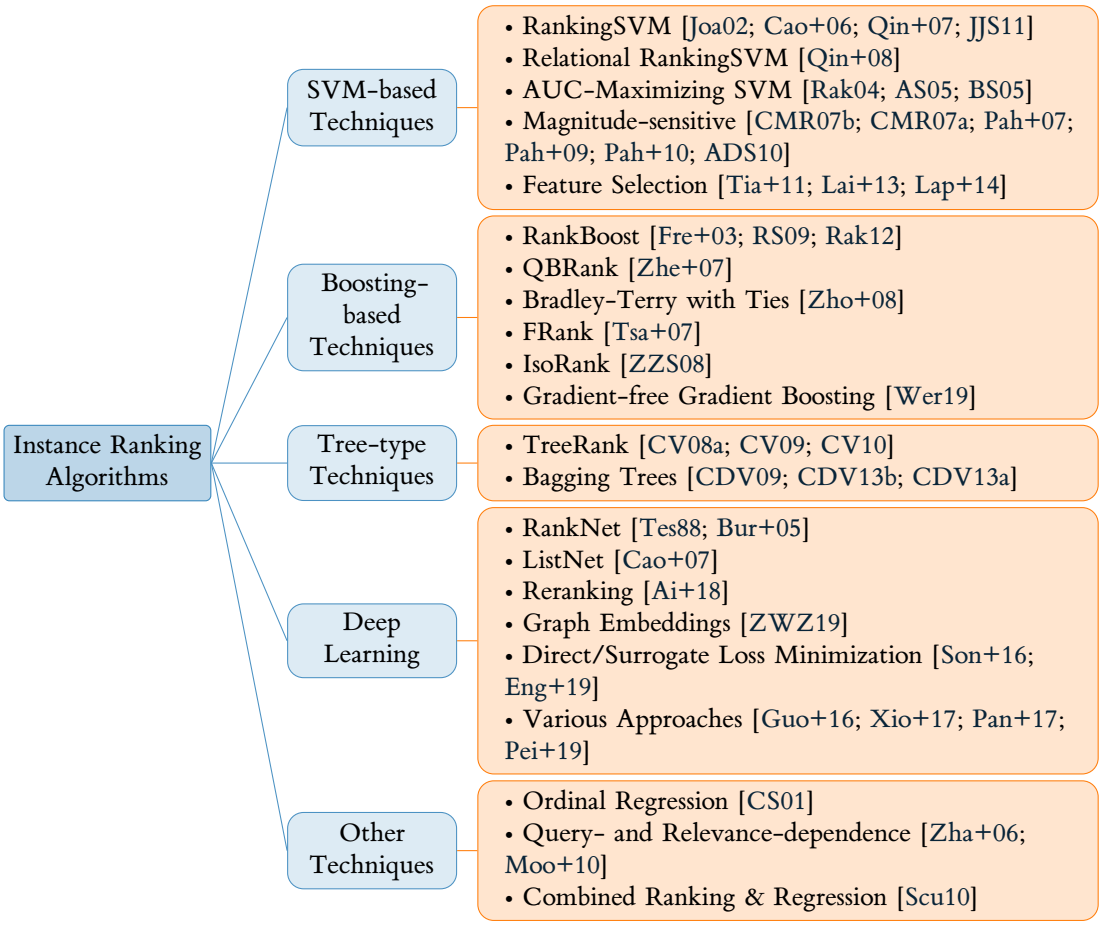


Figure 4.3.: Taxonomy of Instance Ranking Algorithms.

paper Cortes et al. [CMR07a] introduce the variant SVRank that minimizes an ϵ-insensitive loss, which, however, increases the algorithmic complexity. Independently, Pahikkala et al. [Pah+07] introduce the RankRLS algorithm that is very similar to MPRank. Later, they modify the algorithm to be able to handle pairwise preferences [Pah+09] and improve the training complexity using a greedy algorithm [Pah+10]. Agarwal et al. [ADS10] apply different variants of the RankingSVM to rank chemical structures. For the continuous instance ranking setting, they also advocate including the magnitude of the score difference in the loss and defining a corresponding loss function.

Rakotomamonjy [Rak04] and Ataman and Street [AS05] propose a different SVM formulation that maximizes the AUC of a scoring function. Brefeld and Scheffer [BS05] observe that this problem formulation results in an AUC smaller than that of a typical SVM and propose an alternative formulation based on norms.

There is a notable subset of literature dedicated to the challenge of training SVMs that incorporate feature selection. A range of methods has been put forth to address this [Tia+11; Lai+13; Lap+14]. A common thread among these approaches is the modification of the loss function to encourage sparsity.

boosting approaches

Moving on to *boosting approaches*, Freund et al. [Fre+03] transferred the well-known Adaboost algorithm to the setting of ranking (which they call RankBoost). Instead of weighing instances individually as is usual in boosting, they

weigh pairs of instances (see Figure 4.10 for a visualization). Therefore, pairs of instances are weighted more strongly if it is important that the learners rank them correctly. Rudin [Rud09] propose a variant of RankBoost that allows one to put more emphasis on the top elements of a ranking. They do this by employing convex objectives with a p-norm. The case where this becomes the ∞-norm has been investigated by Rakotomamonjy [Rak12].

Zheng et al. [Zhe+07] address an IR setting wherein they consider both relevance scores and pairwise preferences. Since relevance scores alone might not adequately determine a good order of instances, integrating pairwise preferences can be beneficial. They introduce a loss function that integrates these two factors and optimizes it with gradient boosting, naming their algorithm QBRank.

In many applications such as IR, ties often arise, particularly when two documents receive identical relevance scores. To address this, Zhou et al. [Zho+08] have extended both the Bradley-Terry and the Thurstone-Mosteller models to accommodate ties. These modified models are subsequently used to define specific loss functions, which are then optimized using gradient boosting.

Tsai et al. [Tsa+07] argue that the cross-entropy loss used by many algorithms, such as RankNet, does have some undesirable properties. They instead advocate using a fidelity loss instead, a distance measure used in quantum physics to describe the difference between quantum states. They find that their algorithm improves on algorithms such as RankBoost.

A noteworthy boosting method is presented by Zheng et al. [ZZS08]. Iteratively, they construct an additive model through the use of gradient boosted regression trees. In each iteration, deviations between pairs, necessary for the current model to rank every pair correctly, are computed. Following this, a regression tree is trained to predict these deviations, with its output being weighted and incorporated into the current model. To minimize the deviations in each iteration, an isotonic regression optimization problem is employed, hence the name IsoRank.

Werner [Wer19] argues that utilizing surrogate losses as a substitute for the hard ranking loss[1] in the context of hard continuous instance ranking is suboptimal due to the surrogates' unbounded nature and the high computational costs associated with gradient calculations. Instead, they introduce a "gradient-free" gradient boosting method, termed SingBoost, designed to more directly minimize the hard ranking loss. To mitigate overfitting, they incorporate *stability selection* [MB10], a technique that employs subsampling to identify a consistent set of variables. Importantly, this approach is applied at the predictor level to ensure a reliable set of predictors for ranking.

1: With *hard* we are here referring to a loss that penalizes the complete ranking, as opposed to a *weak* ranking loss, where the focus lies on the top-K instances.

Even though we have seen a few tree-based boosting techniques in the preceding paragraphs, there are also dedicated that have been developed for instance ranking. Clémençon and Vayatis [CV08b] consider the bipartite ranking problem and aim to directly approximate the optimal ROC curve. They come up with a recursive approximation scheme, which can readily be represented by a tree-structured decision rule. They call the resulting approach TreeRank. The authors also developed subsequent theoretical and algorithmic advances [CV09; CV10]. Since TreeRank is a decision tree, it is a natural question to ask if it can directly be combined with bagging to arrive at an ensemble

tree-type techniques

algorithm. Clémençon et al. [CDV09; CDV13b] tackle exactly this question. The main difficulty here is the aggregation of multiple rankings (also known as the Kemeny problem). In their RankingForest algorithm [CDV13b], the authors compute the median ranking to solve the aggregation problem.

The increase in popularity of neural networks can also be seen in the context of instance ranking. In the seminal work by Burges et al. [Bur+05], the authors propose the RankNet architecture. It employs an idea that goes back to work by Tesauro [Tes88], featuring a neural network that outputs a real-valued score. To use the network for ranking, one inputs two instances and considers the difference in scores. The weights of the network are trained from pairs of instances using the binary cross-entropy loss function.[2] In IR, one typically puts more emphasis on the top positions of a ranking. RankNet treats each position equally, which is not ideal. Cao et al. [Cao+07] therefore extend RankNet to be able to optimize for the top-k positions. Since the network is now trained from lists of instances, they call the network ListNet (see Page 103 for a detailed description).

2: Refer to Page 98 for a more detailed explanation of RankNet.

While many methods primarily focus on context-independent utility functions, which assign scores to each object irrespective of the query, Ai et al. [Ai+18] introduced an approach capable of reranking search results based on the specific query context. Initially, they generate a context-independent ranking for instances within a given query. Subsequently, a RNN is employed to map the top-k instances into a contextual representation. The instances are then reranked using a model that considers both the context-independent score and the derived context representation.

In the domain of product search, challenges arise due to sparse data and the presence of long-tailed distributions, which hinder the effectiveness of conventional retrieval models. Zhang et al. [ZWZ19] advocate to incorporate the query and product graphs into the ranking algorithm. Within the query graph, two queries are interconnected if they share a common product click. Analogously, in the product graph, two products are connected if they are sought by a mutual query. The researchers utilize graph embedding layers, transforming both graphs into vector representations. In order to estimate the relevance of a product to a specific query, the graph-based representations of both the product and the query are appended to the feature vector.

Typical ranking algorithms do not directly optimize the target loss but instead minimize a surrogate loss. This is done because target losses are often discrete or nonconvex, which makes them hard to optimize. Song et al. [Son+16] employ loss-augmented inference, which allows them to optimize the target loss directly. They apply their approach to the average precision (AP) loss, a very relevant loss function in IR. The researchers demonstrate empirically that their approach outperforms surrogate loss methods as soon as label noise is introduced.

Typical ranking algorithms often minimize a surrogate loss rather than directly optimizing the desired target loss. This approach is prevalent because target losses are frequently discrete or nonconvex, making them hard to optimize. However, Song et al. [Son+16] utilize *loss-augmented inference*, enabling them to optimize the target loss directly. They specifically apply their method to the AP loss, a popular loss function in the realm of IR. Empirical results indicate that their method surpasses surrogate loss techniques when label noise

is present. Engilberge et al. [Eng+19] aim to make it possible to optimize non-differentiable loss functions. They identify the non-differentiable sorting operation as a common challenge in many ranking loss functions. To address this, the authors suggest replacing the sorting operation with a learned, differentiable sorting network that can serve as a surrogate in a variety of loss functions.

other techniques

Finally, the instance ranking problem has also been addressed using *other techniques* that do not fit the previous categories. An early work on the instance ranking problem was the one by Crammer and Singer [CS01], who adapted the perceptron learning algorithm to instance ranking and called it PRank. Zha et al. [Zha+06] identify the problem of query-dependent features varying strongly across queries. To counteract this, they fit two models, one query-independent scoring model and an additional query-dependent, monotone function that transforms the scores returned by the scoring model.

Moon et al. [Moo+10] argue that while typical relative ranking methods are able to capture the overall ranking accurately, they are not able to predict the correct relevance scores. They developed the IntervalRank algorithm that uses isotonic regression to get the correct overall relative ordering. The algorithm further incorporates penalties to make the predicted scores align closely with the ground-truth labels and to separate all labels distinctly. In a similar vein, Sculley [Scu10] consider a regression setting with a long-tailed characteristic, where there are few observations for rare events. They posit that in such scenarios, leveraging relative ranking data is beneficial. Consequently, they propose a hybrid loss function that combines regression and ranking, and their empirical tests highlight its competitive performance in executing both tasks concurrently.

4.1.3. Object Ranking

In object ranking, the instances are sets of *objects*, where an object itself is represented by a feature vector $\boldsymbol{x} \in \mathcal{X}$ with $\boldsymbol{x} := (x_1, \ldots, x_d)^\mathsf{T}$. Here $\mathcal{X}$ is usually a subset of $\mathbb{R}^d$ and called the object space or universal object set. Whereas in label ranking, the target is a ranking of a set of labels, in object ranking, the objects themselves are ranked. We call a set of objects to be ranked a *task* Q with $Q \in \mathbb{Q} \subseteq 2^{\mathcal{X}}$ and $Q := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}$.

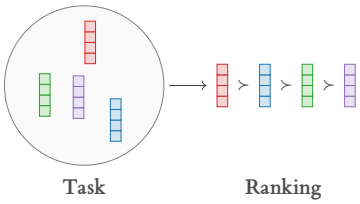


Figure 4.4.: The object ranking setting.

For each task Q of size K we also observe a corresponding order O such that if $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in O$ we have that $\boldsymbol{x}_i \succ \boldsymbol{x}_j$. We say an order is a *total order* if there is a permutation π such that $\boldsymbol{x}_{\pi(1)} \succ \boldsymbol{x}_{\pi(2)} \succ \cdots \succ \boldsymbol{x}_{\pi(K)}$. The learner observes a dataset $\mathcal{D} = \{(Q_n, O_n)\}_{n=1}^N$, where $Q_n \subseteq \mathcal{X}$ is a task and O_n is the previously mentioned corresponding partial order. The goal is to learn a ranking function $\hat{f} : 2^{\mathcal{X}} \to \mathcal{S}$, with $\mathcal{S} := \bigcup_{k=2}^K \mathcal{S}_k$ being the union of all permutation groups of a certain size k.

At this point, it makes sense to mention the setting that arises when label ranking and object ranking are combined. In this new setting, the learner receives an instance comprised of instance features and a set of objects with object features. This can also be represented as a set of *dyads*, where each dyad is a pair of an instance and an object. The learner is then tasked with ranking the dyads. The setting is called *dyad ranking* and is a generalization of both label and object ranking [SH18]. A prototypical example of dyad ranking is

the task of ranking movies for a given user, where the user is the instance and the movies are the objects. As such, dyad ranking as a setting is closely related to the field of recommender systems and collaborative filtering [Gol+92].

Object Ranking Algorithms Given the substantial parallels between object ranking and instance ranking, and considering the field's shift to learning-to-rank as the main application, the term "object ranking" has witnessed a reduced prevalence within the scientific literature. We therefore opt to go into more detail with general learning-to-rank approaches in Section 4.3 since they are relevant for the remainder of the thesis.

4.1.4. Object Choice

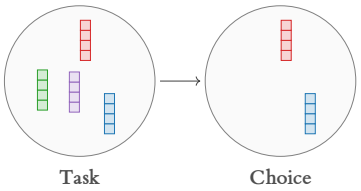


Figure 4.5.: The object choice setting.

Where in the last section we looked at rankings of objects, here we consider the setting where instead we observe *choices*. We use the term "object choice" here to refer to this setting to be consistent with the nomenclature introduced in this thesis. In the preference learning literature, no universal term is used to refer to this setting. The term "discrete choice" is often used, which connects to the nomenclature introduced in classical preference modeling (see Section 2.4).

As before, we have instances that are sets of *objects* $\boldsymbol{x} \in \mathcal{X}$ with $\boldsymbol{x} := (x_1, \dots, x_d)$. The object space $\mathcal{X}$ we assume to be a subset of $\mathbb{R}^d$. We additionally let $\mathbb{Q} \subseteq 2^{\mathcal{X}} \setminus \emptyset$ be the set of admissible sets of objects. A set of objects Q with $Q \in \mathbb{Q}$ we call a *choice task*. Note how we do not yet assume anything about the structure of $\mathbb{Q}$. In particular, no *closure* properties are assumed. In case additional restrictions on $\mathbb{Q}$ are required later on in this thesis, we will mention these at the appropriate places.

choice task

Example 4.1.1 If a family of sets $\mathbb{Q}$ is *closed* with respect to the subset operation $\subset$, then for all sets $A \subset B \in \mathbb{Q}$ it holds that $A \in \mathbb{Q}$.

As for the choices, in the literature, it is typically assumed that a single object is selected. Here, we do not make this assumption and allow choices to be arbitrary subsets. A *choice* is therefore a subset $C \subseteq Q$ with $C \neq \emptyset$. Note that since no closure properties have been assumed, it does not necessarily hold that $C \in \mathbb{Q}$. The typical restriction to singleton sets in the literature corresponds to $|C| = 1$.

choice

Based on a dataset $\mathcal{D} = \{(Q_n, C_n)\}_{n=1}^N$, we seek to find a function $c : \mathbb{Q} \to \mathcal{C}$ that generalizes well to unseen choices. Typical performance measures for this setting are those used in the context of multi-label classification, such as the $F1$ score (see Section 4.2.1).

Object Choice Algorithms As we focused on classical models for preference modeling in Section 2.4, we will here present an overview of the approaches that were developed in the field of preference learning specifically. As is common with machine learning models, these are usually focused on predictive performance as opposed to the existing normative, prescriptive, or descriptive models. The most important distinction is whether the approaches are designed to handle only singleton or also subset choices. This is because it substantially changes the way the choices have to be modeled. For example, if one naïvely were to model each possible subset as one possible outcome, then the space of of possible outcomes becomes exponential in the number of objects, which makes it unfeasible computationally.

We begin with the *singleton choice approaches*. Early work included models based on restricted Boltzmann machines (RBMs) that were proposed by Osogami and Otsuka [OO14] and Otsuka and Osogami [OO16], where choices are modeled using a generative model. RBMs are a flexible model class that can model a diverse range of probability distributions. They are, however, difficult to train reliably and efficiently, which limited their applicability in practice [Cho+15]. In the context of flight itineraries Mottini and Acuna-Agost [MA17] develop a choice model that operates on *sequences* as input (as opposed to sets), such that the predicted outputs are positions in the sequence.

singleton choice approaches

Rosenfeld et al. [ROS20] adopt ideas from the multi-self literature and propose an approach based on neural networks. The overall network architecture decomposes as [ROS20]:

multi-utility models: Section 2.4.2

$$g_{\boldsymbol{f}}(\boldsymbol{x} \mid Q; w, r, \mu) = \sum_{i=1}^{\ell} w(\{f_i(\boldsymbol{x})\}_{\boldsymbol{x}\in Q})\mu\big(f_i(\boldsymbol{x}) - r(\{f_i(\boldsymbol{x})\}_{\boldsymbol{x}\in Q})\big) \tag{4.1}$$

where $\boldsymbol{f}$ consists of ℓ linear functions, and w, r are permutation-invariant neural networks. The idea is that w maps each score set to a set-dependent *weight*. Similarly, r maps each score set to a set-dependent reference value. The function μ is defined as follows

permutation-invariant neural networks: Section 3.3.5.3

$$\mu(z; c) \coloneqq \tanh(z)\big(c[\![z < 0]\!] + [\![z \geq 0]\!]\big),$$

and implements a loss aversion function with $c \geq 1$ controlling how strong the effect is. Rosenfeld et al. [ROS20] prove that the model complexity can be flexibly controlled by changing the number of linear functions ℓ, and that this has diminishing returns with respect to the approximation error. Similarly, the authors show that the estimation error for different aggregators scales well with ℓ, implying that the model can be learned efficiently. The decomposition proposed by the authors subsumes many existing models as special cases, making it an important baseline for our empirical evaluation.

We move on to *subset choice approaches*. For the setting *without* object features, Benson et al. [BKT18] define a probabilistic choice model for subset choices. They define it as a mixture of MNL models of a fixed choice set size. Each object is assumed to have a certain value, and higher-order correction terms are applied where necessary. They show, however, that it is $\mathcal{NP}$-hard to find the optimal set of correction terms.

subset choice approaches

Due to the lack of other work on subset choice, we propose a general way to turn existing ranking models into singleton and subset choice models using thresholding [PH20; Pfa+22]. Please refer to Chapter 5 and specifically Section 5.2.1 for a description of this method. We will use this method to adapt a variety of models, including our own, to this learning task.

4.1.5. Other Important Settings

As mentioned before, preference learning encompasses many more settings than the supervised learning settings we introduced in the preceding sections. Interest in preference learning has increased substantially in recent years due to its application in the training of LLMs [Ouy+22; Raf+23]. This is why we will take a broader look at important application areas in this section to

showcase the breadth of the field. The recent survey by Hüllermeier and Słowiński [HS24] provides an excellent overview of the field and was used as the main reference here.

[HS24]: Hüllermeier et al. (2024), "Preference Learning and Multiple Criteria Decision Aiding: Differences, Commonalities, and Synergies—Part II"

Online Preference Learning One important area we want to highlight here is *online preference learning*, where online machine learning is combined with preference-based feedback. In online learning, the data arrives incrementally, and the learner has to make predictions at each step. The multi-armed bandit setting is a specific online learning setting where a learner is repeatedly confronted with the choice of so-called bandit arms. If one arm is pulled, the learner receives a (potentially stochastic) reward. Roughly speaking, the goal is to maximize the cumulative reward, which makes it necessary to explore different arms to find the best arm quickly but also to pull the best arm found so far often enough. This is the so-called *exploration-exploitation tradeoff*. In many settings, we do not directly observe rewards when pulling an arm, but we can compare two arms and observe a preference. Typical examples include sports tournaments, where teams play each other with the goal of quickly determining an overall winner, and scenarios involving human feedback where absolute ratings for options are difficult to express, but relative preferences are easier to provide. Yue and Joachims [YJ09] formalize this setting as the *dueling bandits problem* and propose a gradient-based method to solve it. The authors show that the algorithm can readily be applied to a web search setting, demonstrating its relevance. Haddenhorst et al. [HBH21] generalize the setting by letting the learner select more than two arms. Then, the winner of these arms is observed as preference, or singleton choice in our terminology. They call it the *multi-dueling bandits problem*.

online preference learning

exploration-exploitation tradeoff

dueling bandits problem

A related problem is *preference-based optimization*, which shares similarities with HPO (see Section 3.2) in its goal of finding the global optimum of a target function. However, the key distinction lies in the evaluation method. In preference-based optimization, we cannot directly query the output value of a single point. Instead, we select two points and receive a preference between them. This approach can be viewed as a continuous version of the dueling bandits setting, highlighting the close relationship between these optimization techniques. To make learning feasible, one usually assumes a certain kind of smoothness of the target function. González et al. [Gon+17] introduce this setting and propose a Thompson sampling-based approach using GPs as surrogate models.

preference-based optimization

Another setting in online preference learning is *preference-based reinforcement learning*. In classical reinforcement learning, a learner interacts with an environment and sometimes receives a reward. A central problem in reinforcement learning is that it is not clear how to define a reward function in a way that accurately captures what the learner is supposed to do, cannot be gamed by the learner, and is given often enough to facilitate reward attribution [Wir+17]. In preference-based reinforcement learning, instead of specifying a reward function, we express preferences on the level of actions, states, trajectories, or even policies [Bus+14; Wir+17; Kau+23]. The advantage being that the learner can use the preference feedback as implicit feedback about what should be rewarded. Some approaches, such as the one by Christiano et al. [Chr+17], try to infer an explicit reward model from the preference feedback. Christiano et al. show human annotators pairs of segments of a full trajectory produced by

preference-based reinforcement learning

[Wir+17]: Wirth et al. (2017), "A Survey of Preference-Based Reinforcement Learning Methods"

a policy and ask them for their preference. Using a Bradley–Terry–Luce (BTL) model, they fit a reward model that is then used in a classical reinforcement learning loop to train the next policy.

Ouyang et al. [Ouy+22] use preference-based reinforcement learning to align a LLM, that was pretrained on a large body of text and finetuned using supervised examples, to be more useful for human instructions. For a given query, the researchers let the model output four different outputs. These outputs are then ranked by humans for quality and adherence to the prompt. This preference data is collected and used to train a reward model. The reward model is then used as a proxy in proximal policy optimization to produce a language model adept at following prompts. Rafailov et al. [Raf+23] note that training the reward model as an intermediate step is unnecessary and propose DPO. Using the preference data, they optimize the model directly to produce more preferred outputs.

Unsupervised Preference Learning

Unsupervised Preference Learning Until now, we only looked at settings where some kind of supervision is available. Whether it is direct supervised learning or indirect supervision, such as in the case of reinforcement learning. In unsupervised preference learning, we are given preference data, which is the object of study itself, meaning we want to find patterns in the preferences that give us additional insight.

clustering

A typical task in unsupervised learning is *clustering*. The objective of clustering is to partition a dataset into subsets where objects inside a cluster are similar, and objects of different clusters are dissimilar. In the context of preference data, models such as Mallows' model [Mal57] can be used to infer the representative ranking for a given dataset. Busse et al. [BOB07] propose to find several clusters using a mixture model. Using an expectation-maximization algorithm, they are able to approximate the parameters of this model efficiently. Bayesian mixture models have been introduced by Vitelli et al. [Vit+18] for the Mallows model. As a probabilistic model for rankings, the PL model can also be used in a clustering context. Caron et al. [CTM14] use the PL model in a Bayesian nonparametric regime, where an infinite number of mixture components is assumed. Mollica and Tardella [MT17] then provide an efficient inference algorithm for finite PL mixtures.

pattern mining

Another unsupervised learning task is *pattern mining*, where one aims to find statistically significant, "interesting" patterns in a dataset. There are different measures for what constitutes an interesting pattern, which depends on the kind of data that is present. Henzgen and Hüllermeier [HH19] are the first to transfer this problem to preference data. They propose an algorithm for mining rank data, which can identify frequent subrankings. Given label ranking data, Sá et al. [Sá+18] aim to do *exceptional preferences mining*. Their goal is to find subgroups of instances that have preferences deviating from the norm. They propose several quality measures that measure how much the preference matrix of a subgroup deviates from the global preference matrix.

4.2. Evaluation Measures

In the realm of preference learning, there is a wide variety of evaluation measures and loss functions. These strongly depend on the specific setting one is solving and, therefore, the format of the given data. For a given setting, the 0/1-loss is typically recognized as the canonical loss function, primarily because minimizing it theoretically yields perfect prediction accuracy. Nevertheless, it is rarely used in practice. This stems from its inherent optimization difficulties and its inability to reflect model confidence, making it less effective for differentiating the performance of different models. Frequently, there are additional considerations that are important in the choice of a target measure. For example, when addressing the issue of significant imbalances between the number of chosen and unchosen objects, it is advisable to consider metrics such as the F-measure or the A-mean because they mitigate the bias towards the majority outcome. In ranking, one must decide on the relative importance of each rank. Is it essential to treat each position with equal weight or to ensure the highest accuracy for the top-K elements? For the former approach, Kendall's τ is often the measure of choice. In contrast, for scenarios prioritizing the top positions, IR metrics such as normalized discounted cumulative gain (NDCG) or expected reciprocal rank (ERR) are typically more suitable.

The primary focus of this thesis is addressing choice problems. Accordingly, we will comprehensively examine the measures prevalent in and useful for choice-related research. To provide context for the methodologies introduced in Section 4.3, an overview of the principal evaluation measures for ranking will also be included, albeit in a more succinct manner.

4.2.1. Choice Measures

singleton and subset choice: Section 2.1

In the choice setting, we consider two settings in this thesis: singleton and subset choice. Recall that with $Q \in \mathbb{Q}$, we denote a choice task, which is a subset of the set of all objects $\mathcal{X}$. A choice is then expressed as a subset $C \subseteq Q$. In the following, we simplify notation by omitting Q from the function signature of the loss but assume that it is passed implicitly. Depending on the field, application, or convention, the following measures can be defined such that the measure should be maximized or minimized. For example, one can maximize the 0/1-accuracy or equivalently minimize the 0/1-loss.

categorical 0/1-loss

Canonical 0/1-Loss We will begin with the more straightforward singleton choice setting, where $|C| = 1$. The canonical *categorical 0/1-loss*, is then defined as

$$L_{0/1}(C, C') := [\![C \neq C']\!], \tag{4.2}$$

where C is the ground-truth singleton set and C' the prediction we want to score. It is easy to see that any false prediction will result in a loss of exactly 1, while predicting the correct object yields a loss of 0, hence the name. Inverting this loss with the formula $1 - L_{0/1}(C, C')$ leads to what is termed *categorical accuracy*, then we get what is called the *categorical accuracy*.

categorical accuracy

In applications where choices are noisy or ambiguous, meaning many top items have a similar quality, or when the number of options is large, or the

choice problem is hard to learn, using *top-k accuracy* can be beneficial. Top-k accuracy is defined as the fraction of times the ground-truth chosen object is found within the top k positions according to the predicted scores [Cho+17; BL18].

One important baseline when evaluating different learners is always the performance of a random guesser to gauge whether the learner is actually learning something. An immediate drawback of the categorical 0/1-loss is that the choice task becomes harder with increasing size of the task Q. If a learner guesses uniformly at random, they will achieve an 0/1-loss of $1 - \frac{1}{|Q|}$. Therefore, the performance of a random guesser is not constant, which makes assessing the performance of a learner more difficult, especially when averaging across a variety of task sizes. This is what the *normalized (categorical) 0/1-loss* is trying to solve. It is defined as

normalized (categorical) 0/1-loss

$$L_{\text{norm } 0/1}(C, C') := \frac{|Q| L_{0/1}(C, C')}{|Q| - 1} \,. \tag{4.3}$$

It has been known as "correction for guessing" for a long time [DE73]. The nominator can be regarded as the expected number of objects the learner chooses incorrectly from a set of size $|Q|$. The denominator specifies the actual number of incorrect objects in the singleton choice setting. Normalization results in a loss where 0 denotes a perfect learner, 1 indicates performance equivalent to random guessing, and values greater than 1 suggest learners that are outperformed by random guessing. Similar to the 0/1-loss, we can define the *normalized accuracy* as $1 - L_{\text{norm } 0/1}(C, C')$.

normalized accuracy

It is possible to use definition (4.2) for the subset choice setting as well. In other words, a prediction C' would only be considered correct if the entire set equals the ground truth set C. We will call this the *subset 0/1-loss*.

subset 0/1-loss

As mentioned in the introduction of Section 4.2, there are a few issues with the 0/1-loss and the subset 0/1-loss. The more general problem is that the losses are not continuous, hence not differentiable, are constant almost everywhere, and non-convex. Therefore, they are difficult to minimize directly.

As for the subset 0/1-loss, it is clear that it becomes exceedingly less useful the larger Q and C become since any error by the learner results in a loss of 1. An alternative would therefore be to evaluate the object-wise 0/1-loss $\frac{1}{|Q|} \sum_{\boldsymbol{x} \in Q} [\![\boldsymbol{x} \in C]\!] [\![\boldsymbol{x} \in C']\!]$. It is also referred to as the *Hamming loss* in the context of multi-label classification [Wae+14]. This loss, commonly referred to as the *Hamming loss* in the context of multi-label classification [Wae+14], addresses the aforementioned issue by scoring each object in Q independently.

object-wise 0/1-loss/Hamming loss

However, the Hamming loss is not without its limitations. A critical shortcoming is its equal weighting of false positives and false negatives. This characteristic can lead to skewed evaluation in scenarios with a strong imbalance between positive and negative instances. For instance, consider a subset choice problem where choice sets with more than one object ($|C| > 1$) only emerge when the decision-making process is indifferent or uncertain. In such cases, singleton sets are frequent, and the dataset has a significant disparity between the number of negative and positive instances.

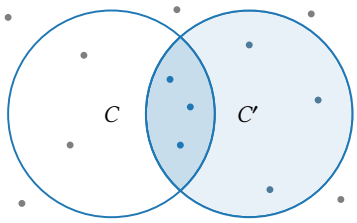


Figure 4.6.: *Precision* is the proportion of correctly chosen objects among those chosen by the model.

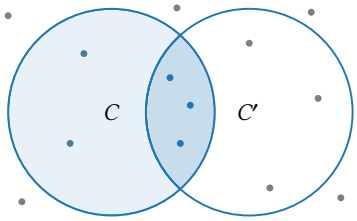


Figure 4.7.: *Recall* is the proportion of objects correctly chosen by the learner among all relevant objects.

F_1-measure

The F_1-Measure Consequently, measures commonly used in multi-label classification in the presence of strong imbalance, are more suitable in the subset choice setting as well. One particularly popular measure is the F_1-*measure*, the harmonic mean of the *precision* and *recall*. These constituents are defined as

$$\text{Precision}(C, C') = \frac{|C \cap C'|}{|C'|} \qquad \text{Recall}(C, C') = \frac{|C \cap C'|}{|C|} \tag{4.4}$$

and depicted in Figure 4.6 and Figure 4.7. The precision represents the proportion of objects that were correctly chosen by the learner among all chosen by the learner. A low precision is an indicator for a learner that chooses too many objects for a given task Q. Solely maximizing the precision of a model is not ideal since it favors models that are very conservative, potentially missing many of the true chosen objects. Conversely, recall is the fraction of objects the learner chooses correctly among all relevant objects. A learner with a low recall is one that chooses too few of the true chosen objects. Similarly to before, it is not useful to optimize recall directly since a learner that always chooses all objects in Q will have maximum recall.

In essence, we want a choice model that achieves a good performance in both of these measures. Such a measure is the F_1-*measure*

$$F_1(C, C') := \frac{2\,\text{precision}(C, C') \cdot \text{recall}(C, C')}{\text{precision}(C, C') + \text{recall}(C, C')} = \frac{2|C \cap C'|}{|C| + |C'|}\,. \tag{4.5}$$

The harmonic mean is dominated by small values, and therefore, both precision and recall need to be high such that the F_1-measure is high as well. Note that a very similar measure is the *Jaccard index*, which is represented by the expression

$$\text{Jaccard}(C, C') := \frac{|C \cap C'|}{|C \cup C'|}.$$

The Jaccard index is especially popular in the field of computer vision, where it is utilized, among other applications, as a measure for the similarity of bounding boxes or regions [Eve+06]. Both measures are monotonically related, but this no longer holds when summed across a dataset, with the Jaccard index being harder to optimize [Wae+14]. We get the corresponding F_1-*loss* by inverting:

F_1-loss

$$L_{F_1}(C, C') := 1 - F_1(C, C') \tag{4.6}$$

Similar to the 0/1-loss, this loss assumes values in $[0, 1]$.

When moving from one instance to a complete dataset, the computation of the F_1-measure/loss is not uniquely defined in multi-label classification [Koy+15]. This is because the confusion matrix can be computed at different levels of averaging. If we calculate the F_1-measure/loss for each instance individually, as shown in (4.5), we are employing a method known as *instance-averaging*:

instance-averaging

$$L_{\text{inst}}(h, \mathcal{D}) := \frac{1}{N} \sum_{i=1}^{N} L_{F_1}\big(C_i, h(Q_i)\big) \tag{4.7}$$

Alternatively, we could first compute the average confusion matrix for each label, compute the corresponding F_1-measure, and finally average across all labels. This is called *macro-averaging*. Lastly, we could compute the average confusion matrix across both labels and instances and use that to evaluate a global F_1-measure. This is referred to as *micro-averaging*.

macro-averaging

micro-averaging

Each of these different methods has their own benefits and tradeoffs. With macro-averaging we place equal importance on each label, which can be advantageous, if the model should also perform well for rare labels. Instance-averaging places equal weight on each instance instead, which is particularly useful in situations where the number of labels varies. Finally, micro-averaging can be useful when the dataset is imbalanced. However, not all of these apply to the problem of object choice. In this setting, we do not have a fixed vector of labels; instead, each instance is comprised of a set of objects only identified by their feature vector. The set of objects $\mathcal{X}$ could even be uncountably large since it is a subset of $\mathbb{R}^d$. Therefore, macro-averaging is not possible in the context of object choice unless there are many reoccurring objects across instances. For this thesis, we therefore use instance-averaging.

The A-Mean However, the downside to the F_1-measure/loss is that it depends on the prevalence of chosen objects and is therefore biased [Pow11]. This can be problematic since one cannot easily see whether the output of a learner is random guessing just by looking at the F_1-measure/loss. This raises the question of whether there exist measures that are unbiased with respect to the prevalence. For the rating of medical tests Youden [You50] proposed the J statistic, which is now attributed to his name as "Youden's J statistic". When applied to the multi-class or multi-label setting, this measure is known as the *Informedness* [Pow11]. It is defined as recall + specificity − 1 or in terms of choices:

Informedness

$$\text{Informedness}(C, C') := \frac{|C \cap C'|}{|C|} + \frac{|\overline{C} \cap \overline{C'}|}{|\overline{C}|} - 1 \tag{4.8}$$

Here $\overline{C}$ denotes the complement of the choice set C with respect to the task Q, i. e., $\overline{C} := Q \setminus C$. The measure assumes values in the domain $[-1, 1]$ and is unbiased with respect to the prevalence [Pow11]. A value of 0 indicates that a learner is randomly guessing.

In the field of multi-class/label classification, the measure is more widely known as the *A-mean* (or AM for short) [Men+13], which is the informedness linearly rescaled to $[0, 1]$.[3] In other words, it is defined as the arithmetic mean of recall and specificity (hence the name):

A-mean

3: This measure is also known as *balanced accuracy* [Bro+10].

$$\text{AM}(C, C') := \frac{1}{2}\left(\frac{|C \cap C'|}{|C|} + \frac{|\overline{C} \cap \overline{C'}|}{|\overline{C}|}\right) = \frac{1}{2}(\text{Informedness}(C, C') + 1) \tag{4.9}$$

After the rescaling, a value of 0.5 again represents random guessing. For datasets with a balanced number of chosen and not chosen objects, the measure is simply the usual accuracy.

Area under the ROC Curve The ROC curve is a popular tool to compare the performance of binary classifiers, visualizing the trade-off between the true positive rate (TPR) and false positive rate (FPR) [Faw06; McC89]. To this end, we utilize the predicted scores of a classifier, which are real values indicating the likelihood of each instance belonging to the positive class. By varying the threshold at which these scores are deemed to indicate a positive classification, we can generate a series of TPR and FPR pairs, each corresponding to a different threshold value. Plotting these pairs produces the ROC curve (see Figure 4.8),

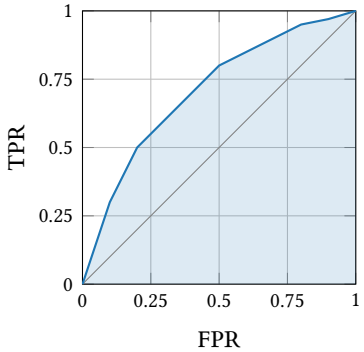


Figure 4.8.: The performance of a model depicted using a receiver operating characteristic (ROC) curve. The shading represents the area under the curve.

providing a visual representation of classifier performance across all possible thresholds.

Computing the area under the ROC curve (AUC) is a way to condense the complete curve into a single number in $[0, 1]$. A perfect classifier would be located at the point $(0, 1)$ where the FPR is 0 % and the TPR is 100 %, and its AUC is 1. If a model guesses randomly, its curve will be the diagonal, and the resulting AUC is 0.5.

When moving to the choice setting, similar to the F_1-measure/loss, we have to account for the fact again that there are multiple objects in each task for which we need to make a binary decision. Since the set of objects in our setting is variable from task to task, a straightforward instance-based averaging is sensible. For a given task, we would, therefore, calculate the AUC for each instance separately, and then average the AUCs:

$$\mathrm{AUC}_{\mathrm{inst}}(\mathrm{u}, \mathcal{D}) := \frac{1}{N} \sum_{i=1}^{N} \mathrm{AUC}\big(C_i, \mathrm{u}(Q_i)\big) \tag{4.10}$$

where u is the utility (or scoring) function of the model. We will adopt this averaging method for the AUC results presented later in this thesis.

The aforementioned measures are useful in comparing the performance of different learners across a variety of datasets. In addition, they can be used as the target measure used during HPO. However, since they are not differentiable, they cannot be used in a gradient-based optimization algorithm to fit a choice model. In the upcoming chapters we will see different approaches on how to work around this problem. That is, we employ surrogate losses based on a probabilistic model in Chapter 5 and a hinge surrogate loss based on constraints in Chapter 6.

HPO: Section 3.2

4.2.2. Ranking Measures

As this thesis focuses on learning from choice data, we will not present the ranking measures in detail here. It is, however, useful to give an overview of the most popular measures here since it will help contextualize the solution approaches in Section 4.3. Recall that in learning to rank, the ranking information may be given in different ways depending on the specific setting. One possibility is that we observe a complete order or permutation of the objects, which corresponds to the object ranking setting. Or we could observe objects with (ordered) relevance degrees assigned to them, as is common in instance ranking or ordinal regression. This specifies a partial order and is particularly common in IR. It is possible to convert a complete ranking into relevance degrees, which is necessary to compute some measures like the NDCG, as we will see in the following. However, the converse direction is not uniquely possible, and measures are required to handle ties.

The ranking measures can be roughly grouped into two main categories based on how they weigh different positions in the ranking. Measures that are agnostic to the positions are *rank correlation* measures such as Kendall's τ [Ken38] or Spearman's ρ [Spe04]. In IR, models usually need to identify the most relevant documents first. Therefore, one weighs the top positions much more highly than the rest. Popular measures in this direction are, for instance, the

NDCG, the mean average precision (MAP), and the mean reciprocal rank (MRR).

Position-Agnostic Ranking Measures Similar to the choice setting, we can also define a 0/1-loss that is 0 if and only if the complete predicted ranking matches the observed ranking. As there are $M!$ many permutations, this becomes exceedingly difficult for even moderate ranking sizes. A more fine-grained way to compare rankings is *rank correlation measures*, such as Kendall's τ [Ken38] and Spearman's ρ [Spe04]. Kendall's τ counts the number of concordant and discordant pairs of objects. A pair of object $\boldsymbol{x}_i, \boldsymbol{x}_j$ is considered concordant, if π and π' agree on their order. Otherwise, it is discordant. As a measure for correlation, it assumes values between -1 and 1, with the latter indicating a perfect match of both rankings and 0 no correlation. If Kendall's τ is negated and scaled to $[0, 1]$ it is known as the *ranking* 0/1*-loss* or simply ranking loss:

position-agnostic ranking measures

rank correlation measures

ranking 0/1-loss

$$L_{0/1}(\pi, \pi') \coloneqq \frac{2}{M(M-1)} \sum_{1 \leq i < j \leq M} [\![(\pi(i) - \pi(j))(\pi'(i) - \pi'(j)) < 0]\!] \,. \quad (4.11)$$

In practice, we often observe incomplete rankings due to the way data is collected (e. g., implicit feedback on top-K lists); therefore, it becomes necessary to be able to account for *ties*. Both for Kendall's τ and Spearman's ρ there are modifications that are able to handle ties.[4]

4: For τ refer to [Ken38; Ken45; Stu53; Cal+10; MB13; MB14; DFF23] and for ρ consider [AT15].

Top-Focused Ranking Measures As for ranking measures that are geared towards IR in that they weigh the top positions much more heavily, there are a variety of different methods. A general way to weigh the top positions more is only to compute a measure for the top-K objects. An example of this is the *precision at K*, which is the fraction of relevant objects that were among the K highest rated objects returned by the learner [MRS08]. This measure, however, ignores the order of the K best objects, which is why the MAP was introduced.[5] For a given task, it takes the ranking predicted by the learner and iteratively computes the precision of the top-i objects, starting with $i = 1$ up to $i = M$. Then, it averages only those precision values achieved for relevant objects (which yields the AP). Finally, these values are averaged across the complete dataset to arrive at the MAP.

top-focused ranking measures

5: Note, that this measure is *not* the average of the simple precision, a frequent source of confusion.

A very similar measure to MAP is the *normalized discounted cumulative gain (NDCG)* [JK00; JK02; Wan+13]. Roughly speaking, the *cumulative gain* sums up the relevance scores of the top-K objects predicted by the learner. The *discounted cumulative gain* reduces the impact of low-relevance objects by downweighing later positions in the predicted ranking π' and can be written as

normalized discounted cumulative gain (NDCG)

$$\mathrm{DCG}_K = \sum_{i=1}^{K} g(y_{\pi'^{-1}(i)}) d(i) \,,$$

where $y_{\pi'^{-1}(i)}$ is the relevance degree of the object the learner placed on position i and d is a monotone discount function and g is a monotonically increasing function. The typical discount function used in practice is $d(x) = \frac{1}{\log_2(1+x)}$. For g, both $g(x) = x$ and $g(x) = 2^x - 1$ are used in practice, where the latter gives even more emphasis towards highly relevant objects. Finally, the NDCG normalizes the discounted cumulative gain to assume values in $[0, 1]$.

reciprocal rank

Lastly, measures based on the *reciprocal rank* have been proposed [KV05; GS19]. We assume that we have a binarization of relevance, i. e., we have relevant and irrelevant objects. For each task Q_i, one records the rank r_i of the first relevant object in the predicted ranking. The *mean reciprocal rank* is then simply the average across all instances [KV05]:

mean reciprocal rank

$$\text{MRR} := \frac{1}{N} \sum_{i=1}^{N} \frac{1}{r_i} \tag{4.12}$$

where h is the ranking function produced by the learner, which gives rise to the ranks $\boldsymbol{r}$. Chapelle et al. [Cha+09] introduce the *expected reciprocal rank (ERR)*, which changes (4.12) to be a weighted average. Each reciprocal rank is weighted by the (estimated) probability that a user stops at that particular rank. This probability is proportional to the relevance degree of each object and the relevance degrees of the objects that came before it on the predicted ranking. In case the relevance degrees are binary, ERR coincides with MRR.

expected reciprocal rank (ERR)

4.3. Solution Approaches

While the distinction of the different preference settings in Section 4.1 is precise, there are some aspects that suggest that a different perspective may be worthwhile. Even though settings may differ, solution strategies might share similarities or be directly transferable. For example, we can convert an object ranking instance that consists of K objects that are ranked to an instance ranking problem, where each instance is assigned K different relevance scores. As another example, consider a choice problem instance where we observed that one object was chosen. This is a setting we often observe in practice since click data on search engines or e-commerce stores can be considered choice data and is routinely used to train ranking algorithms. By decomposing the choice into pairwise comparisons, where the chosen object is preferred over all unchosen objects, we obtain an object ranking problem. Therefore, it may be more useful to establish a universal nomenclature that applies to a variety of settings and more accurately characterizes the data used to train and what kind of model is learned.

[Liu11]: Liu (2011), *Learning to Rank for Information Retrieval*

Liu [Liu11] does so in their influential book. They argue that there are three fundamental approaches that can be distinguished: The *pointwise*, the *pairwise* and the *listwise* approach. In the pointwise approach, one learns a model that maps from a feature vector in $\mathcal{X}$ to a relevance degree in $\mathcal{Y}$. A pairwise approach learns a function that takes in two instances or objects and predicts the relative order. And finally, a listwise approach aims to infer a function that predicts a complete order of the instances/objects in question.

This distinction may appear sensible, but in practical applications, it can be less clear. This boils down to nuances in the definitions of what constitutes a pointwise, pairwise, or listwise approach. For example, we often see approaches in the literature where the data used to train a model is of a pairwise or listwise nature, but the model we actually train is pointwise. Consider RankNet, which utilizes pairwise training data and a pairwise loss function to train a pointwise latent utility model. This becomes problematic when the ground-truth function is inherently pairwise or listwise and cannot be accurately represented by

a pointwise function.[6] Therefore, we argue that approaches should not only be categorized based on their input data and training method but also based on the representational power of the model they infer.

6: We will later discuss *context-dependency*, a characteristic of such functions.

In the following section, we follow the standard delineation of these three categories, but will discuss it in more detail when a particular approach could also fit into a different category and explain the reasoning. We will focus on highly representative approaches and those used later during the experiments.

In Section 4.3.1, we begin by introducing pointwise approaches to which the typical classification and regression approaches belong, but also ordinal regression. Next, in Section 4.3.2, we move our attention to pairwise algorithms. We will introduce fundamental concepts and examine Rankboost, RankSVM, SortNet, and RankNet in more detail. Lastly, we explore listwise approaches in Section 4.3.3. Here, loss functions are particularly important, and we give an overview of their use. With ListNet, we introduce one particular approach in more detail. The remainder of this section follows the structure outlined in Liu [Liu11], but it provides a more detailed description of the methods utilized later in this thesis.

4.3.1. Pointwise Approaches

Pointwise approaches are very straightforward approaches to solve learning to rank problems. Whether an approach is classified as pointwise is usually decided based on the type of input data and the final prediction function that is produced. The most common data we encounter are instances (for example, documents in IR) that are assigned a relevance degree. The latter is typically given on an ordinal scale. Hence, formally we observe tuples $(\boldsymbol{x}_i, y_i)$ with $\boldsymbol{x} \in \mathcal{X}$, $y_i \in \mathcal{Y}$ and $|\mathcal{Y}| = M$. We assume that the relevance degrees are ordered, i. e., there exists a permutation π such that $y_{\pi(1)} \succ y_{\pi(2)} \succ \cdots \succ y_{\pi(M)}$.

Comment

With that, we could cover many preference learning settings simply by transforming the input data into a flattened representation using relevance degrees. To list a few examples: Assume we observe pairwise comparisons of instances, i. e., of the form $\boldsymbol{x}_i \succ \boldsymbol{x}_j$. Then, we can convert them using two relevance degrees: $(\boldsymbol{x}_i, y_1), (\boldsymbol{x}_j, y_2)$. As for rankings of a certain maximal size, they can be mapped to an equal number of relevance degrees. It should be apparent that these transformations are very crude. In the pairwise example, by transforming it into a pointwise representation, we essentially claim that $\boldsymbol{x}_i$ is universally relevant and $\boldsymbol{x}_j$ is universally irrelevant, even though we only have the very local information that $\boldsymbol{x}_i$ is more relevant in the presence of $\boldsymbol{x}_j$.

Pointwise approaches can roughly be categorized into regression, classification, and ordinal regression approaches. We will begin with the *regression approaches*, where the idea is to treat the relevance degrees as numeric values or to encode them (e. g., using a one-hot encoding) and then use standard regression approaches to solve the resulting regression problem. Fuhr [Fuh89] is one of the first to study regression-based approaches for ranking. They look at both

regression approaches

one-hot encoded and numeric regression targets. In their approach, the relevance degrees can also be assigned different costs that can be incorporated when computing the regression targets. In general, as a loss function, we can use any loss function typically used in regression, like the square loss $L(y, \hat{y}) = (y - \hat{y})^2$ (see Page 42). In the context of ordinal regression, Herbrich [Her99] introduce how the problem can efficiently be solved using a support vector machine. The key insight is to assume that the hyperplanes that separate the relevance degrees are parallel. This allows for a very efficient optimization algorithm because only the thresholds between the relevance degrees need to be learned. Cossock and Zhang [CZ06] investigate how well a regression-based approach based on a square loss would fare when evaluated based on the discounted cumulative gain (DCG). They show that the square loss is an upper bound for the DCG, i.e., asymptotically a regression-based learner is consistent for the NDCG.

Another way to define regression targets called *expected rank regression*, has been proposed by Kamishima et al. [KKA05]. This method is particularly useful in case rankings are of varying sizes, which is often the case in practice. The *rank* $\mathrm{r}(\boldsymbol{x}, \pi)$ specifies the position of object $\boldsymbol{x}$ in a ranking π. Say we observe the ranking $\pi = (3, 1, 2)$, then the rank of object $\boldsymbol{x}_3$ is 1. With the mild assumption that we selected the observed set of objects uniformly at random[7], the expected rank can be estimated using the formula [KKA05]:

7: This is also called the missing-at-random assumption.

$$\mathbb{E}[\mathrm{r}(\boldsymbol{x}, \pi^*) \mid \pi] \propto \frac{\mathrm{r}(\boldsymbol{x}, \pi)}{|Q| + 1}$$

where Q is the task for which we observed ranking π and π^* is the assumed ranking of all possible objects.

classification approaches

The next category of pointwise approaches is *classification approaches*. Here, one either reduces the preference learning problem to a binary problem, which simplifies the problem considerably, or one uses techniques from multi-class classification. The former methodology is more widely known as *binary decomposition*. The simplest binary decomposition separates objects into the two classes "relevant" and "not relevant". Several works have used this methodology over the years [CGD92; Gey94; Nal04].

With binary decompositions, we are, however, not limited to using only one learner. One idea is to split the original preference learning problem into several binary classification problems. Assume we have M many relevance classes ordered in increasing order of the relevance degree. The approach by Frank and Hall [FH01] will then train $M - 1$ learners. The first one predicts the probability that the true relevance class is above y_1, i.e., $P(y \succ y_1 \mid \boldsymbol{x})$ and analogously for $y_2, y_3, \ldots$. With that, it is straightforward to compute the probability of one specific relevance class y_m using one pair of learners:

$$P(y_m) = P(y \succ y_{m-1}) \cdot \big(1 - P(y \succ y_m)\big) \tag{4.13}$$

for $m \in [2, M-1]$. For $m = 1$ or M it suffices to query one learner. Since the difference in Equation 4.13 can become negative in practice, it may be useful to apply an additional $\max(\cdot, 0)$ to keep the probability positive.

8: Note that here we are *not* referring to a pairwise preference learning approach. Instead of pairs of objects, this method decomposes the relevance degrees into pairs.

In multi-class classification an influential decomposition method, *learning by pairwise comparisons*[8], was proposed by Fürnkranz [Für02]. A binary classifier is learned for each pair of labels (y_i, y_j) with $1 \leq i < j \leq M$. Then, for a new

instance $\boldsymbol{x}$, we evaluate the output of all learners and aggregate the outputs. The label with the highest aggregated score is predicted. This approach can straightforwardly also be applied to the problem of preference learning, as Fürnkranz et al. [FHV09] point out.

Generally, multi-class classification lends itself well for use in preference learning. Li et al. [LBW07] use multi-class classification to learn a probabilistic model for each relevance label y_m. These probabilities are then used to compute a weighted mean of monotonically transformed relevance degrees. Veloso et al. [Vel+08] employ a very similar weighted mean where they compute probabilities using normalized scores. The scores are obtained by aggregating association rules, i. e., the average confidence of the association rules predicting a certain relevance class is used as the score of that class.

ordinal regression approaches

Finally, an important category of pointwise approaches is *ordinal regression approaches*. These explicitly exploit the fact that a natural ordering of the relevance degrees exists. The idea is to learn thresholds $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_{M-1}$ that in conjunction with a scoring function $\mathrm{u} : \mathcal{X} \to \mathbb{R}$ are able to assign instances to relevance degrees. A seminal approach in this space called "PRanking" (or perceptron-based ranking) was proposed by Crammer and Singer [CS01]. The algorithm adapts a weight vector $\boldsymbol{\theta}$ of a linear model (i. e., $\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$) and the vector of thresholds $\boldsymbol{\theta}$ using a perceptron learning algorithm. Shashua and Levin [SL02] try to improve generalization by introducing large-margin principles. To this end, they specify two different strategies. The "fixed margin" strategy maximizes the margin of the closest relevance degrees, while the "sum of margins" strategy maximizes the overall sum of the $M-1$ margins. Rennie and Srebro [RS05] generalizes this approach by permitting varied margin loss functions and by also allowing one to penalize margins between relevance classes that are not neighboring.

In summary, pointwise approaches offer both merits and limitations. One key advantage is their simplicity; existing machine learning algorithms can easily be applied to transformations of the preference learning problem. This adaptability and the ability to process each instance independently make these algorithms highly scalable. However, these methods also have inherent drawbacks. Specifically, they overlook the relative ordering of data, thereby sacrificing a potentially valuable source of information. While the absolute score of an instance may not be crucial in practice, maintaining the correct order—particularly among the top-K instances—can be of significant importance. Moreover, in common IR scenarios where the number of results per query is very large, pointwise approaches may produce loss functions that are disproportionately influenced by an abundant number of irrelevant results. Consequently, we will move on to approaches that more directly solve the ranking problem, beginning with the category of pairwise approaches.

4.3.2. Pairwise Approaches

As the name suggests, the pairwise approach to preference learning considers pairs of instances or objects and their relative order. We assume that the data consists of ordered pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{X}^2$, which express that $\boldsymbol{x}_i \succ \boldsymbol{x}_j$. The goal is to learn a hypothesis $h : \mathcal{X}^2 \to \{-1, 1\}$ that can accurately predict the order between two instances. From this, we can already glean the first problem: if

we care about a ranking of the instances, how does h help us? If we have a collection of predicted pairwise preferences, we cannot always find a (unique) ranking that satisfies all. A simple example is a preferential cycle, i. e., if we treat each pairwise preference as an edge in a graph, then there could be a cycle in the graph. A ranking is a strict order of the instances that can only exist if the preference graph is cycle-free.

We will see two main strategies to tackle this problem. The more widely used strategy is *latent-utility*, where a function $\mathrm{u} : \mathcal{X} \to \mathbb{R}$ is inferred from the pairwise comparisons. With this it is straightforward to compute a total order of all instances, simply by passing all instances into the utility function and sorting the resulting utilities. We will explore the various approaches of this strategy in Section 4.3.2.1. The other main strategy is to learn an actual pairwise order relation that receives two instances as input and outputs the order between them. In a second step, this pairwise relation is used to come up with an overall ranking, and we will describe the methodologies employed there and their tradeoffs in Section 4.3.2.2.

4.3.2.1. Latent-Utility Approaches

The main objective of latent-utility approaches for pairwise ranking is to decompose the final hypothesis h as follows: $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = f\big(\mathrm{u}(\boldsymbol{x}_i), \mathrm{u}(\boldsymbol{x}_j)\big)$ where $\mathrm{u} : \mathcal{X} \to \mathbb{R}$ is an (unobserved) utility function and $f\colon \mathbb{R}^2 \to \{-1, 1\}$ is a decision function. Often the decision function f is defined in terms of the utility difference $\mathrm{u}(\boldsymbol{x}_i) - \mathrm{u}(\boldsymbol{x}_j)$. Consequently, we have

$$h(\boldsymbol{x}_i, \boldsymbol{x}_j) = f\big(\mathrm{u}(\boldsymbol{x}_i) - \mathrm{u}(\boldsymbol{x}_j)\big)\,. \tag{4.14}$$

From this, we can already see how pairwise approaches using latent utility are essentially pointwise approaches in disguise. Why is that? Assume we have trained a latent-utility model u using pairwise comparisons. We could in principle pass in all instances $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ to obtain utilities $u_1, \ldots, u_N$. With those we can construct a dataset $\mathcal{D}' = \{(\boldsymbol{x}_i, u_i)\}_{i \in [N]}$. We, therefore, transformed the original dataset into a pointwise form. To be more precise, this could now be treated as a continuous instance ranking problem (see Section 4.1.2) or simply a regression problem.

This observation should not be interpreted as a critique of the approach; there are indeed contexts where it is fully justified. A discerning reader may recall a similar issue discussed earlier in Chapter 2. In that chapter, the focus was on identifying the necessary assumptions about preferences required for the existence of a utility function. In our current context, to put it simply, we can impose certain assumptions on the ground truth pairwise relation h. Under specific assumptions, a unique mapping to utilities is guaranteed to exist (typically up to some invariances).

Axioms of utility theory: Section 2.2.1

Several sets of assumptions can assure the existence of an underlying utility function. Interestingly, most of these assumptions can be shown to be equivalent to some form of *transitivity*. Therefore, if the ground truth pairwise relation is transitive (according to an appropriate definition), a latent-utility approach is well-motivated.

transitivity: Page 26

RankSVM As SVMs were and still are a popular model class, they were also applied to the preference learning problem. Joachims [Joa02] are the first to use it in a ranking context, while Herbrich et al. [HGO00] look at a similar approach in the context of ordinal regression two years prior. Both methods utilize a linear latent utility framework, expressed as $\mathrm{u}(\boldsymbol{x}) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}$. Yet, by employing the kernel trick, we are also able to learn nonlinear functions.

For the RankSVM, each instance (with the index i) of a dataset is broken into pairs of objects $\{\boldsymbol{x}_j^{(i)}, \boldsymbol{x}_k^{(i)}\}$ and the corresponding pairwise preference is denoted by $y_{j,k}^{(i)} \in \{-1, 1\}$. Assuming we observe complete rankings for each instance (as is the case in object ranking), the optimization problem of the SVM can be formalized as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \lambda \sum_{i=1}^{N} \sum_{j,k\,:\; y_{j,k}^{(i)}=1} \xi_{i,j,k} \\ \text{subject to} \quad & \boldsymbol{\theta}^T\left(\boldsymbol{x}_j^{(i)} - \boldsymbol{x}_k^{(i)}\right) \geq 1 - \xi_{i,j,k} \quad \text{if } y_{j,k}^{(i)} = 1, \\ & \xi_{i,j,k} \geq 0, \;\; \forall i = 1, \ldots, N \end{aligned}$$

With the minimization, we try to make both the parameters $\boldsymbol{\theta}$ and the slack variables ξ as small as possible. The former for regularization purposes and the latter to minimize the *hinge loss* (depicted in Figure 4.9) when used with the inequality constraints. It is a convex upper bound of the 0/1-loss and, therefore, easier to optimize. The parameter λ controls the tradeoff between both terms and is usually tuned on a hold-out set. As is typical for SVMs, the loss exhibits a *margin effect*, meaning that it penalizes misclassifications in a way that encourages the decision boundary to be as far as possible from any data point while still correctly classifying the training data. By maximizing this margin, we obtain the clearest separation between classes, and hence, the model has good generalization properties. The optimization problem above is specified for the setting where we have full rankings, but can straightforwardly be applied to partial rankings as well.

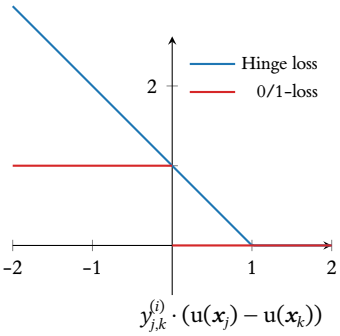


Figure 4.9.: Comparison of hinge loss and 0/1-loss as a function of the score difference $y_{j,k}^{(i)} \cdot (\mathrm{u}(\boldsymbol{x}_j) - \mathrm{u}(\boldsymbol{x}_k))$.

hinge loss

RankBoost RankBoost, as the name suggests, is a boosting algorithm designed for ranking proposed by Freund et al. [Fre+03]. It is an adaptation of AdaBoost (adaptive boosting), a well-known classification algorithm by the same authors [FS97]. The idea is to iteratively construct a weighted ensemble of weak rankers, each being trained on a reweighted dataset.

Algorithm 1 Learning algorithm for RankBoost [Fre+03].

Require: training data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \mid \text{if } \boldsymbol{x}_i \succ \boldsymbol{x}_j\}$
Ensure: Initialize distribution D_1 on $\mathcal{X}^2$.

1: **for** $t = 1$ to T **do**
2: Train weak ranker $f_t : \mathcal{X} \to \mathbb{R}$ based on distribution D_t.
3: Choose α_t based on an appropriate formula.
4: **for** $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathcal{D}$ **do**
5: Update $D_{t+1}(\boldsymbol{x}_j, \boldsymbol{x}_i) = \frac{1}{Z_t} D_t(\boldsymbol{x}_j, \boldsymbol{x}_i) \exp\left(\alpha_t(f_t(\boldsymbol{x}_j) - f_t(\boldsymbol{x}_i))\right)$
6: **return** $f(\cdot) = \sum_t \alpha_t f_t(\cdot)$

The algorithm is shown in Algorithm 1. As usual, it receives a dataset consisting

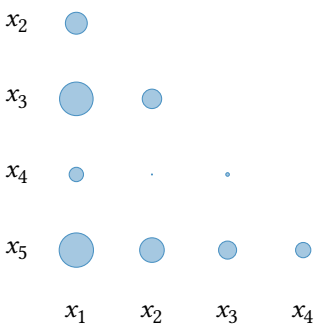


Figure 4.10.: Matrix representation of RankBoost weights D_t for instance pairs (x_i, x_j). Circle size indicates weight magnitude.

of pairs of instances $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. It maintains a probability distribution D_t over all pairs (see Figure 4.10). It is initialized using a uniform distribution over all observed pairs, i. e., we set the probability to 0 for all remaining pairs. In each iteration t a weak ranker f_t is trained using the distribution D_t over pairs as weights, i. e., the weak ranker should focus on getting pairs that have a high weight correctly. Each weak ranker is assigned a weight α_t that is used in the final weighted average. Freund et al. [Fre+03] detail several methods on how to compute α_t, which we will not show here. Then, in each iteration, the next distribution D_{t+1} is computed using an exponential weighting approach. The algorithm will, therefore, quickly focus on pairs that are hard to rank correctly.

RankNet Using pairwise comparisons to train a neural network is an approach that has been proposed as early as 1988 by Tesauro [Tes88]. It was later popularized in the year 2005 by Burges et al. [Bur+05] under the name *RankNet*.

The basic idea is to instantiate Equation 4.14 with u being a neural network and f being a nonlinearity like the sigmoid (mapping to $[0, 1]$)

$$\text{sigmoid}(x) := \frac{1}{1 + \exp(-x)},$$

the tanh function (mapping to $[-1, 1]$) or even a real-valued output. The advantage of using the sigmoid nonlinearity is that the output of the network can be interpreted as the probability that object $\boldsymbol{x}_i$ is preferred to object $\boldsymbol{x}_j$, i. e., $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = P(\boldsymbol{x}_i \succ \boldsymbol{x}_j)$. An example instantiation of the RankNet architecture

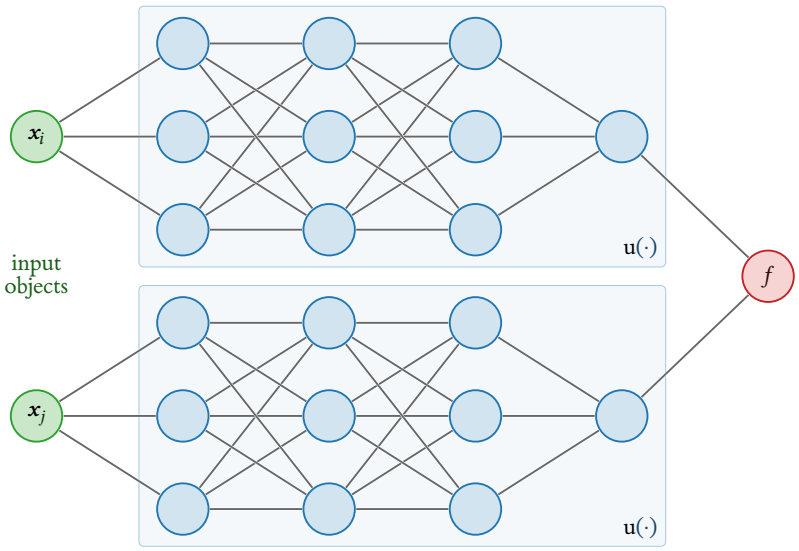


Figure 4.11.: Example instantiation of the RankNet architecture. The input objects of the network are green, the utility network u(·) is shown in blue, and the nonlinearity f is depicted in red.

is shown in Figure 4.11. Two objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, each represented by a feature vector, are separately passed into the network u(·). The number of hidden layers and units inside u(·) are hyperparameters subject to optimization. The output of the network (a real-valued scalar) is then fed into f. However, negative weight sharing is employed, i. e., if we have

$$h(\boldsymbol{x}_i, \boldsymbol{x}_j) = f\big(\theta_1 \mathrm{u}(\boldsymbol{x}_i) + \theta_2 \mathrm{u}(\boldsymbol{x}_j)\big)$$

then $\theta_1 = -\theta_2$ is enforced. Therefore, we have $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = f\big(\theta_1(\mathrm{u}(\boldsymbol{x}_i) - \mathrm{u}(\boldsymbol{x}_j))\big)$. And since the weight θ_1 can just be encoded in the previous layer, the expression

simplifies to $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = f\big(\mathrm{u}(\boldsymbol{x}_i) - \mathrm{u}(\boldsymbol{x}_j)\big)$ which is exactly Equation 4.14.

This yields a straightforward reduction to a classification problem and if we let f be the sigmoid, the usual classification loss functions like the binary cross-entropy loss (in case the nonlinearity is the sigmoid) can be used:

$$L_{\mathrm{CE}}(p, \hat{p}) = -p \log \hat{p} - (1 - p) \log(1 - \hat{p}) \tag{4.15}$$

where p is the ground-truth probability that a given object is ranked before another given object, and $\hat{p}$ is the probability predicted by the model f. The cross-entropy loss function is shown in Figure 4.12. As is apparent, it smoothly approaches infinity the closer the prediction is to the wrong outcome. The parameters of the neural network can then be trained using the classic backpropagation algorithm [RHW86]. The weight update can be factorized, which allows for a significant speedup in practice [Bur10].

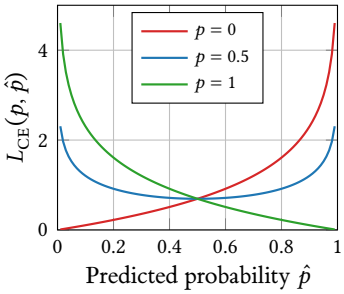


Figure 4.12.: The cross-entropy loss function $L_{\mathrm{CE}}(p, \hat{p})$.

backpropagation: Section 3.3.2

Other algorithms The main advantage of the pairwise latent-utility approaches is that they allow one to learn a pointwise model (i. e., the utility function) from pairwise examples. This is well-motivated in case the underlying pairwise preferences are transitive, which guarantees the existence of a utility function. Predicting a ranking is also straightforward since it suffices to sort the predicted utilities for a given list of objects. As we have seen in Section 2.3, there are situations where preferences are intransitive, often because the preference context influences the preferences observed. In those, it becomes necessary to learn a "true" pairwise function that cannot be reduced to a function of utilities (4.14).

4.3.2.2. True Pairwise Approaches

Recall that the goal with pairwise approaches was to model the function $h : \mathcal{X}^2 \to \{-1, 1\}$ from pairwise preference observations. Instead of decomposing this function, we will attempt to learn it directly or to learn a pairwise scoring function $\mathrm{u} : \mathcal{X}^2 \to [-1, 1]$. Unlike latent-utility approaches, which straightforwardly compute an overall ranking, pairwise scoring functions require an additional step. This step involves aggregating the predicted pairwise preferences into a consensus ranking. There are various methods to achieve this aggregation, each with its own set of tradeoffs.

CmpNN & SortNet Rigutini et al. [Rig+11] propose a pairwise neural network architecture called CmpNN, that is able to learn a certain pairwise preference relation. Such a pairwise preference relation is then used to produce a ranking by using it as a comparator in a sorting algorithm. To solve the problem of learning to rank, the authors employ active learning to produce a CmpNN model that performs well in terms of a global ranking measure (as opposed to only the pairwise performance).

The use of a pairwise relation in conjunction with a sorting algorithm is commonly done in the rank aggregation literature [ACN08]. In Example 4.3.1, we can see how a sorting algorithm can produce a ranked list of objects using a nontransitive comparator.

Example 4.3.1 (Sorting with a nontransitive comparator) Let us consider a sorting problem with four objects $\{A, B, C, D\}$. We have a comparator $\succ$ that produces the following nontransitive preferences: $A \succ B$, $B \succ C$, $C \succ A$, $A \succ D$, $B \succ D$, $C \succ D$. These are nontransitive since A, B, and C form a cycle.
We now run *merge sort* on the input list $[D, C, B, A]$. The algorithm will divide it into the lists $[D]$, $[C]$, $[B]$, $[A]$ and attempt to merge the first two and the last two. Since $C \succ D$ and $A \succ B$, we get the lists $[C, D]$ and $[A, B]$. Now, we merge these lists and compare C and A. The comparator yields $C \succ A$ and we put C as the first element of the final list. Then A follows due to $A \succ D$, and then B due to $B \succ D$ and finally D. Therefore, the resulting list is $[C, A, B, D]$. Another initial order of the objects could have resulted in another final list. However, object D would always be ranked last.

As the CmpNN architecture used to learn the comparator is central to the approach and will also be used later in Section 5.4.1, we are going to explain it in detail here. The network is learning a function $h : \mathcal{X}^2 \to \mathbb{R}^2$. For a given pair of objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the authors propose to use the target vector $(1, 0)$ in case $\boldsymbol{x}_i \succ \boldsymbol{x}_j$ and $(0, 1)$ otherwise. They call the first element of this vector $N_\succ$, the evidence for the first object being preferred, and analogously the second element $N_\prec$. Then, a simple loss function like the mean squared error (MSE) can be used to train the network.

permutation-equivariance: Definition 3.3.2

Until now, the two outputs are not coupled, but we would expect some kind of *permutation-equivariance*. That means it should hold that $h(\boldsymbol{x}_i, \boldsymbol{x}_j) = h(\boldsymbol{x}_j, \boldsymbol{x}_i)$. The authors achieve this by employing weight sharing. Let N be the shared pairwise neural network architecture with one output neuron. Then we pass in a given pair of objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ to obtain $N_\succ(\boldsymbol{x}_i, \boldsymbol{x}_j) := N(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $N_\prec(\boldsymbol{x}_i, \boldsymbol{x}_j) := N(\boldsymbol{x}_j, \boldsymbol{x}_i)$. Now it is clear that we have $N_\succ(\boldsymbol{x}_i, \boldsymbol{x}_j) = N(\boldsymbol{x}_i, \boldsymbol{x}_j) = N_\prec(\boldsymbol{x}_j, \boldsymbol{x}_i)$. It can then be shown that the network is *reflexive* (both outputs are exactly equal if we pass the same object in twice) and that $\succ$ and $\prec$ implemented by the network are *equivalent*. However, the architecture does not prescribe properties such as *transitivity*.

SortNet algorithm

With this comparator network, we are able to learn a pairwise ranking model. Since it can also represent nontransitive relations, it is not possible to use it directly for ranking. That is where the authors introduce the *SortNet algorithm*. The goal is to optimize a global ranking criterion by adaptively changing the training set available to the CmpNN training process. Initially, a CmpNN with random weights is created and used as a comparator to sort the elements of the complete training and validation set. All pairs that the model ordered incorrectly are added to the adaptive training and validation sets. In each iteration, the algorithm also evaluates a ranking criterion on the complete ranking produced on the validation set. If the current CmpNN model achieves a better global ranking score, it is saved and returned after the algorithm terminates. In each subsequent iteration, the CmpNN model is trained on the current adaptive training and validation sets, and the procedure terminates when these sets do not change anymore or if a maximum iteration is reached. The authors remark that one could also use the outputs of the CmpNN model as a score to compute an overall score for each object, which is similar to the idea we investigate in Section 5.3.1.

Other Pairwise Approaches One notable pairwise approach in document retrieval is called *duoBERT* and was proposed by Nogueira et al. [Nog+19]. In document retrieval, it is nowadays common to employ multi-stage retrieval, i. e., one starts with a fast pointwise model to filter out irrelevant documents to end up with a manageable number of documents to be ranked by more expressive models. In the mentioned paper, the authors go through 3 stages. In the first stage, a very fast model called BM25 [Rob+95] does the aforementioned preselection. Then, a more expressive pointwise model called monoBERT further reduces the amount of documents. Finally, the pairwise duoBERT model takes in the query and two documents and produces a pairwise probability. These are then aggregated using an aggregation function to determine a single score for each document, which is then used to rerank the list of documents. The researchers compare different aggregation functions (i. e., the sum of scores, the sum of binary scores, the minimum, and the maximum) and find that the sums perform best overall.

duoBERT

As we can see, the literature contains fewer true pairwise approaches than latent-utility ones. A big reason is that latent-utility approaches are easier to work with and produce a total order out of the box. A major consideration for pairwise approaches was to increase their focus on the top elements or, in general, to make them more aware of the complete global ranking they have to address. This raises the question of whether it is possible to tackle the ranking task directly, i. e., instead of single objects or pairs of objects, to look at the complete list of objects instead. This is what we will look at in the upcoming section.

4.3.3. Listwise Approaches

This section will follow the categorization introduced by Liu [Liu11]. Since most of the approaches proposed in the literature are latent-utility approaches, we will only have a small section about true listwise approaches at the end of this section.

4.3.3.1. Direct Optimization Approaches

Directly optimizing the target evaluation measure is not straightforward since most evaluation measures are neither differentiable nor smooth. This makes them hard to optimize using standard optimization algorithms. There are several ways to address this problem. A popular methodology is to approximate or bound the target measure and then use standard optimization approaches to optimize the approximate measures. The second solution is to use optimization algorithms that can handle non-differentiable and discontinuous functions.

Approximating the Target Measure SoftRank by Taylor et al. [Tay+08] treats the scores of each object as Gaussian probability distributions, which lets them define a probabilistic distribution of the ranks and subsequently a probabilistic version of the NDCG. The expectation of this soft NDCG can then be optimized using standard gradient descent. This methodology was later expanded by learning more of the parameters [GS08] or to be able to optimize other ranking metrics [YR10]. Zoeter et al. [Zoe+08] generalize this idea using a

Approximating the Target Measure

decision-theoretic framework. They define an expected utility, where the utility is a typical non-differentiable target measure, and the expectation is taken with respect to the probability distribution over scores.

Qin et al. [QLL10] and Chapelle and Wu [CW10] realize that the main problem with the non-differentiable target measures is that changing the scores of the objects slightly, does not result in a change of the target measure. Therefore, they propose smoothed versions of popular target measures. Their approaches differ in how the smooth version of the target measure is defined. In both cases, the smoothing requires the specification of a smoothing parameter that impacts the difficulty of the resulting optimization problem and also how well the smoothed measures approximate the true target measure.

Bounding the Target Measure

Bounding the Target Measure Yue et al. [Yue+07] introduce an SVM variant that is able to optimize the average precision (and in later works also NDCG and MRR) directly. Since the resulting optimization problem still requires an exponential number of constraints, the authors propose to minimize the maximum violation instead, which can be done efficiently. Xu et al. [Xu+08] derive the specific expression that is upper bounded by the aforementioned SVM formulation. In addition, they propose PermuRank, which incrementally updates two sets of permutations, those perfect ones that the model judges lowest and imperfect ones that it judges highest. In addition, a hinge-loss is used to upper-bound the expression they derived earlier.

non-smooth optimization

Non-Smooth Optimization The algorithm AdaRank, proposed by Xu and Li [XL07], applies the AdaBoost idea to the problem of learning to rank. The core idea is to take any target measure that operates on the scale $[-1, 1]$ and use it directly in the computation of the ensemble and instance weights. Since the algorithm does not require gradients or a smooth target measure in general, it is able to optimize an exponential version of the target measure directly and efficiently. The authors prove a lower bound for the target measure the algorithm achieves that depends on the measure achieved by the ensemble and the weak rankers in each iteration.

Metaheuristics are popular approaches applied to difficult-to-optimize functions. It is, therefore, no surprise that they were also used to optimize ranking target measures. To this end Yeh et al. [Yeh+07] create RankGP, a genetic programming method that evolves a (linear) ranking function based on primitive operations and constants. The target measure then acts as the fitness function for the genetic programming algorithm, and since it is a black-box optimization algorithm, any such measure can be used. The downside, as with any metaheuristic, is the considerable computational cost. This is because many generations of ranking functions have to be constructed and evaluated.

4.3.3.2. Probabilistic Approaches

Instead of approximately optimizing the target measure, another possibility is not optimizing for the target measure at all and instead fitting a probability distribution over rankings. Often times this results in quite a good performance in terms of the target measure.

ListNet A seminal work in this regard is the *ListNet* approach by Cao et al. [Cao+07]. Since we will use this approach as a baseline in the following chapters, we explain it in more detail here. The authors propose to use the following probability distribution on rankings

ListNet

$$P_{\boldsymbol{u}}(\pi) := \prod_{i=1}^{M} \frac{\phi(u_{\pi(i)})}{\sum_{j=i}^{M} \phi(u_{\pi(j)})}$$

which is exactly the well-known PL distribution [Pla75; Luc59]. The function $\phi : \mathbb{R} \to \mathbb{R}_+$ (usually an exponential function) is increasing and ensures that the result is a proper probability distribution. It is useful since it allows the model to output real-valued utilities. Instead of directly using the PL distribution as the likelihood for observed rankings, the authors aim to compute two distinct probability distributions: one resulting from the predicted scores and another from the observed ranking or relevance scores. Subsequently, the cross-entropy between these distributions can be utilized as a loss function.

Since the number of permutations is $M!$, Cao et al. [Cao+07] contend that such a computation could be intractable and suggest to use the top-1 probability

$$P_{\boldsymbol{u}}(i) := \frac{\phi(u_i)}{\sum_{j=1}^{M} \phi(u_j)}$$

instead. With $h_{\boldsymbol{\theta}}$ being the neural network parametrized by weight vector $\boldsymbol{\theta}$, the resulting cross-entropy loss for an instance $i \in [N]$ is

$$L(y_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) = -\sum_{j=1}^{M_i} P_{y_i}(j) \log\left(P_{h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)}(j)\right)$$

and can easily be optimized using gradient descent.

The neural network architecture is the same as that used in RankNet. The only difference is that RankNet is trained by pairwise comparisons and ListNet by rankings (a distinction that extends to the corresponding loss functions). Complexity-wise, RankNet runs in time quadratic in the maximum number of objects in a ranking, while ListNet only needs linear time.

RankNet: Page 98

One might contend that ListNet does not entirely represent a listwise approach, given its emphasis solely on the probability of the highest-ranked object. Nevertheless, ListNet does incorporate the entire set of objects and is trained on complete rankings. The focus on the highest-ranked object can be seen as advantageous, particularly since the accurate ordering of the top items is often more critical in practical applications. To this end, the authors note an empirical correlation between a decrease in top-1 cross-entropy loss and an increase in the target metric NDCG, a trend not observed in RankNet, which relies on pairwise loss.

Other Probabilistic Approaches Xia et al. [Xia+08] propose to use the negative log PL likelihood directly as a loss, instead of the cross-entropy loss. They show that this loss is the only one that is consistent (converges to the true distribution with infinite data) and sound (worse rankings always increase the loss, and better rankings decrease it). In addition, it can be optimized efficiently, with

its complexity being linear in the maximum number of objects in each ranking. They use the PL likelihood in a learning algorithm and call it *ListMLE*.

ListMLE

Generalizing RankNet, ListNet and ListMLE, Huang and Frey [HF08] apply the framework of cumulative distribution networks (CDNs) to the problem of ranking. It decomposes any partial order expressed on objects into pairwise preferences. Each pairwise preference is then a *node* in a CDN, which itself is an undirected graph. Such a CDN is constructed for each observed ranking. For the case where there are relevance labels are available for each object, Huang and Frey [HF08] propose a corresponding model called *StructRank*.[9] Kernel regression is used to model the utility function for each node. As RankNet, ListNet, and ListMLE can be represented using a specific CDN, and the framework can deal with both partial and full rankings, it is more general than the previously introduced methods.

StructRank

9: Note: "StructRank" here differs from the virtual screening algorithm of the same name, which is used in drug discovery to rank chemical compounds based on SVMs.

A similar idea to CDNs is the method *BoltzRank* proposed by Volkovs and Zemel [VZ09]. They apply the Boltzmann distribution to rankings, specifying a utility function consisting of an individual potential and a sum of pairwise potentials.[10] Precise calculation of this distribution is computationally demanding, leading to the researchers evaluating three approximation strategies. The strength of BoltzRank lies in generating a probability distribution over rankings, from which the expected target measure can be calculated. This expectation is differentiable, making it amenable to optimization using gradient-based methods. Consequently, BoltzRank can optimize various target measures, provided these measures have sufficient variance to generate a meaningful training signal.

BoltzRank

10: This utility function parallels the pairwise FETA strategy, which will be elaborated in Section 5.3.1. In this chapter, we will also elucidate its pros and cons.

Three Approaches to Modeling Context-Dependent Preferences

5. Utility-Based Preferences

In Chapter 2, we examined the classical axiomatic approach to modeling preferences, which begins by identifying the essential properties that real-world preferences should exhibit. The goal is to define a set of reasonable axioms that guarantee a unique assignment of utilities. Recall, for instance, the utility system proposed by Von Neumann and Morgenstern [VM47], where we assume the existence of a (deterministic) weak order relation on a set of alternatives. This order is closed under arbitrary mixtures of the alternatives. Additional axioms ensure that the mixture operation is well-behaved. Under these conditions, we can show that a unique assignment of utilities (up to positive linear transformations) exists for each alternative. The assigned utilities maintain the same weak order relation, and the utility of a mixture can be calculated using the convex combination of individual utilities.

Instead of pairwise relation, Luce [Luc59] defines axioms on the level of probabilistic choices[1]. If all choice probabilities are nonzero, then axiom (i) states that choice probabilities on supersets can be decomposed multiplicatively into choice probabilities on its subsets. While this restriction sounds innocuous, as we have seen in Section 2.2.2, it has major consequences. The most important consequence was the independence of irrelevant alternatives (IIA) property, where for any two alternatives, the ratio of their choice probabilities is constant across all sets containing them. It can also be shown that the axiom implies strong stochastic transitivity (SST).

1: Axiom 2.2.5

IIA: Lemma 2.2.6

SST: p. 26

There are several issues with this methodology. Firstly, the axioms may be violated in practice. As discussed in Section 2.3, studies with human subjects have identified various context effects where observed choice probabilities diverge from those predicted by classical choice models. For example, the "attraction effect" is a context effect where the presence of a third, less attractive object can influence the choice between two other objects. In classical utility models, the addition of an irrelevant alternative should not affect the preference between the original options. However, in reality, people often change their preferences due to the introduction of this decoy, violating the IIA property.

attraction effect: Page 30

Another source of violations may arise from the application area itself. Classical utility theory primarily focuses on human or rational decision makers, which is logical given its development within the fields of psychology and economics. In computer science, machine learning, and other algorithm- or data-driven fields, we often encounter problems that can be framed as preference problems. These include, among others, combinatorial optimization problems [Sch02], where the goal is to find an optimal solution fulfilling certain constraints, or structure prediction problems [Liu+23], where the goal is to rank structures based on costly to evaluate criteria.

Consider, for example, the well-known knapsack problem from combinatorial optimization shown in Figure 5.1, where a metaphorical burglar is confronted with a set of items $[n]$ they could take into their backpack. Each item $i \in [n]$ has a certain value v_i and a weight w_i, and the burglar wants to maximize the

Figure 5.1.: A burglar is confronted with a choice between different items to steal. Their goal is to maximize the total value of stolen goods while obeying the capacity of their knapsack.

sum of the values of the items they chose:

$$C^* = \arg\max_{C \subseteq [n]} \sum_{i \in C} v_i$$

The problem is that the backpack has a limited capacity W, and therefore, any solution C has to obey

$$\left(\sum_{i \in C} w_i \right) \leq W.$$

While the decision version of the knapsack problem is $\mathcal{NP}$-complete, there is no known polynomial-time algorithm, which can check if a given solution is optimal. At the same time, it is known that many problem instances encountered in practice can be solved efficiently [MT90; Pis05]. Pisinger [Pis05] compare the performance of several knapsack solvers on a set of benchmark instances and find that the best algorithms are able to solve all instances with a low running time. This prompted the authors to look for more difficult instances. There are many related problems to the knapsack problem, such as the bin-packing problem, the change-making problem, or the subset sum problem. We could frame this problem as a preference learning problem by considering sets of objects with their values and weights as features and the corresponding optimal solution as the choice sets. The goal here would be to learn a general model that can solve the knapsack problem for unseen combinations of objects.

It is immediately clear that for this problem, no constant assignment of utilities to items can exist such that the choices based on these utilities are always optimal. As an example, assume we have three objects $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ with values $(2, 3, 2)$ and weights $(0.5, 2, 1.5)$ and let the allowed capacity be 2. Let c^* be the choice function that selects an optimal subset of the objects to solve the knapsack problem. Then it is easy to see that $c^*(\{\boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{y}\}$, since $\boldsymbol{y}$ has a higher value of 3. But adding $\boldsymbol{z}$ to the set changes the choice set completely, as we have $c^*(\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}) = \{\boldsymbol{x}, \boldsymbol{z}\}$. The objects $\boldsymbol{x}$ and $\boldsymbol{z}$ can use the allowed capacity more efficiently and achieve a combined value of 4. If we were to try to define a utility function $u : \mathcal{X} \to \mathbb{R}$, which assigns a constant value to each individual object, we would run into trouble. From the first choice, we know that $\mathrm{u}(\boldsymbol{y}) > \mathrm{u}(\boldsymbol{x})$, since $\boldsymbol{y}$ was chosen. As for the second choice, we know that $\boldsymbol{y}$ was not chosen, while $\boldsymbol{x}$ was included in the choice set. Therefore, $\mathrm{u}(\boldsymbol{y}) < \mathrm{u}(\boldsymbol{x})$ which contradicts the first choice. Note that we have not yet specified how to construct a choice set from utilities. This is not necessary since we can see that

any meaningful construction will run into this contradiction.

As we have seen, when framing the knapsack problem as a choice problem, it is impossible to find a global assignment of utilities to objects such that optimal choice sets can be computed. Let us now take a look at typical assumptions on the level of choices and see if they hold for our running example. In Chapter 2 we introduced the relatively weak *regularity condition* which states that the probability of objects in a set is not allowed to increase if the size of the set is increased. The above example violates this condition, since object $\boldsymbol{x}$ starts with 0 % probability in the set $\{\boldsymbol{x}, \boldsymbol{y}\}$ and adding $\boldsymbol{z}$ increases the probability to 100 %. What about *transitivity*? Here, we can construct a counterexample with three objects as follows: Let the values of $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ be $(2, 4, 3)$ and their weights be $(2, 2, 3)$. The knapsack is allowed to hold a total weight of 4. We then have $c^*(\{\boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{x}, \boldsymbol{y}\}$, $c^*(\{\boldsymbol{y}, \boldsymbol{z}\}) = \{\boldsymbol{y}\}$ and $c^*(\{\boldsymbol{x}, \boldsymbol{z}\}) = \{\boldsymbol{z}\}$. For transitivity to hold, it would have been necessary for $\boldsymbol{x}$ to be in $c^*(\{\boldsymbol{x}, \boldsymbol{z}\})$; thus, it does not hold.

regularity condition: Page 25

transitivity: Page 26

We can see from the aforementioned example that there are choice functions that do not obey the classical choice and utility axioms. Now, the astute reader may ask if such choice functions are even common or realistic enough to warrant a move away from the classical axioms. If most of the preference data encountered in practice (approximately) satisfies the axioms, it would not be necessary to accommodate the few scenarios where richer preference models are required. Here, we argue for a shift in perspective. It is clear from experiments with human subjects (see Section 2.3) that context-based violations of the axioms are common. While there exist extensions of classical models (see Section 2.4.2), which are able to take some context effects into account, it would be desirable to have models that can adapt to the data at hand. This way, it is not necessary to know which specific context effects need to be modeled or if there even exists a specific model, which can model the choice function (e. g., in the case of the knapsack problem).

We will take on this perspective for the remainder of the thesis. We will introduce what we call a *generalized utility function*, which will not only depend on an object to assign a utility but also the complete object context, i. e., the complete set of objects present when the preference is expressed. With this change, we gain a substantial increase in representational power since we could, in principle, assign different utilities for each set of objects. Certainly, this increase in flexibility comes with its own problems. Assuming that we only have a finite number N of objects in total, we would need to learn an exponential number of parameters.[2] In addition, the preferences for each object set are independent, thus we cannot generalize at all to unseen object sets.

2: $\sum_{k=2}^{N} k\binom{N}{k} = \frac{1}{2}(2^N - 2)N$

We will, therefore, propose two complementary methodologies on how to represent such a generalized utility function. The first, called First Evaluate Then Aggregate (FETA) we introduce in Section 5.3.1, decomposes the generalized utility function into a sum of simpler utility functions, each of which captures the utility contribution of a certain subset. Similar to concepts like the Taylor series, we can omit contributions of larger subsets to make the computation more tractable. Our second approach (called First Aggregate Then Evaluate (FATE)) is described in Section 5.3.2. It first maps the complete set of objects to a representative vector. Then, each individual object is passed into a utility function, which receives both the object representation and the set

representation as input. In Chapter 6, we will introduce an approach that generalizes utility differently, by moving to multi-dimensional utility functions. For this chapter, generalized utility will refer only to context-dependent utility functions that are scalar. We will now introduce the concept of generalized utility formally and explain how it can be used to model a variety of preference modalities.

5.1. Generalized Utility

We can now introduce what we mean by *utility* in general. Regarding the necessary notation for this chapter, refer back to Section 2.1 in the preliminaries. The goal is to assign a real value to each object so that they can be compared based on this value. In measurement theory, different types of scales are distinguished[3], all resulting from different assumptions specified for the measurement process at hand. In this work, we consider utilities as fundamental elements from which preferences are derived through either deterministic or stochastic processes. A classical utility function can be expressed as a mapping $\mathcal{X} \to \mathbb{R}$, where each $\boldsymbol{x} \in \mathcal{X}$ is mapped to one constant value. As we remarked earlier, this is not flexible enough in the case of more contextual preferences. We, therefore, extend this function to also include the object context as additional input. A *generalized utility function* (for a preference structure $\mathbb{Q}$) is defined as

3: The typical scales are [Kir08]:

- Nominal (categorical, without order)
- Ordinal (rank order)
- Interval (allows comparisons of lengths, additive)
- Ratio (allows ratios)

generalized utility function

$$\mathrm{u} : \{(\boldsymbol{x}, Q) : \boldsymbol{x} \in Q \in \mathbb{Q}\} \to \mathbb{R}, \quad (\boldsymbol{x}, Q) \mapsto \mathrm{u}(\boldsymbol{x}, Q), \tag{5.1}$$

context-independent
context-dependent

We call a utility function *context-independent* in case $\mathrm{u}(\boldsymbol{x}, Q) = \mathrm{u}(\boldsymbol{x}, Q')$ holds for any $Q, Q' \in \mathbb{Q}$ with $\boldsymbol{x} \in Q \cap Q'$ and *context-dependent* otherwise. It is easy to see that this is a generalization since we can let $\mathrm{u}(\boldsymbol{x}) \coloneqq \mathrm{u}(\boldsymbol{x}, Q)$ for some arbitrary $Q \in \mathbb{Q}$ with $\boldsymbol{x} \in Q$ for a given context-independent utility function. McFadden et al. [MTT77] were the first authors who proposed to augment the utility computation of an object by the features of the other objects. They called this the *universal logit model*.

universal logit model

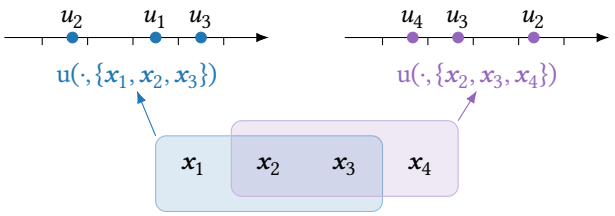


Figure 5.2.: Example how a generalized utility function can map the same objects to different utilities, based on the context.

In Figure 5.2, we see how a generalized utility function can map two overlapping sets of objects to entirely different utility values. The relative utility of $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ is different in the two contexts, meaning $\mathrm{u}(\boldsymbol{x}_2, \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}) < \mathrm{u}(\boldsymbol{x}_3, \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\})$ but $\mathrm{u}(\boldsymbol{x}_2, \{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\}) > \mathrm{u}(\boldsymbol{x}_3, \{\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4\})$. As we can see from this example, with generalized utility, there is no longer a clearly defined value assigned to each object. Instead, utility can only be defined in the context of a preference task Q. This flexibility comes with an obvious downside. There is no coupling between different tasks, which means that without further assumptions, we are not able to generalize to unseen tasks. As for seen tasks, the learning problem comes down to learning independent utility functions,

one for each preference task. The main strategy of the approaches introduced later on will be to decompose the general utility function into simpler parts such that generalization to unseen tasks is possible.

5.2. Generating Preferences from Utilities

In our model, the (context-dependent) utilities are the atomic building blocks from which the observed preferences are derived. For a given preference task $Q \in \mathbb{Q}$ the utility function $\mathrm{u}(\cdot, Q)$ assigns utilities to each object $\boldsymbol{x} \in Q$. In the next step, these utilities can be used by a *preference process* to generate preferences (see Figure 5.3 for an overview of the complete process). This process can be deterministic or probabilistic, and we will begin introducing the former next.

preference process

5.2.1. Deterministic Preferences

In the case where our preferences are choices, a simple deterministic choice process is the *singleton choice*

singleton choice

$$C_{\text{singleton}}(\mathrm{u}, Q) \coloneqq \arg\max_{\boldsymbol{x} \in Q} \mathrm{u}(\boldsymbol{x}, Q)\,. \tag{5.2}$$

Here, we require that the utility function does not produce ties (i. e., $\boldsymbol{x} \mapsto \mathrm{u}(\boldsymbol{x}, Q)$ is injective). Otherwise, the singleton choice function would return a subset instead of a single object. In the following comment box, we discuss the importance of injectivity in the wider context of social choice theory.

Comment

While injectivity seems like a minor technicality here, it can have major consequences in certain settings. Consider social choice theory, where the goal is to aggregate the preferences of several individuals into a collective decision. It is assumed that each individual has their own utility function mapping from $\mathcal{X}$ to $\mathbb{R}$. A *social choice rule* is then employed to choose one of the elements $\boldsymbol{x} \in \mathcal{X}$, such that a certain notion of collective utility is maximized. If such a social choice rule always selects exactly one element, it is called *resolute*.
The Gibbard-Satterthwaite theorem [Gib73; Sat75] assumes a resolute choice rule and shows that either one of these properties holds: (1) one of the individuals is a dictator (2) there can only be two outcomes ($|c(\mathcal{X})| = 2$) or (3) individuals may misrepresent their utilities to improve their outcome. As we can see, resoluteness is a rather strong restriction on the types of admissible social choice rules, and it is also central in several other impossibility results [KH90; OS21].

One way to model subset choices (and indeed the approach we will employ in this thesis) is to define a threshold $t \in \mathbb{R}$ such that

$$C^t_{\text{subset}}(\mathrm{u}, Q) \coloneqq \{\boldsymbol{x} \in Q : \mathrm{u}(\boldsymbol{x}, Q) \geq t\} \tag{5.3}$$

is a *subset choice* function. Although it is a simple mechanism, we will use it

subset choice

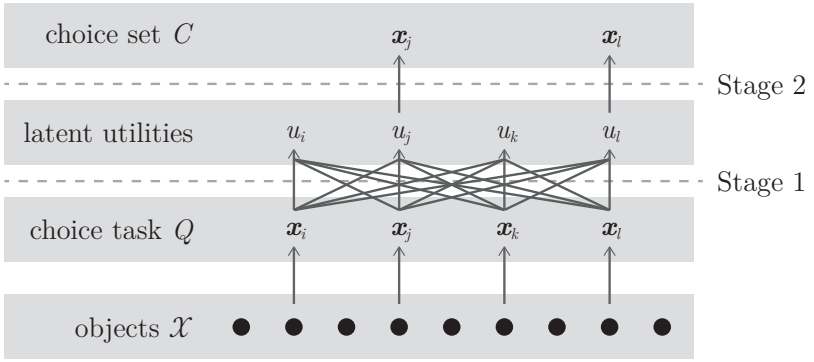


Figure 5.3.: Overview of the data-generating process: First, a task Q is sampled (with probability $p(Q)$) from $\mathcal{X}$. The objects in Q are assigned latent utility degrees and the choice set is finally constructed on the basis of these scores.

to convert utilities to choice sets in the approaches we develop in this chapter. It is also used extensively in related settings like multi-label classification [Koy+15] or multi-criteria sorting [AIM21]. Another possibility is to realize that singleton choice is a special case of the more general top-k choice. While this allows us to drop the threshold t as a parameter, the disadvantage is that the size of the subsets is then fixed to k, whereas thresholded subset choice can express all possible choice set sizes. Top-k choice is conceptually related to ranking, a connection that we will discuss in more detail in the next section.

5.2.2. Probabilistic Preferences

While deterministic preferences are useful to study representability or other related theoretical properties (see Section 2.2), they are less useful when considering real-world data. There are several ways in which real-world data can be probabilistic in nature. First, a decision maker could be uncertain when utilities are close in value, and if we were to query them at different points in time, we would get a different answer. Secondly, we could observe the preferences of multiple decision makers such that the stochasticity in their preferences can be regarded as deviations from the population utilities. Lastly, the preferences could be stochastic by design. An example could be that we model game-theoretic problems as a choice problem, with the objects being the rows of the payoff matrix. Typically, we consider the (not necessarily unique) Nash equilibrium to be the optimal solution for a given zero-sum game. In many cases, the Nash equilibrium is not deterministic but a stochastic strategy that randomizes between several strategies. In opponent modeling, we could then use the observed choices to infer the strategy of the opponent [NZ22].

Inference from probabilistic preferences necessitates the specification of a probability distribution $p(\cdot \mid Q)$, $Q \in \mathbb{Q}$ defined on $\mathcal{C}$, which depends on the utilities. An advantage of probabilistic models is that they allow us to use their likelihoods to infer the parameters of the model. While several mappings can relate utilities to probabilities, we will focus on a few canonical ones, which we will describe here and then use in subsequent sections.

singleton choice

Singleton Choice The setting of *singleton choice* is the most well-studied one in the literature and dates back to work done by Thurstone [Thu27]. A straightforward way to define probabilities is to assume that choices are made based on a multinomial distribution. To compute the probabilities for this distribution, we first apply an exponential transformation to the utilities, ensuring that they

are positive. Then the probabilities are simply defined to be the proportion of each transformed utility of the total sum of transformed utilities. The resulting model is called the *multinomial logit (MNL)* model [Ber44; GLD60; Cox66; Man66; The69]. Formally, we have the following definition

multinomial logit (MNL)

$$p_{\text{MNL}}(\boldsymbol{x} \mid Q) \coloneqq \frac{\exp(\mathrm{u}(\boldsymbol{x}, Q))}{\sum_{\boldsymbol{x}' \in Q} \exp(\mathrm{u}(\boldsymbol{x}', Q))} \,. \tag{5.4}$$

Additionally, we require that the probability of all non-singleton choice sets is 0, i.e., $p(C \mid Q) = 0$ for any $C \in 2^Q$ of size ≥ 2. The defining property of multinomial logit (MNL) models, is that for any $\boldsymbol{x}, \boldsymbol{y} \in Q$ we have that their log odds ratio satisfies:

$$\begin{aligned} \log \frac{p_{\text{MNL}}(\boldsymbol{x} \mid Q)}{p_{\text{MNL}}(\boldsymbol{y} \mid Q)} &= \log \frac{\frac{\exp(\mathrm{u}(\boldsymbol{x},Q))}{\sum_{\boldsymbol{x}' \in Q} \exp(\mathrm{u}(\boldsymbol{x}',Q))}}{\frac{\exp(\mathrm{u}(\boldsymbol{y},Q))}{\sum_{\boldsymbol{x}' \in Q} \exp(\mathrm{u}(\boldsymbol{x}',Q))}} = \log \frac{\exp(\mathrm{u}(\boldsymbol{x}, Q))}{\exp(\mathrm{u}(\boldsymbol{y}, Q))} \\ &= \mathrm{u}(\boldsymbol{x}, Q) - \mathrm{u}(\boldsymbol{y}, Q) \,. \end{aligned}$$

That means we can reason about the relative probability ratio of pairs of objects by simply looking at their utility difference without having to compute the complete probability vector. Relatedly, if we restrict the choice tasks Q to pairs of objects ($|Q| = 2$ for all $Q \in \mathbb{Q}$), we obtain the Bradley-Terry-Luce model [BT52], which is an important special case when studying contests with pairwise comparisons.

Subset Choice Moving on to *subset choices*, the first aspect to notice is that the number of possible subsets is exponential in the number of objects. We, therefore, want to avoid models that have to operate on that set explicitly. One example of such a model is the one proposed by Benson et al. [BKT18]. If we assume that objects are independently chosen based on their utilities for a given task Q, we arrive at a very simple model:

subset choices

$$p(C \mid Q) \coloneqq \gamma(\mathrm{u}, Q) \prod_{\boldsymbol{x} \in Q} \frac{\exp([\![\boldsymbol{x} \in C]\!]\mathrm{u}(\boldsymbol{x}, Q))}{1 + \exp(\mathrm{u}(\boldsymbol{x}, Q))} \,, \tag{5.5}$$

where $C \in 2^Q$ is a non-empty choice set and $Q \in \mathbb{Q}$ the given choice task. $\gamma(\mathrm{u}, Q)$ is a normalization constant that ensures that the sum over all possible choice sets $C \in 2^Q$ is exactly 1.[4] The choice of each object is determined independently. Therefore, if u is a context-independent utility function, the choice probabilities $p(C \mid Q)$ are context-independent as well. As we can see, the following ratio

4: $\sum_{C \in 2^Q \setminus \{\emptyset\}} p(C \mid Q) = 1$

$$\frac{p(C \mid Q)}{p(C' \mid Q)} = \prod_{\boldsymbol{x} \in Q} \frac{\exp([\![\boldsymbol{x} \in C]\!]\mathrm{u}(\boldsymbol{x}))}{\exp([\![\boldsymbol{x} \in C']\!]\mathrm{u}(\boldsymbol{x}))} = \prod_{\boldsymbol{x} \in C \cup C'} \frac{\exp([\![\boldsymbol{x} \in C]\!]\mathrm{u}(\boldsymbol{x}))}{\exp([\![\boldsymbol{x} \in C']\!]\mathrm{u}(\boldsymbol{x}))}$$

is completely independent of Q. This way of handling subset choice is similar to how multi-label classification is usually tackled [ZZ06; Nam+14]. The difference between these settings is that in multi-label classification, the set of labels remains constant across instances, and we do not have label features.

Choices based on Rankings While we will not pursue it further in this thesis, we still want to mention an interesting choice model based on rankings. Instead of directly going from utilities $u(\cdot, Q)$ to choices, we assume that we first sample a ranking π on Q first. Then a choice process $g : \pi \mapsto g(\pi) \in 2^Q$ selects elements based on the ranking. In general, we can express the overall choice probability as

$$p(C \mid Q) := \sum_{\pi} p(\pi)\ p(g(\pi) = C) \ , \tag{5.6}$$

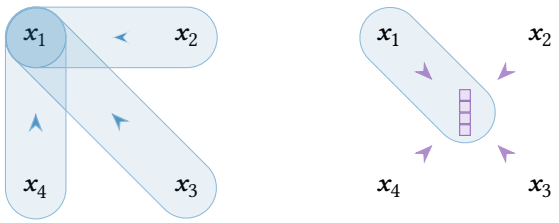
summing over all possible rankings π over Q. While this introduces additional complexity—we now have to model two probabilistic processes—it is of some interest since probability distributions on rankings have been investigated extensively in the literature. Ranking distributions can roughly be categorized into distance-based ranking models [FV86] and multistage ranking models [FV88]. The Mallows model [Mal57] is the most well-known representative of the former category. It is characterized by a central ranking, a distance function, and a dispersion parameter. As for multistage models, the prototypical example is the Plackett-Luce model [Pla75], where the probability of a ranking arises from repeated choices from an urn.

An important special case of the choice process g is *top-k choice*, where the first k objects are chosen deterministically. [5] The advantage is that it is conceptually simple. However, the drawback is that k is fixed across choice tasks, which may only be applicable for some settings (e. g., information retrieval). We can, of course, generalize this approach by assuming that the size k is not fixed but random. In this context, Fahandar et al. [FHC17] introduce a very general model that does not restrict choices to top-k sets.

5: $g(\pi) = \{\pi^{-1}(1), \dots, \pi^{-1}(k)\} \subseteq Q$ holds with probability 1 for any ranking π

To summarize, we have introduced the two-stage process of assigning utilities to objects and then generating preferences based on these utilities. We have seen different models that can be used to generate preferences from utilities, both deterministic and probabilistic. What remains is the need to find suitable decompositions for the generalized utility function such that generalization and inference are possible. This will be the focus of the next sections.

5.3. Decomposing Generalized Utility



(a) FETA (b) FATE

Figure 5.4.: Conceptual visualization of our two decomposition methods. For each method, we show a task consisting of 4 objects. Highlighted in blue is the computation of the utility for node x_1

In this section, we introduce and examine two approaches for decomposing the generalized utility function (5.1). In the previous section, we mentioned that directly modeling the generalized utility function requires one to model an exponential number of utility functions, one for each preference task. Additionally, since there is no coupling between these utility functions, we are unable to generalize to new preference tasks.

In Figure 5.4, we show a conceptual visualization of both decomposition methods applied to a task comprised of four objects: $Q = \{x_1, x_2, x_3, x_4\}$. The first approach, which we will introduce in Section 5.3.1 and which is illustrated in Figure 5.4a, decomposes the generalized utility function into a sum of lower-order sub-utility functions (here pairwise). For a given object $x_1 \in Q$, we first compute sub-utilities (shown in blue) and then aggregate them to form the overall utility of x_1. We refer to this approach as First Evaluate Then Aggregate (FETA).

The second approach, shown in Figure 5.4b, uses a representation function that maps the entire preference task Q to a vector, depicted in purple. Utilities for each object are then computed by evaluating the utility in the presence of this context vector (here shown in blue for x_1). Since the order is reversed here, i. e., we first aggregate all objects and only then compute utilities, we call this method First Aggregate Then Evaluate (FATE), with more details provided in Section 5.3.2.

In the following, we will introduce and explain both decompositions. In addition, we will show some interesting theoretical properties of both of these decompositions. More precisely, for FETA, we analyze the second-order case and show that it is identifiable up to the choice of the context-independent utility term. We will also show that the second-order FETA model is ultimately limited in expressiveness (i. e., there exist some choice functions, which it cannot represent). As for FATE, we prove that the model is flexible enough to represent any choice function. This flexibility, however, comes at the expense of identifiability, which is no longer satisfied for this model.

5.3.1. First Evaluate Then Aggregate (FETA)

As we have previously sketched, our goal will be to decompose the generalized utility function (5.1) into a sum of sub-utilities, evaluated on contexts of varying sizes. We will denote the context size by k. As an example, a context size of 2 would mean that the sub-utility of an object x is evaluated in the presence of 2 other objects. Formally, we will define *sub-utility functions* $u_0, \dots, u_K$ of the form $u_0 : \mathcal{X} \longrightarrow \mathbb{R}$ and

sub-utility functions

$$u_k : D_k \longrightarrow \mathbb{R}, \quad D_k := \{(x, A) : x \in \mathcal{X} \text{ and } A \subseteq \mathcal{X} \setminus \{x\} \text{ with } |A| = k\}$$

where $1 \leq k \leq K \leq |Q|$. To illustrate, assume we have $\mathcal{X} = \{x, y, z\}$, then $u_0(x) = 1$ would express that the latent utility of object x is 1. With $u_2(x, \{y, z\}) = -0.1$ we could express that the utility of x is decreased by 0.1, if objects y and z are present. Now we can represent (5.1) as an aggregation

$$u(x, Q) := u_0(x) + \sum_{k=1}^{K} \overline{u}_k(x, Q), \tag{5.7}$$

with $\overline{u}_k(x, Q)$ being the average over all sub-utilities $u_k(x, Q')$ for subsets Q' of $Q \setminus \{x\}$ consisting of k distinct elements:

$$\overline{u}_k(x, Q) = \frac{1}{\binom{|Q|}{k} - \binom{|Q|-1}{k-1}} \sum_{Q' \subseteq Q \setminus \{x\} : \, |Q'| = k} u_k(x, Q') \,.$$

It should be pointed out that the subsets Q' are not restricted to be valid preference tasks (i. e., $Q' \in \mathbb{Q}$). The sum is always taken with respect to all k-sized subsets Q' of $Q \setminus \{\boldsymbol{x}\}$ and should be considered a sum over all contributions to the utility the other objects have. Thus we can think of $\mathrm{u}_k(\boldsymbol{x}, Q')$ as a measure to which extent an object $\boldsymbol{x}$ is preferred if evaluated in the context of objects Q'. Then $\overline{\mathrm{u}}_k(\boldsymbol{x}, Q)$ is an indicator of how much $\boldsymbol{x}$ is on average valued when in the context of k distinct elements from $Q \setminus \{\boldsymbol{x}\}$. Finally, we call u defined in (5.7) the *FETA utility function with sub-utility functions* $\mathrm{u}_0, \dots, \mathrm{u}_K$ and denote it by $\mathrm{u}_{\mathrm{FETA}}^{\mathrm{u}_0,\dots,\mathrm{u}_K}$.

Recall that the main incentive of the decomposition is to make the computation of the generalized utility function (5.1) feasible. Replacing it with an exponential number of sub-utility functions, it may appear that we have not gained anything. The main motivation behind the above decomposition, however, is that we gain control over the order $K + 1$, up to which dependencies and interaction effects can occur. To gain insight into the concept of "order" in this context, examine the first-order model, which is represented by $K = 0$. This model simplifies to $\mathrm{u}_0(\boldsymbol{x})$ and, therefore, only accounts for the inherent utility of each object. In contrast, a second-order model, represented by $K = 1$, introduces pairwise terms. As a result, it can more closely approximate the generalized utility model because it considers the immediate relationships between objects in addition to their inherent utility. With control over the order of FETA, we are able to find the right level of flexibility for the preference data at hand. In Section 5.3.1.1, we will further investigate the properties of the second-order FETA model.

5.3.1.1. Pairwise Approximation of FETA

Following our earlier discussion, we know that the general FETA-decomposition (5.7) is overly flexible and computationally infeasible due to the exponential number of sub-utilities. Therefore, our goal is to restrict the order of the model to make it more tractable and to find a suitable inductive bias for typical preference data. It is reasonable to assume that most dependencies between objects are captured by lower-order interactions, with diminishing returns as the order of FETA increases.

For this reason, we study the special case $K = 1$, which can be seen as a second-order approximation to the more general FETA model (5.7) and the generalized utility function (5.1). Consequently, for a FETA model $\mathrm{u}_{\mathrm{FETA}}^{(\mathrm{u}_0,\mathrm{u}_1)}$ we need the sub-utility functions $\mathrm{u}_0 : \mathcal{X} \longrightarrow \mathbb{R}$ and $\mathrm{u}_1 : D_1 \longrightarrow \mathbb{R}$ and the FETA utility function then simplifies to

$$\mathrm{u}(\boldsymbol{x}, Q) = \mathrm{u}_0(\boldsymbol{x}) + \frac{1}{|Q| - 1} \sum\nolimits_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} \mathrm{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \,. \tag{5.8}$$

The value $\mathrm{u}_0(\boldsymbol{x})$ can be seen as a kind of inherent, context-independent utility of $\boldsymbol{x}$, whereas the scores $\mathrm{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\})$, $\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}$, serve as "corrections" of this utility in the context of the task Q. To see how this simple pairwise decomposition can achieve a context-dependent utility function, consider Example 5.3.1. As previously mentioned, the context-dependent random utility model (CDM) proposed by Seshadri et al. [SPU19] is a similar approximation, but instead of averaging the task context, the authors sum up all utilities and impose sum-to-zero constraints to guarantee identifiability.

Example 5.3.1 (FETA: Context-dependence) As a simple illustration, suppose $\mathcal{X}$ to consist of 4 elements $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$, let $\mathcal{Q} = 2^{\mathcal{X}} \setminus \{\emptyset\}$ and u be the FETA utility function with sub-utility functions u_0, u_1 defined as follows:

	$u_0(\cdot)$	$u_1(\cdot, \boldsymbol{a})$	$u_1(\cdot, \boldsymbol{b})$	$u_1(\cdot, \boldsymbol{c})$	$u_1(\cdot, \boldsymbol{d})$
$\boldsymbol{a}$	-0.8	—	1.2	0.8	0.0
$\boldsymbol{b}$	-0.7	0.0	—	1.2	1.4
$\boldsymbol{c}$	-0.7	0.6	0.0	—	0.2
$\boldsymbol{d}$	-0.8	1.0	0.0	0.8	—

Then, the utilities for the tasks $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$ and $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{d}\}$ are given as

	$u(\cdot, \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\})$	$u(\cdot, \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{d}\})$
$\boldsymbol{a}$	0.2	-0.2
$\boldsymbol{b}$	-0.1	0.0
$\boldsymbol{c}$	-0.4	—
$\boldsymbol{d}$	—	-0.3

For the task $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$ the item $\boldsymbol{a}$ has a higher utility score than $\boldsymbol{b}$, whereas $\boldsymbol{b}$ is preferred over $\boldsymbol{a}$ for the task $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{d}\}$, i. e., the preference between $\boldsymbol{a}$ and $\boldsymbol{b}$ changes depending on whether the third item in the task set is $\boldsymbol{c}$ or $\boldsymbol{d}$.

We will now move on to analyzing interesting properties of the pairwise FETA model $u_{\text{FETA}}^{(u_0,u_1)}$. More precisely, we are going to look at the identifiability of the sub-utility functions and the expressivity of the approximation.

5.3.1.2. Theoretical Properties of the pairwise FETA model

identifiability

An important property of statistical models is *identifiability* [WP97]. It refers to the ability to determine the parameters of a statistical model from data uniquely. If identifiability is violated, it can lead to a number of problems, such as unreliable parameter estimates. As for the FETA model $u_{\text{FETA}}^{(u_0,u_1)}$, we will now see that it is identifiable up to the choice of u_0.

Proposition 5.3.2 *Suppose $|\mathcal{X}| \geq 4$ and $\mathcal{Q}$ to be such that for any distinct $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$ there is some $Q \in \mathcal{Q}$ with $\{\boldsymbol{x}, \boldsymbol{y}\} \subseteq Q \not\ni \boldsymbol{z}$. Let $u_0, \tilde{u}_0 : \mathcal{X} \to \mathbb{R}$ and $u_1, \tilde{u}_1 : D_1 \to \mathbb{R}$ be arbitrary. Then, we have $u_{\text{FETA}}^{(u_0,u_1)} = u_{\text{FETA}}^{(\tilde{u}_0,\tilde{u}_1)}$ if and only if*

$$\forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{y} \in \mathcal{X} \setminus \{\boldsymbol{x}\} : \; \tilde{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) = u_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \tilde{u}_0(\boldsymbol{x}) + u_0(\boldsymbol{x}).$$

Proof. The converse implication $\Leftarrow$ is clear, since

$$\tilde{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) + \tilde{u}_0(\boldsymbol{x}) = u_1(\boldsymbol{x}, \{\boldsymbol{y}\}) + u_0(\boldsymbol{x})$$

holds for all pairs of distinct pairs $\boldsymbol{x}$, $\boldsymbol{y} \in \mathcal{X}$. In order to prove the remaining implication $\Rightarrow$, suppose that $u_{\text{FETA}}^{(u_0,u_1)} = u_{\text{FETA}}^{(\tilde{u}_0,\tilde{u}_1)}$.

Claim 5.3.2.1 *For any distinct $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$ we have*

$$u_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - u_1(\boldsymbol{x}, \{\boldsymbol{z}\}) = \tilde{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \tilde{u}_1(\boldsymbol{x}, \{\boldsymbol{z}\}).$$

Proof. For arbitrary $Q, Q' \subseteq \mathcal{X}$ with $|Q| = |Q'|$, $\{x, y\} \subseteq Q \not\ni z$ and $\{x, z\} \subseteq Q' \not\ni y$ we have

$$u_{\text{FETA}}^{(u_0,u_1)}(x, Q) - u_{\text{FETA}}^{(u_0,u_1)}(x, Q') = \frac{1}{|Q| - 1} \left(u_1(x, \{y\}) - u_1(x, \{z\})\right).$$

Since this holds for arbitrary (u_0, u_1) (and thus also for $(\tilde{u}_0, \tilde{u}_1)$), Claim 5.3.2.1 follows. □

Now, let $x_0 \in \mathcal{X}$ be fixed for the moment and define $b : \mathcal{X} \to \mathbb{R}$ via

$$b(x) := \begin{cases} \tilde{u}_0(x) - u_0(x), & \text{if } x = x_0, \\ u_1(x, \{x_0\}) - \tilde{u}_1(x, \{x_0\}), & \text{if } x \neq x_0. \end{cases}$$

According to Claim 5.3.2.1 we have for any distinct $x, y \in \mathcal{X} \setminus \{x_0\}$ the identity

$$\tilde{u}_1(x, \{y\}) = u_1(x, \{y\}) - (u_1(x, \{x_0\}) - \tilde{u}_1(x, \{x_0\})) = u_1(x, \{y\}) - b(x).$$

Moreover, the definition of b assures that $\tilde{u}_1(x, \{x_0\}) = u_1(x, \{x_0\}) - b(x)$ holds for any $x \neq x_0$, i. e., b already fulfills

$$\forall x \in \mathcal{X} \setminus \{x_0\} : \ \forall y \in \mathcal{X} \setminus \{x\} : \ \tilde{u}_1(x, \{y\}) = u_1(x, \{y\}) - b(x). \tag{5.9}$$

For $x \in \mathcal{X} \setminus \{x_0\}$ we may choose a query set $Q \subseteq \mathcal{X} \setminus \{x_0\}$ and then (5.9) assures us

$$\begin{aligned} &\tilde{u}_0(x) - u_0(x) \\ &= u_{\text{FETA}}^{(\tilde{u}_0,\tilde{u}_1)}(x, Q) - u_{\text{FETA}}^{(u_0,u_1)}(x, Q) \\ &\quad + \frac{1}{|Q| - 1} \sum\nolimits_{y \in Q \setminus \{x\}} \left(u_1(x, \{y\}) - \tilde{u}_1(x, \{y\})\right) \\ &= \frac{1}{|Q| - 1} \sum\nolimits_{y \in Q \setminus \{x\}} \left(u_1(x, \{y\}) - \tilde{u}_1(x, \{y\})\right) \\ &= \frac{1}{|Q| - 1} \sum\nolimits_{y \in Q \setminus \{x\}} b(x) \\ &= b(x). \end{aligned}$$

Since $\tilde{u}_0(x_0) = u_0(x_0) + b(x_0)$ holds by definition of b, we thus have shown

$$\forall x \in \mathcal{X} : \ b(x) = \tilde{u}_0(x) - u_0(x). \tag{5.10}$$

With regard to (5.9) it remains to show

$$\forall y \in \mathcal{X} \setminus \{x_0\} : \ \tilde{u}_1(x_0, \{y\}) = u_1(x_0, y) - b(x_0). \tag{5.11}$$

For this, note that the same argumentation as before with x_0 replaced by some arbitrary $x_1 \in \mathcal{X} \setminus \{x_0\}$ shows us that b also fulfills (5.9) with x_0 replaced by x_1. In particular, (5.11) holds. Combining (5.9), (5.10) and (5.11) completes the proof. □

Intuitively, we proved that if we know that two models output the same utilities $u_{\text{FETA}}^{(u_0,u_1)} = u_{\text{FETA}}^{(\tilde{u}_0,\tilde{u}_1)}$, then we can find a constant $b(x)$ for each object $x \in \mathcal{X}$ that specifies the differences for u_0, $\tilde{u}_0$ and u_1, $\tilde{u}_1$. For example, if we are given u_0 and we want $\tilde{u}_0(x)$ to be 5 higher, we need to decrease $\tilde{u}_1(x, \cdot)$

by the same amount (when compared to $\mathrm{u}_1(\boldsymbol{x}, \cdot)$). Finally, we can restate Proposition 5.3.2 in terms of injectivity as follows:

Corollary 5.3.3 *Suppose $\mathcal{X}$ and $\mathcal{Q}$ are as in Proposition 5.3.2 and let $\mathrm{u}_0 : \mathcal{X} \to \mathbb{R}$ be fixed. Then, the mapping $\mathrm{u}_1 \mapsto \mathrm{u}_{\mathrm{FETA}}^{(\mathrm{u}_0, \mathrm{u}_1)}$ is injective.*

Expressiveness of pairwise FETA Another interesting theoretical question concerns the *expressiveness* of the FETA decomposition. We will take a closer look at choice functions of the form $c : \mathcal{Q} \to \mathcal{C}$ now. The space of all possible choice functions is vast, and this raises the question, if our FETA model is able to express them all. As we introduced in (5.2) and (5.3), we are able to create a choice function by selecting certain objects based on their utilities. For example, let c be an arbitrary choice function, then we may want to know if there exists a choice function $\hat{c}(\cdot) \coloneqq C^t_{\mathrm{subset}}(\mathrm{u}^{\mathrm{u}_0,\mathrm{u}_1}_{\mathrm{FETA}}, \cdot)$ which produces the same choices as c, i. e., $c(Q) = \hat{c}(Q)$ for all tasks $Q \in \mathcal{Q}$.

expressiveness

Given that the pairwise FETA model (5.8) does not incorporate higher-order interactions between objects as compared to the general FETA model (5.7). We would, therefore, hypothesize that its expressiveness is limited. The following result shows that the decomposition into pairwise utilities is a restriction on the number of choice functions we can represent. More specifically, we can construct a counterexample with 7 objects.

Proposition 5.3.4 *If $|\mathcal{X}| \geq 7$, not every singleton choice function on $\mathcal{X}$ can be expressed via the second-order FETA model. More precisely: For distinct $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x}', \boldsymbol{y}' \in \mathcal{X}$ there do not exist sub-utility functions $\mathrm{u}_0 : \mathcal{X} \to \mathbb{R}$, $\mathrm{u}_1 : D_1 \to \mathbb{R}$ and $t \in \mathbb{R}$ such that the choice function $c : \mathcal{Q} \to \mathcal{C}$ defined either via $c(\cdot) \coloneqq C_{\mathrm{singleton}}(\mathrm{u}^{\mathrm{u}_0,\mathrm{u}_1}_{\mathrm{FETA}}, \cdot)$ or via $c(\cdot) \coloneqq C^t_{\mathrm{subset}}(\mathrm{u}^{\mathrm{u}_0,\mathrm{u}_1}_{\mathrm{FETA}}, \cdot)$ fulfills*

$$c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{x}\}) = \{\boldsymbol{a}\}, \quad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{y}\}) = \{\boldsymbol{a}\}, \quad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{b}\}, \tag{5.12}$$

$$c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{x}'\}) = \{\boldsymbol{b}\}, \quad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{y}'\}) = \{\boldsymbol{b}\}, \quad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}', \boldsymbol{y}'\}) = \{\boldsymbol{a}\}. \tag{5.13}$$

Proof. We prove the statement indirectly. To this end, fix distinct $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{x}'$, $\boldsymbol{y}' \in \mathcal{X}$ and assume there were some $\mathrm{u}_0, \mathrm{u}_1$ and $t \in \mathbb{R}$ such that c defined either via $c(\cdot) \coloneqq C_{\mathrm{singleton}}(\mathrm{u}^{\mathrm{u}_0,\mathrm{u}_1}_{\mathrm{FETA}}, \cdot)$ or via $c(\cdot) \coloneqq C^t_{\mathrm{subset}}(\mathrm{u}^{\mathrm{u}_0,\mathrm{u}_1}_{\mathrm{FETA}}, \cdot)$ fulfills both (5.12) and (5.13). With the convenient abbreviations $u_{\boldsymbol{r}} \coloneqq \mathrm{u}_0(\boldsymbol{r})$ and $u_{\boldsymbol{r},\boldsymbol{s}} \coloneqq \mathrm{u}_1(\boldsymbol{r}, \{\boldsymbol{s}\})$, the following constraints for (5.8) immediately follow from (5.12):

$$\begin{aligned}
u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{c}} + u_{\boldsymbol{a},\boldsymbol{x}}\right) &> u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{c}} + u_{\boldsymbol{b},\boldsymbol{x}}\right), \\
u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{c}} + u_{\boldsymbol{a},\boldsymbol{y}}\right) &> u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{c}} + u_{\boldsymbol{b},\boldsymbol{y}}\right), \\
u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{x}} + u_{\boldsymbol{b},\boldsymbol{y}}\right) &> u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{x}} + u_{\boldsymbol{a},\boldsymbol{y}}\right).
\end{aligned}$$

Summing up the first two inequalities and then applying the third one yields

$$\begin{aligned} &2u_a + \frac{1}{3}\left(2u_{a,b} + 2u_{a,c} + u_{a,x} + u_{a,y}\right) \\ &> u_b + \frac{1}{3}\left(u_{b,a} + u_{b,x} + u_{b,y}\right) + u_b + \frac{1}{3}\left(u_{b,a} + 2u_{b,c}\right) \\ &> u_a + \frac{1}{3}\left(u_{a,b} + u_{a,x} + u_{a,y}\right) + u_b + \frac{1}{3}\left(u_{b,a} + 2u_{b,c}\right), \end{aligned}$$

from which we obtain via subtracting common terms

$$u_a + \frac{1}{3}\left(u_{a,b} + 2u_{a,c}\right) > u_b + \frac{1}{3}\left(u_{b,a} + 2u_{b,c}\right). \tag{5.14}$$

Exactly the same argumentation (with the roles of a and b interchanged and x resp. y replaced by x' resp. y') lets us infer from (5.13)

$$u_b + \frac{1}{3}\left(u_{b,a} + 2u_{b,c}\right) > u_a + \frac{1}{3}\left(u_{a,b} + 2u_{a,c}\right),$$

which contradicts (5.14). This completes the proof. □

It is important to note here that limited expressivity has several potential advantages. High expressivity, or the capacity of the underlying hypothesis space, can lead to overfitting and, therefore, poor generalization. However, even more importantly, a higher-order model can misrepresent the way humans express preferences. Humans rely on absolute evaluations and pairwise comparisons rather than considering complex, higher-order interactions [Tru22]. As the number of objects, $|Q|$, increases, expecting individuals to account for these interactions imposes a significant cognitive load. We also know from experiments that context effects are less pronounced with less deliberation time [Pet12; CC19]. Thus, a model that aligns with simpler, more intuitive human decision-making processes may be more effective and realistic.

Overall, we expect FETA to work well for all choice functions that (approximately) decompose into a pairwise relation between objects. If we want to increase the expressiveness of the model, we can increase the order $K + 1$ of the model but incur an increased computational burden due to the exponential growth of the number of contexts we have to evaluate. Naturally, this leads to the question of whether it is possible to incorporate more of the set-based context without ultimately increasing computational complexity. This question motivated our other decomposition called FATE.

5.3.1.3. Related Models

In the context of market share modeling, Batsell and Polking [BP85] propose a decomposition of the market share of products across subsets. Their idea is to use log-odds ratios

$$\beta_{xy}^{Q} = \log\left(\frac{p_Q(x)}{p_Q(y)}\right)$$

as their main building blocks and show that they can be decomposed into the sum

$$\beta_{xy}^{Q} = \sum_{Q' \subseteq Q \setminus \{x,y\}} \alpha_{xy}^{Q'}.$$

As an example, let $Q = \{x, y, z\}$, then $\beta_{xy}^{Q} = \alpha_{xy}^{\emptyset} + \alpha_{xy}^{\{z\}}$. Here $\alpha_{xy}^{\emptyset}$ can be interpreted as the base log-odds ratio between x and y, while $\alpha_{xy}^{\{z\}}$ measures how the log-odds ratio is adjusted, if z is present. Batsell and Polking further show that there exists a set of equivalent parameters of the form $s_x^{Q'}$, such that the probabilities can be expressed as

$$p_Q(x) = \frac{\prod_{Q' \subseteq Q} \exp\left(s_x^{Q'}\right)}{\sum_{y \in Q} \prod_{Q' \subseteq Q} \exp\left(s_y^{Q'}\right)} .$$

We can see that these terms $s_x^{Q'}$ are conceptually similar to our sub-utilities $u_k(x, Q')$, with the main difference being that they are equally weighted to produce the final probability while we do the aforementioned averaging for each context size.

Seshadri et al. [SPU19] then build on this general model to define their CDM. Similar to the model we will analyze in Section 5.3.1.1, it is a pairwise approximation to the general utility model. Both Batsell and Polking [BP85] and Seshadri et al. [SPU19] assume various sum-to-zero constraints to make the model identifiable. These are no longer an option when we assume $\mathcal{X}$ to be infinite or if we want to generalize to previously unseen objects.

5.3.2. First Aggregate Then Evaluate (FATE)

Recall that our overall goal is still to find suitable decompositions of the generalized utility function (5.1). Decomposing the function into an aggregation of sub-utility functions has been a fruitful endeavor, but we have seen that with increasing order, the computation quickly becomes infeasible. Ideally, we would have a utility function, which we only have to evaluate once for each object x in a given preference task Q, while still taking the task context into account[6]. As the title of this section suggests, the goal will be to first aggregate the task context, as given by Q, into a context vector (of constant size). The context vector can be thought of as a representation of the essential features of the complete input set Q. Then, each object $x \in Q$ is evaluated in the presence of the context feature to arrive at the final utility.

6: Since first-order FETA fulfills the former but does not take into account the task context.

The first ingredient we need is an embedding function ϕ, which maps from the object space $\mathcal{X}$ to an m-dimensional embedding space $\mathcal{Z} \subseteq \mathbb{R}$. The primary objective of this function is to ensure that the context vector will be of constant size m, regardless of the set size $|Q|$. To achieve this, we define the context vector to be the average of all embedding vectors, i.e., $z := \frac{1}{|Q|} \sum_{y \in Q} \phi(y)$. There are a few observations we can make here. Since the arithmetic mean is a commutative operation, the resulting context vector will always be the same, no matter the order in which we do the computation. This property, called *permutation-invariance*, will be important later on when we instantiate the model using neural networks, which are not able to deal with set-valued input directly. The final ingredient to the FATE-model is a context-dependent sub-utility function $u' : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$. With this, we decompose the utility as

$$u(x, Q) := u'\left(x, \frac{1}{|Q|} \sum_{y \in Q} \phi(y)\right) , \tag{5.15}$$

and call this the *FATE utility function with sub-utility function* u' *and transforma-*

FATE utility function

tion ϕ and denote it by $\mathrm{u}_{\mathrm{FATE}}^{\mathrm{u}',\phi}$.

Consider Example 5.3.5 for an example which demonstrates that FATE is a context-dependent model. As before, we want to now take a look at related models that were proposed in the literature.

Example 5.3.5 (FATE: Context-dependence) Similar to Example 5.3.1, suppose $\mathcal{X}$ is comprised of four elements $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$, let $\mathcal{Z} \coloneqq \mathbb{R}$ and $\phi\colon \mathcal{X} \to \mathcal{Z}$ and $\mathrm{u}'\colon \mathcal{Z} \to \mathbb{R}$ be such that

$\mathcal{X}$	$\phi(\cdot)$	$\mathrm{u}'(\cdot, 2)$	$\mathrm{u}'(\cdot, 3)$
$\boldsymbol{a}$	1	0.5	−0.1
$\boldsymbol{b}$	2	−0.1	0.5
$\boldsymbol{c}$	3	−0.2	−0.2
$\boldsymbol{d}$	6	−0.3	−0.3

and $\mathrm{u}'(\cdot, z)$ be arbitrary for any $z \in \mathbb{R} \setminus \{2, 3\}$. For $Q_1 \coloneqq \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$ and $Q_2 \coloneqq \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{d}\}$ the quantity $\frac{1}{|Q_i|}\sum_{\boldsymbol{y}\in Q_i}\phi(\boldsymbol{y})$ is 2 if $i = 1$ and 3 if $i = 2$. Consequently, we have $\mathrm{u}(\boldsymbol{a}, Q_1) = \mathrm{u}'(\boldsymbol{a}, 2) = 0.5 > -0.1 = \mathrm{u}'(\boldsymbol{b}, 2) = \mathrm{u}(\boldsymbol{b}, Q_1)$ and at the same time $\mathrm{u}(\boldsymbol{a}, Q_2) = \mathrm{u}'(\boldsymbol{a}, 3) < \mathrm{u}'(\boldsymbol{b}, 3) = \mathrm{u}(\boldsymbol{b}, Q_2)$, i. e., the preference between $\boldsymbol{a}$ and $\boldsymbol{b}$ changes depending on whether the third item in the set is $\boldsymbol{c}$ or $\boldsymbol{d}$.

5.3.2.1. Theoretical Properties of FATE

expressiveness of pairwise FETA: Page 121

For the pairwise FETA model, we analyzed expressiveness and showed that it cannot represent all possible choice functions (we were able to find a counterexample with 7 objects). Our FATE model appears to be limited in expressiveness as well: The objects of a given task Q are mapped into some vector space, and the average of these vectors acts as the context. Thus, we may expect this to be a bottleneck which prevents the model from being able to express all choice functions as well. However, it turns out that since we are free in our choices of ϕ and u', we are indeed able to do so, and it even suffices to use a one-dimensional function $\phi\colon \mathcal{X} \to \mathbb{R}$. The proof of the upcoming result is similar to the proof of Theorem 2 by Zaheer et al. [Zah+17].

Proposition 5.3.6 *Suppose $\mathcal{X}$ to be countable and $\mathcal{Q} \subseteq \{Q \subseteq \mathcal{X} : |Q| < \infty\}$. There exists a parametrization $\phi\colon \mathcal{X} \to \mathbb{R}$ with the following property:*

(i) For any singleton choice function c on $\mathcal{Q}$, there is a utility function $\mathrm{u}_c'\colon \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ such that $C_{\mathrm{singleton}}(\mathrm{u}_{\mathrm{FATE}}^{\mathrm{u}_c',\phi}, Q) = c(Q)$ holds for any $Q \in \mathcal{Q}$.

(ii) For any subset choice function c on $\mathcal{Q}$ there exists a utility function $\mathrm{u}_c'\colon \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ with $C_{\mathrm{subset}}^{1/2}(\mathrm{u}_{\mathrm{FATE}}^{\mathrm{u}_c',\phi}, Q) = c(Q)$ for any $Q \in \mathcal{Q}$.

Proof. Since $\mathcal{X}$ is countable, there exists an injective function $\delta\colon \mathcal{X} \to \mathbb{N}$. For $\boldsymbol{x} \in \mathcal{X}$ define

$$\phi(\boldsymbol{x}) \coloneqq \ln\left(p_{\delta(\boldsymbol{x})}\right),$$

wherein $p_i \in \mathbb{N}$ denotes the i-th prime number for any $i \in \mathbb{N}$. Before proving (i) and (ii), we show that the mapping

$$\Phi\colon \mathbb{Q} \to \mathbb{R}, \quad Q \mapsto \frac{1}{|Q|}\sum_{\boldsymbol{x}\in Q}\phi(\boldsymbol{x})$$

is injective. For this, let $Q, Q' \in \mathbb{Q}$ with $\Phi(Q) = \Phi(Q')$. Then,

$$\frac{|Q'|}{|Q|} = \frac{\ln\left(\prod_{x \in Q'} p_{\delta(x)}\right)}{\ln\left(\prod_{x \in Q} p_{\delta(x)}\right)} = \log_b(a)$$

holds for the integers $a \coloneqq \prod_{x \in Q'} p_{\delta(x)}$ and $b \coloneqq \prod_{x \in Q} p_{\delta(x)}$, i. e., $a^{|Q|} = b^{|Q'|}$. As a and b are both products of distinct primes, the uniqueness of the prime factorization lets us infer $a = b$ and thus also $Q = Q'$.

We proceed with proving (i) and (ii) simultaneously. For this, suppose any choice function c on $\mathbb{Q}$ to be fixed. Since Φ from above is injective, there exists a mapping $\Psi \colon \mathbb{R} \to \mathbb{Q}$ such that $\Psi(\frac{1}{|Q|} \sum_{x \in Q} \phi(x)) = Q$ holds for any $Q \in \mathcal{Q}$. Note, that $\Psi \restriction_{\Phi(\mathcal{Q})}$ is the inverse function of Φ. Thus, the claim follows with the choice

$$\mathrm{u}'_c(x, z) \coloneqq [\![x \in c(\Psi(z))]\!].$$

□

As we can see, with arbitrary functions ϕ and u', we can fully express any choice function since we are able to encode the full choice task Q without loss of information. Clearly, this expressiveness, while desirable in general, comes at a cost. With a very simple counterexample we can show that $\mathrm{u}_{\mathrm{FATE}}^{(\phi, \mathrm{u}')}$ without further assumptions is *not identifiable*: Suppose $\mathrm{u}' \colon \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ is of the form $\mathrm{u}'(x, z) \coloneqq f(x) + \|z\|_2$ for some function $f \colon \mathcal{X} \to \mathbb{R}$, where $\|\cdot\|_2$ denotes the standard euclidean norm in $\mathbb{R}^d \supseteq \mathcal{Z}$. For arbitrary $\phi_1 \colon \mathcal{X} \to \mathcal{Z}$, we obtain with $\phi_2 \coloneqq -\phi_1$ that

identifiability of FATE

$$\begin{aligned}
&\mathrm{u}_{\mathrm{FATE}}^{(\phi_1, \mathrm{u}')}(x, Q) - \mathrm{u}_{\mathrm{FATE}}^{(\phi_2, \mathrm{u}')}(x, Q) \\
&= \left\| \frac{1}{|Q|} \sum\nolimits_{y \in Q} \phi_1(x) \right\|_2 - \left\| -\frac{1}{|Q|} \sum\nolimits_{y \in Q} \phi_1(x) \right\|_2 = 0
\end{aligned}$$

for any $Q \in \mathcal{Q}, x \in Q \subseteq \mathcal{X}$, i. e., $\mathrm{u}_{\mathrm{FATE}}^{(\phi_1, \mathrm{u}')} = \mathrm{u}_{\mathrm{FATE}}^{(\phi_2, \mathrm{u}')}$ holds.

Now, in practice, we are certainly not using arbitrary functions ϕ and u' in FATE. These will be instantiated using functions of which the parameters can be inferred from given data like linear models or neural networks (see Section 5.4). This, in conjunction with stochasticity resulting from the inference process (e. g., stochastic gradient descent), results in an effective restriction on the expressiveness of the model. Therefore, it is reasonable to expect that one can achieve a good out-of-sample performance, even with such a flexible model.

5.3.2.2. Related Models

FATE is related to recent advances in dealing with set-valued inputs in neural networks [Zah+17; RSP17; Bat+18]. The methodology here is to construct a neural network in such a way that it is *permutation-equivariant* with respect to its inputs, i. e., if the input objects are permuted, the corresponding outputs are permuted in the same way. This can be used to learn mappings from sets to target scores directly. Rosenfeld et al. [ROS20] propose a method for learning set-dependent aggregation functions with an inductive bias towards principles

Section 3.3.5.3: encoding sets using neural networks

from behavioral choice theory. In contrast to more general models like Deep Sets [Zah+17], which try to approximate set functions using a permutation-invariant neural network, this approach aims to reduce the model's "violation capacity," or flexibility to change its choices when objects are removed from the choice task.

The FATE approach, on the other hand, first condenses the task context Q into a representative vector and only then scores each object. The resulting model has an inductive bias that favors functions for which the object utility depends on such a set-global reference object. This could be advantageous for datasets where the set of objects as a whole can be summarized by suitable *global* properties (e. g., choosing that element from a set, which is closest to the centroid of all elements in the set), such that the task to score the objects with this context becomes easy. FETA on the other hand, incorporates task-information through *local* interactions.

5.3.3. Linear Sub-Utility Functions

As linear models are popular in statistics and machine learning, we are interested in what happens if they are used in FATE and FETA.

FATE For FATE, the two functions we have to instantiate are the embedding function $\phi : \mathcal{X} \to \mathcal{Z}$ and the sub-utility function $\mathrm{u}' : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$. Doing so we have $\phi(\boldsymbol{x}) \coloneqq \boldsymbol{A}\boldsymbol{x}$ for some matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$, and $\mathrm{u}'(\boldsymbol{x}, \boldsymbol{z}) \coloneqq \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}^\mathsf{T}\boldsymbol{z}$ for any $(\boldsymbol{x}, \boldsymbol{z}) \in \mathcal{X} \times \mathcal{Z}$, $\boldsymbol{c} \in \mathbb{R}^d$ and $\boldsymbol{d} \in \mathbb{R}^m$. Inserting these into (5.15) we get

$$\mathrm{u}_{\mathrm{FATE}}^{\mathrm{u}',\phi}(\boldsymbol{x}, Q) = \boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t\left(\frac{1}{|Q|}\sum\nolimits_{\boldsymbol{y}\in Q}\boldsymbol{A}\boldsymbol{y}\right).$$

Note how the second summand does not depend on $\boldsymbol{x}$ for a given $Q \in \mathbb{Q}$. Therefore, we can infer that the term $\boldsymbol{c}^\mathsf{T}\boldsymbol{x}$ is the only part of the utility function having an influence on the relative order of the objects. Since the singleton choice $C_{\mathrm{singleton}}(\mathrm{u}_{\mathrm{FATE}}^{\mathrm{u}',\phi}, Q)$ is only affected by $\boldsymbol{c}^\mathsf{T}\boldsymbol{x}$, it is independent of Q. So we can only hope to have a context-dependent FATE model if at least one of u' and ϕ is nonlinear.

FETA We may be inclined to assume now that a linear version of FETA will have the same problem, and the resulting choice model is context-independent. However, we shall see that due to a small difference in how the averaging is done in FETA, we do get a nonlinear, context-dependent utility model. To this end, let u_0 and u_1 be linear sub-utility functions of a pairwise FETA model, i. e., $\mathrm{u}_0(\boldsymbol{x}) \coloneqq \boldsymbol{b}^\mathsf{T}\boldsymbol{x}$ and $\mathrm{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) = \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}^\mathsf{T}\boldsymbol{y}$ for any distinct pair of objects $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and weight vectors $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$. Inserting into (5.8), we obtain

$$\begin{aligned}\mathrm{u}(\boldsymbol{x}, Q) &= \boldsymbol{b}^t\boldsymbol{x} + \frac{1}{|Q|-1}\sum\nolimits_{\boldsymbol{y}\in Q\setminus\{\boldsymbol{x}\}}\left(\boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t\boldsymbol{y}\right)\\ &= (\boldsymbol{b}+\boldsymbol{c})^t\boldsymbol{x} + \frac{1}{|Q|-1}\sum\nolimits_{\boldsymbol{y}\in Q\setminus\{\boldsymbol{x}\}}\boldsymbol{d}^t\boldsymbol{y}\end{aligned}$$

for any object $\boldsymbol{x} \in Q$ and task $Q \in \mathbb{Q}$. This time, the sum in the second summand depends on $\boldsymbol{x}$ (it is excluded from the sum), which is why it is no

longer constant for a given Q. We can deduce that u can not be represented by a linear function.

These considerations are important because they inform us about which functions can be used to create context-dependent decompositions. So far, we have proposed decompositions without addressing the question of how to properly select the sub-utility functions in practice. In the following section, we will examine neural networks that can be utilized to implement the aforementioned models and algorithms for estimating their parameters.

5.4. Generalized Utility using Neural Networks

The goal in this section will be to instantiate the sub-utility functions which are part of the FETA and FATE models. To this end, we chose to use neural networks as our model class because they have the ability to learn any function, given enough data [HSW89]. This means that, in principle, we are able to get arbitrarily close to the ground truth sub-utility functions. Additionally, neural networks are a flexible model class, as their complexity can be controlled through the choice of network architecture and regularization. This allows us to use simple or complex models, depending on the needs of the problem at hand. Another benefit of using neural networks is that they are easy to train using gradient descent without the need to design a completely new training algorithm. In the following, we will define two neural network architectures called FETA-Net and FATE-Net, which implement the corresponding decomposition approaches we introduced earlier.

Our design goals for both neural networks are twofold. First, they should be end-to-end trainable using (stochastic) gradient descent, allowing for a more efficient and automated training process and enabling them to be integrated into larger network architectures. End-to-end trainability can help prevent overfitting and lead to better performance, as the entire network is optimized as a whole rather than relying on suboptimal pre-trained components. It also allows for more flexibility, enabling the entire network to be fine-tuned for new tasks rather than having to retrain pre-trained components. Consequently, the outputs of the networks should be differentiable almost everywhere with respect to the weights and the loss functions employed in conjunction with a regularization term for the weights should also be differentiable almost everywhere and convex with respect to the utilities. Second, the architectures should be able to generalize beyond the task sizes encountered in the training data, since, in practice, it is unreasonable to expect all choice tasks to be of the same size. Third, the architectures should be able to generalize beyond the task sizes encountered in the training data. This is because task sizes can vary greatly in practice, and a neural network that can only perform well on a specific task size may not be useful in real-world applications.

5.4.1. FETA-Net Architecture

We will begin with FETA and will implement the pairwise version as defined in (5.8) in Section 5.3.1.1. Recall that it was defined as follows:

$$\mathrm{u}(x, Q) = \mathrm{u}_0(x) + \frac{1}{|Q| - 1} \sum_{y \in Q \setminus \{x\}} \mathrm{u}_1(x, \{y\}) . \tag{5.16}$$

The two sub-utility functions we need to implement and subsequently learn are u_0 and u_1. We are going to instantiate those using feedforward neural networks. The resulting architecture we call FETA-Net, and it is depicted in

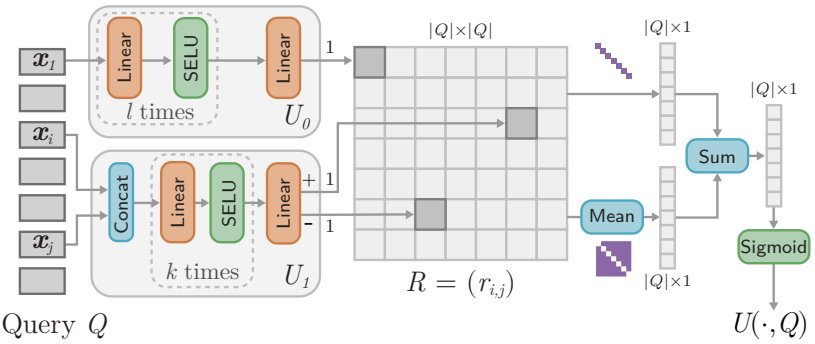


Figure 5.5.: The FETA-Net architecture implementing the FETA approach. Layers with trainable weights are shown in orange, while operations without trainable weights are drawn in blue, and nonlinearities are depicted in green.

Figure 5.5. While the architecture can be applied to a variety of preference learning tasks (e. g., ranking, singleton choice, and subset choice), the network shown here is suitable for subset choice problems, which we can see from the fact that the outputs are scaled using a sigmoid function.[7] For example, if we exchange the sigmoid function for a softmax function—ensuring that the utilities sum to 1—the architecture can be used for singleton choice instead.

7: A sigmoid function smoothly maps the real number line to the interval $[0, 1]$.

The figure depicts a prototypical forward pass through the FETA-Net architecture and we will begin with a high-level view. From left to right, we begin with a query preference task Q, which is passed into the networks u_0 and u_1. The resulting sub-utility scores will be used to construct the $|Q| \times |Q|$ matrix R, where the diagonal is filled using the outputs of u_0 and all off-diagonal elements result from u_1. We compute the mean of the off-diagonal elements along the second axis to obtain a $|Q| \times 1$ vector, which is then added to the diagonal to get the raw utility vector. Finally, an element-wise sigmoid function is applied to make the utility scores usable for the task of subset choice modeling.

We will now take a closer look at the two main networks u_0 and u_1. Beginning with u_0, we can see that it is a typical feedforward neural network using ℓ hidden layers, alternating linear layers, and nonlinear activation functions. Here, we use the self-normalizing linear unit (SELU) as proposed by Klambauer et al. [Kla+17], since it allows us to use deeper networks without having to resort to batch normalization [IS15]. However, any of the contemporary activation functions should work here. As for the pairwise utility function u_1, we also use a simple pairwise feedforward neural network but additionally employ a weight-sharing scheme in order to make the evaluation more efficient. More specifically, we use the CmpNN approach by Rigutini et al. [Rig+11], which allows us to compute both $\mathrm{u}_1(x, y)$ and $\mathrm{u}_1(y, x)$ in a single forward pass (see Page 99 for more details). To this end, we let the network output two sub-utility scores u_1^+ and u_1^-. Then, using *weight-sharing*[8] we ensure

8: Weight-sharing refers to the concept of using the same set of weights in multiple locations in a neural network.

that $\mathrm{u}_1^+(\boldsymbol{x},\boldsymbol{y}) = \mathrm{u}_1^-(\boldsymbol{y},\boldsymbol{x})$ and $\mathrm{u}_1^-(\boldsymbol{x},\boldsymbol{y}) = \mathrm{u}_1^+(\boldsymbol{y},\boldsymbol{x})$ hold. With that, it suffices to iterate over all combinations (with replacement) of objects once, and to construct the matrix R as follows:

$$r_{i,j} = \begin{cases} \mathrm{u}_1^+(\boldsymbol{x}_i,\boldsymbol{x}_j) & \text{if } i<j \\ \mathrm{u}_1^-(\boldsymbol{x}_i,\boldsymbol{x}_j) & \text{if } i>j \\ \mathrm{u}_0(\boldsymbol{x}_i) & \text{otherwise} \end{cases} \tag{5.17}$$

Then, each row of this relation R is averaged, and the diagonal is added to arrive at the final utility. Therefore we have $\mathrm{u}(\boldsymbol{x}_i, Q) = r_{i,i} + \frac{1}{|Q|-1}\sum_{1\le j\neq i\le |Q|} r_{i,j}$ for each object $\boldsymbol{x}_i \in Q$.

Algorithm 2 FETA-Net training algorithm

Require:
Dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N$ with $Q_i \subset \mathbb{R}^d$
Pairwise network $\mathrm{u}_1 : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^2$, parametrized by θ_1
Diagonal network $\mathrm{u}_0 : \mathbb{R}^d \longrightarrow \mathbb{R}$, parametrized by θ_0
Batch size $b \in \mathbb{N}$, Number of epochs $E \in \mathbb{N}$
Step size schedule $\boldsymbol{\eta} = (\eta_1, \eta_2, \dots)$ with $\eta_i \in \mathbb{R}_{>0}\ \forall i \in \mathbb{N}$
Loss function $L : \mathcal{C} \times \bigcup_{k\in\mathbb{N}} \mathbb{R}^k \longrightarrow \mathbb{R}$

1: **procedure** Train-FETA-Net($\mathcal{D}, \mathrm{u}_0, \mathrm{u}_1, b, E, \boldsymbol{\eta}, L$)
2: Initialize random weight vectors θ_0, θ_1
3: **for** Epoch $ep \in [E]$ **do**
4: $\mathcal{D} \leftarrow$ Shuffle($\mathcal{D}$)
5: $T \leftarrow \left\lceil \frac{N}{b} \right\rceil$
6: Construct mini-batches $\mathcal{B}_1, \dots, \mathcal{B}_T$
7: **for** Iteration $t \in [T]$ **do**
8: $\ell_t \leftarrow 0$
9: **for all** $(Q, C) \in \mathcal{B}_t$ **do**
10: **for** $1 \le i \le j \le |Q|$ **do**
11: **if** $i < j$ **then**
12: $r^{\mathrm{tmp}} \leftarrow \mathrm{u}_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$
13: $r_{i,j} \leftarrow r_0^{\mathrm{tmp}}, \quad r_{j,i} \leftarrow r_1^{\mathrm{tmp}}$
14: **else**
15: $r_{i,i} \leftarrow \mathrm{u}_0(\boldsymbol{x}_i)$
16: $\boldsymbol{u} \leftarrow \left(r_{i,i} + \frac{1}{|Q|-1}\sum_{1\le j\neq i\le |Q|} r_{i,j}\right)_{i=1}^{|Q|}$
17: $\ell_t \leftarrow \ell_t + L(C, \boldsymbol{u})$
18: $\theta_0 \leftarrow \theta_0 - \frac{\eta_{ep\cdot T+t}}{|\mathcal{B}_t|}\nabla_{\theta_0}\ell_t$
19: $\theta_1 \leftarrow \theta_1 - \frac{\eta_{ep\cdot T+t}}{|\mathcal{B}_t|}\nabla_{\theta_1}\ell_t$

Now that we have defined the neural network architecture, it is time to estimate its parameters from data. A standard training algorithm for this task is demonstrated in Algorithm 2. This algorithm uses stochastic gradient descent to find the best set of parameters for our FETA-Net model. Since we want to tackle a choice setting here, we will assume that there is a dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N$ consisting of choice tasks Q_i and the corresponding choice sets $C_i \in 2^{Q_i}\setminus\{\emptyset\}$. The weight vectors of the networks u_0 and u_1 will be represented by θ_0 and θ_1, respectively. To avoid problems with exploding or vanishing gradients, it is important that we initialize the weight vectors appropriately at the start of the training process [GB10; He+15]. Then for each epoch, the algorithm shuffles the given dataset and constructs mini-batches $\mathcal{B}_1, \dots, \mathcal{B}_T$ with $\mathcal{B}_i \subset \mathcal{D}$ for all iterations $i \in [T]$. In lines 10 to 15, the pairwise relation

is constructed as outlined above. The utilities $\boldsymbol{u} = (u_1, \ldots, u_{|Q|})$ of the objects within task Q are calculated in line 16 by summing the pairwise relation $r_{i,j}$ across the columns of the matrix. The loss is computed in line 17 and added to the cumulative loss of the batch. After the last batch has been processed, we compute the gradients of the loss with respect to the weight vectors θ_0 and θ_1 using backpropagation. The weight vectors are then updated in lines 18–19 using an iteration-dependent learning rate.

An important consideration in practice is the runtime complexity of a model, since it dictates the sizes of datasets that we can realistically handle.

The training runtime complexity per epoch of FETA-Net, including the computation required for backpropagation, is $\mathcal{O}\left(Ndq^2\right)$. Here, N is the number of training instances, d is the number of features per object, and q is an upper bound on the number of objects in each choice task, defined as $\max_{(Q,Y)\in\mathcal{D}} |Q|$. This means that the runtime complexity is linear in the number of instances and features but quadratic in the number of objects in each choice task. For a new task Q, the prediction time complexity is in $\mathcal{O}\left(d|Q|^2\right)$, therefore quadratic in the number of objects as well.

5.4.2. FATE-Net Architecture

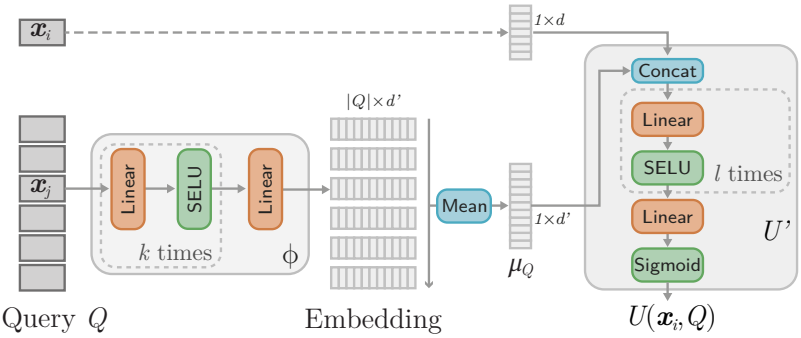


Figure 5.6.: The FATE-Net architecture that implements the FATE approach. Here, we show the score head for object $\boldsymbol{x}_i$. Layers with trainable weights are shown in orange, while operations without trainable weights are drawn in blue, and nonlinearities are depicted in green.

With FATE, the goal is first to embed each object into a m-dimensional embedding space. These embeddings are averaged to obtain a task representative μ_Q. The utility is then computed by a function that receives the object $\boldsymbol{x}_i$ and the task representative μ_Q. We will instantiate this approach using a neural network architecture called FATE-Net.

The architecture is shown in Figure 5.6, depicting the flow from input query Q to output utility $\mathrm{u}(\boldsymbol{x}_i, Q)$ for one object $\boldsymbol{x}_i \in Q$. Recall that we need to learn the embedding function ϕ and the utility function u' of the FATE decomposition (5.15). As before, the query task consists of n objects $Q = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. Each object is independently passed through a deep, densely connected embedding layer ϕ. Note that weight sharing is employed, meaning the same embedding is used for each object. We end up with $|Q|$ embedding vectors of length d'. These embeddings are averaged to arrive at the task representative μ_Q. Then the vector $\boldsymbol{x}_i$, of the object for which we want to compute the utility, is concatenated with μ_Q and both are passed into the network u', which finally outputs a utility $\mathrm{u}(\boldsymbol{x}_i, Q)$. Both networks ϕ and u' are feedforward neural networks with k and ℓ hidden layers respectively. As already mentioned in Section 5.4.1, we use SELU nonlinearities in the hidden layers here, but this should not be seen as a prescription.

Algorithm 3 FATE-Net training algorithm

Require:
Dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^{N}$ with $Q_i \subset \mathbb{R}^d$
Embedding network $\phi : \mathbb{R}^d \longrightarrow \mathbb{R}^{d'}$, parametrized by θ_ϕ
Utility network $\mathrm{u}' : \mathbb{R}^d \times \mathbb{R}^{d'} \longrightarrow \mathbb{R}$, parametrized by θ_{u}
Batch size $b \in \mathbb{N}$, Number of epochs $E \in \mathbb{N}$
Step size schedule $\boldsymbol{\eta} = (\eta_1, \eta_2, \dots)$ with $\eta_i \in \mathbb{R}_{>0}\ \forall i \in \mathbb{N}$
Loss function $L : \mathcal{C} \times \bigcup_{k \in \mathbb{N}} \mathbb{R}^k \longrightarrow \mathbb{R}$

1: **procedure** Train-FATE-Net($\mathcal{D}, \phi, \mathrm{u}', b, E, \boldsymbol{\eta}, L$)
2: Initialize random weight vectors $\theta_\phi, \theta_{\mathrm{u}'}$
3: **for** Epoch $ep \in [E]$ **do**
4: $\mathcal{D} \leftarrow \mathrm{Shuffle}(\mathcal{D})$
5: $T \leftarrow \left\lceil \frac{N}{b} \right\rceil$
6: Construct mini-batches $\mathcal{B}_1, \dots, \mathcal{B}_T$
7: **for** Iteration $t \in [T]$ **do**
8: $\ell_t \leftarrow 0$
9: **for all** $(Q, C) \in \mathcal{B}_t$ **do**
10: $\mu_Q \leftarrow \frac{1}{|Q|} \sum_{\boldsymbol{x} \in Q} \phi(\boldsymbol{x})$
11: $\boldsymbol{u} \leftarrow \left(\mathrm{u}'(\boldsymbol{x}, \mu_Q)\right)_{\boldsymbol{x} \in Q}$
12: $\ell_t \leftarrow \ell_t + L(C, \boldsymbol{u})$
13: $\theta_{\mathrm{u}'} \leftarrow \theta_{\mathrm{u}'} - \frac{\eta_{ep \cdot T + t}}{|\mathcal{B}_t|} \nabla_{\theta_{\mathrm{u}}} \ell_t$
14: $\theta_\phi \leftarrow \theta_\phi - \frac{\eta_{ep \cdot T + t}}{|\mathcal{B}_t|} \nabla_{\theta_\phi} \ell_t$

Moving on to the training algorithm (shown in Algorithm 3), similar to the algorithm used to train FETA-Net, we again have a variant of stochastic gradient descent here. We will refer to the weight vectors for our networks ϕ and u' as θ_ϕ and $\theta_{\mathrm{u}'}$, respectively. These weight vectors are initialized using a suitable initialization scheme, analogously to the procedure described in Section 5.4.1 (line 2). The shuffling of the dataset and mini-batch construction continues as before. In lines 10 to 12, the representative μ_Q is computed for the current mini-batch, the vector of utilities $\boldsymbol{u}$ is obtained and then the cumulative loss is updated with the loss on the current mini-batch. Once all mini-batches have been processed, the weight vectors $\theta_{\mathrm{u}'}$ and θ_ϕ are adjusted by computing the gradient of the cumulative loss with respect to both weight vectors and modifying both with an appropriate step size (lines 13 and 14).

The runtime complexity per epoch of the FATE-Net architecture is worth mentioning. Just looking at the training runtime complexity of FATE-Net, we still get $\mathcal{O}\left(Ndq^2\right)$ with N being the number of preference tasks in total, d being the number of features per object and q being an upper bound of the number of objects per task. The reason is that during backpropagation, the utility of each object depends on μ_Q and thus on each other object in Q, resulting in a quadratic complexity. During prediction time, however, it suffices to compute μ_Q once per task. Therefore, for a new task, the complexity reduces to $\mathcal{O}(dq)$, or time *linear* in the number of objects.

5.5. Empirical Evaluation

At this point, it makes sense to take stock and remind ourselves what we have accomplished so far. We set off by arguing that context-independent utility is too limited in modeling human and non-human preferences alike, which motivated the use of the *generalized utility function* (5.1). Since learning this function directly is not feasible, we proposed two decompositions (pairwise) FETA (5.8) and FATE (5.15), which make it more amenable to learning. Finally, we proposed two neural network architectures FETA-Net and FATE-Net as instantiations of these models. What remains now is to evaluate the efficacy of these models on actual preference data to ascertain if they are suitable for learning context-dependent preference functions. In this section, we will explore this question in more detail, focusing on the investigation of singleton and subset choice functions.

We will focus the evaluation on answering the following *research questions* (short RQ):

FF-RQ1: Are the decompositions FATE and FETA suitable for learning context-dependent choice functions?

FF-RQ2: How important is the complexity/expressiveness of the underlying model class for the performance of the models on the datasets?

FF-RQ3: How important is the ability of the model to represent context-dependent choice functions?

FF-RQ4: How well does our approach generalize to different task sizes? Can we make accurate predictions on tasks of a particular size, even if that size was not included in the training data?

FF-RQ5: What happens in the limit if we increase the amount of training data?

To address FF-RQ1, we will train and evaluate FETA-Net and FATE-Net on a range of datasets. These datasets will include synthetic and semi-synthetic datasets where we can ensure that context-dependence is crucial in accurately modeling the observed choices, as well as common real-world datasets. In addition, we will compare both approaches to existing models to see how competitive they are.

Research questions FF-RQ2 and FF-RQ3 deal with the expressiveness of our models in a practical setting. To elaborate on FF-RQ2, we want to know if deep neural networks (i. e., FATE-Net and FETA-Net) are really needed or whether simpler (e. g., linear) models would suffice. We will attempt to answer both questions by contrasting the performance of FETA-Net and FATE-Net with specific baselines. For FETA, we will include a linear variant in the experiments.[9] In addition, to answer FF-RQ3, we will have a context-independent neural network as a baseline, which will allow us to ascertain the importance of the context for each dataset as well.

9: Recall that for FATE we showed in Section 5.3.3 that a linear variant will be context-independent.

As for FF-RQ4, the methodology will be straightforward: We will train the models on a fixed task size and then evaluate them on a range of sizes during test time. If we notice a significant decrease in performance with only slight alterations to the task size, it suggests that the model has become dependent on the specific set size we provided. This is undesirable, as it indicates that the model is not generalizing well to new sizes.

Finally, for FF-RQ5, we will pick out one of the context-dependent semi-synthetic choice problems, such that we are able to generate an arbitrary amount of training data. We will then generate a stream of fresh batches to train our models with, simulating what happens if the dataset size approaches infinity. From this, we will able to deduce if the models are able to represent the underlying choice functions exactly, or if they approach a limit below 100 % accuracy.

We will start by discussing the experimental details in Section 5.5.1, including the general setup, baseline models, loss functions, and datasets. In Section 5.5.2, we will present the results of our experiments and use them to address the research questions we posed.

5.5.1. Experimental Setup

We will start by introducing the general experimental methodology. It is important that we compare the different models at close to their peak performance, meaning we want to avoid drawing incorrect conclusions because models were using subpar hyperparameters. Therefore, in all of the experiments (if not explicitly mentioned otherwise), we will apply hyperparameter optimization using nested validation with identical splits and number of iterations for each model. To be more precise, we use Bayesian optimization with Gaussian processes to optimize the hyperparameters. To accurately assess the performance of our model on unseen data, we will implement an additional outer cross-validation loop. Inside this loop, we will use the hyperparameters that performed best on the inner validation set and apply them to the training on the full training set. The model's performance will then be evaluated on the test set.

hyperparameter optimization: Section 3.2
nested validation: Section 3.1.2.3
Gaussian processes: Section 3.2.2.1

We conducted experiments on a compute cluster that included a mix of NVIDIA GTX 1080 Ti and RTX 2080 Ti GPUs, averaging 15-20 of each, as well as Intel Xeon E5-2670 processors. On average, a job comprising a single outer split with complete hyperparameter optimization on the validation set took 8 hours to complete. The training of FATE-Net and FETA-Net on average (across datasets) required 11 hours.

A job comprising a single outer split complete hyperparameter optimization on the validation set typically took 8 hours to complete. On average, training FATE-Net and FETA-Net across datasets required 11 hours. The combined total of all experiments was roughly 11 400 GPU hours and 6000 CPU hours. All experiments are implemented in Python, and the code and the dataset generators are publicly available.[10]

10: `https://github.com/kiudee/cs-ranking/tree/sda`

In the following, we motivate and present the set of baseline models we will use in the experiments in Section 5.5.1.1. Then, it is important that we connect the neural network architectures defined in Section 5.4 to the choice tasks at hand via appropriate loss functions. We will introduce these loss functions in Section 5.5.1.2. Specifically, we will distinguish between target losses, which are the true losses we seek to minimize, and surrogate losses, which are, in a loose sense of the word, approximations of the target losses that we can optimize during training. The datasets we consider for the experiments are introduced in Section 5.5.1.3.

5.5.1.1. Baseline Models

In order to answer research questions FF-RQ1, FF-RQ2 and FF-RQ3 in particular, we want to use a variety of different baseline models to compare to. This set should contain classical models like the MNL model and variations thereof to see how well these perform on the different context-dependent learning problems. Further, we also want to compare ourselves to neural network baselines from the literature. Since these models differ in their level of linearity and context-sensitivity, we can gain insight into which aspects the datasets require and where our approaches perform particularly well or not so well.

Random utility models (RUMs): Section 2.4.1

The first set of baselines we consider are the logit models, which belong to the class of random utility models (RUMs). The most well-known representative of this class is the multinomial logit (MNL) model [McF74]. The linear and context-independent nature of this model makes it an excellent choice for establishing a baseline against which to compare other models. Many improvements to the original model have been proposed in the literature, with the aim of making it more flexible and/or better able to incorporate contextual information. We choose to include the mixed logit (ML) [Tra09], the nested logit (NL) [Wil77] and the generalized nested logit (GNL) [WK01] models here. The former allows for more flexibility in modeling heterogeneous preferences in the data. The nested logit (NL) and the generalized nested logit (GNL) models seek to learn the correlations between the objects in a given set, which enables them to capture some context effects, particularly the similarity effect [BL18; Tve72].

Support vector machines (SVMs) have been used to model choice data since Evgeniou et al. [EBZ05] proposed to apply them in conjoint analysis in 2005 [MMW15]. Their robustness to noise is useful when dealing with human preference data. Cui and Curry [CC05] demonstrated in experiments that it outperforms the MNL model. We, therefore, choose to include it in our experiments. Since it is trained by breaking the choices down into pairwise comparisons, we call it PairwiseSVM.

We also include more recent neural network baselines in the experiments. The first one is the RankNet model, which is a context-independent network that is trained from pairwise comparisons [Tes88; Bur+05]. As a nonlinear yet still context-independent model, it can serve as a good baseline from which we can determine the maximum performance that can be achieved by context-independent models. To round the field off, we implement the set-dependent aggregation (SDA) approach, which was proposed by Rosenfeld et al. [ROS20] (see Page 83). This state-of-the-art context-dependent neural network has been specifically designed for use with human preference data.

It is important to note that most of the approaches mentioned earlier are designed for use in a singleton choice setting. We therefore employ thresholding of the utilities, as described in (5.3), to make all approaches compatible with the subset choice setting. The threshold is tuned on a small (nested) validation set for all these learners, which includes our own FETA-Net and FATE-Net.

5.5.1.2. Loss Functions

In Section 5.4, we introduced the neural network architectures FETA-Net and FATE-Net, but did not yet go into which loss functions we will use to train them. As loss functions provide the link between the raw outputs of a network and the data of a particular setting, it is important to specify our learning target. Our goal is to minimize a suitable *target loss function* $L : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ for the given data. This loss function is the one we actually want to optimize, like the F_1-measure in our case. Often, these target losses cannot be optimized directly using gradient descent because we require the loss function to be differentiable almost everywhere. This is why we will instead minimize a differentiable *surrogate loss function*. There are various ways to define such a function. One way we will use is to take the (log-)likelihoods as defined in Section 5.2.2. Another way is to define an upper bound on a target loss. We will discuss both methodologies in the following. At the end of this section, we will take a small detour to investigate the convexity of the surrogate losses, which is an important consideration when trying to optimize them.

target loss function

surrogate loss function

Target Loss Functions For the target loss functions or measures we refer the reader to Section 4.2.1, where we introduced the functions we use throughout the thesis. For this chapter, the main target loss we use for the singleton choice experiments is the categorical 0/1-loss. For the subset choice experiments, we will use the F_1-loss since it can deal with the imbalance of relevant to irrelevant objects that we observe in this setting. These losses are used to guide the hyperparameter optimization (HPO) process.

Choice measures: Section 4.2.1

Additionally, we compute supplementary measures for a comprehensive analysis. In the singleton choice context, we report both top-3 and top-5 accuracies. For subset choices, we include the subset 0/1-loss, informedness, and the area under the ROC curve (AUC). All results are presented such that higher values denote better performance, such as reporting categorical 0/1-accuracy instead of loss.

Surrogate Losses In Section 5.2.2, we introduced various ways one can define probabilistic models for different preference modeling settings. In doing so, we also specified the corresponding likelihood functions. This suggests a very natural approach to derive surrogate loss functions:

1. We use the negative log-likelihood of a probabilistic model as a loss function to train a learner. This has several immediate advantages: First of all, the resulting loss function is differentiable.[11] Secondly, the final model outputs probabilities on the choice space $\mathcal{C}$ and these are calibrated, i. e., the loss functions are minimal at the true (conditional) probabilities.
2. Given a new task Q, we can let the learner output a probability distribution on the choice space $\mathcal{C}$. With that, we can produce a choice set prediction that minimizes a desired target loss in expectation.

11: For the models we consider here.

Recall that the target loss functions received the ground truth choice set and the predicted choice set as input. For the surrogate loss functions defined here, instead of the latter we will pass in the raw utilities predicted by the learner. More formally, let $\mathrm{u}(\cdot, Q)$ be the vector of utility scores $\mathrm{u}(\boldsymbol{x}, Q), \forall \boldsymbol{x} \in Q$, which the learner predicts for a task $Q \in \mathbb{Q}$. Then for the singleton choice setting, we

can make the assumption that the choices are generated by a multinomial logit model, i.e., $p^{\tilde{u}}_{\text{MNL}}(x \mid Q) = p_{\text{MNL}}(x \mid Q)$. The utility function $\tilde{u}$ is assumed to be unknown. The probabilities can be computed using the likelihood (5.4). If we take the log of the expression and negate the formula, we get the negative log-likelihood, also called the *categorical cross-entropy loss*

categorical cross-entropy loss

$$\begin{aligned} L_{\text{CE}}(\{x\}, u(\cdot, Q)) &:= -\log\left(p^{u}_{\text{MNL}}(x \mid Q)\right) \\ &= \log\left(\sum\nolimits_{y \in Q} \exp(u(y, Q))\right) - u(x, Q)\,, \end{aligned} \tag{5.18}$$

where $C = x \in \mathcal{C}$ is the observed choice set. It is easy to see that this expression is minimized when $x = \arg\max_{y \in Q} u(y, Q)$.

Applying the same methodology in the subset choice setting, we start with the assumption that our choice data is generated using the probability distribution defined in (5.5) in Section 5.2, i.e., $p^{u}(C \mid Q) = p(C \mid Q)$. Computing the negative log-likelihood, we get the well-known *binary cross-entropy loss*

binary cross-entropy loss

$$\begin{aligned} L_{\text{BE}}(C, u(\cdot, Q)) &:= -\log\left(p^{u}(C \mid Q)\right) \\ &= \sum\nolimits_{y \in Q} \log\left(1 + \exp(u(y, Q))\right) - [\![y \in C]\!]\, u(y, Q)\,. \end{aligned} \tag{5.19}$$

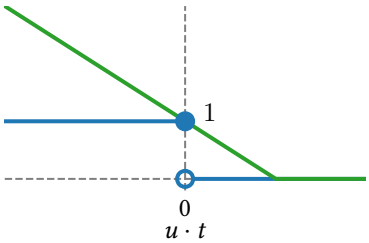


Figure 5.7.: The hinge loss (green) for one class. The 0/1-loss is shown in blue.

Another big class of surrogate loss functions are "hinge-variants" known from SVMs. There a 0/1-loss is upper bounded in score-space by a convex, piecewise linear function. We can see a visualization of the hinge loss $\max(1 - u \cdot t)$ for a one-class problem in Figure 5.7, where u is the score predicted for the class and $t \in \{-1, 1\}$ is the class. On the horizontal axis, we show the product of $u \cdot t$. Therefore, if the learner predicts a score that coincides with the correct sign, then the result is a 0/1-loss (shown in blue) of 0. The hinge loss (green) upper bounds the 0/1-loss. It is also apparent that it is non-zero in the interval $[0, 1)$, which leads to the "margin-effect" [BB00].

We will apply this idea in the singleton choice setting, where the corresponding hinge loss can be defined via

$$L_{\text{CH}}(\{x\}, u(\cdot, Q)) := \max\left(1 + \max\nolimits_{y \in Q \setminus \{x\}} u(y, Q) - u(x, Q), 0\right), \tag{5.20}$$

with $x \in Q \in \mathbb{Q}$. It is inspired by the hinge loss used in multi-class classification [DGI16; MD11] and more widely known under the name *categorical hinge loss*. While the surrogate losses we derived from likelihoods have favorable theoretical properties, the categorical hinge loss often works well in practice and yields stable results across training runs.

categorical hinge loss

To bring our discussion of surrogate loss functions to a close, we will outline the specific loss functions we will use in our experiments. During the training stage of FETA-Net and FATE-Net, we will use the *binary cross-entropy loss* (5.19) for the subset choice and the *categorical hinge loss* for the singleton choice setting. As is common in machine learning, L_2-regularization will additionally be employed to penalize the magnitude of the weights.

Convexity of the Surrogate Losses We have already mentioned convexity as an important property of the hinge loss. In general, this is because optimizing a convex optimization problem is guaranteed to yield a unique global optimum

[BV04]. In addition, it also improves the convergence speed and stability of the optimization process, and we do not incur the risk of getting stuck in a bad local optimum. Therefore, we will now check if our surrogate losses are convex with respect to the utility scores $\mathrm{u}(\boldsymbol{x}, Q)$. As we shall see in the following, all three surrogate losses are convex.

The categorical cross-entropy loss L_{CE} as defined in (5.18), can be written as

$$\log\left(\sum_{\boldsymbol{y}\in Q} \exp\big(\mathrm{u}(\boldsymbol{y}, Q) - \mathrm{u}(\boldsymbol{x}, Q)\big)\right),$$

where the difference of utilities is a linear function and clearly convex. What remains is the outer function, also known as *LogSumExp*. It is convex and strictly increasing in each argument, and therefore, it follows that the categorical cross-entropy (5.18) as a whole is convex as well.

$\mathrm{LSE}(\boldsymbol{x}) \coloneqq \log\left(\sum_{j\in[m]} \exp(x_j)\right)$

For the binary cross-entropy L_{BE} (5.19) we first look at the inner real-valued function $\log(1 + \exp(x))$, which is widely known as *softplus* [Dug+00]. It is smooth with strictly positive first and second derivatives, and hence, convex and non-decreasing. The difference $\mathrm{softplus}(x) - x$ is convex and strictly decreasing as well, which is why we can conclude that the binary cross-entropy (5.19) is convex.

$\mathrm{softplus}(x) \coloneqq \log(1 + \exp(x))$

The categorical hinge (5.20) consists of two max operations. The inner max is a maximum of convex functions and, therefore, convex. Then we have the outer maximum $\max(1 + x, 0)$, which is convex for the same reason, and the overall composition (5.20) is consequently convex.

Recall that the pairwise FETA model (5.8) is a decomposition of the utility $\mathrm{u}(\boldsymbol{x}, Q)$ into sub-utility functions, which are aggregated. We may, therefore, ask whether convexity still holds with respect to the sub-utility functions u_0 and u_1. It is easy to see that this is indeed the case since the aggregation in FETA is a simple weighted sum. We could then go one step further and ask if the complete learning problem of instantiations of FETA and FATE is convex, meaning that we want to know whether the loss minimization problem with respect to the parameters θ of the learner is convex. This, however, is not the case here since we use neural networks to instantiate our models, which are known to be non-convex. While we lose the guarantee that our optimization procedure will find a global optimum, this is not much of a problem in practice. The loss landscape, while being non-convex, is often remarkably smooth and well-behaved, as Li et al. [Li+18] demonstrate. They find that optimization trajectories of stochastic gradient descent optimizers often traverse a very low-dimensional space because the loss landscape of typical neural networks contains regions that are almost convex. In addition, we are using hyperparameter optimization to improve parameters related to the optimization procedure, which helps to make the process even more stable. With all of this in mind, we can expect our learning algorithms to find reasonable solutions in the upcoming experiments.

5.5.1.3. Datasets

We will use a mix of synthetic, semi-synthetic, and real-world datasets for the experiments. The (semi-)synthetic datasets allow us to evaluate the learners on problems where we know that the choices are context-dependent. This

Table 5.1.: Overview of the choice datasets used in the experiments. Bracket notation is used to denote the range of values.

Problem	Dataset	# Train	# Test	# Features	$\lvert Q \rvert$
Singleton Choice	Medoid	10 000	100 000	5	10
	Hypervolume	10 000	100 000	2	10
	MNIST-Mode	10 000	100 000	128	10
	MNIST-Unique	10 000	100 000	128	10
	Tag Genome Dissimilar Movie	10 000	100 000	1128	10
	Tag Genome Similar Movie	10 000	100 000	1128	10
	LETOR-MQ2007-list	[1353, 1356]	[336, 339]	46	[257, 1346]
	LETOR-MQ2008-list	[627, 628]	[156, 157]	46	[204, 1831]
	Expedia	78 041	312 229	17	[5, 38]
	Sushi	7000	3000	7	10
Subset Choice	Pareto-front-2D	10 000	100 000	2	30
	Pareto-front-5D	10 000	100 000	5	30
	MNIST-Mode	10 000	100 000	128	10
	MNIST-Unique	10 000	100 000	128	10
	LETOR-MQ2007	[1160, 1172]	[283, 295]	46	[6, 147]
	LETOR-MQ2008	[442, 459]	[105, 122]	46	[5, 121]
	Expedia	79 855	319 489	17	[5, 38]

gives us a good opportunity to check which learners are able to represent and learn the underlying choice functions. The semi-synthetic datasets are used on existing real-world datasets, but the choice process itself has been designed to be context-dependent. Finally, we have a set of real-world preference learning datasets that we adapt to our settings. We now introduce the learning problems used for the empirical comparison as follows:

Synthetic datasets:

(a) The Medoid problem, where the task is to predict the medoid of a set of points in a Euclidean space.
(b) The Pareto-front problem, in which the learner has to predict the set of points, which are Pareto-optimal.
(c) The Hypervolume singleton choice problem, where the task is to select the point of the Pareto-front which contributes the most to the hypervolume.

Semi-synthetic datasets:

(d) Different choice problems defined on the well-known MNIST dataset.
(e) Similarity/dissimilarity-based movie selection using the MovieLens Tag Genome dataset [VSR12].

Real-world datasets:

(f) The LEarning TO Rank (LETOR) MQ2007 and MQ2008 datasets [QL13] consisting of query-document pairs, with the goal of selecting the relevant documents.
(g) The Expedia hotel dataset featuring search results and relevance labels for each hotel with the goal to select booked/considered hotels [Exp13].
(h) The Sushi dataset, where the task is to choose the most preferred sushi from a set of 10 options provided to a user.

In Table 5.1 we group the datasets, including variants, by setting (some can only be used for singleton or subset choice, respectively). We report the number of training and testing instances, the number of features, and the task sizes for each dataset. Brackets are used to specify a range of values.

The Medoid Problem Finding the most representative element for a set is a task often encountered in practice. For example, when we apply k-means clustering to a dataset we obtain a set of k centroids, which usually are not part of the original set of elements. We may, therefore, be interested in selecting a single element of each cluster as its representative. One application among many is color quantization, where the goal is to reduce the number of distinct colors of an image while preserving overall similarity to the original image. The *medoid* of a set Q is the object $\boldsymbol{x} \in Q$ that has the smallest cumulative dissimilarity to all other objects within the same set. One advantage of the medoid over the centroid is that it can be computed for any structured space for which we can define a dissimilarity measure, e. g., graphs, images, etc. [VPB03; ZC05].

The centroid of a set Q is defined as:

$$\operatorname{centroid}(Q) = \frac{1}{Q} \sum_{\boldsymbol{x} \in Q} \boldsymbol{x}$$

For our experiments, we will treat the medoid computation as a choice function $c_{\text{medoid}} : \mathbb{Q} \to \mathcal{C}$ defined as

$$c_{\text{medoid}}(Q) \coloneqq \operatorname*{arg\,min}_{\boldsymbol{x} \in Q} \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{y} \in Q} \|\boldsymbol{x} - \boldsymbol{y}\|,$$

with $\|\cdot\|$ being the usual euclidean norm. Note, how c_{medoid} computes the norm for all pairs of objects. The resulting choice function is, therefore, context-dependent and small changes to the objects contained within a choice task can cause the choice set to change. This makes it a good choice problem to assess the capabilities of our learners.

$$\|z\| = \sqrt{z^t z}$$

In particular, we can even try to predict how well FETA and FATE should be able to learn the medoid problem. For FETA let $\mathrm{u}_0(\boldsymbol{x}) \coloneqq 0$ and $\mathrm{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \coloneqq -\|\boldsymbol{x} - \boldsymbol{y}\|$ then we can rewrite c_{medoid} as

$$\begin{aligned} c_{\text{medoid}}(Q) &= \operatorname*{arg\,min}_{\boldsymbol{x} \in Q} \frac{1}{|Q| - 1} \sum_{\boldsymbol{y} \in Q} \|\boldsymbol{x} - \boldsymbol{y}\| \\ &= \operatorname*{arg\,max}_{\boldsymbol{x} \in Q} \mathrm{u}_0(\boldsymbol{x}) + \frac{1}{|Q| - 1} \sum_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} \mathrm{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \end{aligned}$$

and it is easy to see that FETA can represent the medoid choice function exactly.

We can try the same for FATE, but there does not appear to be an obvious choice for ϕ and u' that, when combined, can perfectly model the medoid choice function. It is possible to approximate c_{medoid} using $\mathcal{Z} \coloneqq \mathcal{X}$, $\phi \coloneqq \mathrm{id}_{\mathcal{X}}$ and $\mathrm{u}'(\boldsymbol{x}, \boldsymbol{z}) \coloneqq -\|\boldsymbol{x} - \boldsymbol{z}\|$. Inserting into (5.15), we get

$$\mathrm{u}_{\text{FATE}}^{\mathrm{u}', \phi}(\boldsymbol{x}, Q) = -\|\boldsymbol{x} - \operatorname{centroid}(Q)\|\,,$$

with $\operatorname{centroid}(Q) \coloneqq \frac{1}{|Q|} \sum_{\boldsymbol{y} \in Q} \boldsymbol{y}$ being the centroid of Q. Instead of choosing the object that minimizes the pairwise distance with respect to every other object, we pick the object closest to the centroid. As these should often coincide, we can expect this approximation to be close.

To have a reference value for the experiments, we can assume that choice tasks Q are sampled from a uniform distribution ν on $\{A \subseteq [0,1]^d : |A| = r\}$ for a fixed $r \in \mathbb{N}$. With the above approximation, we can then achieve an accuracy of at least

$$P_{Q \sim \nu}\left(c_{\text{medoid}}(Q) = \operatorname{arg\,min}_{\boldsymbol{x} \in Q} \|\boldsymbol{x} - \operatorname{centroid}(Q)\|\right)$$

in expectation. Doing a numerical computation, we can find that for $r = 10$ and $d = 5$, we get a probability of 89.56 %. For more technical details on how we construct the medoid dataset, consult Appendix A.3.1.

Pareto-front dataset

The Pareto-Front Problem The concept of Pareto-optimality is central to multi-objective optimization and has a wide range of applications [Gei+07]. It is only relevant in settings where there are multiple objectives that need to be optimized because with only one objective, there exists a clear linear order by which we can sort the objects. As soon as we have multiple objectives, there can be situations where one object is better than another in one objective but worse in another.

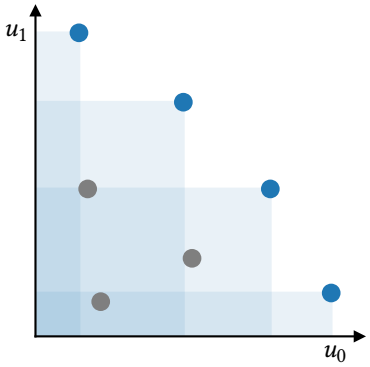


Figure 5.8.: A Pareto-front in two dimensions. The Pareto-optimal points are shown in blue.

We formalize this by introducing a concept called *dominance*. We say $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ is *dominated by* $\boldsymbol{y} \in \mathbb{R}^d$ (short: $\boldsymbol{y} \succ \boldsymbol{x}$) if $x_i \leq y_i$ holds for any $1 \leq i \leq d$ and $x_j < y_j$ for at least one $1 \leq j \leq d$. For any set $Q \in \mathbb{Q}$ we define the *Pareto-set* or *Pareto-front* of Q as

$$c_{\text{Pareto}}(Q) \coloneqq \{\boldsymbol{x} \in Q : \boldsymbol{x} \text{ is not dominated by any element } \boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}\}.$$

An exemplary Pareto-front in two dimensions is shown in Figure 5.8. The Pareto-optimal points are depicted in blue, and the region they dominate is shaded in blue. As you may notice, c_{Pareto} is a valid choice function, and we want to know which approaches are able to learn it from choice data. It is clearly a context-dependent choice problem since only by looking at all other objects we are able to decide whether one particular object is dominated. In addition, the size of the choice set is not constant across choice tasks $Q \in \mathbb{Q}$. We will therefore only use it as a dataset for the subset choice setting.

As before, we may ask if we can predict how well our approaches should be able to represent this problem. For pairwise FETA, we can set $\mathrm{u}_0(\boldsymbol{x}) \coloneqq 0$ and $\mathrm{u}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \coloneqq -[\![\boldsymbol{y} \succ \boldsymbol{x}]\!]$. Inserting into (5.8) we get

$$\mathrm{u}_{\text{FETA}}^{\mathrm{u}_0,\mathrm{u}_1}(\boldsymbol{x}, Q) = -\sum\nolimits_{\boldsymbol{y} \in Q} [\![\boldsymbol{y} \succ \boldsymbol{x}]\!] \in \begin{cases} (-\infty, -1], & \text{if } \boldsymbol{x} \notin c_{\text{Pareto}}(Q), \\ \{0\}, & \text{otherwise.} \end{cases}.$$

In words, that means we count the number of times a given object $\boldsymbol{x} \in Q$ is dominated by any other object $\boldsymbol{y} \in Q$. It follows that every object which achieves a utility of 0 is Pareto-optimal, and $c_{\text{Pareto}}(Q) = \arg\max_{\boldsymbol{x} \in Q} \mathrm{u}_{\text{FETA}}^{(\mathrm{u}_1)}(\boldsymbol{x}, Q)$ holds for all tasks $Q \in \mathbb{Q}$. We consequently expect FETA to be able to learn this problem.

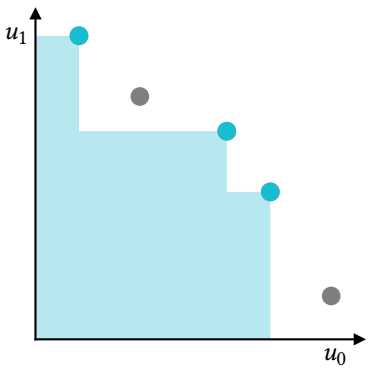


Figure 5.9.: Visualizing the volume (shaded in teal) of the set of three points maximizing the hypervolume.

For the experiments, we generate 30 uniformly distributed points in $\mathbb{R}^2$ and $\mathbb{R}^5$. We vary the dimensionality to see if there is a detectable difference in performance. We use c_{Pareto} to compute the corresponding choice sets. See Appendix A.3.2 for more technical details.

Hypervolume In multi-objective optimization, the goal is to find the best Pareto-front for a given optimization problem. There are a variety of approaches that have been proposed for solving this task, but one particular class of algorithms is multi-objective evolutionary algorithms (MOEAs), where a Pareto-front is created and then iteratively evolved such that the set of points improves over time. One important goal is to cover the ground-truth

Pareto-front evenly, which is a property not enforced by simply optimizing the objectives. Therefore, typical algorithms also try to optimize the *diversity* of the objects [ZDT00; GFP22]. Quality indicators are measures that try to take into account closeness to the Pareto-front, diversity, and overall spread. One particularly important quality measure is the *hypervolume* [LY20; GFP22], which is the volume of the union of the subspaces dominated by each object in the Pareto-front. In Figure 5.9, we can see a visualization of this concept. The volume spanned by the three points is shaded in teal. The hypervolume is important because it is the only known quality measure, that is *strictly Pareto-compliant*, i. e., if one Pareto-front strictly dominates another, then the quality measure of the former will also be strictly larger than the latter. Unfortunately, Bringmann and Friedrich [BF10, Theorem 1] show that it is #P-hard to calculate the hypervolume, which motivated us to use it as a challenging choice problem.

strictly Pareto-compliant

To create a corresponding choice problem, we will look at *hypervolume contributions*. Let us begin by formally introducing the hypervolume. The hypervolume $\lambda_{\mathrm{HypVol}}(Q)$ of a subset $Q \subseteq \mathbb{R}^d$ describes the volume of the union of the subspaces dominated by each individual point $\boldsymbol{x} = (x_1, \dots, x_d)$ in the Pareto set of Q and can formally be defined as

$$\begin{aligned}\lambda_{\mathrm{HypVol}}(Q) &:= \lambda\left(\bigcup\nolimits_{\boldsymbol{x}\in c_{\mathrm{Pareto}}(Q)} [0, x_1] \times \cdots \times [0, x_d]\right) \\ &= \lambda\left(\bigcup\nolimits_{\boldsymbol{x}\in Q} [0, x_1] \times \cdots \times [0, x_d]\right)\end{aligned}$$

where λ denotes the Lebesgue measure of $\mathbb{R}^d$. With that, we can define the *hypervolume contribution* $\lambda_{\mathrm{HypVol}}(Q) - \lambda_{\mathrm{HypVol}}(Q \setminus \{\boldsymbol{x}\})$ of an object $\boldsymbol{x}$ to the overall hypervolume. In other words, we want to know how much the hypervolume is reduced if $\boldsymbol{x}$ is removed from the Pareto-front. We define the problem of learning the corresponding hypervolume choice function $c_{\mathrm{HypVol}} : \mathbb{Q} \to \mathcal{C}$, which picks that object $\boldsymbol{x} \in Q$ with the *smallest* contribution to the overall hypervolume, i. e.,

hypervolume contribution

$$\begin{aligned}c_{\mathrm{HypVol}}(Q) &:= \arg\max_{\boldsymbol{x}\in Q} \lambda_{\mathrm{HypVol}}(Q) - \lambda_{\mathrm{HypVol}}(Q \setminus \{\boldsymbol{x}\}) \\ &= \arg\min_{\boldsymbol{x}\in Q} \lambda_{\mathrm{HypVol}}(Q \setminus \{\boldsymbol{x}\})\,.\end{aligned}$$

For the experiments, we generate sets of 10 objects uniformly at random in $\mathbb{R}^2$ and compute the corresponding singleton choice.

MNIST Number Problems For the set of problems, we are going to define tasks using handwritten digits. We will define choice functions on the level of digits, but the learners will only observe a feature vector based on the underlying handwritten digit. To be more precise, we will use the well-known MNIST dataset, which consists of 70 000 grayscale images, each with dimensions of 28×28. Since we do not want to make this task a computer vision problem, we ensure a fair comparison between all approaches by first training a convolutional neural network (CNN) using a subset of 10 000 instances and then applying it to the remaining 60 000 images to extract feature vectors (see Appendix A.3.3 for technical details). Then we sample sets of 10 images and make choices according to the following selection methods:

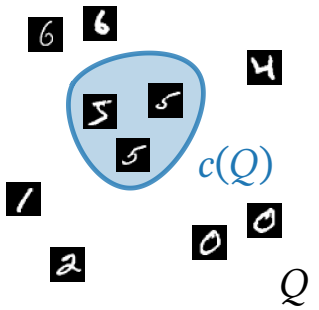


Figure 5.10.: An exemplary task of the Modified National Institute of Standards and Technology (MNIST) Mode dataset.

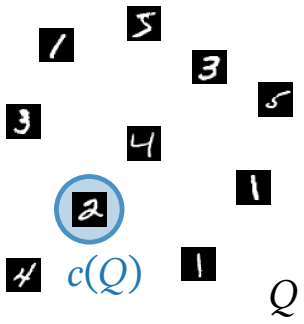


Figure 5.11.: An exemplary task of the MNIST Unique dataset.

Mode: For the Mode dataset, we choose the digits that occur most often in the choice task Q. For example, given a multiset of digits {0, 0, 1, 2,4, 5, 5, 5,6, 6}, we choose all instances with a value equal to the mode value 5. For the singleton choice task, we only output one of the digits by utilizing a tiebreaker. We pick the digit whose representation has the least angle to a predefined vector.

Unique: Here, we choose all numbers that occur only once in the set of sampled label values. For example, given a multiset of numbers {1, 1, 1, 2, 3, 3, 4, 4, 5, 5}, we choose the number {2}. For the singleton choice problem, we ensure that exactly one of the digits is unique.

Both of these problems are impossible to solve without using the task context, because the digit itself is not indicative of whether the object is chosen or not. Only when considering the complete set the learners are able to ascertain which digits appear most often/only once.

MovieLens Tag Genome dataset

MovieLens Tag Genome Recommender systems are an important part of today's economy, allowing companies to personalize the user experience and, therefore, increase customer loyalty. The MovieLens dataset is a collection of movie ratings collected by the MovieLens movie recommendation service [HK15]. The dataset is widely used in recommender systems research.

The MovieLens *Tag Genome* dataset (2014) consists of a large collection of movies and tag relevance scores [VSR12]. The authors base the tag relevance scores on community-curated tags, reviews, comments, blogs and ratings on the MovieLens website and were computed using a supervised learning algorithm that predicts tag relevance based on the sparse tagging data. Each movie's tag genome is a vector of real-valued relevance scores in $[0, 1]$.

We use this semantically rich vector space in the experiments to define similarity-based choice problems. For a given set of movies, we ask the learners to pick the movie that is most similar/dissimilar to the other movies. For the former problem, we compute the medoid with respect to the pairwise similarity of the movies in tag genome space. As for the latter problem, we pick the movie that maximizes the pairwise similarity scores. We use the similarity measure proposed by Vig et al. [VSR11], which is a weighted cosine similarity. Since the choices we compute this way are always singleton sets, we only use this dataset for singleton choice.

LETOR dataset

LETOR Moving on to the category of real-world datasets, we have the LETOR dataset, which is a collection of benchmark datasets for various learning-to-rank problems [QL13]. It is based on the GOV2 web page collection which is a crawl of `.gov` sites in 2004 comprised of 25 205 179 documents with a total size of 426 GiB. The organizers of the Text REtrieval Conference (TREC) 2007 and 2008 used this corpus in conjunction with search queries obtained from a search engine to host the Million Query (1MQ) track [TRE07; TRE08]. Only those queries that resulted in a click event on a document in the corpus were used in the 1MQ track and of those a subset of 10 000 was chosen. The goal of the participants was to submit up to 1000 documents for each of the 10 000. Then, during the judging phase, the submissions were assessed on a subset of 1700 and 782 queries (for TREC 2007 and 2008, respectively) by human judges. The relevance of each document was judged on an ordinal

scale from highly to not relevant. Finally, for the LETOR 4.0 dataset, Qin and Liu [QL13] use the TREC 2007 and 2008 query sets and additionally extract useful features from the documents to level the playing field. Then, they define various learning-to-rank tasks (i.e., supervised, semi-supervised, and listwise ranking as well as rank aggregation) and provide 5-fold cross-validation splits. Each query-document pair is, therefore, defined by a vector of 46 features.

For our experiments, we use the *supervised* version of the dataset, where each query-document pair is assigned an ordinal relevance score to construct a subset choice dataset. For a given query, we pick all documents that were labeled as relevant or highly relevant (i.e., they have a score of 1 or 2) as the choice set. As for singleton choice, we can utilize the *listwise ranking* dataset and let the singleton choice set be the document that achieves the highest score for a given query. For additional technical details, consult Appendix A.4.1.

Expedia Expedia is an online travel company that allows users to search for and book flights, hotels, etc. As a contest, the company uploaded a dataset to the Kaggle website, where researchers were able to participate for two months. The groups participating were also invited to submit a paper describing their approach at the International Conference on Data Mining (ICDM) 2013 [Exp13].[12]

12: Note that there are two similar looking Expedia datasets on Kaggle. We are *not* referring to the more recent one from 2016 here [Exp16].

The dataset is comprised of 665 574 searches on the Expedia website divided into train (399 344 searches) and test set (266 230 searches). Each search result consists of 5 to 38 hotels, each of which is represented by a vector of 45 features. The target is a relevance score ranging from 0 to 2, where 0 indicates that the hotel was not clicked on, 1 that it was clicked on and 2 that the hotel was booked.

Since only ever one hotel is booked per search, we can easily define singleton choice sets for this dataset. For the subset choice task, we use all hotels with a relevance score of at least 1 as the subset choice set. Please refer to Appendix A.4.2 for additional technical details.

SUSHI The SUSHI dataset is a dataset that was collected to facilitate comparisons of preference learning methods by Kamishima and Fujiki [KF03; KF03]. They aggregated names of sushi dishes from 25 sushi restaurants found online and selected the 100 most frequent ones. These sushi dishes were split into several sets of 10 sushis (see Appendix A.4.3 for details on the particular sets used), which were then provided to participants of an online survey to rank and score. The result was a data set of 1025 responses with order information for multiple sets of sushis. The size of the dataset was later increased to 5000 [Kam16]. Each sushi is represented by a set of 7 features (e.g., style of sushi, major & minor group, heaviness/oiliness etc.). The authors also recorded some features of the users responding to the survey, which we will not need in our experiments.

SUSHI dataset

In the experiments, we will pool the responses for both object sets used in the survey to get a dataset of 10 000 instances. For each instance comprised of 10 objects, we let the choice set be the sushi that was preferred by the user. Therefore, the dataset is only used for singleton choice. Further technical details are located in Appendix A.4.3.

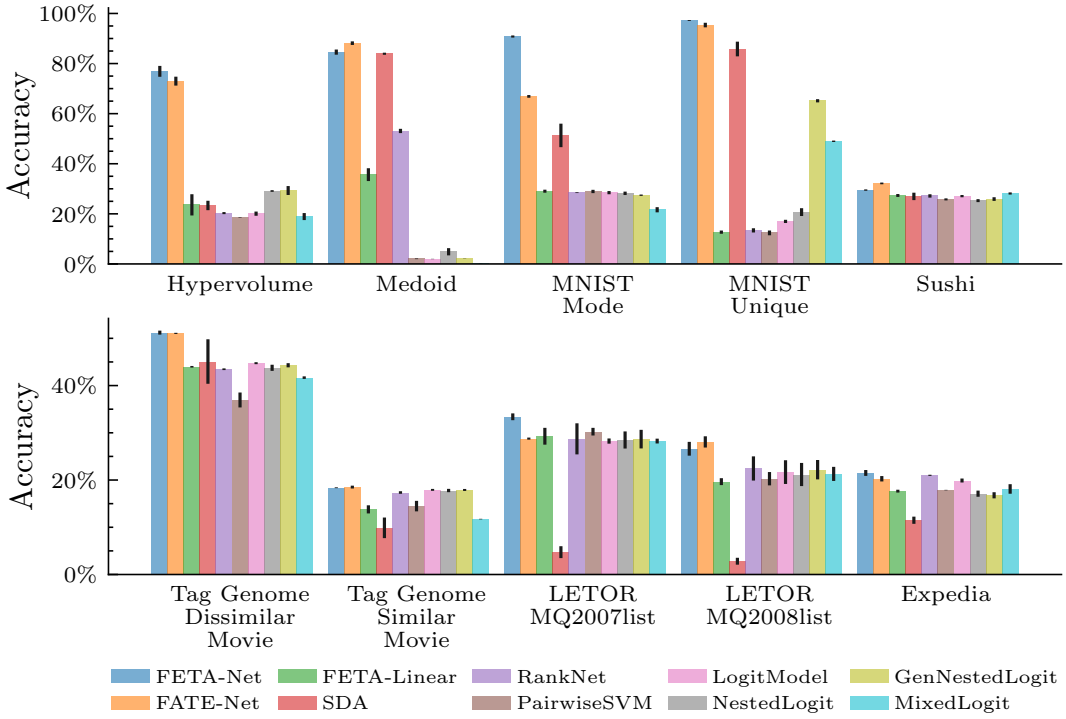


Figure 5.12.: Categorical accuracies and standard deviations (vertical bars) of the singleton choice models on different singleton choice tasks (measured across 5 outer cross-validation folds).

5.5.2. Results and Discussion

We will now move on to the results of the experiments. In Section 5.5.2.1 and Section 5.5.2.2, we compare the approaches on singleton and subset choice problems. Then in Section 5.5.2.3, we evaluate how well the different approaches are able to generalize to task sizes that they were not trained on. In these sections, we will focus on reporting the raw results while we relate the results to our original research questions in Section 5.5.2.4. Note that for reasons of conciseness, we only report the results for the target losses. The comprehensive results, where we report all evaluation metrics, are shown in Tables A.7 to A.9 of Appendix A.5. Finally, it is always important to highlight the limitations of an empirical evaluation, which we will do in Section 5.5.2.5.

5.5.2.1. Singleton Choice

The main experimental results for the singleton choice models are shown in Figure 5.12. The target loss for the singleton choice experiments is the *categorical accuracy*. In the figure, each bar depicts the mean categorical accuracy across all outer validation splits. The small, vertical, black bar shows the estimated standard deviation of the accuracies on these splits.

categorical accuracy: Page 86

We can observe that both FETA-Net and FATE-Net outperform most other baselines in tasks where the choice function depends on context-specific information, such as the Hypervolume, Medoid, and MNIST datasets. This demonstrates that these models effectively utilize the context and learn the underlying choice functions better. In contrast, the context-sensitive SDA network, while expected to achieve comparable results, only improves on the

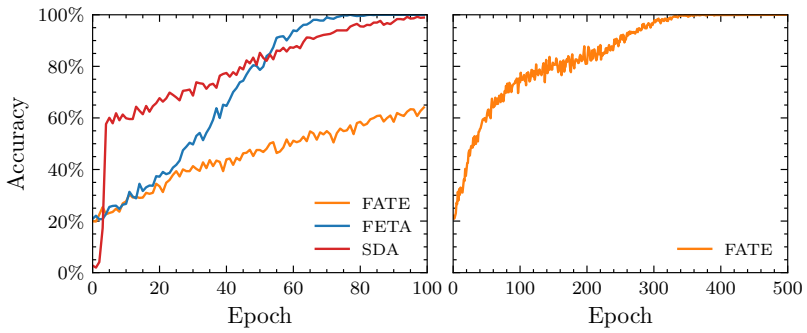


Figure 5.13.: Result of the infinite data experiment for FATE-Net, FETA-Net and SDA on the synthetic unique problem. The left plot displays the neural networks calibrated to have comparable numbers of parameters. The right plot shows a repetition of the experiment, with FATE-Net granted a higher epoch and parameter budget.

baselines on the Medoid and MNIST datasets. On the Hypervolume dataset, however, it performs at the same level as the remaining baselines.

To gauge the importance of nonlinearity and context-sensitivity for solving these tasks, it is interesting to take a closer look at the performance of the baselines FETA-Linear and RankNet. On most datasets, these perform on par with the other baselines, indicating that a combination of context-sensitivity and nonlinearity is required to solve the tasks well. One notable exception is the Medoid dataset, where they both outperform the other baselines by a large margin. For RankNet, this is sensible because, for this task, it is possible to learn that the typical medoid is located closer to the center of the object space. In addition, there appears to be a context-dependent but linear component that can be represented by FETA-Linear.

Looking at the MNIST-Unique dataset specifically, we can see that FETA-Net and FATE-Net are able to achieve a mean categorical accuracy of over 90 % with SDA being close with over 80 %. The underlying choice function has an extreme version of the *similarity effect* [HP83]. Recall that the similarity effect pertains to cases where the introduction of a highly similar object redirects probability weight primarily from the existing similar object instead of being proportionally divided among all objects based on preference. Here, we are choosing the digit that occurs only once; hence, any duplicated digit results in a choice probability of 0 for all of them. The GNL and mixed logit (ML) models can account for the similarity effect, resulting in a substantially better performance compared to the other baselines.

similarity effect: Page 29

Examining the results of the MNIST-Unique problem raise the question of whether the high accuracies achieved by FETA-Net, FATE-Net and SDA are their limit performance, or if they are able to achieve comparable performance given a large enough dataset. To investigate this, we generate a fully synthetic version of this dataset, where we represent each digit $i \in \{0, \dots, 9\}$ by the corresponding standard unit vector $\boldsymbol{e}_i$, i. e., a vector where the i-th position is 1 and the remaining positions are 0. We do this to condense the task down to only the context-dependent choice. Then, we calibrate each network to have roughly the same number of parameters (2870 for FATE-Net, 2849 for FETA-Net and 2850 for SDA). The remaining hyperparameters were set to the same values for all three learners. For the experiment, we let the learners train on a stream of batches, each consisting of 1024 instances with sets of 10 objects.

Infinite data experiment (FF-RQ5)

The result of the experiment is shown in Figure 5.13. On the x-axis, we show the epoch number, where each epoch corresponds to one training pass on one

batch of training instances. On the y-axis, we have the categorical accuracy achieved on a holdout set of instances. We can see that both FETA-Net and SDA reach the 100 % accuracy level. FATE-Net is only able to get to slightly above 60 %, but the accuracy appears to be trending upwards. That is why we decided to run the experiment again where FATE-Net is allowed a higher epoch and parameter budget. The result of this run is shown on the right side of Figure 5.13. Now with an increased budget of 5985 parameters, FATE-Net reaches the 100 % accuracy level within 400 epochs. To sum up, we can see that all three architectures are indeed able to learn this choice function perfectly. We can, however, see a difference in parameter- and data-efficiency between the models. The SDA architecture quickly reaches accuracies above 60 %, but is eventually overtaken in epoch 55 by FETA-Net which is then able to reach 100 % first. This could indicate that the inductive bias of FETA-Net is very suitable for this particular problem. Since FATE-Net needs many more parameters and epochs to converge, we can infer that the decomposition is not particularly well-suited for the dataset. Despite this, it is able to eventually learn the choice function, because it belongs to a very flexible model class.

real-world datasets

Finally, we will examine the *real-world datasets* Sushi, Movielens Tag Genome, LETOR, and Expedia. We find that the difference between the performance of the baseline approaches and our FETA-Net and FATE-Net is less pronounced. This is certainly to be expected since preferences expressed in real-world data tend to be much less context-dependent than in synthetic data. On the Movielens Tag Genome dataset with most dissimilar movie selection and LETOR MQ2008 both approaches outperform the other baselines. On LETOR MQ2007, only FETA-Net achieves higher accuracy, while FATE-Net is on par with the baseline approaches. On the remaining datasets, our context-sensitive approaches achieve competitive results but do not outperform context-independent learners like RankNet by a significant margin. This indicates that the preferences expressed on these datasets are close to being context-independent.

Surprisingly, SDA achieved the lowest accuracy on LETOR and Expedia. We suspect that this is due to the models having been trained on fixed choice task sizes in our experiments but evaluated on choice tasks of varying sizes during testing. As SDA learns a set-dependent aggregation function, it may not generalize well to the larger choice tasks present in real-world datasets.

5.5.2.2. Subset Choice

We continue with the results of the experiments where the choices are expressed as subsets of the choice task. Here, we use the F_1-measure (4.5) as the target measure since we want the models to find as many of the chosen objects as possible while also penalizing them for indiscriminately enlarging the predicted subsets. The results are shown in Figure 5.14. To see if the models have learned anything, we also display the performance of the baseline, which always predicts positive (i. e., all objects are chosen).

Looking at the results, we can see that the general observations we have made for the singleton choice results still hold true here. Our approaches FETA-Net and FATE-Net, as well as the context-sensitive baseline SDA, surpass the remaining baselines on datasets that are highly dependent on context. Namely

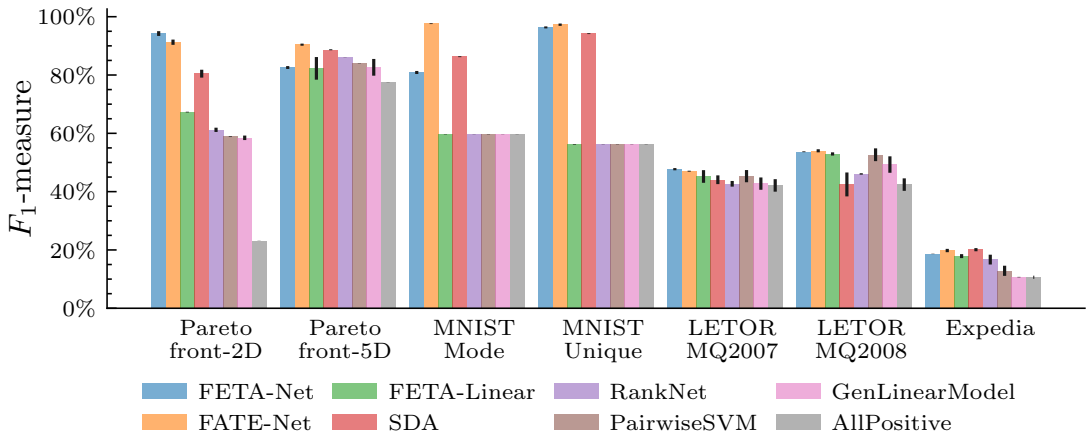


Figure 5.14.: Average F_1-measure and standard deviation (vertical bars) of the subset-choice models on the different choice tasks (measured across 5 outer cross-validation folds).

the 2D version of the Pareto-front dataset and both MNIST datasets. Our approaches work well on the real-world datasets (LETOR and Expedia), but the performance difference to the other baselines is less pronounced, as we have seen before.

There are, however, a few interesting observations we can make. The performance of the FETA-Linear baseline, depicted in green, is close to that of FETA-Net and FATE-Net on the real-world datasets. This indicates that linear, context-sensitive models are effective in these scenarios. However, on the MNIST problems, the FETA-Linear baseline is unable to surpass the baseline that always predicts positives, implying that a nonlinear component is crucial for solving these choice tasks. This holds true for the other linear and/or context-independent baselines as well. The inability of RankNet to outperform the simple baseline underscores the fact that the choice function can only be accurately approximated by a nonlinear, context-dependent model.

An interesting observation can be made regarding the Pareto problem: the context-sensitive models FETA-Net, FATE-Net, and SDA surpass all baseline models on the 2D version of the dataset. However, on the 5D version, the performance of all approaches converges to a similar level. This suggests that as the dimensions of the dataset increase, the task of selecting the Pareto-front becomes increasingly less reliant on context and more reliant on the distance of a point from the center, as this information becomes increasingly informative. Furthermore, the high F_1-measure of the AllPositive baseline is indicative of the fact that there are more points on the Pareto-front overall.

Overall, our results show that FETA-Net and FATE-Net significantly outperform context-independent baselines on tasks that depend heavily on context. They also compare favorably with the state-of-the-art approach SDA. This improvement is due to the models' sensitivity to the context and their ability to model nonlinear relationships between inputs and outputs. However, on real-world datasets, the improvement is not as pronounced, which suggests that either the context-dependence is weaker or that the models have yet to capture it fully. The former is more plausible due to the relatively large lists of objects given to the human evaluators. Research has shown that context effects become less pronounced as the task size grows [Haw+12].

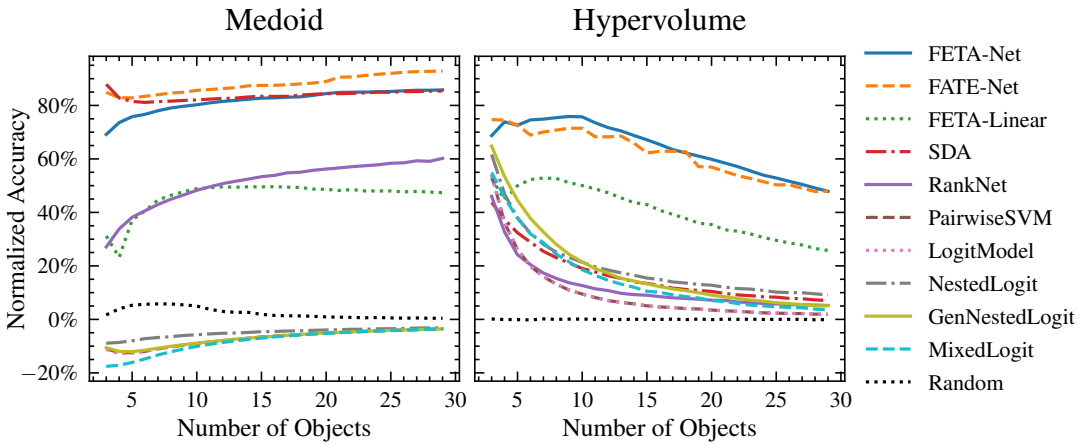


Figure 5.15.: Normalized Accuracy of the singleton choice models (SCMs) trained on queries of size 10, and predicting on queries of a varying size.

5.5.2.3. Generalization Across Task Sizes

FF-RQ4

For the last round of experiments, we want to address FF-RQ4. The goal is to evaluate the ability of the learners to generalize task sizes they did not encounter during training. This is an important aspect since the complexity of computing choices is typically lower during prediction time. A straightforward methodology to handle datasets with larger task sizes could, therefore, be to train on task sizes of a manageable size and use the learned model to predict choices for larger tasks during deployment. Here we will focus on the results on the datasets Medoid and Hypervolume to showcase the key insights we observe across all datasets. Each model is trained on instances consisting of 10 objects and subsequently tested on instances with 3 to 29 objects. It is important to keep in mind that when dealing with singleton choice, comparing accuracy across varying task sizes can be misleading. However, we can resolve this issue by using the *normalized accuracy* metric, which is explained in more detail in Section 4.2.1. By using this metric, we can ensure that random guessing will always achieve a score of 0.

The results for the two datasets are shown in Figure 5.15. Overall, the models demonstrate robust generalization capabilities, extending their performance to task sizes beyond their training regime. However, the exact generalization behavior varies by dataset. In the Medoid dataset, models such as FETA-Net, FETA-Linear, and RankNet show improved performance with increasing task size. This is expected because a context-independent model, which assigns the maximum score to objects in the center, can more effectively solve the problem as more points fill the space.

Conversely, on the Hypervolume dataset, the normalized accuracy declines as the number of objects in the choice task increases. This suggests that choosing the point with the highest contribution to the hypervolume becomes harder with an increasing number of objects. We can also see a notable difference in generalization behavior between FETA-Net, FATE-Net and FETA-Linear, and the remaining baselines. The latter achieve their highest performance on sets of three objects, despite being trained on sets of 10, and it monotonically drops with increasing number of objects. Our three approaches exhibit a more gradual decrease in accuracy. Remarkably, FETA-Net and FETA-Linear assume the maximum accuracy in the range of 7 to 10 objects. This indicates

that FETA-learners adapted more strongly to the specific task size encountered during training, even though the choice problem is inherently easier with fewer objects.

5.5.2.4. Discussion

At the beginning of Section 5.5, we specified the five guiding research questions (FF-RQ1 to FF-RQ5) for the empirical evaluation. We will now relate the results obtained in the previous sections to the research questions. With FF-RQ1 we wanted to know whether our proposed models FETA and FATE, and in particular FETA-Net and FATE-Net are suitable to represent and learn context-dependent choice functions. In the experiments, across both the singleton and the subset choice setting, FETA-Net and FATE-Net consistently outperform the context-independent baselines on the clearly context-dependent tasks. Therefore, we can definitively answer FF-RQ1 with "yes". FF-RQ1

FF-RQ2 and FF-RQ3 asked how the performance of the models on the tasks was related to the overall expressiveness and the ability to model context-dependent choice functions. We introduced the FETA-Linear baseline to tease these factors apart and closely monitored the context-independent but very flexible RankNet. Although there are datasets, like Medoid, where FETA-Linear and RankNet on their own already achieve a better performance than the other baselines, we find that in most cases, they perform on par with the other baselines, and only the context-dependent, nonlinear baselines are able to achieve a high accuracy or F_1-measure on the context-dependent tasks. On the real-world datasets, RankNet is often competitive with our models, suggesting that the task context plays a smaller role in these datasets. At the same time, we find that the linear baselines often achieve a performance close to that of RankNet on these tasks, indicating that a linear, context-independent model is sufficient to model the underlying choice function. Overall, we can say that it depends on the specific task, whether nonlinearity or context-dependence is more important. On the context-dependent datasets, it is clearly important to have a combination of both. FF-RQ2 and FF-RQ3

Next, we wanted to know how well the models and, specifically, our approaches generalize to task sizes that they were not trained on (FF-RQ4). Here, we again observe that the performance depends on the choice problem (some get easier with increasing task size, some harder). For the former problems, our approaches do not suddenly become worse with an increasing number of objects in the task. From this, we can deduce that the learned representation is not overly adapted to the training task size. For problems that get harder with a larger set size (e. g., Hypervolume), the performance degrades only gradually as compared to the baselines, which is another hint that they can be used to predict on larger task sizes. In practice, one should always check what kind of regime the dataset belongs to. FF-RQ4

Finally, we wanted to know if the context-sensitive, nonlinear learners (i. e., FETA-Net, FATE-Net and SDA) are flexible enough to represent the context-dependent exactly (FF-RQ5). Since we observe very high accuracies for some of the choice datasets, it is not clear whether this is the limiting performance of the learners or whether this is simply due to a limited amount of data. We picked the Unique dataset and gave the learners enough new data until FF-RQ5

they reached convergence. As all three approaches were able to reach 100 % of out-of-sample performance, we can deduce that they belong to a flexible enough model class such that this particular choice function can be represented exactly. We were able to see differences in data efficiency, however, as FATE-Net needed a much higher parameter- and data-budget to reach the same performance level.

In conclusion, the empirical evaluation has shown that our proposed models, FETA-Net and FATE-Net, are effective in representing and learning context-dependent choice functions. The models also exhibit good generalization and representational flexibility, although the specific performance depends on the context-dependence of the dataset. We do, however, see that the models perform at worst on par with the context-independent baselines, demonstrating that we do not lose performance by incorporating the task context.

5.5.2.5. Limitations

Our research has provided useful insights into the performance of our two approaches FETA-Net and FATE-Net for modeling human and algorithmic decision-making. As with any research, it is essential to acknowledge the limitations of our study to guide future work.

Firstly, while our experiments utilized synthetic, semi-synthetic, and real-world datasets, the real-world datasets featured large task sizes. This, as we know from Section 2.3.2, can lead to less context-dependent choices. In addition, for LETOR (and the underlying 1MQ dataset), participants rated relevance in absolute terms, leading to even fewer context-dependent choices. Future research could benefit from collecting real-world datasets with smaller task sizes, where objects are presented in an easily comparable form. This might encourage more contextual choices and improve the comparison of context-sensitive models.

Secondly, all approaches were trained on fixed task sizes. Although the models demonstrated good generalization to various task sizes in our generalization experiments (see Section 5.5.2.3), exploring training on varying task sizes could potentially lead to more size-independent representations. This direction of research could lead to improvements in model robustness and applicability across different task sizes.

Lastly, no clear pattern has emerged indicating which particular type of dataset each of our approaches works best on. The infinite data experiment demonstrated that both models can represent the underlying choice function given enough data. The FETA-Net model did, however, converge faster than FATE-Net, which may indicate that it is more data-efficient. More research is needed to understand how the models perform on different types of datasets and how they can be improved to better capture the nuances of decision-making in various contexts. It may be valuable to explore how the models can be adapted or improved to perform better in specific contexts and to investigate which factors affect their performance on different datasets.

5.6. Conclusion and Future Work

In this chapter, we argued that classical utility models are limited in modeling contextual preferences. Since context effects are common in human choice behavior and essential in algorithmic choice problems, it is important to have models of choice that can accommodate these effects. Although the literature has documented a few specific context effects, along with solutions to model them, our goal is for our approaches to be flexible enough to handle any context effects (even those currently unknown) present in a given dataset. We therefore introduced the *generalized utility function* (5.1), where the set of objects is explicitly taken as an additional input. This allows it to model any context-dependent choice function, but this ability comes at a price: it is infeasible to estimate it directly without further assumptions. That is why we proposed two decompositions, FETA and FATE, that aim to approximate the generalized utility function and enable it to be learned.

generalized utility function: Page 112

FETA decomposes (5.1) as a sum of sub-utilities (5.7), where each sub-utility computes an average of a certain set size k. We further propose the *pairwise* FETA model, where we limit the aggregation to at most pairwise interactions. The FATE decomposition (5.15) first computes a representative vector for the input set by mapping each object into an embedding space and taking the mean of the resulting vectors. Then a utility is computed for each object by using the representative vector as context. We conduct a theoretical analysis of the representational capacities of the two models, revealing that FATE is able to approximate any choice function, whereas the representational power of pairwise FETA is limited. Thus, both approaches are complementary and have differing inductive biases. We propose two neural network architectures, FETA-Net and FATE-Net, that are able to learn both decompositions and conduct experiments on a variety of datasets to gauge their practical performance. We see that although the representational power is different between both models, both are competitive on the set of datasets we evaluated outperform the other baselines on the context-dependent datasets. These results are promising, suggesting that our proposed neural network architectures have the potential for practical applications in various domains.

FETA: Page 117
Pairwise FETA: Page 118
FATE: Page 123

From here, we can identify several promising directions for future work. These can be grouped into three broad topics. The first topic is the *expansion of the empirical evaluation* to gain an improved understanding of the existing approaches and their particular properties. One possible course of action is to test the efficacy of context-dependent methods on progressively harder algorithmic choice problems, with the aim of identifying their strengths and weaknesses. In addition, it may be worthwhile to investigate datasets from real-world contexts where the likelihood and strength of context effects are greater.

expansion of the empirical evaluation

Secondly, one may focus on the *development of model improvements*. Especially when evaluating harder choice problems, it may be necessary to increase the representational power of the models. FATE is already able to approximate any choice function given enough data. For FETA, it could be fruitful to investigate incorporating higher-order interactions. A sampling-based approach could work well while keeping the computational overhead in check. In the same vein, the network architectures could be improved to handle larger datasets, allowing them to be more useful in webscale information retrieval.

development of model improvements

The ability to interpret a model is crucial in practical applications. Hence, many practitioners continue to utilize linear models, which are more straightforward and easier to comprehend [LL17]. This is why investigating methods to make our approaches more interpretable is another important area to focus on. With interpretability, we not only mean the typical feature-based explanation of the prediction [Alt+10; LL17]. In fact, with context-dependent models, it can be interesting to investigate the role of the set-valued context in a particular choice and to identify which objects had the greatest influence on that choice.

applications and integrations

Finally, the third topic is the exploration of interesting *applications* of context-dependent choice and possible *integrations* within other frameworks. The ability to approximate hard choice problems could be useful in settings like multi-objective optimization [Hao+22] and reinforcement learning [DLP23; BT24], where the learned choice function could be used as a policy to more efficiently guide an overlying search algorithm. If used as part of a gradient-based policy, it could even be useful to backpropagate the reward signal through the complete choice architecture. Another application area could be online learning and specifically the bandit setting. For instance, in the preselection bandit setting, the goal is to collect a suitable subset of bandit arms for the user [BH20]. The user then selects (possibly stochastically) one of the presented options, and the learner incurs a regret. Here, it could be interesting to see what happens if the underlying choice model is context-dependent and does not fit any of the classical preference models. Causal inference has been on the rise recently [Yao+21] since it allows one to estimate the true effect certain inputs have on an observed output. Identifying causal context effects in choice data could be a promising direction.

As we bring this chapter to a close, we have opened the door towards modeling choices that are dependent on the set-based context, using two natural decompositions of the generalized utility function. This raises the intriguing question of whether we can achieve context-dependent choices without the need to approximate generalized utility explicitly. Our exploration of this topic will continue in the next chapter, where we will examine the use of Pareto-embeddings for choice modeling.

6. Subset Choices using Pareto-Embeddings

In Chapter 5 we introduced a framework for modeling preferences and in particular choices using generalized utilities. The goal in a given context, established by the given set of objects, was to map each object to a utility dependent on that context. With the utilities of the objects, we were then able to produce choices by picking the object with the maximal utility (to get the singleton choice) or the set of objects exceeding a utility threshold (in the subset choice setting). While the former is quite a natural operation, thresholding of the utilities has no inherent meaning and requires the threshold to be tuned for optimal performance. This is certainly undesirable and we may want to have a natural operation similar to how picking the object with the maximal utility is used in singleton choice. One potential solution is to choose the top-k objects according to their utilities. However, this method has a significant limitation: the size of the chosen subset is fixed, making it only applicable in very specific settings.

In this regard, it may be fruitful to look at how humans typically make decisions. Consider a scenario in which you are faced with a decision between several options, each of which can be characterized by attributes that have to be minimized or maximized. Would you first look at all attributes, condense them down into one value and then see which of the options is still viable by applying a threshold? What would even be a useful threshold in this case? A much more intuitive methodology would be to directly compare the options based on their attributes. Maybe one of the options is better in every attribute than another. In this case we may simply remove the latter option. Continuing with this procedure, we may pick the set of options remaining as our chosen subset. This is quite a natural choice, since for the remaining options it is not possible to identify a clear winner. That means it for any pair of options it is the case that one object is better along one axis, while it is worse on another.

Figure 6.1.: A preference expressed on a pair of images. What are the underlying attributes by which this preference was made? Artworks by Steve Johnson.

This informal strategy is the motivation for the approach developed in this chapter. However, before we can develop it into a general methodology, we must tackle the following challenge. Specifically, the attributes we have at our disposal for a given option do not always exhibit monotonic behavior with respect to their utility. In other words, increasing or decreasing the value of an attribute may not necessarily lead to a corresponding increase or decrease in its utility. Often it is the case that only combinations of multiple attributes will result in the most preferred options. As an example consider a setting where one has to decide between different works of art (see Figure 6.1). One artwork could be represented by a matrix containing the raw pixel values in RGB. It is

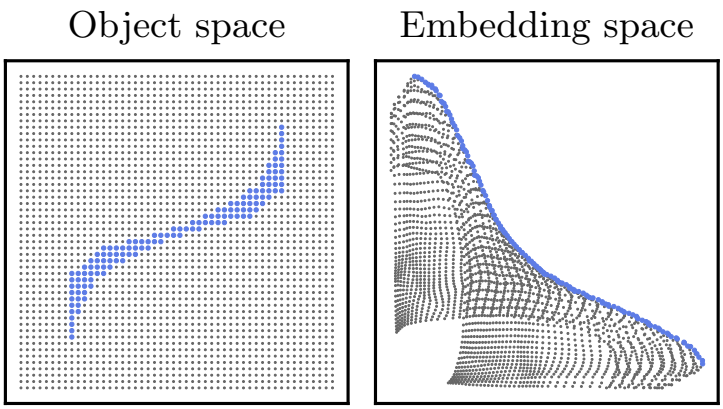


Figure 6.2.: Visualization of a Pareto-embedding learned on the two parabola (TP) problem. We evaluate an evenly spaced grid in the 2D object space (left) and pass it as a choice task to the trained 2D embedding function. The resulting embedding is shown on the right side. The Pareto-front predicted by the model is shown in blue.

clear that simply increasing or decreasing certain pixel values will not mean that an artwork is preferred, unless one has a faible for monochrome art. In other words, only very specific combinations of pixels will result in an image even recognizable by humans. Identifying attributes that describe why certain images would be preferred over others would require a deep understanding of the semantic meaning of images. This property is of course not only limited to images, which were only used as an instructive example here.

It is clear that constructing such a mapping from objects to monotone attributes[1] is desirable. However, unless the choice settings are extremely simple, it is not feasible to manually create such a mapping, which the image example should make clear. Thankfully, with embedding methods/representation learning we may have a way to learn this mapping from data [GBC16]. The high-level process will look as follows: (1) We specify a differentiable model that takes an object as input and outputs a low-dimensional representation of the object. This will be our embedding model. (2) A differentiable loss function ensures that the features of the this embedding are monotone, as explained earlier. To be more precise, it will ensure that the chosen subsets are *Pareto-optimal*[2] in this embedding space. (3) To predict the chosen subset for a given task Q, we compute the embeddings for each object and select the set of Pareto-optimal objects. Due to how we embed the objects in such a way that Pareto-optimality holds for our chosen subsets, we call these *Pareto-embeddings*.

1: We will later just call these attributes *utility* dimensions, which captures the essence of what they are used for.

2: An object is *Pareto-optimal* in a given set, if there is no other object that dominates it. An object *dominates* another, if it is equal or better in every criterion and strictly better in at least one.

Pareto-embeddings

As an example of what such a mapping could look like, consider Figure 6.2. On the left side, we depict a grid of points in the two-dimensional object space. The target function consists of two parabola. Then we train an embedding model that transforms the original dimensions in such a way that the embedding dimensions obey a certain set of constraints. As we can see, the points highlighted in blue form the Pareto-front in the embedding space. These correspond to the points close to the two parabola in the object space. The remaining points of the grid are "wrapped around" and placed below the Pareto-front. The farther away points are from the two parabola in the object space, the closer they are to the origin in the embedding space.

In this chapter we will address the following questions: How can we learn such an embedding and what kind of constraints/loss functions are necessary? Which theoretical properties do the embeddings have? And finally, what neural network architectures are suitable to learn the embedding and how do

they perform on some actual datasets when compared to other approaches. To this end, we will start by consolidating the necessary notation in Section 6.1. We then introduce our Pareto-embedding approach in Section 6.2, explain how to learn it from data, analyze theoretical properties and propose two neural architectures. The empirical evaluation we describe and perform in Section 6.3 and then we end with a conclusion and outlook in Section 6.4.

6.1. Modeling Choice

The preceding chapters have already established the overall notation (see Section 2.1 for an overview). We will, therefore, focus on quickly summarizing the necessary notation for this chapter here to make it self-contained.

Assuming a reference set of objects $\mathcal{X} \subseteq \mathbb{R}^d$, we consider subsets $Q = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\} \subseteq \mathcal{X}$ as our *choice tasks*, where m is arbitrary yet finite. The preference indicated on the choice task is represented by the *choice set* $C \subseteq Q$. In Chapter 5, we imposed a structure $\mathbb{Q} \subseteq 2^{\mathcal{X}}$ on the choice tasks. For this chapter, however, we assume for simplicity that $\mathbb{Q} = 2^{\mathcal{X}}$, i.e., all subsets of objects are valid choice tasks, which simplifies some of the proofs later on. A convenient way to represent a choice set is by using a binary vector $\boldsymbol{c} \in \{0,1\}^m$, where $c_i = 1$ if $\boldsymbol{x}_i \in C$ and $c_i = 0$ if $\boldsymbol{x}_i \notin C$, for every $i \in [m] := \{1, \dots, m\}$.

Particularly important for this chapter will be the definition of a *Pareto-set* and the notion of *dominance* [Par06; Koo51]. We say that one vector $\boldsymbol{x}$ dominates another vector $\boldsymbol{y}$, if $y_i \geq x_i$ for all $i \in [d]$ and if there exists $i \in [d]$ such that $y_i > x_i$. We also denote this by the relation $\boldsymbol{x} \succ \boldsymbol{y}$. The Pareto-set for a set of vectors $Z \subset \mathbb{R}^d$ is then formally defined as

Pareto-set
dominance

$$\begin{aligned} \mathrm{Par}(Z) &:= \{\boldsymbol{x} \in Z \mid \nexists \boldsymbol{y} : \ \boldsymbol{y} \succ \boldsymbol{x}\} && (6.1) \\ &= \{\boldsymbol{x} \in Z \mid \nexists \boldsymbol{y} : \ \forall i \in [d] : \ y_i \geq x_i \text{ and } \exists i \in [d] : \ y_i > x_i\} && (6.2) \end{aligned}$$

We refer to every vector $\boldsymbol{z}$ belonging to the set $\mathrm{Par}(Z)$ as *Pareto-optimal*.

Pareto-optimal

A function $c : \ 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ is called a *choice function* if $c(Q) \subseteq Q$ for all tasks $Q \subseteq \mathcal{X}$ and $|c(Q)| \geq 1$, meaning choice sets are subsets of the choice task and cannot be empty. We are tackling the learning problem, where we are given a set of training instances $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^n$, and we want to learn the choice function that best models the choice process. More specifically, we aim to find a choice function $\hat{c}$ that minimizes the risk, expressed as

choice function

$$R(c) = \int_{2^{\mathcal{X}} \times 2^{\mathcal{X}}} L(C, c(Q)) \, d\, P(Q, C) \,, \tag{6.3}$$

where $L : \ 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to \mathbb{R}$ represents an appropriate loss function that compares predicted and ground-truth choice sets. P is a (unknown) probability distribution that represents the underlying process generating the data.

6.2. Pareto-Embeddings

Now, we can formally define what we mean by *Pareto-embeddings*. As we illustrate in Figure 6.3, we need a function $\varphi : \ \mathcal{X} \to \mathcal{Z}$ that maps objects from

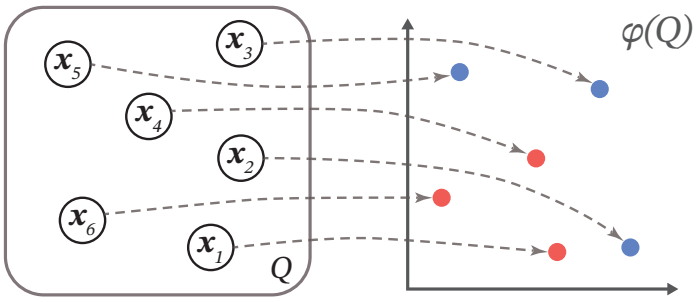


Figure 6.3.: A Pareto-embedding $\varphi(\cdot)$ maps a given set of objects Q into a higher-dimensional utility space $\mathcal{Z}$. The Pareto-optimal points in this space (here depicted in blue) we define to be the choice set C.

their original feature space $\mathcal{X}$ to an embedding space $\mathcal{Z} \subseteq \mathbb{R}^{d'}$. Suppose we have a choice task $Q = \{x_1, \dots, x_m\} \subseteq \mathcal{X}$, then the corresponding embedding is defined as $Z = \varphi(Q) := \{\varphi(x_1), \dots, \varphi(x_m)\} = \{z_1, \dots, z_m\}$. More importantly, this mapping should be defined in such a way that for each choice task Q the corresponding choice set $C \subseteq Q$ forms the *Pareto-set* in the embedding space. Formally, this means that $\text{Par}(\varphi(Q)) = \varphi(C)$. A visual representation of the idea is shown in Figure 6.3.

Pareto-set

For the following, recall that we use the symbol $\succ$ to denote the dominance relation between two objects. In case we have two objects $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and the dominance relation only holds in the embedding space, we will denote this as $\boldsymbol{x} \succ_\varphi \boldsymbol{y}$.

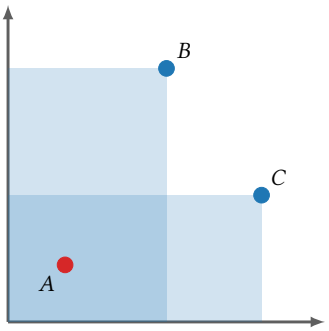


Figure 6.4.: A situation with two criteria and $A \prec B \sim C$.

A perceptive reader might wonder if this new representation offers any advantages over the one-dimensional utility approaches we have seen before. Does this approach allow us to capture a broader range of preferences? We will illustrate the difference using an example. Refer to the situation shown in Figure 6.4. It shows three objects A, B and C within a two-dimensional utility space. We can see that A is dominated by both B and C, while there is no clear preference order between B and C. We are able to define a one-dimensional utility function for this situation. We simply set $u(A) < u(B)$ and $u(B) = u(C)$.

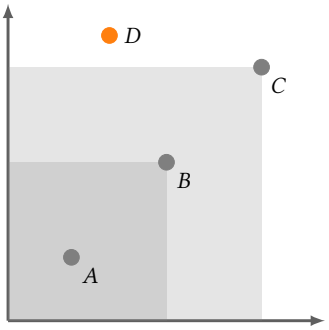


Figure 6.5.: A situation with two criteria and $A \prec B \prec C$ where $B \sim D$ and $C \sim D$.

Now look at the altered situation in Figure 6.5. Here we have four objects A, B, C and D. It is clear that $A \prec B \prec C$ holds. Object D is better than B and C in the second utility dimension but worse in the first. We can, therefore, state that $B \sim D$ and $C \sim D$. In this situation, we cannot state a consistent utility function. It is clear that $u(A) < u(B) < u(C)$ always has to hold, but we are not able to specify a utility for D. The Pareto-embedding is capable of modeling any of these situations since it can model any *partial order*. It is, therefore, possible to model richer preference structures with more layers of indifference.

6.2.1. Learning a Pareto-Embedding from Data

In the introductory sections, we motivated Pareto-embeddings as a natural way to model subset choices. In practice, this is only useful if we can induce such a model from observed choices. Since the Par operation is fixed, we only need to learn the Pareto-embedding φ to have a full subset choice model. Our goal in learning the Pareto-embedding φ is to predict choices that are as close as possible to the original choices, for some definition of closeness as discussed in Section 4.2.1. Generally, for a given dataset $\mathcal{D}$, the goal is to minimize the

expected loss (6.3) or an empirical version thereof. For ease of notation, we will assume that the loss function L receives the objects in the embedded form as its first input, instead of the predicted subset $\mathrm{Par}_\varphi(Q_n)$. With this, the empirical risk minimizer is defined as:

empirical risk: Page 43

$$\varphi^* = \arg\min_{\varphi \in \Phi} \frac{1}{N} \sum_{n=1}^{N} L\big(\varphi(Q_n), C_n\big)\,. \tag{6.4}$$

We therefore have to specify two components: (i) a parametric model for φ and (ii) a suitable loss function L, which, when minimized, produces a correct Pareto-embedding.

We will begin with the latter component. Before we define the loss function itself, it is useful to formalize which properties need to hold for a valid Pareto-embedding.

Definition 6.2.1 *A function φ is a Pareto-embedding (of dimensionality d') for a choice function c if, and only if for all $Q \in 2^{\mathcal{X}}$ we have that:*

(i) For all $\boldsymbol{x} \in c(Q)$, $\boldsymbol{y} \in Q$, $\boldsymbol{y} \neq \boldsymbol{x}$ there exists a dimension $i \in [d']$ such that $\varphi(\boldsymbol{x})_i > \varphi(\boldsymbol{y})_i$.

(ii) For all $\boldsymbol{x} \in Q \setminus c(Q)$ there exists $\boldsymbol{y} \in Q$ such that for all $i \in [d']$ it holds that $\varphi(\boldsymbol{y})_i \geq \varphi(\boldsymbol{x})_i$ and the inequality is strict for at least one such i.

In other words, property (i) expresses that all chosen objects need to be Pareto-optimal, while (ii) states that all non-chosen objects have to be dominated by at least one object within the embedding space. This definition subsumes singleton choice if we set d' to 1. Property (i) would then state that the utility of the chosen object $\boldsymbol{x} \in c(Q)$ is strictly greater than the utility of all other objects, and property (ii) would be equivalent to (i) in this special case. Therefore, the order resulting from the one-dimensional utilities is a strict linear order.

While we could work with these properties directly—using them as constraints in an optimization program—our goal will be to learn an end-to-end trainable model. That entails that an almost everywhere differentiable loss is computed, and we determine the gradient of this loss with respect to the model parameters. The gradient can then be used in a gradient descent setting to optimize the model parameters.

Consequently, we are looking for a (almost everywhere) differentiable loss function that encodes properties (i) and (ii). Methodologically, we will first treat the properties as sums of certain 0/1-losses. In the next step, we define a convex upper bound for these 0/1-losses. Let C denote the chosen subset of a choice task Q and let $\boldsymbol{c} \in \{0,1\}^{|Q|}$ be a vector representing the subset C.[3] Let $\boldsymbol{z}_j \coloneqq \varphi(\boldsymbol{x}_j) = (z_{j,1}, \ldots, z_{j,d'})$ be the embedding vector of $\boldsymbol{x}_j$, $Z \coloneqq \varphi(Q)$ be the corresponding $|Q| \times |d'|$ matrix for the choice task Q and $\mathcal{Z} \subseteq \mathbb{R}^{d'}$ be the embedding space. We will now introduce the loss functions L_{PO} and L_{DOM} that are designed to enforce the properties (i) and (ii) (when minimized).

3: $c_i = 1$ iff. $\boldsymbol{x}_i \in C$ and $c_i = 0$ otherwise.

Beginning with (i), we want to guarantee that all chosen objects are mapped into our embedding space in such a way that no other object dominates them. Therefore, if we look at the difference in the coordinates of a chosen object with the coordinates of any other object, then we want to avoid that the differences are all greater or equal to 0, i. e., $\min_{1 \leq k \leq d'} z_{i,k} - z_{j,k} \geq 0$, with one difference

being strictly greater than 0, i. e., $\max_{1\leq k\leq d'} z_{i,k} - z_{j,k} > 0$. We can express this for a given task Q and $Z := \varphi(Q)$ as the following sum:

$$\sum_{1\leq i\neq j\leq |Z|} c_j \cdot \left[\!\!\left[\left(\min_{1\leq k\leq d'} z_{i,k} - z_{j,k} \right) \geq 0 \right]\!\!\right] \left[\!\!\left[\left(\max_{1\leq k\leq d'} z_{i,k} - z_{j,k} \right) > 0 \right]\!\!\right]$$

where the indicator functions encode (i) negatively using min and max. We can not use this expression directly since the resulting gradient lacks the requisite directional information for gradient descent. It is non-convex, and it allows for embeddings where all chosen points are mapped to a singular point.

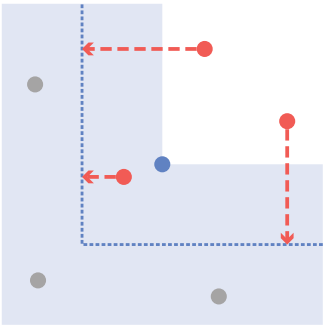


Figure 6.6.: Visualization of the effect of the loss terms L_{PO} in $\mathcal{Z}$ space. The blue point was chosen. Points depicted with a red arrow are penalized.

To solve this issue, we will apply a hinge loss on the first indicator function, obviating the need for the second one:

$$L_{\text{PO}}(Z, \boldsymbol{c}) = \sum_{1\leq i\neq j\leq |Z|} \max\left(0, c_j \cdot \min_{1\leq k\leq d'} (1 + z_{i,k} - z_{j,k})\right) \tag{6.5}$$

The effect of this loss is shown in Figure 6.6. The blue point represents the embedding of the chosen object. The shaded blue region marks the points dominated by the chosen object, and points in the white quadrant in the upper right corner would dominate the chosen object. By adding 1 to the difference of the coordinates, we shift the blue region to the dotted blue line. The red points are all those in Q for which their term in the sum is non-zero. These points are penalized linearly based on their distance to the dotted blue line. Evidently, it is no longer a valid solution to map all chosen objects to a single point since the difference in any dimension needs to be at least 1 to reach a loss of 0. This effect is analogous to the *margin effect* observed for SVMs when used in conjunction with the hinge loss.

We will apply the same methodology to (ii). In this case, we need to penalize a non-chosen point if it is not dominated by another point. The corresponding sum using 0/1-losses can be formalized as

$$\sum_{i=1}^{|Z|} (1 - c_i) \min_{j\neq i} \left(\left[\!\!\left[\left(\max_{1\leq k\leq d'} z_{i,k} - z_{j,k} \right) > 0 \right]\!\!\right] \right) .$$

This loss is only evaluated for the non-chosen objects and is 0 otherwise. The inner indicator function tests whether there exists a dimension in the embedding space for which the current non-chosen object is strictly better than another object (and hence not dominated by it). By minimizing this expression, we return 0 only if there exists at least one object that dominates the non-chosen object. Otherwise, we return 1 and penalize it.

When translating this into a suitable surrogate loss, it becomes apparent that the inner maximization provides a rather weak signal for optimization. This is because it imposes an equal penalty on a point that barely is surpassed by the non-chosen object in only one dimension and a point that is completely dominated across all dimensions. To address this issue, we will replace the inner maximization with a sum that penalizes points for each dimension the non-chosen object is better in. Then, we apply the same hinge loss approach as before, penalizing linearly based on the distance to the boundary and including

a margin of 1. We then end up with the loss

$$L_{\text{DOM}}(Z, \boldsymbol{c}) = \sum_{i=1}^{|Z|}(1 - c_i) \min_{j \neq i}\left(\sum_{k=1}^{d'} \max\left(0, 1 + z_{i,k} - z_{j,k}\right)\right) \tag{6.6}$$

The effect of this loss on a non-chosen object is depicted in Figure 6.7, where we show a set of points in the embedding space.

The red dot in the center represents the non-chosen object under consideration, which in the current situation is not dominated. In order for it to be dominated, a point would need to be moved into the region shaded in red. Due to the margin, we additionally require that the region marked with the dotted line needs to be reached. The point shown in blue is the one closest to dominating the red point and penalized based on its distance towards the dotted line (shown using the dashed blue arrow).

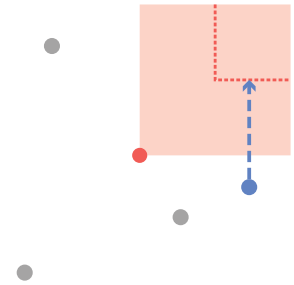


Figure 6.7.: Visualization of the effect of the loss terms L_{DOM} in $\mathcal{Z}$ space. The red point was *not* chosen. The point depicted with a blue arrow is penalized.

Combining the loss functions L_{PO} and L_{DOM} we are almost ready to specify an overall loss function. Already, we can guarantee that an embedding φ that achieves a loss of 0 across all choice tasks Q on both loss functions is a valid Pareto-embedding (see Section 6.2.2 for the proof).

The loss function we have defined so far is invariant to shifting and scaling. Formally, that means that for all $Z \in \mathcal{Z}$: $\text{Par}(Z) = \text{Par}(Z + a)$ with $a \in \mathbb{R}$ (invariance to shifting) and $\text{Par}(Z) = \text{Par}(Z \cdot a)$ with $a > 0$ (invariance to scaling). The implication of this is that a model learned with this loss function is not identifiable, which could cause issues during the optimization process. This can be solved using an $L_{2,1}$ norm, i.e., $\sum_{i=1}^{|Z|} \|\boldsymbol{z}_i\|_2$, as a penalty for the embedded points that encourages points to be as close to 0 as possible. Therefore a shift or scale operation, which takes a set of objects away from 0 is penalized. Other norms could be used as well if they are shift- and scale-invariant. The margin of the hinge losses has the added benefit of preventing the points from collapsing to 0.

Combining all terms using a convex combination, we end up with

$$L(Z, \boldsymbol{c}) = \alpha L_{\text{PO}}(Z, \boldsymbol{c}) + \beta L_{\text{DOM}}(Z, \boldsymbol{c}) + \gamma \sum_{i=1}^{|Z|} \|\boldsymbol{z}_i\|_2\,, \tag{6.7}$$

where $\alpha, \beta, \gamma \geq 0$ are positive, real-valued coefficients that have to be set. Typically, these would be treated as hyperparameters and optimized alongside all other hyperparameters in a validation loop.

Note, how L_{PO} and L_{DOM} use the $\max(0, x)$ function to accomplish a one-sided penalization. To also have some loss signal when $x < 0$, it can be worthwhile to utilize a *softplus* function $\log(1 + e^x)$ instead. It is differentiable and has a nonzero gradient everywhere, which makes it much more stable in gradient-based optimization. In the experimental section (Section 6.3), we are going to evaluate both variants of the loss to see if there is a significant difference in performance.

softplus

On Structure-Preserving Losses With the loss (6.7), we are now able to learn Pareto-embeddings φ. However, no restrictions are placed on the function φ. We could, for example, instantiate it using an arbitrary nonlinear function that maps points into an embeddings space in such a way that the observed choices

can be represented, but local and global structures in the original object space are not preserved. Why this might be a useful consideration, we can see by the popularity of *factor analysis* in psychology, economics and other disciplines [Chi06; Mul09]. In factor analysis, the goal is to identify latent factors that explain a certain number of observed variables. Since the number of latent factors is usually assumed to be smaller than the number of observed variables, factor analysis belongs to the category of dimensionality reduction approaches. How the observed variables influence each latent factor is an important consideration for the interpretability. Consider, for example, a healthcare study examining daily exercise, vegetable intake, alcohol consumption, and smoking habits for a population of patients. Using factor analysis, we may end up with a factor having positive weights for daily exercise and vegetable intake, and negative weights for alcohol consumption and smoking habits. We may interpret this factor to represent whether a patient is adhering to a "healthy lifestyle".

In our setting, we observe sets of objects, each characterized by observable variables. The difference is that we are not primarily interested in correlations between variables but in the relation of the variables to the observed choices. For the final embeddings, we may still want to require that local and global structures are preserved. A straightforward approach to accomplish this is to add a multidimensional scaling term [Mea92]:

$$L_{\mathrm{MDS}}(Z, D) = \left(\sum_{i,j} \left(d_{i,j} - \|z_i - z_j\| \right)^2 \right)^{1/2}, \tag{6.8}$$

where $D = (d_{i,j})_{1 \leq i,j \leq |Q|}$ is the pairwise distance matrix that results from evaluating an appropriate metric on pairs of objects in Q. The loss can be added to the overall loss function (6.7) with an additional weight. In the experiments later in this chapter, we run experiments with and without $L_{\mathrm{MDS}}(Z, D)$ to ascertain the impact it has on overall performance.

Of course, having defined a loss function is just one part of being able to learn Pareto-embeddings from data. The next step is defining a parametric model for the function φ that we can use in conjunction with our surrogate loss. Here, we chose neural networks as a flexible and differentiable model class. In the following section, we will introduce two neural network architectures and elaborate on the specific requirements that guided their design.

6.2.1.1. Neural Network Architectures

Before we go into detail on which specific architecture we chose and why, it makes sense to take a step back and recall what our objective is. The goal is to learn a Pareto-embedding of the form $\varphi : \mathcal{X} \to \mathcal{Z}$ such that for a given choice task Q we can predict choices by evaluating $\mathrm{Par}(\varphi(Q))$. Here, the extension of φ from one object to the complete set of objects was done implicitly. More generally, it can be useful to let $\varphi : \bigcup_k \mathcal{X}^k \to \bigcup_k \mathcal{Z}^k$ such that *context-dependence* can be captured as well. Concretely, the sought model maps each object $\boldsymbol{x} \in Q$ to a d'-dimensional vector of utilities.

context-dependence: Page 112

optional requirements

While the signature of the model is a mandatory requirement, there are a few *optional requirements* that we choose to impose on a model for Pareto-embeddings. First of all, it should be (almost everywhere) differentiable, allowing the model

to be trained using our surrogate loss function in an end-to-end manner. Secondly, the mapping from objects to utilities should be able to be nonlinear. For some straightforward situations, a linear model may suffice. For example, this would be the case when the utility function is merely a weighted sum of monotone criteria. However, in real-world choice problems, the relationship between objects and utilities is often far from linear. Recall the earlier example of a choice problem where the objects are images. In such a setting, it is unlikely that a linear function, which merely considers pixel values, could capture and express complex semantic preferences. Lastly, it is common in decision theory to work with positive utilities, which is why we require that $\mathcal{Z} \subseteq \mathbb{R}^{d'}_{\geq 0}$.

Keeping the requirements in mind, we propose two neural network architectures. One context-independent architecture and one context-dependent. Neural networks as a model class fit the requirements well since they are differentiable (almost everywhere), can learn arbitrary nonlinear mappings, and the output dimensionality can easily be set to fit our embedding setting.

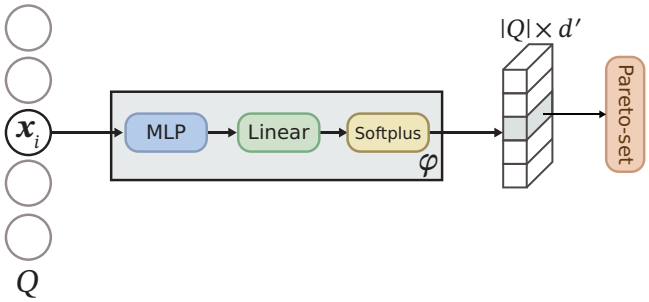


Figure 6.8.: Architecture of the general embedding approach [PH20]. A multi-layer perceptron and a final linear layer map each object into the embedding space, while the softplus ensures positive utilities. The Pareto-set of the points will be used to obtain a choice set C.

Context-independent architecture

Context-Independent Architecture The architecture is shown in Figure 6.8. Each object $x_i \in Q$ is passed through a fully-connected multi-layer perceptron (MLP) (shown in blue) with d input units. The number of hidden units and layers are treated as hyperparameters that should be optimized for the data at hand using validation. Next, the output of the MLP is processed by a final linear layer with d' units to produce the raw embedding, and a softplus nonlinearity is applied to make the output positive. As a result of these transformations, a $|Q| \times d'$ matrix is produced. From this matrix, we can compute the set of Pareto-optimal points using standard algorithms [DZ04]. The worst-case complexity of this operation is $\mathcal{O}(|Q|^2 \cdot d')$, which you can achieve by iterating over all pairs of points and checking for dominance. However, Dai and Zhang [DZ04] present an algorithm that, in expectation, runs in time linear in the number of points. With the Pareto-set of this matrix, we have the predicted choice set. Additional technical details are discussed in Section 6.3.1.

Pairwise Pareto-embeddings As we have seen in Chapter 5, context-sensitive approaches such as FETA and FATE are able to leverage the object context, which refers to all objects present in a given choice. The presence and properties of these objects can affect the final choice. This allows these approaches to capture various context effects present in the data.

In the context of Pareto-embeddings, which are context-independent as defined so far, context-sensitivity is interesting for several reasons. First of all,

we may want to discern the difference in performance between a context-independent mapping and one that takes the object context into consideration. Secondly, we can make a comparative analysis of Pareto-embeddings (both context-dependent and -independent) and other context-sensitive approaches such as FETA and FATE. This comparison enables us to determine which datasets exhibit context-dependence, those that can be effectively represented by Pareto-embeddings, and those that display both characteristics. Furthermore, it offers the potential for improving the choice accuracy of the embedding approach.

That is why we want to explore a context-sensitive Pareto-embedding method here. We could use any of the mechanisms introduced in Chapter 5 to make the embedding context-sensitive. Inspired by the simplicity of pairwise FETA (see Section 5.3.1), we opt to implement a straightforward pairwise decomposition approach. The embedding function ψ is then decomposed as follows:

$$\psi(\boldsymbol{x}, Q) := \frac{1}{|Q|} \sum_{\boldsymbol{y} \in Q} g(\boldsymbol{x}, \boldsymbol{y}), \tag{6.9}$$

where $g : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{d'}$ is a function that maps a point $\boldsymbol{x} \in \mathcal{X}$ into the embedding space in the context of another object $\boldsymbol{y} \in \mathcal{X}$. The representations are then aggregated using the arithmetic mean.

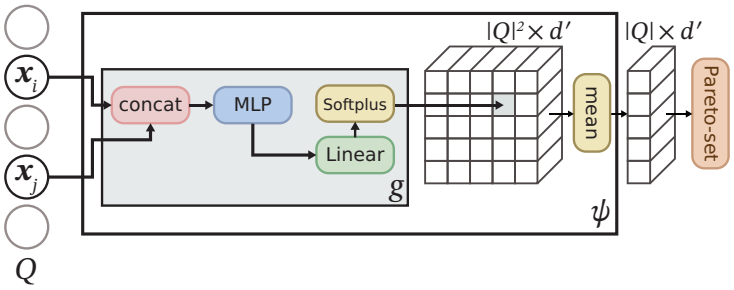


Figure 6.9.: Neural network architecture for the pairwise Pareto-embedding. The pairwise embedding function $g(\cdot, \cdot)$ is evaluated on all pairs of objects of the choice task Q. The $|Q|$ embeddings are averaged for each object to arrive at the final representation of the object.

The corresponding neural network architecture is shown in Figure 6.9. It was purposefully designed to be similar to the context-independent architecture. The main difference is that for a given set of objects we first generate all pairs, which are then each concatenated and passed through network g. As before, network g has a final linear layer with d' output units, which are transformed using a softplus nonlinearity to ensure the output is positive. We end up with $|Q|^2$ embedding vectors, which are averaged for each object, according to (6.9), to arrive at the final embeddings. The Pareto-set and, hence, the prediction can then be computed.

Computational Complexity The main scaling dimension for our setting is the number of objects. The computational complexity of the pairwise Pareto-embedding architecture is higher compared to the context-independent one due to its reliance on evaluating every possible pair of objects. Therefore, at training and test time, the pairwise method has a runtime that scales quadratically in the number of objects. In contrast, the context-independent architecture scales linearly in the number of objects in runtime. In this case, the dominating factor is the algorithm that computes the set of Pareto-optimal points. This algorithm has a time complexity of $\mathcal{O}(m \log m)$ for up to 3 embedding dimensions or $\mathcal{O}(d' m^2)$ in general, where m represents the number of

objects [KLP75]. However, there exist algorithms that can compute the set of Pareto-optimal points in *expected linear time* [BCL93; DZ04].

After establishing a surrogate loss function intended for generating valid Pareto-embeddings, along with corresponding neural network architectures trainable with this loss function, the logical next question is: can we provide a theoretical proof that the loss function is, in fact, suitable? In the following section, we will consider the theoretical properties of Pareto-embeddings and answer this question affirmatively.

6.2.2. Theoretical Properties of Pareto-Embeddings

A question that is central in the analysis of choice models is whether a given choice model is able to *represent* a given set of choice functions or even all possible choice functions. A parametric choice model $\hat{c}_{\boldsymbol{\theta}} : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ can represent a ground truth choice function c, if there exists a parameter $\boldsymbol{\theta}$ such that, for all choice tasks $Q \in 2^{\mathcal{X}}$ it holds that $\hat{c}_{\boldsymbol{\theta}}(Q) = c(Q)$. In the case of Pareto-embeddings, the choice function is induced by the embedding function φ, so $\boldsymbol{\theta} \coloneqq \varphi$. The set of Pareto choice functions is a subset of the set of all possible choice functions. Therefore, it is natural to ask which set of choice functions can be represented by Pareto choice functions and how this set can be characterized.

As a first step, we will ask the simple question of whether Pareto-embeddings are powerful enough to represent *all* choice functions. This, as it turns out, is not the case:

Proposition 6.2.2 *Let $\mathcal{X}$ be a set of objects with $|\mathcal{X}| \geq 3$. There exists a choice function $c : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$, which cannot be represented by a Pareto-embedding.*

Proof. We will construct such a choice function as follows: Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ be two objects for which $c(\{\boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{x}\}$ holds and for which a set $Q \supset \{\boldsymbol{x}, \boldsymbol{y}\}$ exists with $\boldsymbol{y} \in c(Q)$.

For a valid Pareto-embedding $\varphi : \mathcal{X} \to \mathcal{Z}$ with $\mathcal{Z} \subseteq \mathbb{R}^{d'}$, the first choice would require that $\boldsymbol{x} \succ_{\varphi} \boldsymbol{y}$ holds, i.e., for all $k \in [d']$ we have that $\varphi(\boldsymbol{x})_k \geq \varphi(\boldsymbol{y})_k$ and $\varphi(\boldsymbol{x})_k > \varphi(\boldsymbol{y})_k$ for at least one such k. In order for $\boldsymbol{y}$ to be in $c(Q)$, it needs to hold that for all objects $\boldsymbol{z} \in Q$ that $\boldsymbol{z} \not\succ_{\varphi} \boldsymbol{y}$. Since we know that $\boldsymbol{x} \in Q$ and $\boldsymbol{x} \succ_{\varphi} \boldsymbol{y}$, it is not possible for any function φ to produce that choice. □

As we can see, there are simple choice functions that cannot be explained using a Pareto-embedding. The next step will be to find a characterization of the set of choice functions representable by Pareto-embeddings. This we will do as follows: First, we will define an order relation on the set of objects $\mathcal{X}$ using any choice function c. Then, we will define certain properties for this order relation and show that a choice function that obeys these properties can be represented by Pareto-embeddings.

Defining an order relation on the level of choices can be done in several ways and has been a field of study in choice modeling for a long time [Sam38]. In this context, one also speaks of *revealed preferences* because the choices reveal

revealed preferences

the "true" preferences of the decision maker. Preference relations are defined in several levels of strictness, but here, we only need a very weak notion:

Definition 6.2.3 *Let $c\colon 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ be a choice function. We will say that $\boldsymbol{x} \in \mathcal{X}$ is preferred to $y \in \mathcal{X}$ by c (denoted $\boldsymbol{x} \succ_c \boldsymbol{y}$), if and only if*

$$\exists Q \in 2^{\mathcal{X}} \text{ with } y \in Q, \boldsymbol{x} \notin Q, y \in c(Q), y \notin c(Q \cup \{\boldsymbol{x}\}) \text{ and } \boldsymbol{x} \in c(Q \cup \{\boldsymbol{x}\}).$$

In other words, we say that an object $\boldsymbol{x}$ is preferred to $\boldsymbol{y}$ if adding $\boldsymbol{x}$ to at least one subset causes y to be no longer chosen in the resulting larger set. Note that this preference is purely descriptive in that it does not constrain the choice function in any way. Therefore, it is possible that $\boldsymbol{x} \succ_c \boldsymbol{y}$ and $\boldsymbol{y} \succ_c \boldsymbol{x}$ hold simultaneously or that neither hold at all.

Many properties or axioms have been defined on the level of revealed preferences that limit the expressivity of the underlying choice function, but at the same time, make the choices more plausible from a rationality perspective (see Section 2.2 for a thorough discussion). Here, we are seeking the characteristic property of exactly those choice functions representable by Pareto-embeddings.

Pareto-property

We will call this property the *Pareto-property* and define it as follows:

Definition 6.2.4 *A choice function $c\colon 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ satisfies the Pareto-property, if and only if:*

(i) $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$*: If* $\boldsymbol{x} \succ_c \boldsymbol{y}$*, then* $\forall Q \in 2^{\mathcal{X}}$ *with* $Q \supseteq \{\boldsymbol{x}, \boldsymbol{y}\}$*:* $y \notin c(Q)$*, and*
(ii) $\succ_c$ *is transitive.*

asymmetry

Property (i) ensures that the relation $\succ_c$ is *asymmetric* and with (ii) we prevent any kind of preferential cycles. The astute reader may notice that the Pareto-property makes $\succ_c$ a *strict partial order*, which we will use to prove the main theorem.

strict partial order

Comment

Note, that asymmetry of $\succ_c$ does *not* imply property (i). This is easy to see with an example: Let c be a choice function and $\mathcal{X} := \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\}$ such that $c(\{\boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{x}\}$ holds and for all other sets $Q \in 2^{\{\mathcal{X}\}}$ we have $c(Q) = Q$. It is clear that we have $\boldsymbol{x} \succ_c y$. Since $\boldsymbol{x}$ is always chosen when available, we also have $\boldsymbol{y} \not\succ_c \boldsymbol{x}$, hence $\succ_c$ is asymmetric. However, $\succ_c$ is not obeying (i), because $\boldsymbol{y} \in c(\{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}\})$. Therefore, Definition 6.2.4 is *not* equivalent to requiring that $\succ_c$ is a strict partial order.

We can now show that the set of choice functions possessing the Pareto-property precisely corresponds to the set of choice functions that can be represented by Pareto-embeddings. Roughly speaking, the proof idea will be as follows: If there exists a Pareto-embedding, we can show that $\succ_c$ and $\succ_\varphi$ coincide, and we only have to show that $\succ_\varphi$ fulfills the criteria of Definition 6.2.4. For the converse direction, we will exploit that $\succ_c$ is a strict partial order for which we know from existing results in the literature that an embedding has to exist. In preparation for Theorem 6.2.8 and the corresponding proof, we will proceed using the following steps:

1. Equate $\succ_c$ and $\succ_\varphi$ with Lemma 6.2.5.

2. State an existing theorem by Hiraguchi [Hir55] that guarantees the existence of a partial order embedding (see Lemma 6.2.7).
3. Prove the equivalence of a choice function having the Pareto-property and the existence of a Pareto-embedding (see Theorem 6.2.8).

To begin, we will make use of the fact that the dominance relation (as defined in Section 6.2) also defines an order on objects. Let $\succ_\varphi$ be the dominance relation resulting from a Pareto-embedding φ, then we can equate it to $\succ_c$ using the following lemma:

Lemma 6.2.5 *Let $c\colon 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ be a choice function for which a Pareto-embedding $\varphi\colon \mathcal{X} \to \mathbb{R}^{d'}$ with $c(Q) = \mathrm{Par}_\varphi(Q)$ for all $Q \in 2^{\mathcal{X}}$ exists. Then for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$: $\boldsymbol{x} \succ_\varphi \boldsymbol{y}$, if and only if $\boldsymbol{x} \succ_c \boldsymbol{y}$.*

Proof. Let c be an arbitrary choice function for which a Pareto-embedding exists, and φ the corresponding embedding. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ be arbitrary with $\boldsymbol{x} \succ_\varphi \boldsymbol{y}$. Let $Q \coloneqq \{\boldsymbol{y}\}$ be a task, then $c(Q) = \boldsymbol{y}$ and $c(Q \cup \{\boldsymbol{x}\}) = \{\boldsymbol{x}\}$ must hold by definition of φ being a valid Pareto-embedding for c. Thus, it also holds that $\boldsymbol{x} \succ_c \boldsymbol{y}$.

Now let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ be arbitrary with $\boldsymbol{x} \succ_c \boldsymbol{y}$. Let $Q \in 2^{\mathcal{X}}$ be one set for which $\boldsymbol{y} \in Q$, $\boldsymbol{x} \notin Q$, $\boldsymbol{y} \in c(Q)$ and $\boldsymbol{y} \notin c(Q \cup \{\boldsymbol{x}\})$. Then by

$$\begin{aligned} c(Q) &= \mathrm{Par}_\varphi(Q) \\ &= \{\boldsymbol{x} \in Q \mid \nexists \boldsymbol{y} \in Q\colon\ \forall i \in [d]\colon\ \varphi(\boldsymbol{y})_i \geq \varphi(\boldsymbol{x})_i \text{ and } \exists i \in [d]\colon\ \varphi(\boldsymbol{y})_i > \varphi(\boldsymbol{x})_i\} \\ &= \{\boldsymbol{x} \in Q \mid \nexists \boldsymbol{y} \in Q\colon\ \boldsymbol{y} \succ_\varphi \boldsymbol{x}\} \end{aligned}$$

and $\boldsymbol{y} \in c(Q)$ we know that there does not exist a $\boldsymbol{y}' \in Q$ with $\boldsymbol{y}' \succ_\varphi \boldsymbol{y}$. By $\boldsymbol{y} \notin c(Q \cup \{\boldsymbol{x}\})$ we know that there exists a $\boldsymbol{y}' \in Q \cup \{\boldsymbol{x}\}$ with $\boldsymbol{y}' \succ_\varphi \boldsymbol{y}$. Since only $\boldsymbol{x}$ was added to the set, it holds that $\boldsymbol{y}' = \boldsymbol{x}$ and therefore $\boldsymbol{x} \succ_\varphi \boldsymbol{y}$. □

Example: Intersection of linear orders

Let $S = \{(\boldsymbol{x}, \boldsymbol{z}), (\boldsymbol{y}, \boldsymbol{z})\}$ be a partial order. Then with $L_1 = \boldsymbol{x} \succ \boldsymbol{y} \succ \boldsymbol{z}$ and $L_2 = \boldsymbol{y} \succ \boldsymbol{x} \succ \boldsymbol{z}$ we have $L_1 \cap L_2 = S$.

In the second step, we will state a well-known result from the field of *order embeddings*. We are going to utilize the notion of a partial order dimension, which is defined as follows: Let S be a partial order defined on a set $\mathcal{X}$. Then the *dimension* $\dim(S)$ is the minimum number of linear orders such that their intersection is S [DM41]. For us, the more interesting characterization is the following:

order embeddings

Definition 6.2.6 ([Ore62; Yan82]) *Let S be a partial order defined on a set $\mathcal{X}$. Then the dimension $\dim(S)$ is the minimum d' for which there exists an embedding $\pi\colon \mathcal{X} \to \mathbb{R}^{d'}$, such that $S(\pi) = S$, where $S(\pi)$ is a partial order on $\mathcal{X}$ defined by: $(\boldsymbol{x}, \boldsymbol{y}) \in S(\pi)$ if and only if $\pi(\boldsymbol{x})_i > \pi(\boldsymbol{y})_i$ for all $i \in [d']$.*

dimension

In other words, for any given partial order, we are able to assign a finite, minimal dimension number that characterizes the "complexity" of representing that order. Moreover, this number is the dimensionality of an embedding function $\pi\colon \mathcal{X} \to \mathbb{R}^{d'}$, which is a fact we will use in the proof of the main theorem. The fact that this number is always finite was proven by Hiraguchi [Hir55]:

Lemma 6.2.7 (Theorem 8.3 by Hiraguchi [Hir55]) *Let $\mathcal{X}$ be a finite set with $|\mathcal{X}| > 3$ and S any partial order defined on $\mathcal{X}$. Then* $\dim(S) \leq |\mathcal{X}|/2$.

We are now ready to state and prove the main theorem:

Theorem 6.2.8 *Let $\mathcal{X}$ be a finite set. A choice function $c : 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ satisfies the Pareto-property if and only if there exists an embedding $\varphi : \mathcal{X} \to \mathbb{R}^{d'}$, such that $c(Q) = \mathrm{Par}_{\varphi}(Q)$ for all $Q \in 2^{\mathcal{X}}$.*

Proof. Let $\mathcal{X}$ be any finite set. We will begin by letting c be any choice function that has the Pareto-property ((i) and (ii) of Definition 6.2.4 hold). Further let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ be objects such that $\boldsymbol{x} \succ_c \boldsymbol{y}$ holds. By (i) we know that for all $Q \in 2^{\mathcal{X}}$ with $Q \supseteq \{\boldsymbol{x}, \boldsymbol{y}\}$ we have that $\boldsymbol{y} \notin c(Q)$. Therefore, there is no $Q' \subset Q$ with $\boldsymbol{x} \in Q'$, $\boldsymbol{y} \notin Q'$, $\boldsymbol{x} \in c(Q')$, $\boldsymbol{x} \notin c(Q)$ and $\boldsymbol{y} \in c(Q)$. Therefore, $\succ_c$ is asymmetric. Additionally, observe that $\succ_c$ is irreflexive, which is why it forms a *strict partial order*.

By Lemma 6.2.7, it follows that the dimension of this partial order is $\leq |\mathcal{X}|/2$ and there exists an embedding φ into $\mathbb{R}^{|\mathcal{X}|/2}$. It remains to show that φ is a valid Pareto-embedding. From Definition 6.2.6 we know that φ defines a partial order $S(\varphi) = \succ_c$. and $(\boldsymbol{x}, \boldsymbol{y}) \in S(\varphi)$ if and only if $\varphi(\boldsymbol{x})_i > \varphi(\boldsymbol{y})_i$ for all $i \in [d']$. Let $Q \in 2^{\mathcal{X}}$ be any set of objects and $c(Q) \subseteq Q$ the corresponding choices. Let $\boldsymbol{x}$ be any object in Q.

If $\boldsymbol{x} \in c(Q)$, then for any other object $\boldsymbol{y} \in Q$, $\boldsymbol{y} \neq \boldsymbol{x}$ we have that $(\boldsymbol{y}, \boldsymbol{x})$ is not contained in $S(\varphi)$. As a consequence, there exists an index $i \in [d']$ such that $\varphi(\boldsymbol{x})_i \geq \varphi(\boldsymbol{y})_i$. Therefore, $\boldsymbol{x}$ is contained in $\mathrm{Par}_{\varphi}(Q)$.

In case $\boldsymbol{x} \notin c(Q)$, then there exists an object $\boldsymbol{y} \in Q$ such that $(\boldsymbol{y}, \boldsymbol{x}) \in S(\varphi)$. This entails that $\varphi(\boldsymbol{y})_i > \varphi(\boldsymbol{x})_i$ for all $i \in [d]$ and thus $\boldsymbol{x} \notin \mathrm{Par}_{\varphi}(Q)$. We can conclude that φ is a valid Pareto-embedding.

To prove the converse direction, assume that there exists an embedding $\varphi : \mathcal{X} \to \mathbb{R}^{d'}$ with $c(Q) = \mathrm{Par}_{\varphi}(Q)$ for all $Q \in 2^{\mathcal{X}}$. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ be arbitrary with $\boldsymbol{x} \succ_c \boldsymbol{y}$ and therefore $\boldsymbol{x} \succ_{\varphi} \boldsymbol{y}$ (by Lemma 6.2.5). Then for any $Q \in 2^{\mathcal{X}}$ with $\{\boldsymbol{x}, \boldsymbol{y}\} \subseteq Q$ it holds that $\boldsymbol{y} \notin c(Q)$, since $\boldsymbol{x} \succ_{\varphi} \boldsymbol{y}$ implies $\boldsymbol{y} \notin \mathrm{Par}_{\varphi}(Q)$. Therefore, (i) of Definition 6.2.4 holds.

To prove (ii), it suffices to prove that $\succ_{\varphi}$ is transitive. Assume that $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$ with $\boldsymbol{x} \succ_{\varphi} \boldsymbol{y}$ and $\boldsymbol{y} \succ_{\varphi} \boldsymbol{z}$. It follows that for all $i \in [d']$ we have $\varphi(\boldsymbol{x})_i \geq \varphi(\boldsymbol{y})_i$, $\varphi(\boldsymbol{y})_i \geq \varphi(\boldsymbol{z})_i$ and by transitivity of $\geq$, $\varphi(\boldsymbol{x})_i \geq \varphi(\boldsymbol{z})_i$. We know that there exist $i, j \in [d']$ such that $\varphi(\boldsymbol{x})_i > \varphi(\boldsymbol{y})_i$ and $\varphi(\boldsymbol{y})_j > \varphi(\boldsymbol{z})_j$. If $i = j$, $\varphi(\boldsymbol{x})_i > \varphi(\boldsymbol{z})_i$ follows by transitivity of $>$. If $i \neq j$, then we have $\varphi(\boldsymbol{x})_i > \varphi(\boldsymbol{y})_i \geq \varphi(\boldsymbol{z})_i$ and $\varphi(\boldsymbol{x})_j \geq \varphi(\boldsymbol{y})_j > \varphi(\boldsymbol{z})_j$. In either case, $\varphi(\boldsymbol{x})_i > \varphi(\boldsymbol{z})_i$ follows and it holds that $\boldsymbol{x} \succ_{\varphi} \boldsymbol{z}$. Therefore, $\succ_{\varphi}$ and $\succ_c$ are transitive and (ii) of Definition 6.2.4 holds. □

To recap, the main idea of the proof is that Pareto-embeddings are a certain kind of order embeddings, which allows us to reuse existing results in that field, namely that such an embedding always exists for a strict partial order on a finite set. It is known that for partial orders of dimension 2, there exists an efficient algorithm to compute the corresponding embedding [DM41; PLE71]. For higher-dimensional orders, however, it is $\mathcal{NP}$-complete to decide that

the dimension is at most k [Yan82]. Interestingly, characterizing the properties of higher-dimensional partial orders is an ongoing endeavor and could lead to new insights [Qi16; HBG22].

Suitability of Surrogate Loss Function Now we want to analyze the surrogate loss functions we proposed in Section 6.2.1. More concretely, we want to know if minimizing the loss function (6.7) results in an embedding that satisfies the properties of a Pareto-embedding. Note that we are not speaking about empirical risk minimization here but rather about the case when the loss is applied to all possible subsets of the object space. For the sake of simplicity, we will consider the loss function (6.7) with the regularization coefficient γ set to 0.

Suitability of Surrogate Loss Function

Theorem 6.2.9 *Let $c\colon 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ be a choice function with the Pareto-property, and let its dimension be d'. If $\varphi\colon \mathcal{X} \to \mathbb{R}^{d'}$ achieves a loss $L(\varphi(Q), c(Q)) = 0$ for all $Q \subseteq \mathcal{X}$, then φ is a Pareto-embedding of c.*

Proof. Let $\varphi\colon \mathcal{X} \to \mathbb{R}^{d'}$ a function parametrized by θ. Let $Q \subseteq \mathcal{X}$ be arbitrary and $Z = \varphi(Q)$ be the matrix resulting from applying φ to all objects in Q. Assume that $L(Z, c(Q)) = 0$ for any such arbitrarily chosen Q.

It is easy to see that $L_{\text{DOM}} \geq 0$ and $L_{\text{PO}} \geq 0$ and therefore $L \geq 0$ in general. As for L_{PO}, it is clear that all terms in the sum are 0 where $c_j = 0$. Let $\boldsymbol{x} \in c(Q)$ be an arbitrary, chosen object, and let $j \in [|Q|]$ be its index. We know that L_{DOM} only penalizes non-chosen objects, which is why we only have to consider L_{PO} here. Since $L_{\text{PO}}(Z, c(Q)) = 0$, it needs to hold that

$$\begin{aligned} & \min_{1 \leq k \leq d'} (1 + z_{i,k} - z_{j,k}) \leq 0 \\ \Leftrightarrow \quad & \min_{1 \leq k \leq d'} (z_{i,k} - z_{j,k}) \leq -1 < 0 \end{aligned}$$

for all $i \neq j$. That means for all objects $\boldsymbol{y} \neq \boldsymbol{x}$ there exists a dimension $k \in [d']$ such that $\varphi(\boldsymbol{x})_k > \varphi(\boldsymbol{y})_k$ and therefore $\boldsymbol{x} \in \text{Par}_{\varphi}(Q)$.

Now let $\boldsymbol{x} \in Q \setminus c(Q)$ and let $i \in [|Q|]$ be its index. Since L_{PO} only penalizes chosen objects, we now consider L_{DOM}. As before, we know that $L_{\text{DOM}}(Z, c(Q)) = 0$ and therefore

$$\min_{j \neq i} \left(\sum_{k=1}^{d} \max(0, 1 + z_{i,k} - z_{j,k}) \right) = 0$$

Each term in the sum is 0, only if $z_{i,k} - z_{j,k} \leq 1 < 0$. Thus, the complete sum can only be 0, if $z_{i,k} < z_{j,k}$ for all $k \in [d']$. We, therefore, know that there exists an object $\boldsymbol{y} \neq \boldsymbol{x}$ with $\varphi(\boldsymbol{x})_k < \varphi(\boldsymbol{y})_k$ for all $k \in [d']$ and $\boldsymbol{x} \notin \text{Par}_{\varphi}(Q)$. Consequently, it holds that $\text{Par}_{\varphi}(Q) = c(Q)$ and since the choice of Q was arbitrary, φ is a Pareto-embedding. □

We may wonder whether the converse direction is also true, i. e., if all Pareto-embeddings φ have a loss of 0. It is easy to see that this is not directly the case. A simple counterexample is the following: Let $\mathcal{X} = \{\boldsymbol{x}, \boldsymbol{y}\}$ with $\varphi(\boldsymbol{x}) = (1, 0)$ and $\varphi(\boldsymbol{y}) = (0, 0)$. The choice function is defined as $c(\{\boldsymbol{x}, \boldsymbol{y}\}) = \text{Par}_{\varphi}(\{\boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{x}\}$. Thus, φ is a valid Pareto-embedding, however $L(\varphi(\{\boldsymbol{x}, \boldsymbol{y}\}), c(\{\boldsymbol{x}, \boldsymbol{y}\})) = 1$, since

L_{DOM} requires a margin of 1 in every dimension. That being said, if no pair of the embedded points lie on lines parallel to the axes, simply scaling the embedding by an appropriate constant will yield a loss of 0. More formally, we say that two points $z, z' \in \mathbb{R}^{d'}$ are *coaxial*, if there exists a $k \in [d']$ such that $z_k = z'_k$.

Proposition 6.2.10 *Let $c\colon 2^{\mathcal{X}} \to 2^{\mathcal{X}}$ be a choice function with the Pareto-property, and let its dimension be d'. If φ is a Pareto-embedding of c, then for all $Q \subseteq \mathcal{X}$ where no two objects are coaxial, there exists a constant $a^* \in \mathbb{R}$ such that $L(a^* \cdot \varphi(Q), c(Q)) = 0$ for all $Q \subseteq \mathcal{X}$.*

Proof. Let c be a choice function with the Pareto-property and φ a Pareto-embedding of c. Let $Q \subseteq \mathcal{X}$ be arbitrary, but with the property that no two objects are coaxial.

Let $\boldsymbol{x} \in c(Q)$ be an arbitrary, chosen object and $j \in [|Q|]$ be its index. Since φ is a Pareto-embedding, we know that for all objects $\boldsymbol{y} \in Q$ with $\boldsymbol{y} \neq \boldsymbol{x}$ and i its index, there exists a $k \in [d']$ such that $z_{j,k} > z_{i,k}$. It holds that

$$\delta_{ij} \coloneqq \min_{1 \leq k \leq d'} (z_{i,k} - z_{j,k}) < 0 \,,$$

and since the difference δ_{ij} is negative, we can find a constant $a \in \mathbb{R}^+$ such that $a\delta_{ij} < -1$. The loss term for this pair in L_{PO} is then 0.

Now for $\boldsymbol{x} \notin c(Q)$ with i its index, we know that there exists a $\boldsymbol{y} \in Q$ with $\boldsymbol{y} \succ_{\varphi} \boldsymbol{x}$ and j its index. In addition, their embeddings are not coaxial, hence $z_{i,k} < z_{j,k}$ for all $k \in [d']$. Let

$$\delta_{ij} \coloneqq \max_{1 \leq k \leq d'} (z_{i,k} - z_{j,k}) < 0$$

be the smallest difference. Then we can find a constant $a \in \mathbb{R}^+$ such that $a\delta_{ij} < -1$ and the loss term for object $\boldsymbol{x}$ in L_{DOM} is 0.

Let a^* be the maximum of all such constants a. Then it holds that $L(a^* \cdot \varphi(Q), c(Q)) = 0$ for all $Q \subseteq \mathcal{X}$. □

Learning from Pairwise Comparisons

Learning from Pairwise Comparisons As we have seen in Theorem 6.2.8, learning a Pareto-embedding can be reduced to finding an order embedding of $\succ_c$. While c maps from sets of objects to subsets thereof, $\succ_c$ is a relation solely defined on *pairs* of objects. An interesting question, therefore, is whether it suffices to minimize our surrogate loss solely on sets $|Q| = 2$.

Proposition 6.2.11 *Assume for a given $\varphi\colon \mathcal{X} \to \mathbb{R}^{d'}$ and c a choice function with Pareto property, that $L(\varphi(Q), c(Q)) = 0$ holds for any subsets $Q \subseteq \mathcal{X}$ with $|Q| = 2$. Then it holds that $L(\varphi(Q), c(Q)) = 0$ for all subsets $Q \subseteq \mathcal{X}$.*

Proof. Assume that $L(\varphi(Q), c(Q)) = 0$ for all subsets Q of size 2. Assume there exists a Q with $|Q| > 2$ such that $L(\varphi(Q), c(Q)) > 0$. Then $L_{\text{PO}}(\varphi(Q), c(Q)) > 0$ or $L_{\text{DOM}}(\varphi(Q), c(Q)) > 0$.

Case 1 $L_{\text{PO}}(\varphi(Q), c(Q)) > 0$ (see Equation 6.5)

It follows that there exists $\boldsymbol{x} \in c(Q)$ (index j) and $\boldsymbol{y} \in Q$ (index i) such that

$$c_j \cdot \min_{1 \leq k \leq d'} (z_{i,k} - z_{j,k}) > -1 \, .$$

Since $\boldsymbol{x} \in c(Q)$, we have that c_j is equal to 1 here. However, we know that for $\{\boldsymbol{x}, \boldsymbol{y}\}$ the loss is $L(\varphi(\{\boldsymbol{x}, \boldsymbol{y}\}), c(\{\boldsymbol{x}, \boldsymbol{y}\})) = 0$ and since $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are identical, it follows that for the $\{\boldsymbol{x}, \boldsymbol{y}\}$ we have that $c_j = 0$. In other words, $\boldsymbol{x} \notin c(\{\boldsymbol{x}, \boldsymbol{y}\})$ and $\boldsymbol{y} \succ_c \boldsymbol{x}$. By the Pareto property (Definition 6.2.4) this implies that $\boldsymbol{x} \notin c(Q' \cup \{\boldsymbol{y}\})$ for all $Q' \in 2^{\mathcal{X}}$. In particular, $\boldsymbol{x} \notin c(Q)$ which contradicts the initial assumption.

Case 2 $L_{\text{DOM}}(\varphi(Q), c(Q)) > 0$ (see Equation 6.6)

Here it follows that there exist an object $\boldsymbol{x} \in Q$ (index i) with $\boldsymbol{x} \notin c(Q)$. Then

$$(1 - c_i) \min_{j \neq i} \Big(\sum_{k=1}^{d'} \max\big(0, 1 + z_{i,k} - z_{j,k}\big) \Big) > 0$$

Since $\boldsymbol{x}$ is not chosen, $(1 - c_i) = 1$. Therefore, we know that the sum is > 0 for all $\boldsymbol{y} \in Q$. Let $\boldsymbol{y} \in Q$ be *any* of the other objects (with index j). Thus for at least one dimension $k \in [d']$ we have that $1 + z_{i,k} - z_{j,k} > 0$. As before, by assumption we know that $L(\varphi(\{\boldsymbol{x}, \boldsymbol{y}\}), c(\{\boldsymbol{x}, \boldsymbol{y}\})) = 0$ and since $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are identical, it follows that $c_i = 1$ such that $(1 - c_i) = 0$ to make the loss 0. Therefore, $\boldsymbol{x}$ is chosen in the set $\{\boldsymbol{x}, \boldsymbol{y}\}$, which in conjunction with the Pareto property (Definition 6.2.4) implies that $\boldsymbol{y} \nsucc_c \boldsymbol{x}$. Since we did not restrict $\boldsymbol{y}$, this is true for all such $\boldsymbol{y} \in Q$.

It follows that for all proper subsets $Q' \subset Q$ with $\boldsymbol{x} \in Q'$ we have that $\boldsymbol{x} \in c(Q')$ (otherwise by Definition 6.2.3 $\boldsymbol{y} \succ_c \boldsymbol{x}$ would hold). Since $\boldsymbol{x} \notin c(Q)$ but $\boldsymbol{x} \in c(Q \setminus \{\boldsymbol{y}\})$ for all $\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}$, the preference $\boldsymbol{y} \succ_c \boldsymbol{x}$ must hold for all $\boldsymbol{y}$, which is a contradiction.

We conclude $L(\varphi(Q), c(Q))$ could not have been strictly greater than 0, and the proposition holds. □

In conclusion, we have shown that all choice functions with the Pareto-property can be represented by A Pareto-embedding. To do this, we have introduced a loss function that, when minimized, yields a Pareto-embedding. In addition, we do not need to minimize the loss across all possible tasks $Q \in 2^{\mathcal{X}}$, but it suffices to minimize it for all pairs of objects. This is advantageous since it makes the learning problem computationally more tractable in practice.

6.2.3. Related Work

The concept of using an order relation to define choice functions has been explored in the literature [MM07; MN07]. Manzini and Mariotti [MM07] define the set of *P-maximal elements* of a set Q, which is similar to our Pareto-choice. They, however, only use this set as a building block to define a singleton choice function. Masatlioglu and Nakajima [MN07] on the other hand treat the choice of a *subset* as the main output. Instead of using a strict partial order based on dominance, as in our case, Masatlioglu and Nakajima [MN07] introduce an even weaker notion of *elimination orders*. An elimination order is

elimination orders

A relation R on a set X is *negatively transitive* if for all $x, y, z \in X$: $\neg xRy, \neg yRz$ implies $\neg xRz$.

asymmetric and negatively transitive. Note that if we assume completeness, strict partial and elimination orders are equivalent. In addition, the authors introduce a (context-dependent) *comparable* set $\Omega(\boldsymbol{x}, Q)$ that contains all objects that $\boldsymbol{x}$ can be compared to in a given choice context. With these constituents, they can define a choice function

$$c_{\mathrm{ACE}}(Q) := \{\boldsymbol{x} \in Q \mid \nexists \boldsymbol{y} \in \Omega(\boldsymbol{x}, Q) \text{ such that } \boldsymbol{y}E\boldsymbol{x}\},$$

where E is a context-independent elimination order.

For example assume that we observe the choices $c(\{a, b, c\}) = \{a, b\}$ and $c(\{a, b\}) = \{a\}$. This choice behavior is impossible in Pareto-choice, since the first choice implies that neither $a \succ b$ nor $b \succ a$ hold, but the second one implies that $a \succ b$. By letting $a \notin \Omega(b, \{a, b, c\})$, we can achieve the observed choices using $c_{\mathrm{ACE}}(Q)$ in conjunction with the elimination order $\{aEb, aEc\}$. Certainly, the function Ω is doing a lot of heavy lifting here since it is a fully context-dependent function. The authors prove that representability by their model is equivalent to the axiom of choice by elimination (ACE) they define, which they view as the "minimum requirement" for rationality, i. e., the most flexible model for choice functions that do not admit preference cycles.

multi-self framework: Section 2.4.2

The multiple selves (or multi-self for short) framework introduced by Ambrus and Rozen [AR15] unifies several (singleton) choice models where a choice is made by aggregating multiple utilities produced by so-called selves. We already introduced the framework in detail in Section 2.4.2, so we will focus more on the similarities and differences between that framework and our Pareto-embeddings. Both approaches use context-independent utility functions as the basic building blocks for decision making. In the multi-self framework, these are then aggregated into one utility score for a given choice task with the goal of producing a singleton choice. In our case, we use the utilities directly as dimensions in a utility space and use the positions of the objects to compute a subset choice. Despite this, it is possible to encode Pareto-choice as an aggregator in the multi-self framework. We return a utility of 1 for every object contained within the Pareto-set and 0 otherwise. Using thresholding as explained in (5.3) of Chapter 5, we are then able to predict subset choices.

Our work on Pareto-embeddings has already resulted in promising follow-up approaches in the literature. Benavoli et al. [BAP23a; BAP23b] adapt Pareto-embeddings and the loss function to be useful in a probabilistic model. To this end, they propose a likelihood function such that a Gaussian process model can be inferred from subset choice data.

6.3. Empirical Evaluation

In the theoretical analysis, we investigated the suitability of our surrogate loss function to yield Pareto-embeddings, if the loss is 0. However, a crucial next step is to ascertain whether this loss function is effective in a practical setting as well. There, additional considerations arise, such as the ability of the loss function to produce a good target for gradient descent and whether the proposed neural network architectures are appropriate for the task. The primary objective of this empirical evaluation is to assess if our Pareto-embedding approaches can learn a viable model. If that is the case, we want to investigate

if Pareto-embeddings are able to outperform the more general FETA and FATE models on problems that are particularly suitable for them. If that is not the case, it would suffice to always go for general models instead.

We will focus the evaluation on answering the following *research questions* (short RQ):

P-RQ1: Are the Pareto-embedding architectures in conjunction with the surrogate loss able to learn useful embeddings at all?
P-RQ2: How do they compare to existing approaches, in particular our FETA and FATE architectures?
P-RQ3: Does the addition of the multi-dimensional scaling (MDS) term to the loss function improve performance?
P-RQ4: Does the softplus function improve the performance and robustness of the model?
P-RQ5: Will increasing the embedding dimensionality improve the performance of the model?

From these research questions, we derive appropriate experiments that can answer them adequately. To address P-RQ1 and P-RQ2, we make use of a suite of 14 benchmark datasets from the realm of multi-objective optimization (MOO) [ZDT00; Deb+05]. Each problem exhibits different properties, such as nonconvexity and discontinuities. On these problems, we then compare our Pareto-embedding approaches to existing ones. This allows us to ascertain whether viable models are learned and how well our approaches perform against competent baselines.

As for P-RQ3, P-RQ4, and P-RQ5, we perform ablation studies to gauge the effect of different components. To be more specific, we systematically investigate changes to the loss function and the embedding dimension. We test whether an additional MDS term is beneficial for the loss function. Furthermore, we examine the softplus function as an alternative to the standard max operation. The purpose of this is to provide the gradient-based optimization procedure with a more informative target. The ablation studies are done by re-executing the full set of experiments, which includes optimizing hyperparameters while keeping the random seeds consistent.

We will first describe the full experimental setup in Section 6.3.1. This entails a description of the general setup, the baseline models, hyperparameters and datasets. Then, we present the results of the experiments in Section 6.3.2 and relate them to the research questions above.

6.3.1. Experimental Setup

Our primary goal in these experiments is to perform a robust and fair comparison of various learners or our ablation configurations. This comparison needs to be efficient in terms of the computational resources at our disposal. To realize this, we adopt several methods.

First, we utilize Monte Carlo cross-validation, a procedure that involves partitioning our dataset randomly into training and testing subsets. Specifically, this is done 5 times, each time designating 90 % of the data for training and the remaining 10 % for testing. This method allows us to create error bars for our average performance estimates, effectively giving us a range of likely

performance values rather than a single point estimate. Importantly, it does this in a way that is mindful of our computational resource limits.

hyperparameter optimization: Section 3.2

Finally, to ensure a fair comparison across different learners, we employ HPO. This process fine-tunes the hyperparameters of each model (e. g., the learning rate of a gradient-based optimization), ensuring that we are comparing the best possible version of each learner. To facilitate this, the training instances are further split into 1/9 validation instances and 8/9 training instances and the performance on the validation set is used as the optimization target of the HPO process. The details of the process are described in Section 6.3.1.2.

Choice measures: Section 4.2.1

Here, we use the A-mean (also called balanced accuracy) as our main target measure (see Section 4.2.1 for details). It is advantageous because it is unbiased with respect to the prevalence of positive instances, making it well-suited in scenarios with class imbalances [Men+13]. In addition, it is used as the optimization target for HPO procedure.

Moreover, in order to prepare the data for effective learning, we implement a standardization step prior to training. This involves scaling the data to have a mean of 0 and a standard deviation of 1. The parameters (mean and standard deviation) used for this standardization process are kept constant and subsequently applied to both the validation and test sets.

We ensure consistency and reproducibility by setting the global random number generator's seed to 0. To maintain independence between various random processes, we used this global random generator to seed specialized random generators for three separate activities: dataset creation, hyperparameter tuning, and the operation of the learners themselves. By doing so, we ensured that each process was using its own unique random stream, which is a commonly used variance reduction technique [BR17].

After the experiments, we conduct a statistical analysis of the results using the `autorank` library [Her20] to ascertain which performance differences between learners are significant. The approach is based on the methodology proposed by Demšar [Dem06]. We set the family-wise significance level of the tests to $\alpha = 0.05$. To detect whether *any* significant differences[4] exist in the median values of the mean A-mean performances, we utilize the non-parametric Friedman test. If that is the case, we use the post-hoc Nemenyi test to infer which particular differences are significant.

4: The non-parametric Friedman test, an *omnibus test*, checks for any significant differences across the median performances. If detected, further tests can pinpoint *where* these differences occur.

We carried out our experiments utilizing a combination of NVIDIA GeForce RTX 2080 Ti and GTX 1080 Ti GPUs (30 in total). Each 'outer fold' run, which involved comprehensive hyperparameter optimization, had a median runtime of approximately 8.9 hours on the GPU. The cumulative duration for all the experiments, incorporating initial tests, totaled 9891 GPU hours. We implemented the entire source code, which includes components such as the learners, dataset creators, and the experimental execution framework, in Python. This code is open-source and readily accessible to everyone.[5]

5: `https://github.com/kiudee/pareto-embeddings/releases/tag/paper-version`

We will introduce the baseline models we compare to in Section 6.3.1.1. The HPO process and the corresponding hyperparameter ranges of each learner are explained in Section 6.3.1.2. Finally, in Section 6.3.1.3 we introduce our suite of datasets that are used in the experiments before presenting our results in Section 6.3.2.

6.3.1.1. Baseline Models

Recall research questions P-RQ1 and P-RQ2, where we want to evaluate if the Pareto-embedding approaches are able to learn anything useful at all (P-RQ1) and how they compare to existing approaches (P-RQ2). To answer these questions, we need to pick a set of baseline models that are a representative selection. In Table 6.1, we list the learners we use for this purpose.

Table 6.1.: Overview of the baseline models used in the experimental evaluation.

Learner	Context-sensitive	Nonlinear	Source
FETA	Yes	Yes	Section 5.4.1
FATE	Yes	Yes	Section 5.4.2
RankNet	No	Yes	[Tes88; Bur+05]
PairwiseSVM	No	No	[Joa06]

We focus particularly on the context-sensitive learning algorithms we have developed, namely FETA and FATE. These represent the class of general-purpose context-sensitive learners. As demonstrated in Section 5.5, they achieve state-of-the-art performance on a variety of datasets, which makes them ideal baselines for our experimental evaluation.

As for context-independent baselines, we chose RankNet [Tes88; Bur+05] and PairwiseSVM [Joa06]. Here RankNet is our best nonlinear context-independent baseline and PairwiseSVM the corresponding linear baseline.

All these baselines have architectural details and hyperparameters to set that influence their performance in the upcoming experiments. In the next section, we will discuss in more detail how the learners are set up and which specific hyperparameter ranges are used.

6.3.1.2. Hyperparameters

As we have previously explained in the general experimental setup, we are applying HPO using a nested train-validation split of the training data. To reiterate, we first set aside the test data and then divide the remaining data into two parts: 8/9ths of the instances are used for training and 1/9th is for validation.

For the HPO process,we perform 60 iterations of Bayesian optimization. These consist of 20 initialization iterations using the quasirandom Steinerberger sequence [Ste20] and 40 adaptive iterations using max-value entropy search as suggested by Wang and Jegelka [WJ17] with Gaussian processes (GPs) acting as our surrogate model. The library `bask`[6] is used to facilitate this process [PHA22]. After the hyperparameters are optimized on the validation set, the learner is retrained on the complete training set (training + validation) using the best hyperparameters found, and only then will it be used to predict on the test set.

6: https://pypi.org/project/bask/

There are a few hyperparameters that are fixed across the experiments. We use rectified linear units (ReLUs) as the nonlinearities. Batch normalization [IS15] is applied to the output of each hidden layer, before applying the nonlinearity to speed up and stabilize training. The neural networks are trained for 500 epochs using mini-batch stochastic gradient descent with Nesterov momentum

[Nes83]. The batch size is set to 2048 for all neural networks. Lastly, we set the number of embedding dimensions d' of the Pareto-learners to 2.

Table 6.2.: Hyperparameter ranges of the learners optimized using Bayesian optimization.

Type	Hyperparameter	Lower	Upper
Complexity	Hidden layers[1,2]	1	6
	Hidden units[1,2]	16	128
	L_2 penalty (log)[3]	1×10^{-5}	100
Optimizer[1]	Learning rate (log)	1×10^{-5}	1×10^{-1}
	Momentum	0.1	0.999
Pareto-loss weights[4]	Weight for L_{PO}	0.0	2.0
	Weight for L_{DOM}	0.0	2.0
	Weight for L_{MDS}	0.0	2.0
	Weight for L_2	0.0	2.0

[1] Shared by all neural network models.
[2] FATE utilizes two neural networks, we tune the parameters for both.
[3] Only PairwiseSVM.
[4] Only used by the Pareto-embeddings.

As for the tuned hyperparameters, the ranges used for hyperparameter optimization are shown in Table 6.2. These hyperparameters were chosen based on their potential impact on model learning and overall prediction quality. When it comes to the neural network-based models (i.e., FETA, FATE and RankNet), we tune the number of hidden layers and units, the learning rate (on a log scale) and the momentum parameter of the stochastic gradient descent. For the Pareto-embedding models, we also have to optimize the weights of our surrogate loss function (6.7). Finally, our linear model PairwiseSVM only has one parameter governing the strength of the L_2 regularization.

6.3.1.3. Datasets

The methodology by which we generate the datasets is the same across the suite of datasets: Objects are generated randomly in a certain *object space*. The dimensions of this object space we will call the *features* of the objects and are visible to the learners as feature vectors. Then, a set of objective functions maps the objects into an *objective space*. These objectives are assumed to *increase* in utility when *minimized*. Neither the objective functions nor the mapped objects are available to the learners. Subset choices are then constructed by choosing those objects of a given choice task Q that are Pareto-optimal in the objective space.

objective space

In the following, we will describe the generation and properties of the DTLZ and ZDT datasets in more detail. We use the `pygmo` library [BI20] to generate these datasets. We also add a simple two-dimensional dataset, where the underlying Pareto-front consists of two parabola (TP):

$$f_1(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^4$$
$$f_2(x_1, x_2) = (x_1 + 1)^2 + (x_2 + 1)^4$$

In total, we end up with 14 benchmark problems that are summarized in Table 6.3. Each dataset contains 40 960 instances. We divide this dataset into test, training, and validation sets. The test set comprises 10 % of the total dataset, hence it contains $1/10 \times 40960 = 4096$ instances. After setting aside

Table 6.3.: Dataset Characteristics

Datasets	TP	ZDT1–4, 6	ZDT5	DTLZ1–7
# Features	2	6	35	6
# Objectives	2	2	2	5
# Objects	10 (all datasets)			
# Instances	40960 (all datasets)			

the test set, we have $40960 - 4096 = 36864$ instances remaining for the training and validation sets. The training set comprises $8/9$ of this remainder, yielding $8/9 \times 36864 = 32768$ instances. Lastly, the validation set is $1/9$ of the remaining instances, which means it contains $1/9 \times 36864 = 4096$ instances.

ZDT Datasets

ZDT Datasets In the context of studying algorithms for solving multiobjective optimization problems, Zitzler et al. [ZDT00] proposed a suite of test problems that exhibit characteristics that make them hard to be optimized. These problematic characteristics are convexity, nonconvexity, discreteness, multimodality, deceptiveness, and nonuniformity. For each of these characteristics, the authors designed a corresponding test function. Since these features also make it difficult for machine learning algorithms to learn these functions, the test problems are also a good source of datasets.

[ZDT00]: Zitzler et al. (2000), "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results"

The test problems are limited to two objectives and are always constructed using the following template:

$$\begin{aligned} F(\boldsymbol{x}) &= \big(f_1(x_1), f_2(\boldsymbol{x})\big) \\ f_2(\boldsymbol{x}) &= g(x_2, \dots, x_d)\, h\big(f_1(x_1), g(x_2, \dots, x_d)\big) \\ \text{with } \boldsymbol{x} &= (x_1, \dots, x_d) \end{aligned}$$

Here, $\boldsymbol{x}$ is the object represented in object space, and $F(\boldsymbol{x})$ is the mapping to the objective space. The function f_1 only takes feature x_1 as input, while g is a function of the remaining features.

The first test problem, *ZDT1*, is a convex problem and defined as ZDT1

$$\begin{aligned} f_1(x_1) &:= x_1 \\ g(x_2, \dots, x_d) &:= 1 + \frac{9}{d-1} \cdot \sum_{i=2}^{d} x_i \\ h(f_1, g) &:= 1 - \sqrt{f_1/g} \end{aligned}$$

with d fixed at 30 and $x_i \in [0, 1]$ for all $i \in [d]$.

ZDT2 is a minor variant of ZDT1, where $h(f_1, g) := 1 - (f_1/g)^2$. This makes the function nonconvex. ZDT2

Test function *ZDT3* also varies the function h of ZDT1 as follows: ZDT3

$$h(f1, g) := 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$$

This causes the Pareto-front in objective space to be discontinuous.

ZDT4 alters ZDT1 and makes it a highly multimodal test function, i. e., it has ZDT4

21^9 local Pareto-fronts.

$$
\begin{aligned}
f_1(x_1) &:= x_1 \\
g(x_2, \dots, x_d) &:= 1 + 10(d-1) + \sum_{i=2}^{d} \left(x_i^2 - 10\cos(4\pi x_i)\right) \\
h(f_1, g) &:= 1 - \sqrt{f_1 / g}
\end{aligned}
$$

Here $d = 10$, $x_1 \in [0, 1]$ and $x_2, \dots, x_d \in [-5, 5]$.

ZDT5

ZDT5, instead of using real-valued features, uses binary strings as features. With that the problem is defined as

$$
\begin{aligned}
f_1(x_1) &:= 1 + u(x_1) \\
g(x_2, \dots, x_d) &:= \sum_{i=2}^{d} v\big(u(x_i)\big) \\
h(f_1, g) &:= 1 / f_1 \\
v\big(u(x_i)\big) &:= \begin{cases} 2 + u(x_i) & \text{if } u(x_i) < 5 \\ 1 & \text{if } u(x_i) = 5 \end{cases}
\end{aligned}
$$

where $u(x_i)$ counts the number of ones in the bit vector x_i. Here $x_1 \in \{0, 1\}^{30}$ and $x_2, \dots, x_d \in \{0, 1\}^5$.

ZDT6

Finally, *ZDT6* is a nonconvex problem where Pareto-optimal solutions are concentrated in a certain region:

$$
\begin{aligned}
f_1(x_1) &:= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\
g(x_2, \dots, x_d) &:= 1 + 9 \cdot \left(\sum_{i=2}^{d} \frac{x_i}{d-1}\right)^{1/4} \\
h(f_1, g) &:= 1 - (f_1 / g)^2
\end{aligned}
$$

with $d = 10$ and $x_i \in [0, 1]$ for all $i \in [d]$.

DTLZ Datasets

[Deb+05]: Deb et al. (2005), "Scalable Test Problems for Evolutionary Multiobjective Optimization"

DTLZ Datasets We will now introduce the DTLZ datasets and the underlying test problems, as proposed by Deb et al. [Deb+05]. A drawback of the ZDT test problems was that they only used two objective functions. Therefore, the main design goal of the new test problems was to have test functions that could be scaled in terms of the number of objectives and features. The test problems were created to exhibit various characteristics such as convexity, nonconvexity, discontinuity, and multimodality. In the following, all features x_i for $i \in [d]$ are sampled from the interval $[0, 1]$.

DTLZ1

The first problem, *DTLZ1*, constructs d' objective functions

$$
\begin{aligned}
f_1(\boldsymbol{x}) &= \frac{1}{2} x_1 x_2 \dots x_{d'-1}(1 + g(\boldsymbol{x}_{d':d})) \\
f_2(\boldsymbol{x}) &= \frac{1}{2} x_1 x_2 \dots (1 - x_{d'-1})(1 + g(\boldsymbol{x}_{d':d})) \\
&\vdots \\
f_{d'-1}(\boldsymbol{x}) &= \frac{1}{2} x_1 (1 - x_2)(1 + g(\boldsymbol{x}_{d':d})) \\
f_{d'}(\boldsymbol{x}) &= \frac{1}{2} (1 - x_1)(1 + g(\boldsymbol{x}_{d':d})) \,.
\end{aligned}
$$

Here $\boldsymbol{x}_{d':d} \coloneqq (x_{d'}, x_{d'+1}, \dots, x_d)$ and $g(\cdot)$ is a functional that can be chosen as desired. The authors propose

$$
g(\boldsymbol{x}_{d':d}) \coloneqq 100 \left[|\boldsymbol{x}_{d':d}| + \sum_{x_i \in \boldsymbol{x}_{d':d}} (x_i - 0.5)^2 - \cos\big(20\pi(x_i - 0.5)\big) \right], \quad (6.10)
$$

which we use in our experiments.

DTLZ2 is a nonlinear test problem with the following definition: DTLZ2

$$
\left.
\begin{aligned}
f_1(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(x_1 \pi/2) \dots \cos(x_{d'-2}\pi/2) \cos(x_{d'-1}\pi/2) \\
f_2(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(x_1 \pi/2) \dots \cos(x_{d'-2}\pi/2) \sin(x_{d'-1}\pi/2) \\
f_3(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(x_1 \pi/2) \dots \sin(x_{d'-2}\pi/2) \\
&\vdots \\
f_{d'}(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \sin(x_1 \pi/2)
\end{aligned}
\right\} \quad (6.11)
$$

with

$$
g(\boldsymbol{x}_{d':d}) \coloneqq \sum_{x_i \in \boldsymbol{x}_{d':d}} (x_i - 0.5)^2 \,. \quad (6.12)
$$

DTLZ3 combines the objective functions of DTLZ2 (6.11) with the functional g (6.10) from DTLZ1. DTLZ3

DTLZ4 is a variation on DTLZ2 (6.11) DTLZ4

$$
\begin{aligned}
f_1(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(x_1^\alpha \pi/2) \dots \cos(x_{d'-2}^\alpha \pi/2) \cos(x_{d'-1}^\alpha \pi/2) \\
f_2(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(x_1^\alpha \pi/2) \dots \cos(x_{d'-2}^\alpha \pi/2) \sin(x_{d'-1}^\alpha \pi/2) \\
f_3(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(x_1^\alpha \pi/2) \dots \sin(x_{d'-2}^\alpha \pi/2) \\
&\vdots \\
f_{d'}(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \sin(x_{d'-2}^\alpha \pi/2)
\end{aligned}
$$

with g as in DTLZ2 (6.12) and $\alpha = 100$.

DTLZ5 is defined as DTLZ5

$$
\begin{aligned}
f_1(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(\theta_1 \pi/2) \dots \cos(\theta_{d'-2}\pi/2) \cos(\theta_{d'-1}\pi/2) \\
f_2(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(\theta_1 \pi/2) \dots \cos(\theta_{d'-2}\pi/2) \sin(\theta_{d'-1}\pi/2) \\
f_3(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \cos(\theta_1 \pi/2) \dots \sin(\theta_{d'-2}\pi/2) \\
&\vdots \\
f_{d'}(\boldsymbol{x}) &= (1 + g(\boldsymbol{x}_{d':d})) \sin(\theta_1 \pi/2)
\end{aligned}
$$

with

$$\theta_1 = x_1$$
$$\theta_i = \frac{\pi(1 + 2g(\boldsymbol{x}_{d':d})x_i)}{4(1 + g(\boldsymbol{x}_{d':d}))} \quad \text{for all } i = 2, \ldots, d-1$$

and g as in DTLZ2 (6.12).

DTLZ6

DTLZ6 combines the objective functions of DTLZ5 with an alternative functional g:

$$g(\boldsymbol{x}_{d':d}) := \sum_{x_i \in \boldsymbol{x}_{d':d}} x_i^{0.1}$$

DTLZ7

DTLZ7 is constructed to have a set of disconnected Pareto-optimal regions. It is defined as follows:

$$f_1(\boldsymbol{x}) = x_1$$
$$f_2(\boldsymbol{x}) = x_2$$
$$\vdots$$
$$f_{d'-1}(\boldsymbol{x}) = x_{d'-1}$$
$$f_{d'}(\boldsymbol{x}) = (1 + g(\boldsymbol{x}_{d':d}))h(f_1, f_2, \ldots, f_{d'-1}, g)$$

with

$$g(\boldsymbol{x}_{d':d}) := 1 + \frac{9}{|\boldsymbol{x}_{d':d}|} \sum_{x_i \in \boldsymbol{x}_{d':d}} x_i$$

and

$$h(f_1, f_2, \ldots, f_{d'-1}, g) := d - \sum_{i=1}^{d'-1} \left[\frac{f_i}{1+g} \left(1 + \sin(3\pi f_i)\right) \right]$$

In conclusion, our comprehensive methodology for dataset generation and the detailed characterization of the DTLZ, ZDT, and TP datasets provide a robust foundation for evaluating the performance of the learning algorithms. Having established clear definitions for the datasets, we can proceed to present and discuss the results of our empirical evaluation.

6.3.2. Results and Discussion

We first present the raw results and the statistical analysis in Section 6.3.2.1. Following that, the results are interpreted and related to our research questions in Section 6.3.2.2. Finally, it is important to acknowledge the limitations of our experimental design, which we do in Section 6.3.2.3.

research questions: Page 173

6.3.2.1. Results

Our main results are summarized in Table 6.4, where we show the mean aggregated performance of the learners on each of the datasets across the 5

Table 6.4.: Mean and standard error A-mean performance of the learners on the TP, ZDT, and DTLZ datasets. The values are shown on the 10^{-1} scale. The standard error for the mean estimate is shown in parentheses. The bold entries denote the highest obtained mean performance for each dataset.

	Dataset						
Learner	TP	ZDT1	ZDT2	ZDT3	ZDT4	ZDT5	ZDT6
FATE	8.424(1)	8.560(9)	8.82(1)	8.375(6)	7.70(1)	8.101(9)	7.59(1)
FETA	8.31(3)	8.67(3)	8.74(5)	8.28(9)	7.58(3)	7.75(2)	7.589(7)
RankNet	8.17(2)	8.209(1)	8.479(6)	8.14(3)	7.80(1)	8.110(1)	7.9(2)
PairwiseSVM	5.4(2)	7.835(6)	8.215(6)	7.64(1)	7.11(1)	7.593(9)	7.40(1)
PairwisePareto	**9.917(8)**	8.7(7)	**9.974(2)**	**9.68(3)**	**8.24(2)**	**9.541(8)**	**8.8(1)**
Pareto	9.532(7)	**9.538(9)**	9.639(5)	8.91(2)	8.12(6)	9.20(2)	7.42(6)
	Dataset						
Learner	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
FATE	5.02(1)	5.66(6)	5.24(2)	7.16(3)	6.88(4)	6.6(1)	7.22(5)
FETA	5.011(7)	5.46(5)	5.27(4)	7.16(5)	6.97(5)	6.49(5)	7.31(6)
RankNet	5.08(2)	6.35(2)	5.42(5)	7.07(4)	6.89(6)	6.7(2)	6.95(9)
PairwiseSVM	5.007(2)	5.05(2)	5.25(2)	6.27(5)	5.15(9)	6.27(8)	6.84(3)
PairwisePareto	**6.12(4)**	**9.04(4)**	**6.32(4)**	**9.73(2)**	**9.55(4)**	**9.46(3)**	**9.82(2)**
Pareto	5.7(2)	7.0(4)	5.83(4)	7.60(4)	7.25(8)	7.95(6)	7.65(2)

outer folds. We report the A-mean introduced in (4.9) as the evaluation metric. We highlight the entries with the highest obtained mean performance using bold font. The parentheses denote the standard error of the mean estimate. For instance, 9.917(8) means the learner achieved an A-mean of 0.9917 with a standard error of ±0.0008.

Our context-sensitive learner, PairwisePareto, surpasses the other models with the highest average A-mean performance on almost all datasets. The exception is ZDT1, where Pareto exhibits superior performance. In addition, both PairwisePareto and Pareto are able to reach A-means of over 0.9, which none of the other baselines can do. The nonlinear baselines, which include FETA, FATE and RankNet, often get comparable results, with the linear PairwiseSVM lagging behind a bit.

We performed a statistical evaluation of the results to discern statistically significant differences in the performance of the different learning models, following the methodologies by Herbold [Her20] and Demšar [Dem06]. We established the family-wise significance level for the tests at $\alpha = 0.05$.

For the learning model FATE, we reject the null hypothesis that the performance values are normally distributed ($p = 0.002$). Consequently, we use the non-parametric Friedman test as an omnibus test to determine, if there are any significant differences between the median values of the mean A-mean performances.

We use the post-hoc Nemenyi test to infer *which* differences are significant. We report three primary measures: the median performance, the median absolute deviation, and the average rank each model holds across all datasets. These results are displayed in Table Table 6.5.

The differences between the learning models are deemed statistically significant, if the difference in the mean rank surpasses the critical distance (CD = 2.015)

Table 6.5.: Robust summary statistics of the performance of the learners across the datasets. Shown are the mean rank (MR), median (MED), median absolute deviation, confidence interval of the median (CI) and the effect size in comparison to the highest ranking learner using Akinshin's γ [Aki20].

Learner	MR	MED	MAD	CI	γ
PairwisePareto	1.143	0.950	0.065	[0.612, 0.997]	0.000
Pareto	2.143	0.780	0.139	[0.569, 0.964]	1.573
FATE	3.857	0.741	0.133	[0.502, 0.882]	2.007
RankNet	3.857	0.744	0.104	[0.508, 0.848]	2.382
FETA	4.071	0.744	0.127	[0.501, 0.874]	2.048
PairwiseSVM	5.929	0.656	0.169	[0.501, 0.821]	2.305

as specified by the Nemenyi test. We reject the null hypothesis ($p < 0.001$) of the Friedman test that there is no difference in the central tendency of the learners and, therefore, assume that there is a statistically significant difference between the median values of the learners.

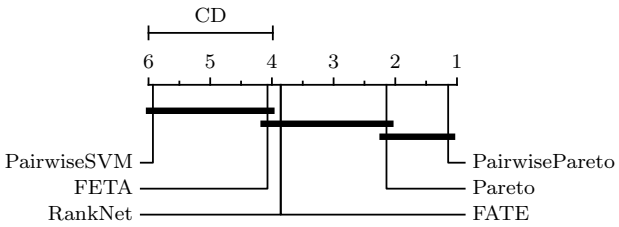


Figure 6.10.: Critical distance diagram for the Nemenyi post-hoc test. The black bars group approaches for which no significant difference could be detected.

Consider the critical distance plot in Figure 6.10, which visualizes which learners significantly differ in performance by comparing their ranks with respect to the *critical distance*. Based on the post-hoc Nemenyi test, there are no significant differences in the groups marked by bold black lines. There are no significant differences within the following groups: PairwisePareto and Pareto; Pareto, FATE, RankNet, and FETA; FETA and PairwiseSVM. All other differences are significant.

Ablations

Ablations To address research questions P-RQ3, P-RQ4 and P-RQ4, we conducted a series of ablation experiments using the same multi-objective datasets as before. Figure 6.11 presents a violin plot that illustrates the distribution of

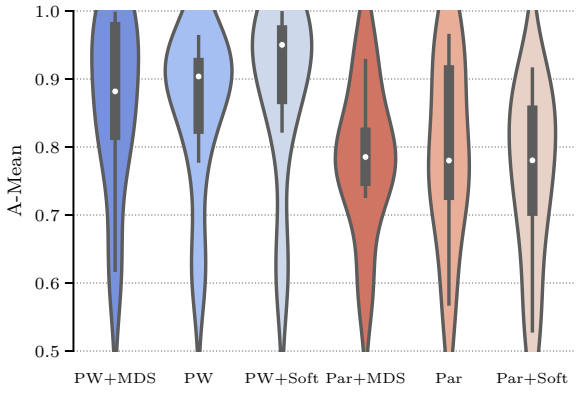


Figure 6.11.: Plot showing the distribution of the mean A-mean performances of the Pareto learners (PW - PairwisePareto, Par - Pareto). Variants include the learners trained with MDS loss and those with softplus instead of ReLU. Violins show the kernel density estimate of the distribution, while the boxplots show the three quartiles, and the whiskers depict 1.5 × inter-quartile range.

the mean A-means obtained from the various datasets. This plot uses violins to represent kernel density estimates of each distribution, mirrored symmetrically for clarity. In these violins, the white dot represents the median value of each distribution. The enclosed boxplots within each violin extend from the lower to the upper quartile of the distribution, depicting the inter-quartile range.

The whiskers on these boxplots indicate 1.5× the inter-quartile range, providing additional information about data spread. Different colors distinguish the variations in our approaches: The context-sensitive PairwisePareto approach is depicted by the violins shaded in blue, while the context-independent Pareto approach is represented by the red violins.

First, we investigate whether including the MDS loss has a positive/adverse effect on the performance. To this end, we repeat the complete evaluation on all datasets for Pareto and PairwisePareto with the MDS loss being used and its weight being optimized during the HPO stage. Our findings indicated no significant differences in performance across various datasets when we set a significance threshold of $\alpha = 0.05$ using Wilcoxon's signed rank test. It is noteworthy that for the Pareto approach, the use of the MDS loss seemed to decrease the variability of the results. However, for PairwisePareto, we observe an increased variance. Consequently, we excluded the MDS loss function for our main comparisons when utilizing the PairwisePareto model.

Next, we check if replacing the $\max(0, \cdot)$ in (6.5) and (6.6) by the softplus function $\log(1+e^x)$ results in a loss function that is easier to train and ultimately achieves a better performance. The results are again shown in Figure 6.11. For PairwisePareto, we observe an increase in median performance from 90.3 % to 95.0 %, and there was a reduction in the median absolute deviation from 0.092 to 0.044. Despite this positive trend, the increase did not reach statistical significance ($p = 0.097$, Wilcoxon signed rank test), suggesting the need for additional experimentation. In contrast, for the Pareto model, we did not observe a discernible difference in performance when switching between the max and softplus functions. For the main experiments, we opt to use the softplus for both approaches since the limited data points to it having a minor positive effect, although this hypothesis warrants further empirical validation.

Next, we present the experimental results examining the effect of the dimensionality of the embedding space d'. For a dimensionality of 1 ($d' = 1$), the model is restricted to representing and learning total orders, thus limiting predictions to singleton sets. Consequently, our focus shifts to comparing the impact of a two-dimensional versus a three-dimensional embedding space.

The resulting violin plot is shown in Figure 6.12. The two distributions appear almost identical, with the median A-mean even slightly lower when $d' = 3$. Following this ablation experiment, we therefore decided to standardize the dimensionality to $d' = 2$ for all learners employing Pareto-embeddings in the primary experiments.

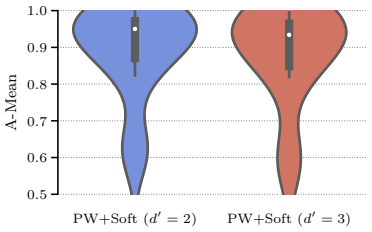


Figure 6.12.: Plot contrasting the distributions of the mean A-mean performances of the PairwisePareto approach, once with $d' = 2$ and once with $d' = 3$. Violins show the kernel density estimate of the distribution, while the boxplots show the three quartiles, and the whiskers depict 1.5 × inter-quartile range.

6.3.2.2. Discussion

Now that we introduced the raw results, we are going to relate them to the four research questions we introduced at the beginning of Section 6.3.

P-RQ1

We begin with research question P-RQ1: We asked if the Pareto-embedding models, Pareto and PairwisePareto, can effectively learn meaningful embeddings when trained with our surrogate loss function. The results confirm that they indeed can. We can see this clearly from the fact that both Pareto and PairwisePareto models have managed to reach A-mean scores exceeding

0.9 on some of the two-dimensional test problems (namely TP, ZDT1 to 3 and ZDT5). This indicates that both approaches can successfully represent and learn the underlying choice function. Specifically, when examining the ZDT1 and ZDT2 problems, the Pareto model, which is context-independent, exhibits an average A-mean of more than 0.95. This suggests an accurate representation of the inherent choice function and, hence, the learned Pareto embedding. Moreover, a closer look at the TP and ZDT2 datasets reveals the PairwisePareto model's exceptional performance, with an average A-mean exceeding 0.99. This score shows that the model is able to model these particular problem sets nearly perfectly.

When we shift our attention to the more complex DTLZ datasets, we can see that the PairwisePareto model attains average A-mean scores of greater than 0.9 on DTLZ2 and DTLZ4 to 7. In contrast, Pareto on these same datasets achieves average scores in the 0.7 to 0.8 range. Notably, these datasets possess a higher underlying dimensionality (i. e., 5), which is greater than the embedding dimensionality employed by both models. In light of this, the superior performance of the PairwisePareto model is intriguing. One possible explanation could be that PairwisePareto is implicitly learning to use a higher-dimensional embedding by taking advantage of its set-based context. However, this intriguing phenomenon necessitates further exploration and additional experiments to validate the hypothesis.

P-RQ2

With the related research question P-RQ2, we wanted to assess the performance of Pareto and PairwisePareto in comparison to existing approaches, in particular our more general FETA and FATE architectures. In the statistical analysis, we observed that PairwisePareto performs significantly better than the other baselines across the datasets we employed for this study. In the case of the Pareto model, we did not see that it outperforms FETA, FATE, and RankNet significantly across the entire dataset collection. Nonetheless, the Pareto model did exhibit remarkable performance in the two-dimensional ZDT test problems, achieving A-mean scores exceeding 0.9 and outperforming the baselines by a notable margin. Therefore, we conclude that the specialized Pareto-embedding models have the potential to exceed the performance of current leading choice models on datasets that have a latent low-dimensional utility structure.

P-RQ3

Moving on to our ablations, with P-RQ3 we wanted to answer the question of whether an additional MDS term in the loss function would improve the performance of the Pareto-embedding models. It is intended to conserve the observed distances in the object space during the embedding learning process. We did not find a significant effect of this term on the overall distribution of average A-mean scores, suggesting that it may not contribute positively to performance. Consequently, we can answer P-RQ3 with a no. We did not investigate whether the MDS term helps with the interpretability of the learned embeddings, which could be an interesting direction for future research.

P-RQ4

As for P-RQ4, in a separate experiment, we replaced the max operation in the loss function with a smoother softplus operation. We hypothesized that this change would make the loss function easier to optimize. Our results did not show a significant benefit from this change. While there was a slight positive effect on the PairwisePareto model, the evidence is insufficient to establish

a definitive trend. Further experiments are necessary to confirm these initial observations. Overall, we will answer P-RQ4 to the negative.

P-RQ5

Finally, we turn our attention to P-RQ5, where we investigated whether increasing the dimensionality of the embedding space improves performance. To provide context, the DTLZ datasets in our evaluation have an inherent dimensionality of 5. Thus, we anticipated that increasing the number of dimensions from $d' = 2$ to $d' = 3$ would increase performance. However, literature on partial order embeddings suggests that finding embeddings becomes challenging for $d' \geq 3$. A priori, it was unclear whether the increase in dimension would provide a benefit. The results show that the distribution of average A-mean scores appears to be identical, and there was no significant difference. This suggests that the learner was unable to leverage the increased dimensionality. Further investigation is needed to understand why this is the case. As for P-RQ5, the answer is no, increasing the dimensionality does not improve performance of the Pareto-embedding approaches.

Overall, the empirical evaluation demonstrates the ability of the Pareto and PairwisePareto models to learn and represent underlying choice functions effectively. Notably, the PairwisePareto model shows potential in handling more complex datasets and even outperforms our more general models FETA and FATE across various datasets. While these findings are very promising, it is essential to recognize certain limitations in our evaluation to pave the way for future investigations.

6.3.2.3. Limitations

We want to highlight a few important limitations to be aware of when contextualizing our work. We will distinguish the limitations of the learner and those of the empirical evaluation.

Our examination of our surrogate loss in Section 6.2.2 established its theoretical suitability. Specifically, we demonstrated that a surrogate loss of 0 (across all possible subsets of objects) implies that a valid Pareto-embedding has been learned. However, we have not seen a performance improvement when increasing the embedding dimensionality. One potential cause is that there are diminishing returns in the datasets investigated. Here, it would be useful to investigate whether this is the case or if there are convergence issues in higher dimensions.

Regarding our empirical evaluation, we adopted several measures to secure a comprehensive and unbiased comparison of all learners. These steps included the application of HPO for all learners, using precisely seeded random generators, nested validation, and subsequent statistical analysis. The datasets were selected due to their reputation as widely recognized test problems within the multi-objective optimization field, exhibiting a variety of characteristics. What additionally could be interesting is an evaluation of experimental human choice data that investigates, whether Pareto-embeddings work well in experiments where context effects are observed. In essence, we want to know if the choice mechanism based on multiple utilities is also a good choice model for human behavior. Therefore, one needs to be aware that the excellent results in the empirical evaluation do not yet imply that it is a plausible model for how humans make decisions.

6.4. Conclusion and Future Work

We are closing this chapter by highlighting what we have accomplished. In utility theory, we usually try to find an underlying utility function that explains a set of preferences. If our target is to predict subset choices, it is unclear how single utilities can be used to model such preferences. In this chapter, we asked what happens if we assume the existence of multiple utility functions. This elegantly allows us to model subset choices since the *Pareto-set* produces a natural subset of non-dominated objects.

This leads us to our main contributions. We proposed to learn this set of utility functions, which we called a *Pareto-embedding*, that is able to produce the original choices when coupled with Pareto-optimal choice. In order to learn the embedding, we proposed a surrogate loss function and proved that the minimization thereof guarantees a valid Pareto-embedding. We also theoretically characterize the exact family of choice functions that can be represented by Pareto-embeddings, namely those whose revealed preferences are a certain partial order.

Finally, we proposed two neural network architectures, the context-independent Pareto and one context-sensitive PairwisePareto. We evaluate them extensively on a suite of benchmark datasets and find that they outperform existing baselines by a large margin. All in all, we conclude that the results are very promising and feel that it is a worthwhile research direction to explore.

Future Work

Potential for Future Work We suggest several prospective areas for future exploration in this field of study. One theoretical line of investigation involves our surrogate loss function. While we have proved that its minimization results in a valid Pareto-embedding, we do not yet know whether it is actually a good loss function to be used during gradient-based optimization. Exploring potential improvements here could be fruitful when investigating more embedding dimensions.

A further theoretical aspect to consider is the understanding of the model class that PairwisePareto is learning. Unlike the context-independent Pareto, which is evaluated independently for each object, PairwisePareto can learn a different embedding for each object set, thus learning a more complex preference structure. The degree to which this improves the representational power of the model remains unclear, making this an area worth investigating.

On the empirical side, we recommend conducting a comprehensive evaluation using real-world datasets, but more importantly, human preference data, to better understand the capabilities of Pareto-embeddings. Such an analysis could answer questions related to the inherent dimensionality of real-world data. This includes determining the necessary number of embedding dimensions for adequate modeling and assessing the rate at which returns diminish with additional dimensions. Furthermore, it is vital to ascertain whether the $\mathcal{NP}$-hardness of the partial order embedding problem remains a constant concern in practice, or if there are viable strategies to scale up to highly dimensional embeddings.

An alternate research trajectory could focus on the *preference elicitation* setting, a form of *active learning*. Until now, we have assumed the existence of a

dataset with choices to be modeled. Preference elicitation, however, involves capturing a decision-maker's preferences by repeatedly presenting them with a purposefully constructed set of objects and observing their choices. The aim is to model their preferences accurately with minimal queries. This approach would require a model that can be used to express its epistemic uncertainty. The probabilistic GP model developed by Benavoli et al. [BAP23b] may serve as a starting point for this endeavor.

A related consideration is the interpretability of the embedding. If we use Pareto-embeddings to learn lower-dimensional utility functions for a given set of objects, we may want to understand how each utility is influenced by the features in the original object space. Similar to how factors are analyzed in factor analysis.

7. Conclusion and Future Work

We conclude this thesis with a reflection on our contributions to preference learning and neighboring fields. In Section 7.1, we refer back to the original research questions that we posed in Section 1.1. We describe how they have been answered and draw conclusions. With Section 7.2, we put the thesis and its findings into a broader context and analyze its impact on the field of preference learning. During the work on this thesis, we identified several promising research directions that could be addressed in future works. In Section 7.3, we discuss each direction in detail. Finally, we close the thesis with a concluding statement in Section 7.4.

7.1. Summary of Key Findings

Recall the main goal of this thesis, which was to develop efficient preference models that can predict subset choices and incorporate context effects. With that goal in mind, we set out to answer four research questions (see Section 1.1). In this section, we will now reflect on the results we obtained throughout the work on this thesis and use them to answer our research questions.

Research Question 1 (RQ1):

> "Which approaches exist in preference learning, and of those, which have a mechanism to account for context effects?"

To answer RQ1, we conduct two literature reviews. One in classical utility theory and preference modeling, and the second one in preference learning as a subfield of machine learning. The results are reported in Chapter 2 and Chapter 4. We identify the big family of random utility models (RUMs) that are still used today to model preferences. Some variants in this family do have mechanisms that let them represent certain context effects. A more recent branch of research in choice modeling are so-called multi-self models, of which SDA [ROS20] is a recent representative. In Chapter 4, we have seen the breadth of research that has been conducted in the field of preference learning. The majority of the approaches we categorize as *context-independent*, demonstrating that these types of learners are underrepresented in the literature. In addition, the number of choice models to which we can compare our approaches is limited. Therefore, we adapt a representative selection of the models used for ranking tasks to a (subset) choice setting for our experimental investigations.

Research Question 2 (RQ2)

> "What are different efficient and novel mechanisms to capture the context in a choice setting?"

generalized utility

We began by defining what it means for a choice to be context-dependent. Identifying *generalized utility* as an idealized target, we derive two approaches FETA and FATE in Chapter 5. Both make learning a generalized utility function feasible by different means. FETA decomposes the utility into lower-order utility contributions, and FATE condenses the task context into a task representative, which is then used as the fixed context in the utility computation. Both approaches are computationally efficient with respect to the number of objects in a task, with a linear complexity at prediction time—achieving our first secondary objective. Furthermore, they can be trained in an end-to-end fashion in conjunction with the loss functions we propose.

Pareto-embeddings

In Chapter 6, we introduce an orthogonal methodology for learning choice functions through *Pareto-embeddings*. This approach inherently supports subset selection without necessitating the thresholding utilized by FETA and FATE. We also propose a differentiable surrogate loss function, enabling the embedding to be seamlessly integrated and trained within a broader end-to-end system architecture. In terms of complexity, our method achieves an expected linear time complexity with respect to the number of objects during prediction. We develop two versions of the embedding: one that is context-independent and another that adapts the embedding based on the task context.

Research Question 3 (RQ3)

> "What is the expressive power of these models, and which other interesting theoretical properties can be shown?"

We analyze all three of the developed models with respect to their expressive power, i. e., how flexible they are in representing a wide variety of choice functions. For FATE we show that it is flexible enough to represent all possible choice functions, at the cost of not being an identifiable model. The pairwise approximation to the complete FETA model we show is limited in its expressive power. There are choice functions defined on 7 objects that cannot be represented. The model is, however, identifiable up to the 0th order utility.

Pareto-property

Similarly, we show that choice functions exist that cannot be represented by Pareto-embeddings. We are then interested in precisely characterizing the representable family of choice functions. We define the *Pareto-property* on the level of choice functions and prove that it is equivalent to choice functions representable by Pareto-embeddings. In addition, we propose and analyze a surrogate loss function for learning Pareto-embeddings. Here, we prove that a loss of 0 guarantees that one has obtained Pareto-embedding, and we also show that it suffices to minimize the loss only on pairwise comparisons.

Research Question 4 (RQ4)

> "How can these models be inferred from data, and how do they compare to existing models?"

We employ a straightforward methodology to answer the first part of this question. We define differentiable loss functions and combine them with a suitable neural network architecture. Then, the losses are minimized using standard stochastic gradient descent (SGD) algorithms. For the neural network architectures, we made sure to let them encode the model we are aiming for to

avoid introducing additional architectural biases. Some architectural choices we test in ablation studies, such as the Pareto-embedding architectures.

We compare our approaches in extensive empirical evaluations to representative baselines from the literature. Care was taken to maximize the performance of each model, to ensure a fair comparison. We observe that FETA-Net and FATE-Net often outperform the baselines by a large margin, especially on the tasks with a strong context-dependence. We saw a relative improvement in categorical accuracy of up to 162 % in the case of FETA-Net on the singleton choice tasks, and an improvement in F_1-measure of up to 17 % on the subset choice tasks. FATE-Net similarly was able to outperform the baselines on many tasks but trailed slightly behind FETA-Net. FETA-Net was also able to converge fastest to an accuracy of 100 % in our infinite data experiment, where we equalized the number of parameters of each model to ascertain which model is the most data efficient. We can see that while FATE (and therefore FATE-Net) is able to represent all possible choice functions, the inductive bias of FETA allows the model to be much more data efficient.

Moving on to the results of our context-independent Pareto and context-dependent PairwisePareto Pareto-embedding architectures. When evaluated on the benchmark suite used in Section 6.3, we see that PairwisePareto significantly outperforms all baselines, including our FETA-Net and FATE-Net, often achieving an A-mean of more than 0.99. This demonstrates that specialist models such as Pareto-embeddings can outperform more generalist choice models in certain domains.

Overall, we can conclude that the research questions we posed at the beginning of this thesis have been answered. As such, our contributions in this thesis are significant and advance the field of preference learning. To explore the potential impact of these contributions, we will now discuss the broader context of this work.

7.2. Contextualization and Impact

This thesis advances the field of preference learning by introducing novel context-dependent models that significantly enhance our ability to model diverse preferences across a spectrum of applications. It is especially useful in the context of choice modeling and is useful in a wide range of fields beyond traditional machine learning. The context-dependent models introduced in this thesis have the potential to impact how choices are modeled in practice. This is true, particularly in sectors where the predictive performance and the flexibility of the models are crucial, such as in e-commerce and information retrieval.

In contrast to existing models for context-dependent choices, our models are general in that they can represent context effects that are not yet known and are not limited to human choices. This allows our models to learn more complicated algorithmic choice functions. This could be particularly useful in scenarios requiring an accurate surrogate of complex, costly-to-evaluate choice functions, such as multi-objective optimization.

efficient prediction of subset choices

We address a critical gap in preference learning: the *efficient prediction of subset choices*, an area that has seen limited development until now. Existing

methodologies in subset choice modeling lacked efficient predictive mechanisms for subset choices. In contrast, our approaches provide a significant improvement, enabling the prediction of subsets within polynomial, and in some instances linear, time (in the number of objects). This efficiency is crucial for handling large datasets, making our models not only more effective but also more scalable. However, this improvement in runtime does not mean that we have to sacrifice the ability of the models to take the task context into account. Through rigorous experimental evaluations, we show that our models perform significantly better than our baselines on context-dependent tasks, underscoring the practical implications of our research.

While this thesis addresses critical gaps in the modeling of choices, the evolving nature of the field presents ongoing challenges. In the next section, we identify key areas for future research, focusing on unresolved challenges and offering a roadmap for addressing them.

7.3. Future Research Directions

During our research on the topics covered throughout this thesis, we identified several promising areas where research could be focused next. We will now present the most impactful directions grouped by category.

7.3.1. Incorporating More Context and Scalability

The methodologies we develop in this thesis allow us to learn a richer set of preferences than before. It is natural to ask how other contexts can be incorporated into a choice model. Adding instance features to the models proposed in this thesis is straightforward, and we will discuss this in the section about personalization. A promising direction for future research is *dynamic context modeling*, i. e., to incorporate how the influence of the context changes over time into the modeling process. On the one hand, this can accommodate changes in the choice behavior of the population, but on the other hand, it can also reflect changes in the data-gathering process. Take, for instance, a hotel booking website that changes how search results are displayed. As we know from research on context effects, this can drastically change the impact of the context on the overall decision.

dynamic context modeling

A major trend in machine learning is multi-modal learning [XZC23], where different types of data (such as text, images, sound, etc.) are combined. *Multi-modal choice modeling* would allow us to model many more relevant context effects. For example, image data could be used to learn what users deem important when comparing pictures of different products. Questions like which influence the background of an image has on the choices could be addressed.

Multi-modal choice modeling

Another important dimension is scalability. Several interesting questions arise in a regime where the number of objects per task becomes large. While the raw computational complexity becomes an issue, additional considerations are worth exploring. We know from studies with human participants that the nature of context effects changes when the number of objects becomes large (see Section 2.3). As it is simply no longer possible to compare all options,

participants turn to *satisficing* to find good enough options to choose. Therefore, an interesting problem is selecting a representative sample of all available objects in a complete task to present to the user. Such a selection should ideally be small enough to encourage the user to find the best objects, contain the overall most promising objects, but also be diverse. In addition, there is also an *active learning* component, where we may want to periodically include options that we are uncertain about to improve our model. In applications where the options compete, such as in a search engine, *fairness* is important. Overall, this setting is similar to the preselection bandit setting considered by Bengs and Hüllermeier [BH20] but adds a few additional dimensions. In information retrieval systems, multi-stage methodologies are common to tackle the issue of computational complexity; however, they do not address all of the aforementioned aspects [Zha+24].

7.3.2. Interpretability and Context Effects

Interpretability and explainable artificial intelligence (AI) have increased in importance and popularity as more and more models in practice are black boxes such as neural networks [BP21]. This becomes particularly important in choice modeling, where we not only want to find a good predictive model but also understand *how* a choice was made. Modeling context effects involves several aspects. As a first step, detecting whether and what kind of context effect is present in the data could be worthwhile. We present initial research in this direction in Section 2.3.1. As previously mentioned, our models are, in principle, able to model context effects that have not been identified in the literature. An impactful research direction here could be to work towards a mechanistic model of the context effect. An idea for this is to decompose the model into a context-independent component and a symbolic model for the context-dependent deviations. This would allow practitioners to gain deep insights into how the actual choice process incorporates the task context.

7.3.3. Personalization and Bias

The research directions we previously presented enable the move towards more fine-grained personalization. By incorporating user and additional context features, we can model different types and strengths of context effects in the data. For example, take the organizer of a movie night that collects a wishlist of movies and has to decide on one. They may be interested in finding a great movie that fulfills several criteria, and they carefully consider all submitted movies, and potentially even past chosen movies, to gauge the overall taste profile. On the other hand, a person coming home from a stressful day of work may want to find a good movie quickly without wanting to use mental capacities to compare different movie options. The level of context-dependence is different for both of these settings. Incorporating additional context can also yield deeper insights into choice behavior. One may be interested in segmenting the user base based on their preferences, i. e., identify if there are clusters of users with similar (contextual) choice behavior. This could inform decisions on how alternatives and their features are communicated to different groups of users. In marketing, one may be interested in which users are most influenced by certain contexts and to what extent.

"debiasing"

For now, we mostly talked about capturing richer preferences by incorporating more context. We can also flip this around and consider how context-dependent models could be used for *"debiasing"*. Say the goal is to train a context-independent model on a large dataset. It is known that the way the data was collected will likely lead to context effects, i. e., there were few objects, the objects were easy to compare, etc. Then, these context effects could lead to bias and, ultimately, worse generalization for a context-independent model. The idea of debiasing would first be to gauge how strong the bias likely will be and, secondly, to train an efficient decomposable context-dependent model on a subset of the data that is then used to "correct" or downweigh choices that likely resulted from the object context.

7.4. Concluding Statement

In this thesis, we develop and evaluate several complementary context-dependent choice models that can predict singleton and subset choices. These contributions advance the field of preference learning and choice modeling as a whole, enabling practitioners to learn richer types of preferences. We identify several promising directions for future research. Firstly, we suggest incorporating a more multi-modal context for a more holistic view of which aspects impact a choice decision. In addition, increasing the degree of personalization and interpretability of the models and the detected context effects could make the models applicable in more domains. We see the biggest impact in applications where precise and efficient models of contextual choice behavior are important. These include e-commerce, market research, information retrieval, and recommender systems since the predictive accuracy is central and the datasets are typically large. Additionally, this impact extends to the domain of evidence synthesis, where choices contain substantial information [BB23].

Through our work on this thesis, which sits at the intersection of several disciplines, we gain an appreciation of the different perspectives each field has on the same problem. However, each field is also, in a sense, "myopic" in that more cross-contamination between them could be fruitful. Therefore, our vision for choice modeling within preference learning is an area of research that seamlessly bridges interdisciplinary gaps. Acknowledging the challenges and developing a comprehensive set of tools to model and understand preferences in all kinds of contexts. Therefore, we call upon the research community in preference learning, choice modeling, and applications to build upon our proposed methods. Integrating our context-dependent choice models with their expertise to solve previously infeasible problems moves us towards a better and more complete understanding of preferences.

Appendix

Glossary

ACE	axiom of choice by elimination
Adam	adaptive moment estimation
AI	artificial intelligence
AP	average precision
AUC	area under the ROC curve
BN	batch normalization
BTL	Bradley-Terry-Luce
CDM	context-dependent random utility model
CDN	cumulative distribution network
CNN	convolutional neural network
DCG	discounted cumulative gain
DPO	direct preference optimization
EUM	expected utility maximization
ERM	empirical risk minimization
ERR	expected reciprocal rank
FATE	First Aggregate Then Evaluate
FETA	First Evaluate Then Aggregate
FPR	false positive rate
GEV	generalized extreme value
GIN	graph isomorphism network
GNL	generalized nested logit
GNN	graph neural network
GP	Gaussian process
HPO	hyperparameter optimization
ICDM	International Conference on Data Mining
IIA	independence of irrelevant alternatives
i. i. d.	independent and identically distributed
IR	information retrieval
LETOR	LEarning TO Rank
LLM	large language model
LSTM	long short-term memory
MAP	mean average precision
MLE	maximum likelihood estimation
MCMC	Markov chain Monte Carlo
ML	mixed logit
MDS	multi-dimensional scaling
MLP	multi-layer perceptron
MNIST	Modified National Institute of Standards and Technology
MOEA	multi-objective evolutionary algorithm
MOO	multi-objective optimization
MNL	multinomial logit
MRR	mean reciprocal rank
MSE	mean squared error
MST	moderate stochastic transitivity
NDCG	normalized discounted cumulative gain

NL	nested logit
NLP	natural language processing
1MQ	Million Query
PL	Plackett-Luce
PSIS-LOO	Pareto smoothed importance sampling leave-one-out
RBM	restricted Boltzmann machine
ReLU	rectified linear unit
RLHF	reinforcement learning from human feedback
RNN	recurrent neural network
ROC	receiver operating characteristic
RUM	random utility model
SCM	singleton choice model
SDA	set-dependent aggregation
SELU	self-normalizing linear unit
SGD	stochastic gradient descent
SST	strong stochastic transitivity
SVM	support vector machine
TREC	Text REtrieval Conference
TPR	true positive rate
VC	Vapnik–Chervonenkis
WL	Weisfeiler-Leman
WST	weak stochastic transitivity

Symbols

Notation	Description
General	
$\mathcal{S}$	Symmetric group
ϕ	An increasing and strictly positive function (e. g., the exponential function)
Machine Learning	
$\boldsymbol{x}$	Instance
$\mathcal{X}$	Instance space
$\mathcal{D}$	Dataset
N	Size of dataset
n	Index of dataset
y	Label
$\mathcal{Y}$	Label space
M	Size of the label space
m	Index of label
θ	Weight
$\boldsymbol{\theta}$	Weight vector
h	Hypothesis/Model
$\mathcal{H}$	Hypothesis/Model space
T	Maximum number of iterations
t	Index of current iteration
L	Loss function
d	Discount function
ξ	Slack variable of an SVM
Preferences	
$\boldsymbol{x}$	An object
$\boldsymbol{x}'$	An alternative object
$\boldsymbol{y}$	An alternative object
$\boldsymbol{z}$	An alternative object
x	Element of an object $\boldsymbol{x}$
$\mathcal{X}$	Object space
$\boldsymbol{\mathfrak{X}}$	Class of all object spaces
$\lvert\mathcal{X}\rvert$	Size of the object space
d	Dimensionality of the instance/object ($\boldsymbol{x} \in \mathbb{R}^d$)
Q	Task
$\mathbb{Q}$	Task space
$2^{\mathcal{X}}$	Set of all subsets of objects
Z	Embedding of a set of objects
$\mathcal{Z}$	Embedding space
d'	Number of embedding dimensions $\lvert\mathcal{Z}\rvert$
K	Size of task

Notation	Description
k	Index of task
K	Size of the set of the best items
φ	Pareto embedding
C	Choice set, which is a subset of a choice task Q
$\boldsymbol{c}$	Choice set, represented as a binary vector
c	Choice function $\mathbb{Q} \to \mathbb{Q}$
$\mathcal{C}$	Choice space
π	Ranking
R	(Partial) ranking represented as a set
π	Permutation
O	(Partial) order defined on a set
$\mathcal{S}$	The set of all symmetric groups
r	Rank $\mathrm{r}(\boldsymbol{x}, \pi)$ of object $\boldsymbol{x}$ in ranking π
ρ	Ranking function
θ	Threshold
M	Size of a ranking
u	Utility function
$\overline{\mathrm{u}}$	Average utility of a certain order
u	Real-valued utility
$\boldsymbol{u}$	Vector of real-valued utilities
U	Random utility
ε	Noise component of a random utility model
S	Individual space
$\boldsymbol{s}$	Individual
ℓ	Dimensionality of the space of individuals
B	Nest, which is a subset of a task
λ	Independence parameter in a nested logit model
K	Number of nests of objects
I	Indexing function $\mathcal{X} \to \mathbb{N}$
α	Allocation parameter of the generalized nested logit model
t	Type of the multi-self framework
$\mathcal{T}$	Space of types of the multi-self framework
$\mathcal{S}$	The set of all selves in the multi-self framework
S	A set of selves in the multi-self framework

Appendix A.

A.1. Additional Experimental Details

In this section, we will now list all experimental details which were excluded from the description in Section 5.5.1 for conciseness reasons.

Empirical Comparison In order to compare all learners fairly, we do nested cross-validation with synchronized random streams for all the learning models, as shown in Figure A.1. The hyperparameters of all models are tuned using extensive Bayesian optimization. We describe the complete procedure in two parts: first, the *hyperparameter optimization* and second, the *out-of-sample evaluation*.

Hyperparameter Optimization The training dataset, denoted as $\mathcal{D}_k$, is used to identify the best hyperparameters through 3-fold stratified cross-validation, followed by training the final model for out-of-sample evaluation.

The hyperparameter optimizer selects hyperparameters from the ranges specified in Table A.1 for each iteration i. Within the inner loop ($1 \leq j \leq 3$), the full training dataset $\mathcal{D}_k$ is split into a training set (D_{kj} 90 % of $\mathcal{D}_k$) and a validation set (V_{kj} 10 % of $\mathcal{D}_k$) using stratified shuffle splitting. For the selected hyperparameters p_i, the model is trained on the training set (D_{kj}) and evaluated on the validation dataset V_{kj} using the target loss function.

We use the F_1-loss for general subset choice and the categorical 0/1-loss for singleton choice as the target loss functions to evaluate the hyperparameter configurations. The mean loss $\ell_i = \text{mean}(l_1, l_2, l_3)$ is calculated for the given hyperparameters p_i. The optimization loop runs for 100 iterations, validating 100 sets of hyperparameters to determine close to optimal parameters p_b for the learning model.

Out-of-Sample Evaluation After HPO, we configure the learners using the best-found hyperparameters p_b and the remaining default parameters p_d. Next, we train the model M on the complete training dataset $\mathcal{D}_k$ and evaluate it on the test dataset $\mathcal{D}_{Tk}$ using various evaluation measures m as defined in Section 4.2.1. To obtain reliable estimates of the mean performance and standard deviation, we repeat this procedure five times using outer cross-validation. For each fold $k \in [K]$, where $K = 5$, we record the evaluated value a_k and subsequently calculate the mean and standard deviation of the performance measure m.

Hyperparameters & Inference We will now describe the specific hyperparameters we optimize and the respective value ranges we consider. An overview can be found in Table A.1. For probabilistic models, we also describe the inference process. For all neural network models, we make use of the following techniques:

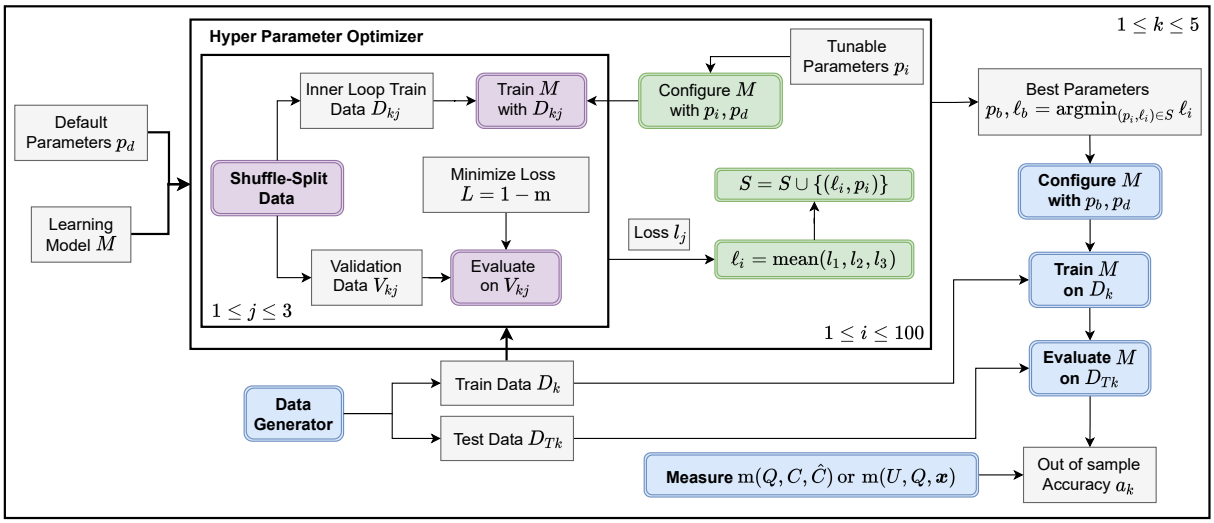


Figure A.1.: Overview of the complete evaluation pipeline.

Table A.1.: Hyperparameter ranges used by the optimizer to select configurations for the learners.

	Architecture Parameters				Other Parameters			LRScheduler		Linear Model Parameters	
Learner	Set Units	Set Layers	Joint Units	Joint Layers	Regularizer Strength	Learning Rate	Batch Size	Epochs Drop e_{drop}	Drop d_r	tol	C
FETA-Net	NA	NA	$[1, 20]$	$[4, 1024]$	$[10^{-10}, 0.1]$	$[10^{-5}, 0.01]$	$[32, 4096]$	$[50, 250]$	$[0.01, 0.5]$	NA	NA
FATE-Net	$[4, 1024]$	$[1, 20]$	$[4, 1024]$	$[1, 20]$	$[10^{-10}, 0.1]$	$[10^{-5}, 0.1]$	$[32, 4096]$	$[50, 250]$	$[0.01, 0.5]$	NA	NA
FETA-Linear	NA	NA	NA	NA	$[10^{-10}, 0.1]$	$[10^{-5}, 0.1]$	$[32, 2048]$	$[10, 150]$	$[0.01, 0.5]$	NA	NA
SDA	$r[4, 64]$	$r[1, 4]$	$w[4, 64]$	$w[1, 4]$	$[10^{-10}, 0.1]$	$[10^{-5}, 0.1]$	$[8, 1024]$	$[50, 250]$	$[0.01, 0.5]$	NA	NA
RankNet	NA	NA	$[1, 20]$	$[4, 1024]$	$[10^{-10}, 0.1]$	$[10^{-5}, 0.01]$	$[64, 8192]$	$[50, 250]$	$[0.01, 0.5]$	NA	NA
RankSVM	NA	NA	NA	NA	NA	NA	NA	NA	NA	$[10^{-4}, 0.5]$	$[1, 12]$

- We use either ReLU nonlinearities in conjunction with batch normalization (BN) [IS15] or SELU nonlinearities [Kla+17] for each hidden layer.
- Regularization: L_2 penalties are applied and the corresponding regularization strength is tuned.
- Optimizer: SGD with Nesterov momentum [Nes83].
- Learning rate annealing: A step-decay function is used for the learning rate schedule, and the decay factor is tuned [DHS11].

The step-decay function reduces the learning rate by a factor after a specified number of epochs [DHS11]. Formally, it is defined as:

$$lr = lr_0 \cdot d_r^{\lfloor \frac{e}{e_{\text{drop}}} \rfloor},$$

where lr_0 is the initial learning rate, $0 < d_r < 1$ is the decay rate, e is the current epoch, and e_{drop} is the number of epochs after which the learning rate is decreased. We set the maximum number of training epochs for the neural networks to 1000.

The hyperparameters of each algorithm were tuned using the scikit-optimize package [Hea+18]. In addition to the number of hidden layers and units, we also tuned the learning rate of the stochastic gradient descent optimizer, the regularization strength, and the batch size (fraction of training examples used for estimating the gradient in one iteration). We also optimize the drop-rate (d_r) and epoch drop e_{drop} for the step-decay function used by the stochastic gradient descent optimizer of the neural networks.

For PairwiseSVM, we tune the penalty parameter C of the error term and the tolerance (*tol* in scikit-learn) for the stopping criterion of the optimization algorithm [Ped+11].

All the generalized extreme value (GEV) models were implemented in PyMC3, a library that facilitates Markov Chain Monte Carlo estimation of the posterior distribution [SWF16]. An overview of all the hyperparameters and their admissible ranges is shown in Table A.1.

Threshold Tuning To set the threshold for the subset choice models (5.3), we tune the threshold for all models using a small validation set. The optimal value for θ depends on the underlying target loss function. Our primary target loss is the (instance-averaged) F_1-measure (4.5), which balances the precision and recall of the predictions [Lew95; Ye+12; Wae+14]. Koyejo et al. [Koy+15] demonstrate that tuning a threshold on a validation set yields a consistent classifier if the estimated marginal instance probabilities (in our case, the choice probabilities) converge in probability to the population-level probabilities. One important difference from the multi-label classification setting is the absence of a fixed set of labels. Instead, we have a dynamically changing set of objects. Therefore, it is only possible to consider instance-averaged performance metrics.

A.2. Design of the Generalization Experiment

In Section 5.5 we raised the research question FF-RQ4, asking how well the approaches generalize to unseen task sizes. To this end, we vary the task sizes from 3 to 30 as shown in Figure A.2.

First, we configure the learning model with the best hyperparameters p_b obtained from the empirical comparison experiment for the given dataset and the remaining default parameters p_d. Then, we generate the training dataset containing task sets of size $K = |Q|$ and train the configured model on the training dataset $\mathcal{D}_K$. Finally, we evaluate the trained model on different test datasets $\mathcal{D}_k$ containing the tasks of sizes in S ($|Q| = k \in S$) as described in Table A.2.

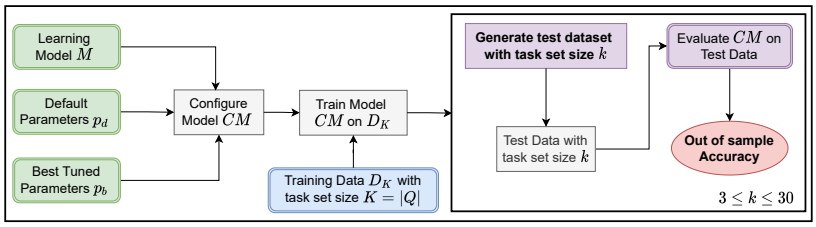


Figure A.2.: Design of the generalization experiments.

Table A.2.: Dataset configurations for generalization experiments. Bracket notation is used to denote the range of values.

Problem	Dataset	# Features	# Train	# Test	Task set sizes S	Task set Size $\|Q\|$
Singleton Choice	Medoid	5	10 000	100 000	$[3, 30]$	10
	Hypervolume	2	10 000	100 000	$[3, 30]$	10

A.3. Synthetic Datasets

In this section, we will describe the process of generating the synthetic datasets for the experimental evaluation in Chapter 5, including objects and tasks.

A.3.1. The Medoid Problem

Recall that we have defined the *medoid* of a set $Q \subset \mathbb{R}^d$ as $c_{\text{medoid}}(Q) = \arg\min_{x \in Q} \frac{1}{|Q|} \sum_{y \in Q} \|x - y\|$, where $\|\cdot\|$ is the standard euclidean norm in $\mathbb{R}^d$. Thus, the medoid of Q may be thought of as the most centrally located object in Q. See the illustration of a choice set Q of size 5 and its medoid in Figure A.3a. As it depends on its distance to any other point from Q, the medoid of a task is sensitive to changes of any points in Q.

For our empirical study, we created a dataset $\mathcal{D} = \{(Q_1, C_1), \ldots, (Q_N, C_N)\}$ by drawing each Q_i independently and uniformly at random from the set

$$\left\{Q \subset [0,1]^d \,:\, |Q| = n \text{ and } |c_{\text{medoid}}(Q)| = 1\right\}$$

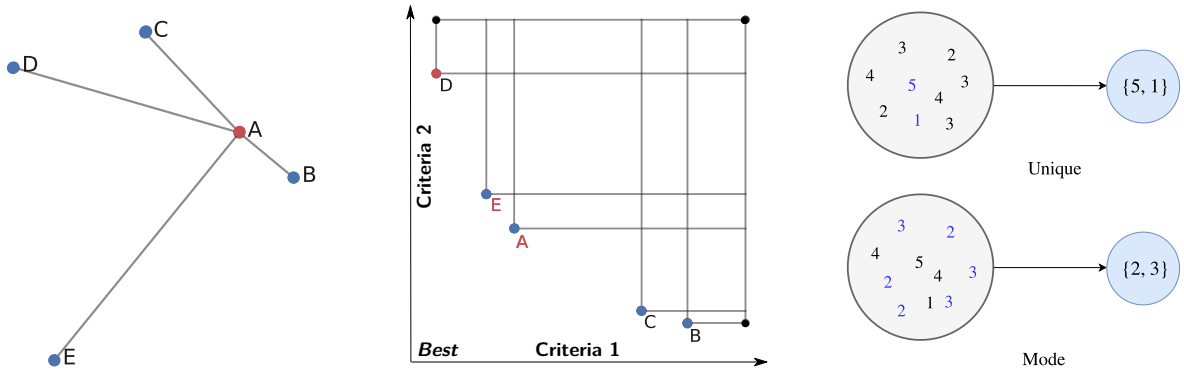
and then choose $C_i := c_{\text{medoid}}(Q_i)$. Here, the sampling step can be performed via the acceptance-rejection method: One may repeatedly sample $x_1, \ldots, x_n$ uniformly at random from $[0,1]^d$ until $Q = \{x_1, \ldots, x_n\}$ has size n and a unique medoid. Regarding that this condition is already fulfilled with probability 1 after sampling $x_1, \ldots, x_n$ only once, this method is efficient.

A.3.2. The Pareto Problem

Above, we introduced the *Pareto set* $c_{\text{Pareto}}(Q)$ of a set $Q \subset \mathbb{R}^d$ as the set of all elements $x \in Q$ which are not dominated by any $y \in Q \setminus \{x\}$, wherein x was said to dominate y if $\forall i \in [d] : x_i \leq y_i$ and $\exists i \in [d] : x_i < y_i$. Figure A.3b shows the Pareto set of a set $Q \subset \mathbb{R}^2$.

With the help of Pareto sets, we create a synthetic dataset $\mathcal{D} = \{(Q_j, C_j)\}_{j=1}^N$ for the subset choice task, where each sample $(Q, C) \in \mathcal{D}$ is generated independently of the others in the following way:

1. Sample $\mu_1, \ldots, \mu_n$ i.i.d. uniformly at random from $\{x \in \mathbb{R}^d : \|x\| \leq 1\}$
2. Draw i.i.d. samples $\xi_1, \ldots, \xi_n$ from $N(\mathbf{0}, I_d)$, the standard Gaussian distribution on $\mathbb{R}^d$, and define $x_i := \mu_i + \xi_i$ for each $i \in [n]$.
3. Choose $Q := \{x_1, \ldots, x_n\}$ and $C := c_{\text{Pareto}}(Q)$.



(a) Random points in $[0,1]^2$ are ranked according to their distance to the medoid (here shown in orange)

(b) Random points in $[0,1]^2$, on Pareto-font ranked according to their contributions

(c) Example for unique and mode function on MNIST digits

Figure A.3.: Examples for the synthetic datasets

Hypervolume In Section 5.5.1.3 we have introduced for $Q \subset \mathbb{R}^d$ the choice set $c_{\text{HypVol}}(Q)$ as the set of all $\boldsymbol{x} \in Q$, which contribute the least among all elements in Q to the *hypervolume* of Q, see Section 5.5.1.3 for the precise definitions and also for the connection of the hypervolume of Q to the Pareto front of Q. As this contribution of each point depends on the position of other points in Q, c_{HypVol} is context-dependent. This is illustrated in Figure A.3b, where all five elements of $Q = \{A, B, C, D, E\}$ lie on the Pareto front of Q. There, the contribution of point A is largest in Q, but removing point D from the choice set increases the contribution of point E to the set. So, the singleton choice changes from A to E, after removing D from Q.

Based on c_{HypVol} we construct a singleton choice dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N$ by sampling each Q_i uniformly at random from the set of all $Q \subseteq \mathbb{R}^d$, which fulfill

$$|Q| = n,\ \forall \boldsymbol{x} \in Q:\ (\|\boldsymbol{x}\| = 1 \text{ and } \forall i \in [d]:\ x_i \leq 0) \text{ and } |c_{\text{HypVol}}(Q)| = 1,$$

and then defining $C_i := c_{\text{HypVol}}(Q_i)$ afterwards. Similarly, the acceptance-rejection method can be used to sample in the construction of the Medoid data set.

A.3.3. MNIST Number Problems

In this section, we will describe the process of generating different semisynthetic datasets using the MNIST dataset [LCB10].

Feature Extraction Since the dataset consists of 2-D image maps, we first train an off-the-shelf CNN to solve the digit multi-class classification task to level the playing field and abstract away from the computer vision context. This architecture of the CNN consists of 2-D Convolutional, 2-D Max-Pooling, and fully-connected dense layers and applied batch normalization to increase the stability of the network by subtracting the batch mean and dividing by the batch standard deviation as shown in Figure A.4 [GBC16; IS15]. The 2-D convolutional layer is of kernel-size 5×5 using rectified linear unit (ReLU) nonlinear activation function and l-2 regularization and 2-D max-pooling

Feature Extraction

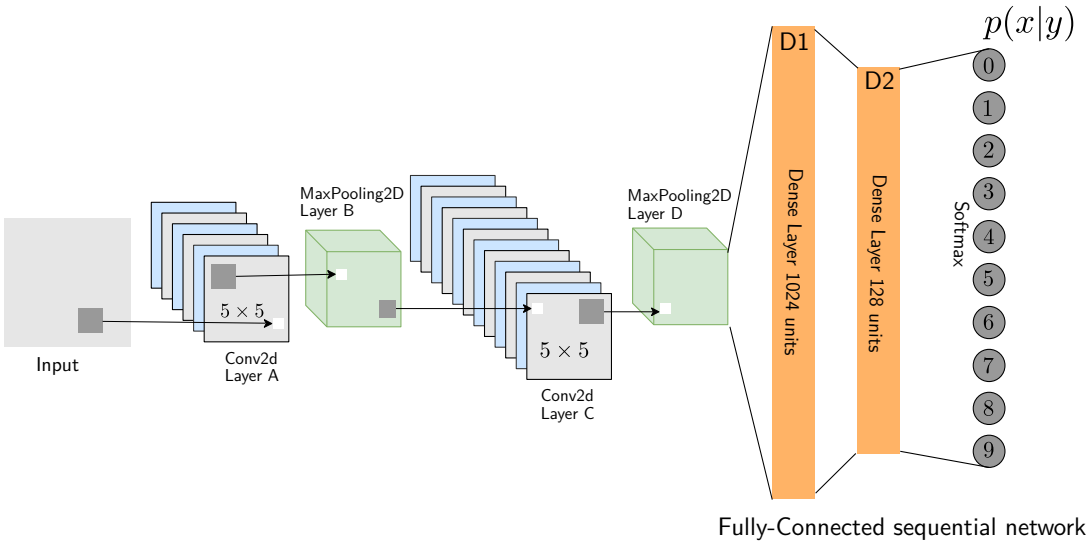


Figure A.4.: CNN that is used to convert MNIST images to high-level features.

layer, with filter of size 2×2 applied with a stride of 2, which down-samples the input by 2 along the width and height, discarding 50 % of the activations by applying max operation over 4 numbers in 2×2 region [GBC16]. The output of these layers is provided as input to a fully-connected sequential network with 10 outputs, where each output predicts the probability of the input image belonging to a particular class using the softmax [GBC16]. We train this network on 10 000 instances, then we transform the remaining 60 000 digits to a high-level feature representation by passing them through the trained CNN and recording the 128 outputs of the last hidden layer (D2).

The transformed MNIST dataset $\mathcal{D}_M = \{(\boldsymbol{x}_1, l_1), \ldots, (\boldsymbol{x}_N, l_N)\}$, is represented as a set of tuples $(\boldsymbol{x}_i, l_i)$, where $\boldsymbol{x}_i$ is the feature vector and l_i represents the corresponding label, such that $|\mathcal{D}_M| = N = 60000$, $\boldsymbol{x}_i \in \mathbb{R}^{128}$, $l_i \in \mathcal{L} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ and $\mathcal{D}_M(\boldsymbol{x}_i) = l_i$ holds for all $i \in [N]$. For constructing the choice datasets, we sample instances $(\boldsymbol{x}_i, l_i) \in \mathcal{D}_M$ from the transformed dataset uniformly at random to construct a task set $Q = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. Based on Q and $\boldsymbol{l} = (\mathcal{D}_M(\boldsymbol{x}_1), \ldots, \mathcal{D}_M(\boldsymbol{x}_n))$, we then select as choice set $C = g(Q, \boldsymbol{l})$, where is an appropriately predefined function g. We consider two variants for g, namely g_{unique} and g_{mode}.

The function g_{unique} outputs the instances corresponding to the numbers which occur only once in the label vector. For example $g_{\text{unique}}(Q, (4, 3, 2, 3, 3, 1, 8, 8, 7, 7)) = \{\boldsymbol{x}_1, \boldsymbol{x}_3, \boldsymbol{x}_6\}$, corresponding to the numbers 4, 2 and 1. For singleton choice choice, we sample only the task sets, whose corresponding label vector $\boldsymbol{l}$ contains a single unique number, to make it identifiable, i. e., for example, $g_{\text{unique}}(Q, (4, 3, 3, 2, 2, 1, 1, 1, 5, 5, 5)) = \{\boldsymbol{x}_1\}$. The section function is g_{mode}, which outputs the instances corresponding to the number that occur most frequently in the label vector. For example $g_{\text{mode}}(Q, (4, 3, 2, 3, 3, 8, 8, 7, 7, 7)) = \{\boldsymbol{x}_8, \boldsymbol{x}_9, \boldsymbol{x}_{10}\}$, corresponding to the mode 7. For singleton choice choice, we choose the instances corresponding to the mode, which are at the least angle from a predefined weight vector $\boldsymbol{w}$.

Both functions that generate choices depend on all other objects in the given task set Q, thus making the datasets highly context-dependent.

MNIST-Unique

Unique In this subsection, we explain the data generation process for the *Unique* choice dataset using the g_{unique} function defined above. For generating the dataset, we select a set of instances from $\mathcal{D}_M$ uniformly at random to construct the task set Q and the label vector $\boldsymbol{l}$. Then, we choose the objects from Q, which corresponds to the unique digit in the label vector $\boldsymbol{l}$ (an example is shown in Figure A.3c). Let us assume we want to generate a dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N$ with N instances.

1. Sample n data points $(\boldsymbol{x}_{i,1}, l_1), \dots, (\boldsymbol{x}_{i,n}, l_n)$ from $\mathcal{D}_M$, let $\boldsymbol{l}_i \coloneqq (l_1, \dots, l_n)$ and $Q_i \coloneqq \{\boldsymbol{x}_{i,1}, \dots, \boldsymbol{x}_{i,n}\}$
2. For each $l \in \mathcal{L}$ let k_l be the number of times the label l appears in the label vector $\boldsymbol{l}_i$ for Q_i, define $\boldsymbol{k} \coloneqq \{k_0, \dots, k_9\}$ and write for convenience $\boldsymbol{k}(l) \coloneqq k_l$ in the following. For example for $\boldsymbol{l} = (1, 2, 4, 4, 4, 5, 5)$ we have $\boldsymbol{k} = (0, 1, 1, 0, 3, 2, 0, 0, 0, 0)$.
3. We create C_i by selecting the objects whose values occur only once in the label vector $\boldsymbol{l}$:

$$C_i \coloneqq \left\{\boldsymbol{x}_{i,j} \in Q_i \,:\, \boldsymbol{k}(l_j) = 1\right\}$$

4. In order to create the corresponding singleton choice or top-1 version of this dataset, we discard Q_i in case $|C_i| > 1$ and repeat steps 1–4. If $|C_i| = 1$ instead, we keep the sample (Q_i, C_i).

MNIST-Mode

Mode In this subsection, we explain the data generation process for the *Mode* choice dataset using the g_{mode} function defined above. For generating the dataset, we select a set of instances from $\mathcal{D}_M$ uniformly at random to construct the task set Q and the label vector $\boldsymbol{l}$ (an example is shown in Figure A.3c). Then, we choose the objects from Q, which corresponds to the mode value of the label vector $\boldsymbol{l}$, to construct the ground-truth set of chosen objects. For creating the corresponding singleton choice or top-1 dataset, we choose the object corresponding to the mode value of the label vector, which is at the least angle to the predefined weight vector $\boldsymbol{w}$. Let us assume we want to generate a dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N$ with N instances. First, we sample the weight vector $\boldsymbol{w} \in \mathbb{R}^{128} \overset{\text{iid}}{\sim} N(\boldsymbol{0}, \boldsymbol{I}_{128})$.

1. Sample n data points $(\boldsymbol{x}_{i,1}, l_1), \dots, (\boldsymbol{x}_{i,n}, l_n)$ uniformly at random from $\mathcal{D}_M$, abbreviate $\boldsymbol{l}_i \coloneqq \{l_1, \dots, l_n)$ and let $Q_i \coloneqq \{\boldsymbol{x}_{i,1}, \dots, \boldsymbol{x}_{i,n}\}$.
2. As for the Unique dataset, write k_l for the number of times the label appears l in the label vector $\boldsymbol{l}_i$ for Q_i, define $\boldsymbol{k} \coloneqq \{k_0, \dots, k_9\}$ and write again $\boldsymbol{k}(l) \coloneqq k_l$.
3. For the case of subset choice define

$$C_i = \left\{\boldsymbol{x}_{i,j} \in Q_i \,:\, \boldsymbol{k}(l_i) = \max_{l \in \mathcal{L}}(\boldsymbol{k}(l))\right\},$$

and in case of singleton choice, select C_i to be that set, which contains only the object with the least angle to vector $\boldsymbol{w}$, i. e.,

$$C_i \coloneqq \left\{\arg\max_{\boldsymbol{x} \in C_i} \cos^{-1} \frac{\boldsymbol{x} \cdot \boldsymbol{w}}{\|\boldsymbol{x}\|\|\boldsymbol{w}\|}\right\}$$

A.3.4. Tag Genome Dataset

The GroupLens Research group released many datasets collected from the MovieLens website[1] for research in the field of recommender systems [HK15]. As of August 2017, the complete dataset collected from this website is comprised of 26 000 000 ratings and 750 000 tags applied to 45 000 movies by 270 000 users [HK15]. One of the datasets is the Tag Genome dataset,[2] which provides real-valued features to characterize the movies [VSR12].

1: https://movielens.org/

2: This dataset is available on https://grouplens.org/datasets/movielens/

Tags are meta-data in the form of keywords, which help to describe an object, such as a movie, song, book, etc. [Smi07]. On the MovieLens website, users create tags to describe a movie. Other users can then use them to filter movies more effectively. Users can also gain more information about a movie with the help of tags applied by other users.

The Tag Genome dataset was generated by applying machine learning algorithms on the information provided by users for a movie in the form of tags, reviews, and ratings [VSR12]. It consists of movies and a set of tags applied to each of them, and a score between 0 and 1 quantifying the *relevance* of each tag to the particular movie (as shown in Figure A.5). This dataset consists of around 12 million relevance scores across 1128 tags applied on 10 993 movies.

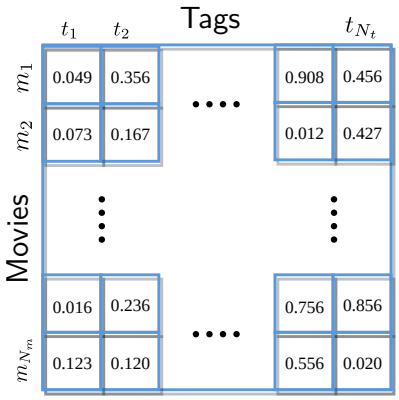


Figure A.5.: Structure of the Tag Genome dataset.

Framework

Framework According to Vig et al. [VSR12] the Tag Genome dataset consists of:

1. M: The set of movies $\{m_1, \ldots, m_{N_m}\}$, where $|M| = N_m = 10993$.
2. T: The set of tags $T = \{t_1, \ldots, t_{N_t}\}$, where $|T| = N_t = 1128$.
3. $R_{rel} : M \times T \longrightarrow [0, 1]$: Relation such that $R_{rel}(m_i, t_j)$ denotes the degree to which extent the tag $t_j \in T$ applies to the movie $m_i \in M$ on a scale of 0 to 1; here 0 indicates no relevance and 1 indicates strong relevance to the movie (as shown in Figure A.5).
4. $\mathcal{M}_f : M \longrightarrow [0, 1]^{N_t}$: Relation mapping each movie to its feature vector in tag-space (vector of tag relevance values across all tags), such that $\mathcal{M}_f(m_i) = \boldsymbol{x}_i := (R_{rel}(m_i, t_1), \ldots, R_{rel}(m_i, t_{N_t}))$.
5. tag-pop $: T \longrightarrow \mathbb{N}$: Function representing the popularity of a tag, measured as the number of users who applied the tag $t_j \in T$.

6. tag-spec : $T \longrightarrow \mathbb{N}$: Function representing the movie frequency of tag $t_j \in T$, i. e., $\text{tag-spec}(t_j) := \sum_{m_i \in M} [\![R_{rel}(m_i, t_j) > 0.5]\!]$ denotes the number of movies for which the relevance of tag t_j is greater than 0.5.
7. P: The set of top 20 most popular-tags $P \subset T$ based on the popularity tag-pop.

The *weighted cosine similarity* is a similarity measure defined to measure the similarity between two movies [VSR12]. The weight vector $\boldsymbol{w}$ is defined so that more weight is assigned to both the popular tags because this implies that more users care about these tags, and also to more specific tags because they can uniquely identify the similarity. For example, if two movies have the *harry potter* tag in common, they are more likely to be similar than the ones that have the tag *fantasy* in common [VSR12]. A log-transform is applied to both values to bring them closer to the normal distribution. The weighted cosine similarity between two movies is defined as:

$$\text{sim}_{\text{ws}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{w}) = \frac{\sum_{k=1}^{N_t} w_k x_{ik} x_{jk}}{\sqrt{(\sum_{k=1}^{N_t} w_k x_{ik}^2)} \cdot \sqrt{(\sum_{k=1}^{N_t} w_k x_{jk}^2)}} \quad , \tag{A.1}$$

where $\boldsymbol{x}_i = \mathcal{M}_f(m_i)$, $\boldsymbol{x}_j = \mathcal{M}_f(m_j)$ and $w_k = \frac{\log(\text{tag-pop}(t_k))}{\log(\text{tag-spec}(t_k))}$ for any $t_k \in T$.

To construct the singleton choice semisynthetic dataset, we sample uniformly at random n movie items from M to create a task set Q, and we choose the medoid $\boldsymbol{r}$ of Q as the reference movie.

We define two tasks based on the reference movie $\boldsymbol{r}$ of the sampled task set Q. The first task is to choose the *most similar movie* to the reference movie in task set Q. The second task is to choose the *most dissimilar movie* with respect to the reference movie $\boldsymbol{r}$ for a given task set Q. This problem is similar to finding the outliers for a given set of objects, which can be used to solve the problem of anomaly detection [AY01; CBK09]. Both tasks used to generate semisynthetic datasets depend on the similarity between all objects in the given task set Q, thus making the datasets highly context-dependent.

Data Generation Process

Data Generation Process We explain the data generation process for the *Tag Genome Similar Movie* and *Tag Genome Dissimilar Movie* datasets. Let us assume we want to generate a singleton choice dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^{N}$ with N instances. Each task set Q_i and its corresponding singleton choices C_i is constructed in the following way:

1. Sample i.i.d. and uniformly at random $m_1, \ldots, m_n$ from M, let $\boldsymbol{x}_{i,n} := \mathcal{M}_f(m_j)$ for each $j \in [n]$ and $Q_i := \{\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,n}\}$.
2. Compute the reference object (movie) for Q_i (medoid):

$$\boldsymbol{r} := \arg\max_{\boldsymbol{x} \in Q_i} \frac{1}{n} \sum_{j=1}^{n} \text{sim}_{\text{ws}}(\boldsymbol{x}, \boldsymbol{x}_{i,j}, \boldsymbol{w})$$

3. Now we define the corresponding singleton choices $C_1, \ldots, C_N$ for *Tag Genome Similar Movie* and *Tag Genome Dissimilar Movie* dataset.
    a) The singleton choice set C_i for Q_i for *Tag Genome Dissimilar Movie* is the set consisting of only that element of Q_i, which is most

dissimilar to $\boldsymbol{r}$, i. e., formally

$$C_i := \left\{ \underset{\boldsymbol{x}_{i,j} \in Q \setminus \{\boldsymbol{r}\}}{\arg\min}\ \mathrm{sim}_{\mathrm{ws}}(\boldsymbol{r}, \boldsymbol{x}_{i,j}, \boldsymbol{w}) \right\}$$

b) For the *Tag Genome Similar Movie* dataset, we select for the task Q_i the singleton choice set

$$C_i := \left\{ \underset{\boldsymbol{x}_{i,j} \in Q \setminus \{\boldsymbol{r}\}}{\arg\max}\ \mathrm{sim}_{\mathrm{ws}}(\boldsymbol{r}, \boldsymbol{x}_{i,j}, \boldsymbol{w}) \right\},$$

which consists of the one element from Q_i, that is most similar to $\boldsymbol{r}$.

A.4. Real-World Datasets

Some widely used benchmark datasets available for solving this task are LETOR and SUSHI [QL13; KKA10]. In the following sections, we briefly describe these datasets and the process we use to generate *singleton* and *subset choice* datasets for the evaluation in Chapter 5.

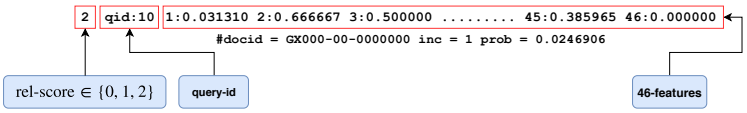
A.4.1. LETOR Datasets

LETOR (in the version 4.0) is a package of benchmark datasets released by Microsoft Research Asia, which are used to compare and evaluate different learning algorithms in the field of preference learning [QL13]. We use the datasets MQ2007 and MQ2008 released for learning the task of partial ranking to create the *subset choice* dataset. There are other datasets MQ2007-list and MQ2008-list released for learning the task of complete ranking[3] to create the *singleton choice* dataset.

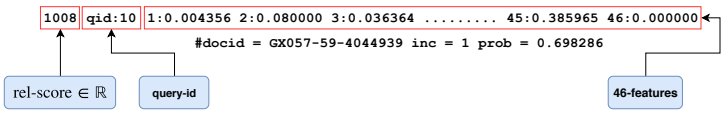
3: These datasets are available on https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/

LETOR Supervised Datasets The datasets (MQ2007 and MQ2008) consist of the queries and retrieved documents, with individual preferences in the form of a relevance for each document with respect to the corresponding query [QL13]. The format of both datasets (MQ2007 and MQ2008) is the same, and there are about 1500 queries in MQ2007 and about 500 in MQ2008 with labeled documents. These datasets consist of 46 features extracted from a query and document constructing an object called *query-document*, and each pair is labeled with a relevance score in $\{0, 1, 2\}$, indicating how relevant the document is to the respective query as shown in Figure A.6a. A relevance score of 0 means that the document is not relevant, 1 means relevant, and 2 means very relevant to the query. The goal of the choice problem for this dataset is to choose all the relevant documents for the given task.

Structure The dataset consists of a universal set of objects $\boldsymbol{x} \in \mathcal{X}$. Each instance of these datasets $\mathcal{D}_S = \left\{(\tilde{Q}_1, l_1), \ldots, (\tilde{Q}_N, l_N)\right\}$, is represented as set of tuples $(\tilde{Q}_i, l_i)$, where $\tilde{Q}_i = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is the task set ($\boldsymbol{x}_i$ features extracted from *query-document*) and $l_i = (l_1, \ldots, l_n)$ represents vector of relevance label for the given set of objects, such that $\boldsymbol{x}_j \in \mathbb{R}^{46}$, $l_j \in \{0, 1, 2\}$ for all $j \in [n]$ and $5 \leq |\tilde{Q}_i| \leq 147$ for every $i \in [N]$.



(a) MQ2007/MQ2008 format



(b) MQ2007-list/MQ2008-list format

Figure A.6.: LETOR dataset formats [QL13]

The size of the universal set of objects in the MQ2007 dataset is 59 570, i. e., $|\mathcal{X}| = 59570$ and the MQ2008 dataset is 564, i. e., $|\mathcal{X}| = 12102$. These datasets have been partitioned into 5 parts by Qin and Liu [QL13], such that $\mathcal{D}_S = \mathcal{D}_{S1} \cup \mathcal{D}_{S2} \cup \mathcal{D}_{S3} \cup \mathcal{D}_{S4} \cup \mathcal{D}_{S5}$. This partition is used to conduct 5-fold cross-validation, and for each fold, we use four parts for training and the remaining part for testing as described in Table A.3.

Table A.3.: 5-folds of the LETOR dataset and the sub-sampled training task sets of size 5.

Dataset			MQ2007			MQ2008		
Fold	Test	Train	# Train	# Test	# Sampled Train	# Train	# Test	# Sampled Train
1	$\mathcal{D}_{S1}$	$\mathcal{D}_S \setminus \mathcal{D}_{S1}$	1172	283	7111	459	105	1187
2	$\mathcal{D}_{S2}$	$\mathcal{D}_S \setminus \mathcal{D}_{S2}$	1160	295	7012	452	112	1083
3	$\mathcal{D}_{S3}$	$\mathcal{D}_S \setminus \mathcal{D}_{S3}$	1163	292	7069	442	122	1122
4	$\mathcal{D}_{S4}$	$\mathcal{D}_S \setminus \mathcal{D}_{S4}$	1160	295	7047	444	120	1203
5	$\mathcal{D}_{S5}$	$\mathcal{D}_S \setminus \mathcal{D}_{S5}$	1165	290	7077	459	105	1201
			# Instances $\lvert\mathcal{D}_S\rvert$	# Features	# Objects $\lvert\mathcal{Q}\rvert$	# Instances $\lvert\mathcal{D}_S\rvert$	# Features	# Objects $\lvert\mathcal{Q}\rvert$
			1455	46	[6, 147]	564	46	[5, 121]

Choice Data Conversion The corresponding choice dataset is created by considering the documents in $\tilde{Q}_i$ as the task sets Q_i and the set of relevant documents $C_i := \left\{\boldsymbol{x}_j \in \tilde{Q}_i : l_j \in \{1, 2\}\right\}$ as the corresponding choice set for each instance $(\tilde{Q}_j, l_j) \in \mathcal{D}_S \setminus \mathcal{D}_{Si}$. For training the choice model, we sub-sample 10 objects from each query instance $\tilde{Q}_i$ to construct the task sets. We still evaluate the models on the corresponding test choice dataset, which consists of all original queries for each fold as described in Table A.3.

LETOR Listwise Datasets The format of both listwise datasets is the same as the supervised one. There are about 1700 queries in MQ2007-list and about 800 queries in MQ2008-list with each *query-document* pair consisting of 46 features. In this dataset, all the documents for each query are labeled with a real-valued relevance score instead of the multi-level relevance judgments as shown in Figure A.6b. The documents on top positions in the ground truth permutation have a larger value of the relevance degree.

Structure The dataset consists of a universal set of objects $\boldsymbol{x} \in \mathcal{X}$. Each instance of these datasets $\mathcal{D}_L = \left\{(\tilde{Q}_1, l_1), \dots, (\tilde{Q}_N, l_N)\right\}$, is represented as a set of tuples $(\tilde{Q}_i, l_i)$, where $\tilde{Q}_i = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ is the task set ($\boldsymbol{x}_i$ features extracted from *query-document*) and $l_i = (l_1, \dots, l_n)$ represents a vector of relevance score for the given set of objects, such that $\boldsymbol{x}_j \in \mathbb{R}^{46}$, $l_j \in \mathbb{R}$ for all $j \in [n]$ and $204 \leq |\tilde{Q}_i| \leq 1831$ for every $i \in [N]$.

Table A.4.: 5-folds of the LETOR MQ2007-list and MQ2008-list dataset and the sub-sampled training task sets of size 5.

Dataset			MQ2007-list			MQ2008-list		
Fold	Test	Train	# Train	# Test	# Sampled Train	# Train	# Test	# Sampled Train
1	$\mathcal{D}_{L1}$	$\mathcal{D}_L \setminus \mathcal{D}_{L1}$	1353	339	97 557	627	157	71 600
2	$\mathcal{D}_{L2}$	$\mathcal{D}_L \setminus \mathcal{D}_{L2}$	1353	339	98 055	627	157	71 908
3	$\mathcal{D}_{L3}$	$\mathcal{D}_L \setminus \mathcal{D}_{L3}$	1353	339	97 580	627	157	72 233
4	$\mathcal{D}_{L4}$	$\mathcal{D}_L \setminus \mathcal{D}_{L4}$	1353	339	98 000	627	157	71 868
5	$\mathcal{D}_{L5}$	$\mathcal{D}_L \setminus \mathcal{D}_{L5}$	1356	336	98 304	628	156	71 847
			# Instances	# Features	# Objects $\lvert Q \rvert$	# Instances	# Features	# Objects $\lvert Q \rvert$
Total			1692	46	[257, 1346]	784	46	[204, 1831]

Singleton Choice Data Conversion The corresponding singleton choice datasets are created by considering the documents in $\tilde{Q}_i$ as the task sets Q_i and the most relevant document $C_i = \left\{\arg\max_{\boldsymbol{x}_j \in \tilde{Q}_i} l_j\right\}$ as the corresponding singleton choice set for each instance $(\tilde{Q}_j, l_j) \in \mathcal{D}_L \setminus \mathcal{D}_{Li}$. For training the SCM, we sub-sample 10 objects from each query instance $\tilde{Q}_i$ to construct the task sets. We still evaluate the models on the corresponding singleton choice test dataset, which consists of all original queries for each fold as described in Table A.3.

A.4.2. Expedia Hotel Dataset

Expedia released a dataset on the Kaggle website as a competition and for research purposes[4]. The dataset includes browsing and booking data as well as information on price competitiveness. The data are organized around a set of search result impressions, the ordered list of hotels that the user sees after they search for a hotel on the Expedia website. In addition to impressions from the existing algorithm, the dataset contains impressions where the hotels were randomly sorted to avoid the algorithm's position bias. The user response is provided as a click on a hotel and/or a purchase of a hotel room. This dataset consists of 399 344 search queries and 45 features extracted from the search query and the hotel constructing an object. Each hotel is labeled with a relevance score of 0, 1, or 2, indicating how relevant it is to the respective query of the user. A relevance score of 0 means that the hotel is not clicked, 1 means it was clicked, and 2 means the hotel was booked by the user. This dataset is very similar to the LETOR dataset as shown in Figure A.6. For this dataset, we define the learning target as the set of relevant hotels (clicked and/or booked). Since the number of hotels displayed for each query is different, this dataset consists of different task sizes.

4: These datasets are available on https://www.kaggle.com/c/expedia-personalized-sort/data

Structure The dataset consists of a universal set of objects $\boldsymbol{x} \in \mathcal{X}$. Each instance of the datasets $\mathcal{D}_E = \left\{(\tilde{Q}_1, l_1), \dots, (\tilde{Q}_N, l_N)\right\}$, is represented as a set of tuples $(\tilde{Q}_i, l_i)$, where $\tilde{Q}_i = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ is the task set ($\boldsymbol{x}_i$ features extracted from *hotel*)

Table A.5.: Properties of the Expedia dataset and the sub-sampled training queries of size 10.

		# Features Missing Values			# Instances				# Objects
Learning Problem	# Features	# All	> 90 %	> 50 %	# Total	# Train	# Test	# Sampled Train	$\lvert Q \rvert$
Choice	45	31	17	28	399 344	79 855	319 489	238 744	[38, 5]
Singleton choice	45	31	17	28	390 270	78 041	312 229	166 940	[38, 5]

and $l_i = (l_1, \dots, l_n)$ represents the vector of relevance label for the given set of objects, such that $\boldsymbol{x}_j \in [-1, \infty]^{45}$, $l_j \in \{0, 1, 2\}$ for each $j \in [n]$ and $5 \leq |\tilde{Q}_i| \leq 38$ for all $i \in [N]$.

The number of instances N in this dataset is 399 344, i. e., $|\mathcal{D}_E| = 399344$ and the size of the universal set of objects (hotels) is 136 886, i. e., $|\mathcal{X}| = 136886$. There are 31 features that have missing values, and we removed the features that consist of more than 50 % missing values. For the remaining 3 features that have missing values, we impute them with a negative value less than -1. The models are trained on the resulting dataset with 17 features.

Data Conversion Process

Data Conversion Process We create 5 folds by shuffle-splitting the dataset randomly into 80 % test and 20 % train instances. The choice dataset is created by considering the hotels in $\tilde{Q}_i$ as the task set Q_i and the set of relevant hotels $C_i := \{\boldsymbol{x}_j \in \tilde{Q}_i : l_j \in \{1, 2\}\}$ as the corresponding choice set for each instance $(\tilde{Q}_i, l_i) \in \mathcal{D}_E$. The models are trained on the sampled training dataset and corresponding test dataset using 5-fold stratified cross-validation as described in Table A.5.

Singleton Choice

Singleton Choice In order to create the singleton choice dataset, we consider the samples where the user booked the hotel, which is the singleton choice for the given query. The singleton choice dataset is created by considering the hotels in $\tilde{Q}_i$ as the task set Q_i and the set of booked hotels $C_i = \{\boldsymbol{x}_j \in \tilde{Q}_i : l_j = 2\}$ as the corresponding choice set for each instance $(\tilde{Q}_i, l_i) \in \mathcal{D}_E$.

The models are trained on the sampled training dataset and corresponding test dataset using 5-fold stratified cross-validation as described in Table A.5. Note the instances where the hotel was not booked were discarded, and only the instances where there was a booking were considered.

A.4.3. SUSHI Dataset

SUSHI[5] was another dataset released for solving the task of *object ranking*. This dataset was collected by surveying 5000 individuals, such that each person was provided with two item sets A and B. Set A consists of the 10 most famous sushis, and B consists of the top 100 sushis in Japan. Individuals were asked to provide preferences in the form of a total order for items in set A and a real-valued score between 0 and 5 for sushi in set B. There were missing rating values for many items in set B, so they extracted the total order for the top 10 preferred items by each user.

5: This dataset can be downloaded from `http://www.kamishima.net/sushi/`

The SUSHI dataset consists of universal set of objects $\boldsymbol{x} \in \mathcal{X}$, with size 100, i. e., $|\mathcal{X}| = 100$, with 10 000 set of object Q of size 10 and each sushi consists of 7 features, i. e., $\boldsymbol{x} \in \mathbb{R}^7$. The instances of the dataset $\mathcal{D}_S = \{(Q_1, \pi_1), \dots, (Q_N, \pi_N)\}$,

Table A.6.: Major Group feature description

Major Group					
Value	Species	Value	Species	Value	Species
0	Aomono (blue-skinned fish)	4	Clam or shell	8	Other seafood
1	Akami (red meat fish)	5	Squid or octopus	9	Egg
2	Shiromi (white-meat fish)	6	Shrimp or crab	10	Meat other than fish
3	Tare (something like baste for eel)	7	Roe	11	Vegetables

are represented as a set of tuples (Q_i, π_i), where $Q_i = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ is the set of objects and π_i represents the underlying orderings for the given set of objects Q_i, such that $N = |\mathcal{D}_M| = 10000$, $\boldsymbol{x}_i \in \mathbb{R}^7$ and $|Q_i| = 10$ holds for all $i \in [N]$.

The dataset contains the following features:

1. Style: This binary feature describes whether the sushi is a Maki or other, where 0 means Maki sushi and 1 means others.
2. Major Group: This is a binary feature, which describes whether it is listed as a seafood (0) or not (1).
3. Minor group: Described the species group used to prepare the sushi. The group is denoted by the categorical value between 0 and 11, i.e., it lies in the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Refer to Table A.6 for a description of each group.
4. Oiliness/Heaviness: the amount of oil or fat present in the sushi, expressed as a real number between 0 and 4, where 0 indicates heavy/oil and 4 oil-free.
5. Demand: The frequency with which the user demands the sushi, expressed as a real number between 0 and 3, where 3 means most frequently and 0 not at all.
6. Normalized Price: The price of sushi normalized over the given 100 sushis.
7. Supply: The frequency of selling the sushi in the shop, expressed as a real number between 0 and 1, where 0 indicates not at all and 1 frequently.

Singleton Choice Data Conversion For using the SUSHI dataset for singleton choice setting, we re-utilize the set of object Q in $\mathcal{D}_S$ and choose the most preferred object as the singleton choice. We created the singleton choice dataset $\mathcal{D}_{SDC} = \{(Q_1, C_1), \dots, (Q_N, C_N)\}$ with $N = |\mathcal{D}_S|$ instances, such that $|Q_k| = 10$ and $C_k := \left\{\boldsymbol{x}_{\pi_i(1)}\right\}$ for all $k \in [N]$. The singleton choice models are evaluated using 5-folds by train-test shuffle-split with 80 % train and 20 % test instances.

A.5. Detailed Experimental Results

The following tables Tables A.7 to A.9 contain all experimental results of the evaluation for Chapter 5, as discussed in Section 5.5.2 in numeric form for all evaluation measures.

Table A.7.: Results for the general subset choice models (mean and standard deviation of different measures, measured across 5 outer cross-validation folds). The best entry for each measure is marked in bold.

Dataset	Choice Model	F_1-measure	Subset 0/1 Accuracy	Informedness	AUC-ROC
Pareto-front-2D	FETA-Net	**0.942±0.008**	**0.680±0.028**	**0.956±0.012**	**0.999**
	FATE-Net	0.912±0.009	0.506±0.037	0.911±0.006	0.996±0.001
	FETA-Linear	0.673±0.001	0.064±0.007	0.694±0.015	0.955
	SDA	0.805±0.014	0.223±0.031	0.806±0.014	0.984±0.002
	RankNet	0.612±0.007	0.060±0.010	0.672±0.014	0.971±0.006
	PairwiseSVM	0.588±0.001	0.044±0.003	0.646±0.007	0.956
	GenLinearModel	0.585±0.008	0.044±0.005	0.633±0.013	0.952±0.007
	AllPositive	0.232	0.000	0.000	0.500
Pareto-front-5D	FETA-Net	0.826±0.005	0.001	0.406±0.032	0.854±0.014
	FATE-Net	**0.904±0.004**	**0.115±0.209**	**0.743±0.094**	**0.958±0.021**
	FETA-Linear	0.823±0.039	0.002±0.002	0.379±0.257	0.808±0.140
	SDA	0.887±0.002	0.013±0.001	0.656±0.012	0.935±0.002
	RankNet	0.859	0.006	0.581±0.006	0.923
	PairwiseSVM	0.839	0.002	0.491±0.021	0.895
	GenLinearModel	0.826±0.029	0.002±0.001	0.402±0.225	0.738±0.351
	AllPositive	0.775	0.000	0.000	0.500
MNIST-Unique	FETA-Net	0.963±0.003	0.814±0.020	0.945±0.005	0.992±0.001
	FATE-Net	**0.973±0.004**	**0.848±0.021**	**0.960±0.006**	**0.995±0.001**
	FETA-Linear	0.562±0.001	0.000±0.001	0.000±0.001	0.517±0.001
	SDA	0.942±0.001	0.702±0.006	0.915±0.002	0.984
	RankNet	0.562	0.000	0.000	0.504±0.001
	PairwiseSVM	0.562	0.000	0.000	0.511±0.006
	GenLinearModel	0.562	0.000	0.000	0.508±0.004
	AllPositive	0.562	0.000	0.000	0.500
MNIST-Mode	FETA-Net	0.809±0.005	0.311±0.032	0.695±0.009	0.981±0.006
	FATE-Net	**0.976±0.001**	**0.883±0.010**	**0.961±0.002**	**0.992±0.001**
	FETA-Linear	0.597±0.001	0.003	0.003±0.002	0.516±0.001
	SDA	0.863±0.002	0.357±0.009	0.807±0.002	0.973±0.001
	RankNet	0.597	0.003	0.000	0.503±0.002
	PairwiseSVM	0.597	0.003	0.000	0.509±0.006
	GenLinearModel	0.597	0.003	0.000	0.497±0.004
	AllPositive	0.597	0.003	0.000	0.500
LETORMQ2007	FETA-Net	**0.477±0.004**	**0.007±0.002**	**0.235±0.009**	**0.729±0.011**
	FATE-Net	0.470±0.002	0.000	0.232±0.002	0.704±0.002
	FETA-Linear	0.452±0.022	0.001±0.002	0.231±0.035	0.694±0.006
	SDA	0.441±0.015	0.001±0.002	0.195±0.022	0.666±0.003
	RankNet	0.427±0.010	0.001±0.012	0.029±0.007	0.610±0.015
	PairwiseSVM	0.453±0.021	0.000	0.220±0.026	0.696±0.007
	GenLinearModel	0.427±0.021	0.001±0.002	0.058±0.029	0.614±0.009
	AllPositive	0.421±0.021	0.001±0.002	0.000	0.500
LETORMQ2008	FETA-Net	0.537±0.001	**0.044±0.001**	**0.440±0.003**	**0.842±0.004**
	FATE-Net	**0.540±0.005**	0.041±0.002	0.431±0.002	0.837±0.006
	FETA-Linear	0.529±0.006	0.026±0.009	0.421±0.012	0.803±0.009
	SDA	0.425±0.041	0.018±0.011	0.287±0.038	0.727±0.023
	RankNet	0.461±0.002	0.017±0.004	0.323±0.002	0.758±0.004
	PairwiseSVM	0.526±0.022	0.042±0.022	0.428±0.016	0.786±0.018
	GenLinearModel	0.493±0.028	0.014±0.010	0.311±0.061	0.739±0.019
	AllPositive	0.424±0.021	0.000	0.000	0.500
Expedia	FETA-Net	0.186±0.001	0.009±0.002	0.322±0.003	0.688±0.001
	FATE-Net	0.198±0.006	0.018±0.002	0.346±0.010	0.707±0.007
	FETA-Linear	0.179±0.007	**0.020±0.002**	0.324±0.006	0.696±0.007
	SDA	**0.201±0.005**	0.013±0.003	**0.352±0.012**	0.708±0.008
	RankNet	0.167±0.017	0.003±0.001	0.278±0.034	**0.716±0.006**
	PairwiseSVM	0.129±0.017	0.004±0.002	0.165±0.097	0.680±0.050
	GenLinearModel	0.107±0.001	0.000	0.004±0.007	0.503±0.102
	AllPositive	0.106	0.000	0.000	0.500

Table A.8.: Mean and standard deviation of the accuracies on the singleton choice data (measured across 5 outer cross-validation folds). The best entry for each measure is marked in bold.

Dataset	SCM	Accuracy	Top-3	Top-5
Medoid	FETA-Net	0.846±0.010	0.994±0.001	**1.000**
	FATE-Net	**0.881±0.007**	**0.996±0.001**	**1.000**
	FETA-Linear	0.356±0.026	0.715±0.007	0.883±0.011
	SDA	0.839±0.004	0.987±0.001	0.998
	RankNet	0.531±0.008	0.873±0.006	0.970±0.004
	PairwiseSVM	0.021±0.001	0.194±0.009	0.501±0.002
	MNL	0.020±0.001	0.191±0.005	0.500±0.001
	NL	0.049±0.014	0.216±0.006	0.463±0.027
	GNL	0.020	0.195±0.004	0.500±0.001
	ML	0.003	0.055±0.012	0.249±0.032
Hypervolume	FETA-Net	**0.769±0.022**	**0.933±0.007**	**0.980±0.001**
	FATE-Net	0.730±0.018	0.920±0.013	0.968±0.006
	FETA-Linear	0.236±0.042	0.404±0.042	0.560±0.028
	SDA	0.233±0.019	0.417±0.029	0.589±0.036
	RankNet	0.203±0.004	0.369±0.006	0.562±0.004
	PairwiseSVM	0.186±0.001	0.340±0.002	0.550±0.002
	MNL	0.201±0.008	0.360±0.010	0.559±0.004
	NL	0.291±0.003	0.511±0.007	0.651±0.006
	GNL	0.293±0.018	0.471±0.021	0.663±0.014
	ML	0.189±0.014	0.451±0.019	0.621±0.014
MNIST-Unique	FETA-Net	**0.972±0.002**	**0.995±0.001**	**0.998**
	FATE-Net	0.954±0.009	0.993±0.001	**0.998±0.001**
	FETA-Linear	0.127±0.006	0.320±0.003	0.505±0.010
	SDA	0.858±0.029	0.935±0.026	0.955±0.018
	RankNet	0.134±0.008	0.307±0.002	0.495±0.002
	PairwiseSVM	0.124±0.010	0.319±0.008	0.502±0.007
	MNL	0.170±0.006	0.325±0.009	0.495±0.002
	NL	0.207±0.016	0.354±0.004	0.502±0.006
	GNL	0.651±0.006	0.763±0.003	0.841±0.001
	ML	0.490±0.003	0.718±0.005	0.784±0.002
MNIST-Mode	FETA-Net	**0.908±0.004**	**0.961±0.003**	**0.978±0.004**
	FATE-Net	0.669±0.005	0.907±0.004	0.943±0.003
	FETA-Linear	0.290±0.006	0.674±0.010	0.877±0.007
	SDA	0.513±0.047	0.806±0.041	0.901±0.061
	RankNet	0.284±0.002	0.668±0.003	0.876±0.003
	PairwiseSVM	0.289±0.007	0.675±0.011	0.881±0.007
	MNL	0.285±0.006	0.652±0.011	0.853±0.010
	NL	0.282±0.007	0.646±0.012	0.848±0.010
	GNL	0.274±0.003	0.641±0.008	0.849±0.006
	ML	0.216±0.010	0.536±0.020	0.765±0.022
Tag Genome Similar Movie	FETA-Net	0.184±0.001	0.481±0.002	0.699±0.002
	FATE-Net	**0.185±0.003**	**0.482±0.006**	0.699±0.004
	FETA-Linear	0.138±0.009	0.391±0.023	0.613±0.030
	SDA	0.099±0.022	0.306±0.050	0.511±0.058
	RankNet	0.174±0.003	0.477±0.002	**0.708±0.003**
	PairwiseSVM	0.145±0.011	0.405±0.019	0.626±0.018
	MNL	0.179±0.002	0.472±0.003	0.694±0.004
	NL	0.178±0.004	0.467±0.006	0.689±0.007
	GNL	0.179±0.002	0.472±0.003	0.694±0.003
	ML	0.117±0.001	0.353±0.009	0.575±0.013

Table A.9.: Cont.: Mean and standard deviation of the accuracies on the singleton choice data (measured across 5 outer cross-validation folds). The best entry for each measure is marked in bold.

Dataset	SCM	Accuracy	Top-3	Top-5
Tag Genome Dissimilar Movie	FETA-Net	**0.512±0.004**	**0.835±0.004**	**0.942±0.002**
	FATE-Net	0.510±0.001	0.830±0.002	0.938±0.002
	FETA-Linear	0.440±0.002	0.759±0.002	0.889±0.001
	SDA	0.451±0.047	0.694±0.072	0.789±0.054
	RankNet	0.435±0.002	0.779±0.001	0.914±0.001
	PairwiseSVM	0.369±0.016	0.712±0.012	0.871±0.008
	MNL	0.447±0.002	0.692±0.005	0.795±0.005
	NL	0.438±0.006	0.671±0.015	0.775±0.018
	GNL	0.443±0.004	0.681±0.010	0.784±0.011
	ML	0.417±0.003	0.763±0.001	0.895±0.005
LETORMQ2007-list	FETA-Net	**0.334±0.007**	**0.577±0.012**	**0.705±0.006**
	FATE-Net	0.288±0.002	0.508±0.006	0.639±0.004
	FETA-Linear	0.293±0.018	0.551±0.007	0.697±0.007
	SDA	0.047±0.013	0.137±0.007	0.211±0.014
	RankNet	0.287±0.033	0.513±0.050	0.627±0.037
	PairwiseSVM	0.302±0.008	0.541±0.031	0.654±0.039
	MNL	0.282±0.006	0.503±0.029	0.622±0.038
	NL	0.285±0.018	0.499±0.030	0.608±0.043
	GNL	0.287±0.020	0.509±0.029	0.625±0.037
	ML	0.282±0.005	0.503±0.038	0.628±0.037
LETORMQ2008-list	FETA-Net	0.266±0.015	0.396±0.019	0.504±0.017
	FATE-Net	**0.281±0.012**	0.369±0.015	**0.544±0.012**
	FETA-Linear	0.197±0.007	0.392±0.027	0.506±0.032
	SDA	0.028±0.007	0.078±0.032	0.124±0.034
	RankNet	0.225±0.026	**0.399±0.020**	0.501±0.023
	PairwiseSVM	0.203±0.014	0.376±0.032	0.497±0.021
	MNL	0.217±0.025	0.362±0.020	0.500±0.027
	NL	0.212±0.024	0.355±0.030	0.472±0.030
	GNL	0.222±0.020	0.366±0.034	0.494±0.026
	ML	0.213±0.015	0.367±0.019	0.501±0.025
Expedia	FETA-Net	**0.215±0.006**	**0.451±0.016**	0.587±0.008
	FATE-Net	0.203±0.006	0.434±0.003	0.576±0.003
	FETA-Linear	0.176±0.003	0.394±0.002	0.543±0.003
	SDA	0.115±0.008	0.288±0.014	0.431±0.015
	RankNet	0.210±0.001	0.445±0.001	**0.590±0.001**
	PairwiseSVM	0.179	0.405±0.001	0.550
	MNL	0.199±0.004	0.423±0.005	0.565±0.004
	NL	0.171±0.006	0.388±0.008	0.534±0.008
	GNL	0.168±0.006	0.385±0.010	0.531±0.009
	ML	0.181±0.010	0.406±0.010	0.551±0.007
SUSHI	FETA-Net	0.295±0.003	0.552±0.003	0.766±0.003
	FATE-Net	**0.322±0.003**	**0.589±0.005**	**0.817±0.005**
	FETA-Linear	0.273±0.006	0.500±0.014	0.680±0.012
	SDA	0.270±0.015	0.498±0.043	0.689±0.043
	RankNet	0.272±0.007	0.559±0.035	0.721±0.016
	PairwiseSVM	0.258±0.004	0.480±0.022	0.679±0.013
	MNL	0.271±0.004	0.502±0.003	0.677±0.010
	NL	0.253±0.006	0.533±0.019	0.730±0.025
	GNL	0.259±0.007	0.562±0.023	0.735±0.016
	ML	0.281±0.004	0.575±0.013	0.777±0.007

B. List of Publications

- Arthur Klippenstein, Christoph Weskamp, Florian Laux, Florian Neuhaus, Karlson Pfannschmidt, Melissa Bülling, and Stephan Kassanke. "A Prototype to Support Business Model Innovation Through Crowdsourcing and Artificial Intelligence." In: *Wirtschaftsinformatik 2023 Proceedings*. WI 2023 (Paderborn, Germany, Sept. 18, 2023). Ed. by Daniel Beverungen, Christiane Lehrer, and Matthias Trier. 47. Sept. 8, 2023
- Michael Dellnitz, Eyke Hüllermeier, Marvin Lücke, Sina Ober-Blöbaum, Christian Offen, Sebastian Peitz, and Karlson Pfannschmidt. "Efficient Time-Stepping for Numerical Integration Using Reinforcement learning." en. In: *SIAM J. Sci. Comput.* 45 (2 Apr. 30, 2023)
- Karlson Pfannschmidt and Eyke Hüllermeier. "A Characterization of Choice Functions Representable by Pareto-Embedding of Alternatives." In: *DA2PL 2022 Proceedings*. DA2PL 2022 (Compiègne, France, Nov. 17, 2022). Nov. 17, 2022
- Karlson Pfannschmidt, Pritha Gupta, Björn Haddenhorst, and Eyke Hüllermeier. "Learning Context-Dependent Choice Functions." In: *Int. J. Approx. Reason.* 140 (2022)
- Karlson Pfannschmidt and Eyke Hüllermeier. "Learning Choice Functions via Pareto-Embeddings." In: *KI 2020: Advances in Artificial Intelligence*. KI 2020 (2020). Ed. by Ute Schmid, Franziska Klügl, and Diedrich Wolter. Lecture Notes in Computer Science. Springer International Publishing, 2020
- Karlson Pfannschmidt, Pritha Gupta, and Eyke Hüllermeier. "Deep Architectures for Learning Context-Dependent Ranking Functions." Mar. 15, 2018. eprint: 1803.05796 (stat.ML)
- Karlson Pfannschmidt, Eyke Hüllermeier, Susanne Held, and Reto Neiger. "Evaluating Tests in Medical Diagnosis: Combining Machine Learning with Game-Theoretical Concepts." In: IPMU 2016 (Eindhoven, The Netherlands, June 20, 2016). Ed. by Joao Paulo Carvalho, Marie-Jeanne Lesot, Uzay Kaymak, Susana Vieira, Bernadette Bouchon-Meunier, and Ronald R Yager. Springer International Publishing, June 20, 2016
- Kalina Jasinska, Krzysztof Dembczyński, Robert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hüllermeier. "Extreme F-Measure Maximization Using Sparse Probability Estimates." In: *Proceedings of The 33rd International Conference on Machine Learning*. ICML 2016 (New York City, NY, USA, June 20, 2016). Ed. by Maria Florina Balcan and Kilian Q Weinberger. Vol. 48. Proceedings of machine learning research. PMLR, 2016

Bibliography

[AML12] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from Data: A Short Course*. AMLbook.com, 2012. 201 pp. (cited on page 44).

[ADS10] Shivani Agarwal, Deepak Dugar, and Shiladitya Sengupta. "Ranking Chemical Structures for Drug Discovery: A New Machine Learning Approach." In: *J. Chem. Inf. Model.* 50 (5 2010). PMID: 20387860, pp. 716–731. doi: 10.1021/ci9003865 (cited on page 78).

[AY01] Charu C Aggarwal and Philip S Yu. "Outlier Detection for High Dimensional Data." In: *SIGMOD Rec.* 30 (2 June 2001), pp. 37–46. doi: 10.1145/376284.375668 (cited on page 213).

[Ai+18] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. "Learning a Deep Listwise Context Model for Ranking Refinement." In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR 2018 (Ann Arbor, MI, USA). 41. New York, NY, USA: Association for Computing Machinery, 2018, pp. 135–144. doi: 10.1145/3209978.3209985 (cited on pages 78, 80).

[ASS10] Artur Aiguzhinov, Carlos Soares, and Ana Paula Serra. "A Similarity-Based Adaptation of Naive Bayes for Label Ranking: Application to the Metalearning Problem of Algorithm Recommendation." In: *Discovery Science*. Springer Berlin Heidelberg, 2010, pp. 16–26. doi: 10.1007/978-3-642-16184-1_2 (cited on page 75).

[ACN08] Nir Ailon, Moses Charikar, and Alantha Newman. "Aggregating Inconsistent Information: Ranking and Clustering." In: *J. ACM* 55 (5 Oct. 5, 2008), pp. 1–27. doi: 10.1145/1411509.1411513 (cited on page 99).

[Aki20] Andrey Akinshin. *Nonparametric Cohen's d-Consistent Effect Size*. June 25, 2020. url: https://aakinshin.net/posts/nonparametric-effect-size/ (visited on 05/14/2022) (cited on page 182).

[All53] M Allais. "Le Comportement de l'Homme Rationnel devant le Risque: Critique des Postulats et Axiomes de l'Ecole Americaine." In: *Econometrica* 21 (4 Oct. 1953), p. 503. doi: 10.2307/1907921 (cited on page 20).

[Alt+10] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. "Permutation Importance: A Corrected Feature Importance Measure." In: *Bioinformatics* 26 (10 May 15, 2010), pp. 1340–1347. doi: 10.1093/bioinformatics/btq134 (cited on page 152).

[AIM21] Pavel Anselmo Alvarez, Alessio Ishizaka, and Luis Martínez. "Multiple-Criteria Decision-Making Sorting Methods: A Survey." In: *Expert Syst. Appl.* 183 (2021), p. 115368 (cited on page 114).

[AR15] Attila Ambrus and Kareen Rozen. "Rationalising Choice with Multi-Self Models." In: *Econ. J. Nepal* 125 (585 2015), pp. 1136–1156. doi: 10.1111/ecoj.12103 (cited on pages 36, 172).

[AT15] Ilaria L Amerise and Agostino Tarsitano. "Correction Methods for Ties in Rank Correlations." In: *J. Appl. Stat.* 42 (12 Dec. 2, 2015), pp. 2584–2596. doi: 10.1080/02664763.2015.1043870 (cited on page 91).

[And03] Amy C Anderson. "The Process of Structure-Based Drug Design." In: *Chem. Biol.* 10 (9 Sept. 2003), pp. 787–797. doi: 10.1016/j.chembiol.2003.09.002 (cited on page 67).

[Arr51] Kenneth J Arrow. *Social Choice and Individual Values*. John Wiley & Sons, 1951 (cited on pages 20, 22).

[AS05] Kaan Ataman and W Nick Street. *Optimizing Area Under the ROC Curve using Ranking SVMs*. 2005. url: https://www.biz.uiowa.edu/faculty/nstreet/research/kdd05kaan.pdf (visited on 06/21/2024) (cited on page 78).

[BHN23] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023 (cited on page 20).

[Bar93] Andrew R Barron. "Universal Approximation Bounds for Superpositions of a Sigmoidal Function." In: *IEEE Trans. Inf. Theory* 39 (3 May 1993), pp. 930–945. doi: 10.1109/18.256500 (cited on page 63).

[Bas84] Kaushik Basu. "Fuzzy Revealed Preference Theory." In: *J. Econ. Theory* 32 (2 Apr. 1984), pp. 212–227. doi: 10.1016/0022-0531(84)90051-6 (cited on page 26).

[BP85] Richard R Batsell and John C Polking. "A New Class of Market Share Models." In: *Marketing Science* 4 (3 1985), pp. 177–198 (cited on pages 122, 123).

[Bat+18] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. "Relational Inductive Biases, Deep Learning, and Graph Networks." June 4, 2018. eprint: 1806.01261 (cs.LG) (cited on page 125).

[Bay+18] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic Differentiation in Machine Learning: a Survey." In: *J. Mach. Learn. Res.* 18 (153 2018), pp. 1–43 (cited on page 59).

[BDM63] Gordon M Becker, Morris H Degroot, and Jacob Marschak. "Probabilities of Choices among Very Similar Objects: An Experiment to Decide between Two Models." In: *Behav. Sci.* 8 (4 1963), pp. 306–311. doi: 10/bnjftm (cited on pages 2, 26, 30).

[BP21] Vaishak Belle and Ioannis Papantonis. "Principles and Practice of Explainable Machine Learning." In: *Front. Big Data* 4 (688969 July 1, 2021). doi: 10.3389/fdata.2021.688969 (cited on page 193).

[Bel55] R Bellman. "Functional Equations in the Theory of Dynamic Programming. V. Positivity and quasi-linearity." In: *Proc. Natl. Acad. Sci. U. S. A.* 41 (10 Oct. 15, 1955), pp. 743–746. doi: 10.1073/pnas.41.10.743 (cited on page 55).

[BB99] Moshe Ben-Akiva and Michel Bierlaire. "Discrete Choice Methods and their Applications to Short Term Travel Decisions." In: *International Series in Operations Research & Management Science*. Boston, MA: Springer US, 1999, pp. 5–33. doi: 10.1007/978-1-4615-5203-1_2 (cited on page 35).

[BL18] Moshe E Ben-Akiva and Steven R Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. Transportation Studies. London, England: MIT Press, Apr. 20, 2018. 412 pp. (cited on pages 35, 87, 134).

[BAP23a] Alessio Benavoli, Dario Azzimonti, and Dario Piga. "Bayesian Optimization for Choice Data." In: *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. GECCO '23 Companion: Companion Conference on Genetic and Evolutionary Computation (Lisbon Portugal). New York, NY, USA: ACM, July 15, 2023. doi: 10.1145/3583133.3596324 (cited on page 172).

[BAP23b] Alessio Benavoli, Dario Azzimonti, and Dario Piga. "Learning Choice Functions with Gaussian Processes." In: *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*. Ed. by Robin J Evans and Ilya Shpitser. Vol. 216. Proceedings of Machine Learning Research. PMLR, 2023, pp. 141–151 (cited on pages 172, 187).

[BH20] Viktor Bengs and Eyke Hüllermeier. "Preselection Bandits." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé Iii and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 778–787 (cited on pages 152, 193).

[BB00] Kristin P Bennett and Erin J Bredensteiner. "Duality and Geometry in SVM Classifiers." In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*. International Conference on Machine Learning (Stanford, CA, USA, June 29, 2000). Ed. by Pat Langley. San Francisco, CA: Morgan Kaufmann, June 29, 2000, pp. 57–64 (cited on page 136).

[BKT16] Austin R Benson, Ravi Kumar, and Andrew Tomkins. "On the Relevance of Irrelevant Alternatives." In: *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*. Ed. by Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y Zhao. ACM, 2016, pp. 963–973. doi: 10.1145/2872427.2883025 (cited on pages 29, 35).

[BKT18] Austin R Benson, Ravi Kumar, and Andrew Tomkins. "A Discrete Choice Model for Subset Selection." In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM 2018 (Los Angeles, California, USA, Feb. 5, 2018). Ed. by Yi Chang, Chengxiang Zhai, Yan Liu, and Yoelle Maarek. ACM, 2018, pp. 37–45. doi: 10.1145/3159652.3159702 (cited on pages 3, 8, 83, 115).

[Ben89] Jeremy Bentham. *An Introduction to the Principles of Morals and Legislation*. London: T. Payne, and Son, 1789 (cited on page 22).

[BCL93] Jon L Bentley, Kenneth L Clarkson, and David B Levine. "Fast Linear Expected-Time Algorithms for Computing Maxima and Convex Hulls." In: *Algorithmica* 9 (2 Feb. 1993), pp. 168–183. doi: 10.1007/bf01188711 (cited on page 165).

[Ber+11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization." In: *Advances in Neural Information Processing Systems*. NIPS 2011 (Granada, Spain, Dec. 12, 2011). Ed. by J Shawe-Taylor, R Zemel, P Bartlett, F Pereira, and K Q Weinberger. Vol. 24. Curran Associates, Inc., 2011 (cited on page 52).

[Ber44] Joseph Berkson. "Application of the Logistic Function to Bio-Assay." In: *J. Am. Stat. Assoc.* 39 (227 1944), pp. 357–365 (cited on page 115).

[Ber51] Joseph Berkson. "Why I Prefer Logits to Probits." In: *Biometrics* 7 (4 Dec. 1951), p. 327. doi: 10.2307/3001655 (cited on page 33).

[BR04] B Douglas Bernheim and Antonio Rangel. "Addiction and Cue-Triggered Decision Processes." In: *Am. Econ. Rev.* 94 (5 Dec. 2004), pp. 1558–1590. doi: 10.1257/0002828043052222 (cited on page 36).

[Ber38] Daniel Bernoulli. "Specimen Theoriae Novae de Mensura Sortis." In: *Commentarii Academiae Scientiarum Imperialis Petropolitanae*. Vol. 5. 1738, pp. 175–192 (cited on pages 1, 19, 22).

[BT24] Matteo Biagiola and Paolo Tonella. "Testing of Deep Reinforcement Learning Agents with Surrogate Models." In: *ACM Trans. Softw. Eng. Methodol.* 33 (3 Mar. 31, 2024), pp. 1–33. doi: 10.1145/3631970 (cited on page 152).

[Bie98] Michel Bierlaire. "Discrete Choice Models." In: *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 203–227. doi: 10.1007/978-3-662-03514-6_9 (cited on page 35).

[BI20] Francesco Biscani and Dario Izzo. "A Parallel Global Multiobjective Framework for Optimization: pagmo." In: *J. Open Source Softw.* 5 (53 2020), p. 2338. doi: 10.21105/joss.02338 (cited on page 176).

[Bis+23] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, Difan Deng, and Marius Lindauer. "Hyper-parameter Optimization: Foundations, Algorithms, Best Practices, and Open Challenges." In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 13 (2 Jan. 16, 2023), e1484. doi: 10.1002/widm.1484 (cited on page 49).

[Bli34] Chester I Bliss. "The Method of Probits." In: *Science* 79 (2037 Jan. 12, 1934), pp. 38–39. doi: 10.1126/science.79.2037.38 (cited on page 33).

[BW67] Daniel A Bloch and Geoffrey S Watson. "A Bayesian Study of the Multinomial Distribution." In: *Ann. Math. Stat.* 38 (5 Oct. 1967), pp. 1423–1435. doi: 10.1214/aoms/1177698697 (cited on page 33).

[Blu+15] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. "Weight Uncertainty in Neural Network." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1613–1622 (cited on page 53).

[Boc69] R Bock. "Estimating Multinomial Response Relations." In: *Contributions to Statistics and Probability: Essays in Memory of S. N. Roy*. Ed. by R Bose. Univ. of North Carolina Press, 1969 (cited on page 33).

[BR17] Zdravko Botev and Ad Ridder. *Variance Reduction*. In: *Wiley StatsRef: Statistics Reference Online*. Chichester, UK: John Wiley & Sons, Ltd, Nov. 15, 2017, pp. 1–6. doi: 10.1002/9781118445112.stat07975 (cited on page 174).

[BV20] Xavier Bouthillier and Gaël Varoquaux. *Survey of Machine-Learning Experimental Methods at NeurIPS2019 and ICLR2020*. Research rep. Inria Saclay Ile de France, Jan. 2020 (cited on page 49).

[BV04] Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge, England: Cambridge University Press, Mar. 8, 2004. 744 pp. (cited on page 137).

[BT52] Ralph Allan Bradley and Milton E Terry. "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons." In: *Biometrika* 39 (3/4 1952), pp. 324–345 (cited on page 115).

[BSC03] Pavel B Brazdil, Carlos Soares, and Joaquim Pinto da Costa. "Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results." In: *Mach. Learn.* 50 (3 Mar. 1, 2003), pp. 251–277. doi: 10.1023/A:1021713901879 (cited on page 75).

[BS05] Ulf Brefeld and Tobias Scheffer. "AUC Maximizing Support Vector Learning." In: *Proceedings of the ICML 2005 Workshop on ROC Analysis in Machine Learning* (Bonn, Germany, 2005). 2005 (cited on page 78).

[Bre01] Leo Breiman. "Random Forests." In: *Mach. Learn.* 45 (1 Oct. 1, 2001), pp. 5–32. doi: 10.1023/A:1010933404324 (cited on page 52).

[BF10] Karl Bringmann and Tobias Friedrich. "Approximating the Volume of Unions and Intersections of High-Dimensional Geometric Objects." In: *Comput. Geom.* 43 (6 2010), pp. 601–610 (cited on page 141).

[BFH06] Klaus Brinker, Johannes Fürnkranz, and Eyke Hüllermeier. "A Unified Model for Multilabel Classification and Ranking." In: *Proceedings of the 17th European Conference on Artificial Intelligence*. European Conference on Artificial Intelligence (ECAI 2006) (Riva del Garda, Italy). NLD: IOS Press, May 22, 2006, pp. 489–493 (cited on page 76).

[Bro+10] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. "The Balanced Accuracy and its Posterior Distribution." In: *Proceedings of the International Conference on Pattern Recognition*. ICPR 2010 (Istanbul, Turkey, Aug. 23, 2010). IEEE, Aug. 2010, pp. 3121–3124. doi: 10.1109/icpr.2010.764 (cited on page 89).

[BB23] Tanja Burgard and André Bittermann. "Reducing Literature Screening Workload with Machine Learning: A Systematic Review of Tools and Their Performance." In: *Z. Psychol.* 231 (1 Feb. 22, 2023), pp. 3–15. doi: 10.1027/2151-2604/a000509 (cited on page 194).

[Bur10] Christopher J C Burges. *From RankNet to LambdaRank to LambdaMART: An Overview*. Tech. rep. MSR-TR-2010-82. Microsoft Research, 2010 (cited on page 99).

[Bur+05] Christopher J C Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N Hullender. "Learning to Rank using Gradient Descent." In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML 2005 (Bonn, Germany, Aug. 7, 2005). Vol. 119. ACM, 2005, pp. 89–96 (cited on pages 78, 80, 98, 134, 175).

[Bus+14] Róbert Busa-Fekete, Balázs Szörényi, Paul Weng, Weiwei Cheng, and Eyke Hüllermeier. "Preference-Based Reinforcement Learning: Evolutionary Direct Policy Search using a Preference-Based Racing Algorithm." In: *Mach. Learn.* 97 (3 Dec. 2, 2014), pp. 327–351. doi: 10.1007/s10994-014-5458-8 (cited on page 84).

[Bus+20] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. "Albumentations: Fast and Flexible Image Augmentations." In: *Information* 11 (2 Feb. 2020), p. 125. doi: 10.3390/info11020125 (cited on page 62).

[BOB07] Ludwig M Busse, Peter Orbanz, and Joachim M Buhmann. "Cluster Analysis of Heterogeneous Rank Data." In: *Proceedings of the 24th International Conference on Machine Learning*. ICML 2007 & ILP 2007 (Corvalis Oregon USA, June 20, 2007). New York, NY, USA: ACM, June 20, 2007. doi: 10.1145/1273496.1273511 (cited on page 85).

[Cal+16] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. "Manifold Gaussian Processes for Regression." In: *Proceedings of the 2016 International Joint Conference on Neural Networks*. IJCNN 2016 (Vancouver, BC, Canada, July 24, 2016). July 2016, pp. 3338–3345. doi: 10.1109/IJCNN.2016.7727626 (cited on page 53).

[Cal+10] Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. "Findings of the 2010 Joint Workshop on Statistical Machine Translation and Metrics for Machine Translation." In: *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*. 2010, pp. 17–53 (cited on page 91).

[Cao+06] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. "Adapting Ranking SVM to Document Retrieval." In: *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2006 (Seattle Washington USA, Aug. 6, 2006). New York, NY, USA: ACM, Aug. 6, 2006. doi: 10.1145/1148170.1148205 (cited on pages 77, 78).

[Cao+07] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. "Learning to Rank: From Pairwise Approach to Listwise Approach." In: *Proceedings of the 24th Annual International Conference on Machine Learning*. ICML 2007. Ed. by Zoubin Ghahramani. Omni Press, 2007, pp. 129–136 (cited on pages 78, 80, 103).

[CTM14] François Caron, Yee Whye Teh, and Thomas Brendan Murphy. "Bayesian Nonparametric Plackett–Luce Models for the Analysis of Preferences for College Degree Programmes." In: *Ann. Appl. Stat.* 8 (2 June 1, 2014), pp. 1145–1181. doi: 10.1214/14-aoas717 (cited on page 85).

[CC19] Andrea M Cataldo and Andrew L Cohen. "The Comparison Process as an Account of Variation in the Attraction, Compromise, and Similarity Effects." In: *Psychon. Bull. Rev.* 26 (3 June 2019), pp. 934–942. doi: 10.3758/s13423-018-1531-9 (cited on pages 32, 122).

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey." In: *ACM Comput. Surv.* 41 (3 2009), p. 15. doi: 10.1145/1541880.1541882 (cited on page 213).

[Cha+09] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. "Expected Reciprocal Rank for Graded Relevance." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM 2009 (Hong Kong China, Nov. 2, 2009). New York, NY, USA: ACM, Nov. 2, 2009. doi: 10.1145/1645953.1646033 (cited on page 92).

[CW10] Olivier Chapelle and Mingrui Wu. "Gradient Descent Optimization of Smoothed Information Retrieval Metrics." In: *Inf. Retr. Boston.* 13 (3 June 2010), pp. 216–235. doi: 10.1007/s10791-009-9110-3 (cited on page 102).

[Cha+17] R Qi Charles, Hao Su, Mo Kaichun, and Leonidas J Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR 2017 (Honolulu, HI, USA, July 21, 2017). July 2017, pp. 77–85. doi: 10.1109/CVPR.2017.16 (cited on page 66).

[CDH10] Weiwei Cheng, Krzysztof Dembczyński, and Eyke Hüllermeier. "Label Ranking Methods based on the Plackett-Luce Model." In: *Proceedings of the 27th International Conference on Machine Learning*. ICML 2010 (Haifa, Israel, June 21, 2010). Madison, WI, USA: Omnipress, June 21, 2010, pp. 215–222 (cited on page 75).

[CHH13] Weiwei Cheng, Sascha Henzgen, and Eyke Hüllermeier. "Labelwise versus Pairwise Decomposition in Label Ranking." In: *Proceedings of Lernen-Wissen-Adaption 2013*. LWA 2013 (Bamberg, Germany). Ed. by Andreas Henrich and Hans-Christian Sperker. Otto Friedrich University Bamberg, Oct. 2013, pp. 129–136 (cited on pages 74, 75).

[CHH09] Weiwei Cheng, Jens Hühn, and Eyke Hüllermeier. "Decision Tree and Instance-Based Learning for Label Ranking." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML 2009 (Montreal, Quebec, Canada, June 14, 2009). New York, NY, USA: Association for Computing Machinery, June 14, 2009, pp. 161–168. doi: 10.1145/1553374.1553395 (cited on page 75).

[CH08] Weiwei Cheng and Eyke Hüllermeier. "Instance-Based Label Ranking using the Mallows Model." In: *Workshop Proceedings of the 9th European Conference on Case-Based Reasoning*. ECCBR 2008 (Trier, Germany, Sept. 1, 2008). Ed. by Martin Schaaf. 2008, pp. 143–157 (cited on page 75).

[CH09] Weiwei Cheng and Eyke Hüllermeier. "A New Instance-Based Label Ranking Approach Using the Mallows Model." In: *Advances in Neural Networks – ISNN 2009*. Springer Berlin Heidelberg, 2009, pp. 707–716. doi: 10.1007/978-3-642-01507-6_80 (cited on page 75).

[Chi06] Dennis Child. *The Essentials of Factor Analysis*. 3rd. A&C Black, 2006 (cited on page 162).

[Chi60] John S Chipman. "Stochastic Choice and Subjective Probability." In: *Decisions, Values and Groups*. Ed. by Dorothy Willner. Pergamon, 1960, pp. 70–95. doi: 10.1016/B978-0-08-009237-9.50010-X (cited on pages 2, 28, 29).

[Cho+15] Kyunghyun Cho, Tapani Raiko, Alexander Ilin, and Juha Karhunen. "How to Pretrain Deep Boltzmann Machines in Two Stages." In: *Springer Series in Bio-/Neuroinformatics*. Cham: Springer International Publishing, 2015, pp. 201–219. doi: 10.1007/978-3-319-09903-3_10 (cited on page 83).

[Cho+17] François Chollet et al. *Keras*. 2017. url: https://keras.io (cited on page 87).

[Chr+17] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep Reinforcement Learning from Human Preferences." In: *Advances in Neural Information Processing Systems*. NIPS 2017 (Long Beach, CA, USA, Dec. 4, 2017). Ed. by I Guyon, U Von Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett. Vol. 30. Curran Associates, Inc., 2017 (cited on pages 3, 84).

[CS21] Stephen Chung and Hava Siegelmann. "Turing Completeness of Bounded-Precision Recurrent Neural Networks." In: *Advances in Neural Information Processing Systems*. NeurIPS 2021 (Virtual, Dec. 6, 2021). Ed. by M Ranzato, A Beygelzimer, Y Dauphin, P S Liang, and J Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 28431–28441 (cited on page 66).

[CV09] Stephan Clémençon and Nicolas Vayatis. "On Partitioning Rules for Bipartite Ranking." In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*. AISTATS 2009 (Clearwater Beach, Florida, USA, Apr. 16, 2009). Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. PMLR, Apr. 16, 2009, pp. 97–104 (cited on pages 78, 79).

[CDV09] Stéphan Clémençon, Marine Depecker, and Nicolas Vayatis. "Bagging Ranking Trees." In: *Proceedings of the International Conference on Machine Learning and Applications*. ICMLA 2009 (Miami Beach, Florida, USA, Dec. 13, 2009). IEEE, Dec. 2009, pp. 658–663. doi: 10.1109/icmla.2009.14 (cited on pages 78, 80).

[CDV13a] Stéphan Clémençon, Marine Depecker, and Nicolas Vayatis. "An Empirical Comparison of Learning Algorithms for Nonparametric Scoring: The TreeRank Algorithm and Other Methods." In: *Pattern Anal. Appl.* 16 (4 Nov. 2013), pp. 475–496. doi: 10.1007/s10044-012-0299-1 (cited on page 78).

[CDV13b] Stéphan Clémençon, Marine Depecker, and Nicolas Vayatis. "Ranking Forests." In: *Journal of Machine Learning Research* 14 (2 2013), pp. 39–73 (cited on pages 78, 80).

[CV08a] Stéphan Clémençon and Nicolas Vayatis. "Approximation of the Optimal ROC Curve and a Tree-Based Ranking Algorithm." In: *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 22–37. doi: 10.1007/978-3-540-87987-9_7 (cited on page 78).

[CV08b] Stéphan Clémençon and Nicolas Vayatis. "Tree-Structured Ranking Rules and Approximation of the Optimal ROC Curve." Apr. 4, 2008. eprint: hal-00268068v2 (cited on page 79).

[CV10] Stéphan Clémençon and Nicolas Vayatis. "Overlaying Classifiers: A Practical Approach to Optimal Scoring." In: *Constr. Approx.* 32 (3 Dec. 2010), pp. 619–648. doi: 10.1007/s00365-010-9084-9 (cited on pages 78, 79).

[Coo58] Clyde H Coombs. "On the Use of Inconsistency of Preferences in Psychological Measurement." In: *J. Exp. Psychol.* 55 (1 1958), pp. 1–7. doi: 10/dr3gwh (cited on pages 2, 28).

[CGD92] William S Cooper, Fredric C Gey, and Daniel P Dabney. "Probabilistic Retrieval Based on Staged Logistic Regression." In: *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 1992 (Copenhagen, Denmark, June 21, 1992). New York, New York, USA: ACM Press, 1992. doi: 10.1145/133160.133199 (cited on page 94).

[CMR07a] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. "An Alternative Ranking Problem for Search Engines." In: *Experimental Algorithms*. Ed. by Camil Demetrescu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–22 (cited on page 78).

[CMR07b] Corinna Cortes, Mehryar Mohri, and Ashish Rastogi. "Magnitude-Preserving Ranking Algorithms." In: *Proceedings of the 24th International Conference on Machine Learning*. ICML 2007 (Corvalis, Oregon, USA, June 20, 2007). New York, NY, USA: Association for Computing Machinery, 2007, pp. 169–176. doi: 10.1145/1273496.1273518 (cited on pages 77, 78).

[CZ06] David Cossock and Tong Zhang. "Subset Ranking Using Regression." In: *Learning Theory*. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 605–619. doi: 10.1007/11776420_44 (cited on page 94).

[Cox66] David Rox Cox. "Some Procedures Connected With the Logistic Qualitative Response Curve." In: *Research Papers in Statistics* Festschrift for J. Neyman (1966), pp. 55–71 (cited on page 115).

[CS01] Koby Crammer and Yoram Singer. "Pranking with Ranking." In: *Advances in Neural Information Processing Systems*. NIPS 2001 (Vancouver, BC, Canada, Dec. 3, 2001). Ed. by T Dietterich, S Becker, and Z Ghahramani. Vol. 14. MIT Press, 2001 (cited on pages 78, 81, 95).

[CC05] Dapeng Cui and David Curry. "Prediction in Marketing Using the Support Vector Machine." In: *Marketing Science* 24 (4 2005), pp. 595–615. doi: 10.1287/mksc.1050.0123 (cited on page 134).

[Cyb89] G Cybenko. "Approximation by Superpositions of a Sigmoidal Function." In: *Math. Control Signals Systems* 2 (4 1989), pp. 303–314. doi: 10.1007/BF02551274 (cited on page 63).

[DZ04] H K Dai and X W Zhang. "Improved Linear Expected-Time Algorithms for Computing Maxima." In: *LATIN 2004: Theoretical Informatics*. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 181–192. doi: 10.1007/978-3-540-24698-5_22 (cited on pages 163, 165).

[DM58] Donald Davidson and Jacob Marschak. *Experimental Tests of a Stochastic Decision Theory*. Research rep. Stanford: Applied Mathematics and Statistics Laboratory, July 25, 1958 (cited on page 26).

[Deb+05] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. "Scalable Test Problems for Evolutionary Multiobjective Optimization." In: *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. London: Springer London, 2005, pp. 105–145. doi: 10.1007/1-84628-137-7_6 (cited on pages 173, 178).

[Deb54] Gerard Debreu. "Representation of a Preference Ordering by a Numerical Function." In: *Decision Processes* 3 (1954), pp. 159–165 (cited on pages 19, 21).

[Deb60] Gerard Debreu. "Review of R. D. Luce, Individual Choice Behavior: A Theoretical Analysis." In: *Am. Econ. Rev.* 50 (1 1960), pp. 186–188 (cited on page 29).

[DSM03] Ofer Dekel, Yoram Singer, and Christopher D Manning. "Log-Linear Models for Label Ranking." In: *Advances in Neural Information Processing Systems*. NIPS 2003 (Vancouver, BC, Canada, Dec. 8, 2003). Ed. by S Thrun, L Saul, and B Schölkopf. Vol. 16. MIT Press, 2003 (cited on pages 74, 75).

[Del+23] Michael Dellnitz, Eyke Hüllermeier, Marvin Lücke, Sina Ober-Blöbaum, Christian Offen, Sebastian Peitz, and Karlson Pfannschmidt. "Efficient Time-Stepping for Numerical Integration Using Reinforcement learning." In: *SIAM J. Sci. Comput.* 45 (2 Apr. 30, 2023), A579–A595. doi: 10.1137/21m1412682 (cited on page 223).

[Dem06] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets." In: *J. Mach. Learn. Res.* 7 (1 2006), pp. 1–30 (cited on pages 174, 181).

[Der21] Lihi Dery. *Multi-Label Ranking: Mining Multi-Label and Label Ranking Data*. Version 1. 2021. eprint: 2101.00583 (cs.LG) (cited on pages 74, 76).

[DMP15] Sébastien Destercke, Marie-Hélène Masson, and Michael Poss. "Cautious Label Ranking with Label-Wise Decomposition." In: *Eur. J. Oper. Res.* 246 (3 Nov. 1, 2015), pp. 927–935. doi: 10.1016/j.ejor.2015.05.005 (cited on pages 74, 75).

[DFF23] Daniel Deutsch, George Foster, and Markus Freitag. "Ties Matter: Meta-Evaluating Modern Metrics with Pairwise Accuracy and Tie Calibration." May 23, 2023. eprint: 2305.14324 (cs.CL) (cited on page 91).

[DE73] James Diamond and William Evans. "The Correction for Guessing." In: *Rev. Educ. Res.* 43 (2 1973), pp. 181–191 (cited on page 87).

[DLP23] Oliver Dippel, Alexei Lisitsa, and Bei Peng. "Deep Reinforcement Learning for Continuous Control of Material Thickness." In: *Artificial Intelligence XL*. Lecture notes in computer science. Cham: Springer Nature Switzerland, 2023, pp. 321–334. doi: 10.1007/978-3-031-47994-6_30 (cited on page 152).

[DGI16] Ürün Dogan, Tobias Glasmachers, and Christian Igel. "A Unified View on Multi-class Support Vector Classification." In: *J. Mach. Learn. Res.* 17 (45 2016), pp. 1–32 (cited on page 136).

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *J. Mach. Learn. Res.* 12 (2011), pp. 2121–2159 (cited on page 206).

[Dug+00] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. "Incorporating Second-Order Functional Knowledge for Better Option Pricing." In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS 2000 (Denver, CO, USA, Nov. 27, 2000). Cambridge, MA, USA: MIT Press, 2000, pp. 451–457 (cited on page 137).

[DM41] Ben Dushnik and E W Miller. "Partially Ordered Sets." In: *Amer. J. Math.* 63 (3 1941), pp. 600–610. doi: 10/c6vwmq (cited on pages 167, 168).

[Egg+21] Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. "HPOBench: A Collection of Reproducible Multi-Fidelity Benchmark Problems for HPO." In: *Proceedings of 35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. NeurIPS 2021 (Virtual, Dec. 6, 2021). Sept. 14, 2021 (cited on page 49).

[Ell61] Daniel Ellsberg. "Risk, Ambiguity, and the Savage Axioms." In: *Q. J. Econ.* 75 (4 Nov. 1961), p. 643. doi: 10.2307/1884324 (cited on page 20).

[Eng+19] Martin Engilberge, Louis Chevallier, Patrick Perez, and Matthieu Cord. "SoDeep: A Sorting Deep Net to Learn Ranking Loss Surrogates." In: *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. CVPR 2019 (Long Beach, CA, USA, June 15, 2019). IEEE, June 2019, pp. 10784–10793. doi: 10.1109/cvpr.2019.01105 (cited on pages 78, 81).

[Eve+06] Mark Everingham, Andrew Zisserman, Christopher K I Williams, Luc Van Gool, Moray Allan, Christopher M Bishop, Olivier Chapelle, Navneet Dalal, Thomas Deselaers, Gyuri Dorkó, Stefan Duffner, Jan Eichhorn, Jason D R Farquhar, Mario Fritz, Christophe Garcia, Tom Griffiths, Frederic Jurie, Daniel Keysers, Markus Koskela, Jorma Laaksonen, Diane Larlus, Bastian Leibe, Hongying Meng, Hermann Ney, Bernt Schiele, Cordelia Schmid, Edgar Seemann, John Shawe-Taylor, Amos Storkey, Sandor Szedmak, Bill Triggs, Ilkay Ulusoy, Ville Viitaniemi, and Jianguo Zhang. "The 2005 PASCAL Visual Object Classes Challenge." In: *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 117–176. doi: 10.1007/11736790_8 (cited on page 88).

[EBZ05] Theodoros Evgeniou, Constantinos Boussios, and Giorgos Zacharia. "Generalized Robust Conjoint Estimation." In: *Marketing Science* 24 (3 2005), pp. 415–429 (cited on page 134).

[Exp13] Expedia. *Personalize Expedia Hotel Searches – ICDM 2013*. Sept. 2013. url: https://www.kaggle.com/c/expedia-personalized-sort (visited on 02/03/2023) (cited on pages 138, 143).

[Exp16] Expedia. *Expedia Hotel Recommendations*. June 2016. url: https://www.kaggle.com/c/expedia-hotel-recommendations/overview (visited on 02/03/2023) (cited on page 143).

[FHC17] Mohsen Ahmadi Fahandar, Eyke Hüllermeier, and Inés Couso. "Statistical Inference for Incomplete Ranking Data: The Case of Rank-Dependent Coarsening." In: *Proceedings of the International Conference on Machine Learning*. ICML 2017 (Sydney, Australia, Aug. 2017). Vol. 70. PMLR, 2017, pp. 1078–1087 (cited on page 116).

[Faw06] Tom Fawcett. "An Introduction to ROC Analysis." In: *Pattern Recognit. Lett.* 27 (8 June 2006), pp. 861–874 (cited on page 89).

[Fec60] Gustav Theodor Fechner. *Elemente der Psychophysik*. Vol. 2. Breitkopf u. Härtel, 1860 (cited on pages 1, 33).

[FV86] M A Fligner and J S Verducci. "Distance Based Ranking Models." In: *J. R. Stat. Soc. Series B Stat. Methodol.* 48 (3 1986), pp. 359–369 (cited on page 116).

[FV88] Michael A Fligner and Joseph S Verducci. "Multistage Ranking Models." In: *J. Am. Stat. Assoc.* 83 (403 1988), pp. 892–901 (cited on page 116).

[FH01] Eibe Frank and Mark Hall. "A Simple Approach to Ordinal Classification." In: *Proceedings of the 12th European Conference on Machine Learning*. ECML 2001 (Freiburg, Germany, Sept. 5, 2001). Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 145–156. doi: 10.1007/3-540-44795-4_13 (cited on page 94).

[Fre+03] Yoav Freund, Raj D Iyer, Robert E Schapire, and Yoram Singer. "An Efficient Boosting Algorithm for Combining Preferences." In: *J. Mach. Learn. Res.* 4 (2003), pp. 933–969 (cited on pages 78, 97, 98).

[FS97] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting." In: *J. Comput. Syst. Sci.* 55 (1 1997), pp. 119–139. doi: 10.1006/jcss.1997.1504 (cited on page 97).

[FL06] Drew Fudenberg and David K Levine. "A Dual-Self Model of Impulse Control." In: *Am. Econ. Rev.* 96 (5 Dec. 2006), pp. 1449–1476 (cited on page 36).

[Fuh89] Norbert Fuhr. "Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle." In: *ACM Trans. Inf. Syst.* 7 (3 July 1, 1989), pp. 183–204. doi: 10.1145/65943.65944 (cited on page 93).

[Für02] Johannes Fürnkranz. "Round Robin Classification." In: *JMLR* 2 (2002), pp. 721–747 (cited on page 94).

[FH03] Johannes Fürnkranz and Eyke Hüllermeier. "Pairwise Preference Learning and Ranking." In: *Proceedings of the 14th European Conference on Machine Learning*. ECML 2003 (Cavtat-Dubrovnik, Croatia, Sept. 22, 2003). Springer Berlin Heidelberg, 2003, pp. 145–156. doi: 10.1007/978-3-540-39857-8_15 (cited on pages 74, 75).

[FH10] Johannes Fürnkranz and Eyke Hüllermeier. *Preference Learning*. Springer Science & Business Media, Nov. 19, 2010. 466 pp. (cited on pages 2, 3, 71, 72).

[FH11] Johannes Fürnkranz and Eyke Hüllermeier. "Preference Learning and Ranking by Pairwise Comparison." In: *Preference Learning*. Ed. by Johannes Fürnkranz and Eyke Hüllermeier. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 65–82. doi: 10.1007/978-3-642-14125-6_4 (cited on page 75).

[FHV09] Johannes Fürnkranz, Eyke Hüllermeier, and Stijn Vanderlooy. "Binary Decomposition Methods for Multipartite Ranking." In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. ECML PKDD 2009 (Bled, Slovenia, Sept. 7, 2009). Lecture Notes in Computer Science. Springer, 2009, pp. 359–374 (cited on page 95).

[Gad33] J H Gaddum. *Reports on Biological Standards. 3*. Special report series (Medical Research Council (Great Britain)). London: H.M.S.O., 1933 (cited on page 33).

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: *Proceedings of The 33rd International Conference on Machine Learning*. ICML 2016 (New York City, NY, USA, June 19, 2016). Ed. by Maria Florina Balcan and Kilian Q Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 20, 2016, pp. 1050–1059 (cited on page 53).

[Gar23] Roman Garnett. *Bayesian Optimization*. in preparation. Cambridge, England: Cambridge University Press, Jan. 31, 2023. 358 pp. doi: 10.1017/9781108348973 (cited on pages 49, 52, 54).

[Gei+07] Marc Geilen, Twan Basten, Bart Theelen, and Ralph Otten. "An Algebra of Pareto Points." In: *Fund. Inform.* 78 (1 2007), pp. 35–74 (cited on page 140).

[Gey94] Fredric C Gey. "Inferring Probability of Relevance Using the Method of Logistic Regression." In: *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 1994 (Dublin, Ireland, July 3, 1994). London: Springer London, 1994, pp. 222–231. doi: 10.1007/978-1-4471-2099-5_23 (cited on page 94).

[GS19] Elham Ghanbari and Azadeh Shakery. "ERR.Rank: An Algorithm Based on Learning to Rank for Direct Optimization of Expected Reciprocal Rank." In: *Appl. Intell.* 49 (3 Mar. 2019), pp. 1185–1199. doi: 10.1007/s10489-018-1330-z (cited on page 92).

[Gib73] Allan Gibbard. "Manipulation of Voting Schemes: A General Result." In: *Econometrica* 41 (4 1973), pp. 587–601. doi: 10.2307/1914083 (cited on page 113).

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks." In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. AISTATS 2010 (Chia Laguna Resort, Sardinia, Italy, May 13, 2010). Vol. 9. JMLR Proceedings. JMLR, 2010, pp. 249–256 (cited on page 129).

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. AISTATS 2011 (Fort Lauderdale, USA, Apr. 11, 2011). Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 11, 2011, pp. 315–323 (cited on page 57).

[Gol+92] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. "Using Collaborative Filtering to Weave an Information Tapestry." In: *Commun. ACM* 35 (12 Dec. 1, 1992), pp. 61–70. doi: 10.1145/138859.138867 (cited on page 82).

[Gon+17] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D Lawrence. "Preferential Bayesian Optimization." In: *Proceedings of the 34th International Conference on Machine Learning*. ICML 2017 (Sydney, Australia, Aug. 6, 2017). Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1282–1291 (cited on page 84).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cited on pages 56, 63, 65, 156, 209, 210).

[Gri12] Andreas Griewank. "Who Invented the Reverse Mode of Differentiation?" In: *Documenta Mathematica: Optimization Stories: 21st International Symposium on Mathematical Programming*. Ed. by Martin Grötschel. Held in Berlin, Germany, 19–24 August 2012. Helsinki, Finland: EMS Press, Aug. 2012, pp. 389–400. doi: 10.4171/DMS/6 (cited on page 59).

[GFP22] Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. "The Hypervolume Indicator: Computational Problems and Algorithms." In: *ACM Comput. Surv.* 54 (6 July 31, 2022), pp. 1–42. doi: 10.1145/3453474 (cited on page 141).

[GS08] John Guiver and Edward Snelson. "Learning to Rank with SoftRank and Gaussian Processes." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2008 (Singapore, July 20, 2008). New York, NY, USA: ACM, July 20, 2008. doi: 10.1145/1390334.1390380 (cited on page 101).

[Guo+16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. "A Deep Relevance Matching Model for Ad-Hoc Retrieval." In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. CIKM 2016 (Indianapolis, Indiana, USA, Oct. 24, 2016). New York, NY, USA: ACM, Oct. 24, 2016. doi: 10.1145/2983323.2983769 (cited on page 78).

[GLD60] John Gurland, Ilbok Lee, and Paul A Dahm. "Polychotomous Quantal Response in Biological Assay." In: *Biometrics* 16 (3 1960), pp. 382–398 (cited on pages 33, 115).

[GFS14] Massimo Gurrieri, Philippe Fortemps, and Xavier Siebert. "Alternative Decomposition Techniques for Label Ranking." In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer International Publishing, 2014, pp. 464–474. doi: 10.1007/978-3-319-08855-6_47 (cited on pages 74, 75).

[Gur+12] Massimo Gurrieri, Xavier Siebert, Philippe Fortemps, Salvatore Greco, and Roman Słowiński. "Label Ranking: A New Rule-Based Label Ranking Method." In: *Advances on Computational Intelligence*. Springer Berlin Heidelberg, 2012, pp. 613–623. doi: 10.1007/978-3-642-31709-5_62 (cited on pages 75, 76).

[HBG22] Pedro Hack, Daniel A Braun, and Sebastian Gottwald. "On a Geometrical Notion of Dimension for Partially Ordered Sets." Mar. 30, 2022. eprint: 2203.16272 (math.CO) (cited on page 169).

[HBH21] Björn Haddenhorst, Viktor Bengs, and Eyke Hüllermeier. "Identification of the Generalized Condorcet Winner in Multi-Dueling Bandits." In: *Advances in Neural Information Processing Systems*. NeurIPS 2021 (Virtual, Dec. 6, 2021). Ed. by M Ranzato, A Beygelzimer, Y Dauphin, P S Liang, and J Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021 (cited on page 84).

[HHK20] Björn Haddenhorst, Eyke Hüllermeier, and Martin Kolb. "Generalized Transitivity: A Systematic Comparison of Concepts with an Application to Preferences in the Babington Smith Model." In: *Int. J. Approx. Reason.* 119 (Apr. 1, 2020), pp. 373–407. doi: 10/gn7nwt (cited on page 26).

[Hal90] A Hald. *History of Probability and Statistics and Their Applications Before 1750*. Nashville, TN: John Wiley & Sons, Jan. 24, 1990. 608 pp. doi: 10.1002/0471725161 (cited on page 19).

[HYL17] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs." In: *Advances in Neural Information Processing Systems*. NIPS 2017 (Long Beach, CA, United States, Dec. 4, 2017). Vol. 30. Curran Associates, Inc., 2017 (cited on page 68).

[Hao+22] Hao Hao, Aimin Zhou, Hong Qian, and Hu Zhang. "Expensive Multiobjective Optimization by Relation Learning and Prediction." In: *IEEE Trans. Evol. Comput.* 26 (5 Oct. 2022), pp. 1157–1170. doi: 10.1109/tevc.2022.3152582 (cited on page 152).

[HRZ02] Sariel Har-Peled, Dan Roth, and Dav Zimak. "Constraint Classification for Multiclass Classification and Ranking." In: *Advances in Neural Information Processing Systems*. NIPS 2002 (Vancouver, BC, Canada, Dec. 9, 2002). Ed. by S Becker, S Thrun, and K Obermayer. 15. MIT Press, 2002 (cited on pages 74, 75).

[HK15] F Maxwell Harper and Joseph A Konstan. "The MovieLens Datasets: History and Context." In: *ACM Trans. Interact. Intell. Syst.* 5 (4 Dec. 2015). doi: 10.1145/2827872 (cited on pages 142, 212).

[Hau95] Daniel M Hausman. "The Impossibility of Interpersonal Utility Comparisons." In: *Mind* 104 (415 1995), pp. 473–490. doi: 10.1093/mind/104.415.473 (cited on page 22).

[Haw+12] Guy Hawkins, Scott D Brown, Mark Steyvers, and Eric-Jan Wagenmakers. "Context Effects in Multi-Alternative Decision Making: Empirical Data and a Bayesian Model." In: *Cogn. Sci.* 36 (3 Apr. 1, 2012), pp. 498–516. doi: 10.1111/j.1551-6709.2011.01221.x (cited on pages 32, 147).

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *Proceedings of the International Conference on Computer Vision*. ICCV 2015 (Santiago, Chile, Dec. 7, 2015). IEEE Computer Society, 2015, pp. 1026–1034 (cited on page 129).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. CVPR 2016 (Las Vegas, NV, USA, June 27, 2016). June 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90 (cited on page 65).

[Hea+18] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, Fcharras, Zé Vinícius, Cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene-rex, Kejia (kj) Shi, Justus Schwabedal, Carlosdanielcsantos, Hvass-Labs, Mikhail Pak, SoManyUsernames-Taken, Fred Callaway, Loïc Estève, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. *scikit-optimize/scikit-optimize: v0.5.2*. Version v0.5.2. Mar. 2018. doi: 10.5281/zenodo.1207017 (cited on page 206).

[Hea+92] Shaun Hargreaves Heap, Albert Weale, Bruce Lyons, Martin Hollis, and Robert Sugden. *The Theory of Choice: A Critical Guide*. London, England: Blackwell, Mar. 26, 1992. 416 pp. (cited on page 15).

[HH19] Sascha Henzgen and Eyke Hüllermeier. "Mining Rank Data." In: *ACM Trans. Knowl. Discov. Data* 13 (6 Dec. 31, 2019), pp. 1–36. doi: 10.1145/3363572 (cited on page 85).

[Her20] Steffen Herbold. "Autorank: A Python Package for Automated Ranking of Classifiers." In: *Journal of Open Source Software* 5 (48 2020), p. 2173. doi: 10.21105/joss.02173 (cited on pages 174, 181).

[Her99] R Herbrich. "Support Vector Learning for Ordinal Regression." In: *Proceedings of the 9th International Conference on Artificial Neural Networks*. ICANN 1999 (Edinburgh, UK, Sept. 7, 1999). Vol. 1. IEEE, 1999, pp. 97–102. doi: 10.1049/cp:19991091 (cited on page 94).

[HGO00] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. "Large Margin Rank Boundaries for Ordinal Regression." In: *Advances in Large-Margin Classifiers*. Ed. by Alexander Johannes Smola, Peter L Bartlett, Bernhard Schölkopf, and Dale Schuurmans. The MIT Press, Sept. 29, 2000, pp. 115–132. doi: 10.7551/mitpress/1113.003.0010 (cited on page 97).

[HHG14] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. "Predictive Entropy Search for Efficient Global Optimization of Black-box Functions." In: *Advances in Neural Information Processing Systems 28*. NIPS 2014 (Montréal, Canada, Dec. 8, 2014). 2014 (cited on page 56).

[HM53] I N Herstein and John Milnor. "An Axiomatic Approach to Measurable Utility." In: *Econometrica* 21 (2 1953), pp. 291–297. doi: 10/cztfdh (cited on page 23).

[Hin+12] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. "Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors." July 3, 2012. eprint: 1207.0580 (cs.NE) (cited on page 62).

[Hir55] Toshio Hiraguchi. "On the Dimension of Orders." In: *Sci. Rep. Kanazawa Univ.* 4 (4 Oct. 1, 1955), pp. 1–20 (cited on pages 167, 168).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural Comput.* 9 (8 Nov. 15, 1997), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735 (cited on page 66).

[Hor96] Stephen C Hora. "Aleatory and Epistemic Uncertainty in Probability Elicitation with an Example from Hazardous Waste Management." In: *Reliab. Eng. Syst. Saf.* 54 (2-3 Nov. 1, 1996), pp. 217–223. doi: 10.1016/s0951-8320(96)00077-4 (cited on page 51).

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks are Universal Approximators." In: *Neural Netw.* 2 (5 Jan. 1989), pp. 359–366. doi: 10.1016/0893-6080(89)90020-8 (cited on pages 57, 63, 127).

[HF08] Jim Huang and Brendan J Frey. "Structured Ranking Learning Using Cumulative Distribution Networks." In: *Advances in Neural Information Processing Systems*. NIPS 2008 (Vancouver, BC, Canada, Dec. 8, 2008). Vol. 21. 2008 (cited on page 104).

[HPP82] Joel Huber, John W Payne, and Christopher Puto. "Adding Asymmetrically Dominated Alternatives: Violations of Regularity and the Similarity Hypothesis." In: *J. Consum. Res.* 9 (1 1982), pp. 90–98 (cited on pages 2, 30).

[HP83] Joel Huber and Christopher Puto. "Market Boundaries and Product Choice: Illustrating Attraction and Substitution Effects." In: *J. Consum. Res.* 10 (1 1983), pp. 31–44 (cited on pages 2, 30, 31, 145).

[Hül+08] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. "Label Ranking by Learning Pairwise Preferences." In: *Artif. Intell.* 172 (16 Nov. 1, 2008), pp. 1897–1916. doi: 10.1016/j.artint.2008.08.002 (cited on pages 74, 75).

[HS24] Eyke Hüllermeier and Roman Słowiński. "Preference Learning and Multiple Criteria Decision Aiding: Differences, Commonalities, and Synergies—Part II." In: *4OR* (Jan. 30, 2024), pp. 1–37. doi: 10.1007/s10288-023-00561-5 (cited on pages 3, 72, 84).

[HW21] Eyke Hüllermeier and Willem Waegeman. "Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods." In: *Mach. Learn.* 110 (3 Mar. 8, 2021), pp. 457–506. doi: 10.1007/s10994-021-05946-3 (cited on page 51).

[Hus+21] Denis Huseljic, Bernhard Sick, Marek Herde, and Daniel Kottke. "Separation of Aleatoric and Epistemic Uncertainty in Deterministic Deep Neural Networks." In: *Proceedings of the 25th International Conference on Pattern Recognition*. ICPR 2021 (Milan, Italy, Jan. 10, 2021). IEEE, Jan. 10, 2021, pp. 9172–9179. doi: 10.1109/icpr48806.2021.9412616 (cited on page 53).

[Hut+14] Frank Hutter, Lin Xu, Holger H Hoos, and Kevin Leyton-Brown. "Algorithm Runtime Prediction: Methods & Evaluation." In: *Artif. Intell.* 206 (Jan. 1, 2014), pp. 79–111. doi: 10.1016/j.artint.2013.10.003 (cited on page 52).

[Huy57] Christiaan Huygens. "De Ratiociniis in Ludo Aleae." In: *Exercitationum mathematicarum libri V*. Ed. by Frans van Schooten. Leiden: Elsevir, 1657, pp. 521–534 (cited on page 19).

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *Proceedings of the 32nd International Conference on Machine Learning*. ICML 2015 (Lille, France, July 7, 2015). Vol. 37. JMLR Workshop and Conference Proceedings. JMLR, 2015, pp. 448–456 (cited on pages 128, 175, 206, 209).

[IL00] Sheena S Iyengar and Mark R Lepper. "When Choice is Demotivating: Can One Desire Too Much of a Good Thing?" In: *J. Pers. Soc. Psychol.* 79 (6 Dec. 2000), pp. 995–1006. doi: 10.1037/0022-3514.79.6.995 (cited on page 32).

[JK00] Kalervo Järvelin and Jaana Kekäläinen. "IR Evaluation Methods for Retrieving Highly Relevant Documents." In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2000 (Athens, Greece, July 24, 2000). New York, NY, USA: ACM, July 2000. doi: 10.1145/345508.345545 (cited on page 91).

[JK02] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated Gain-based Evaluation of IR Techniques." In: *ACM Trans. Inf. Syst. Secur.* 20 (4 Oct. 2002), pp. 422–446. doi: 10.1145/582415.582418 (cited on page 91).

[Jas+16] Kalina Jasinska, Krzysztof Dembczyński, Robert Busa-Fekete, Karlson Pfannschmidt, Timo Klerx, and Eyke Hüllermeier. "Extreme F-Measure Maximization Using Sparse Probability Estimates." In: *Proceedings of The 33rd International Conference on Machine Learning*. ICML 2016 (New York City, NY, USA, June 20, 2016). Ed. by Maria Florina Balcan and Kilian Q Weinberger. Vol. 48. Proceedings of machine learning research. PMLR, 2016 (cited on page 223).

[Joa02] Thorsten Joachims. "Optimizing Search Engines Using Clickthrough Data." In: *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD 2002 (Edmonton, Alberta, Canada). New York, NY, USA: Association for Computing Machinery, July 23, 2002, pp. 133–142. doi: 10.1145/775047.775067 (cited on pages 77, 78, 97).

[Joa06] Thorsten Joachims. "Training Linear SVMs in Linear Time." In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD 2006 (Philadelphia, USA, Aug. 20, 2006). 2006, pp. 217–226 (cited on page 175).

[JG03] Eric J Johnson and Daniel Goldstein. "Medicine. Do Defaults Save Lives?" In: *Science* 302 (5649 Nov. 21, 2003), pp. 1338–1339. doi: 10.1126/science.1091721 (cited on page 32).

[JJS11] Cheolkon Jung, L C Jiao, and Yanbo Shen. "Ensemble Ranking SVM for Learning to Rank." In: *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*. MLSP 2011 (Beijing, China, Sept. 18, 2011). 2011, pp. 1–6. doi: 10.1109/MLSP.2011.6064549 (cited on pages 77, 78).

[KKT90] Daniel Kahneman, Jack L Knetsch, and Richard H Thaler. "Experimental Tests of the Endowment Effect and the Coase Theorem." In: *J. Polit. Econ.* 98 (6 Dec. 1990), pp. 1325–1348. doi: 10.1086/261737 (cited on page 32).

[KT79] Daniel Kahneman and Amos Tversky. "Prospect Theory: An Analysis of Decision Under Risk." In: *Econometrica* 47 (2 Mar. 1979), p. 263. doi: 10.2307/1914185 (cited on page 20).

[KRS02] Gil Kalai, Ariel Rubinstein, and Ran Spiegler. "Rationalizing Choice Functions by Multiple Rationales." In: *Econometrica* 70 (6 2002), pp. 2481–2488 (cited on page 36).

[Kam16] Toshihiro Kamishima. *SUSHI Preference Data Sets*. 2016. url: https://www.kamishima.net/sushi/ (visited on 02/05/2023) (cited on page 143).

[KF03] Toshihiro Kamishima and Jun Fujiki. "Clustering Orders." In: *Discovery Science*. Ed. by Gunter Grieser, Yuzuru Tanaka, and Akihiro Yamamoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 194–207 (cited on page 143).

[KKA05] Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho. "Supervised Ordering — An Empirical Survey." In: *Proceedings of the 5th IEEE International Conference on Data Mining*. ICDM 2005 (Houston, TX, USA, Nov. 27, 2005). IEEE Computer Society, 2005, pp. 673–676 (cited on page 94).

[KKA10] Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho. "A Survey and Empirical Comparison of Object Ranking Methods." In: *Preference Learning*. Springer, 2010, pp. 181–201 (cited on pages 60, 214).

[KV05] Paul B Kantor and Ellen M Voorhees. *Report on the TREC-5 Confusion Track*. Research rep. Oct. 24, 2005 (cited on page 92).

[Kau+23] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. "A Survey of Reinforcement Learning from Human Feedback." Dec. 22, 2023. eprint: 2312.14925 (cs.LG) (cited on page 84).

[KH90] Jerry S Kelly and Maxwell Hall. "Impossibility Results with Resoluteness." In: *Econ. Lett.* 34 (1 1990), pp. 15–19 (cited on page 113).

[Ken38] M G Kendall. "A New Measure of Rank Correlation." In: *Biometrika* 30 (1/2 June 1938), p. 81. doi: 10.2307/2332226 (cited on pages 90, 91).

[Ken45] M G Kendall. "The Treatment of Ties in Ranking Problems." In: *Biometrika* 33 (3 Nov. 1945), p. 239. doi: 10.2307/2332303 (cited on page 91).

[Ker+22] Giancarlo Kerg, Sarthak Mittal, David Rolnick, Yoshua Bengio, Blake Richards, and Guillaume Lajoie. "On Neural Architecture Inductive Biases for Relational Tasks." June 9, 2022. eprint: 2206.05056 (cs.NE) (cited on page 60).

[KB15] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cited on page 59).

[Kir08] "Level of Measurement." In: *Encyclopedia of Public Health*. Ed. by Wilhelm Kirch. Dordrecht: Springer Netherlands, 2008, pp. 851–852. doi: 10.1007/978-1-4020-5614-7_1971 (cited on page 112).

[KD09] Armen Der Kiureghian and Ove Ditlevsen. "Aleatory or Epistemic? Does It Matter?" In: *Struct. Saf.* 31 (2 Mar. 1, 2009), pp. 105–112. doi: 10.1016/j.strusafe.2008.06.020 (cited on page 51).

[Kla+17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. "Self-Normalizing Neural Networks." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS 2017 (Long Beach, California, USA, Dec. 4, 2017). Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 972–981 (cited on pages 128, 206).

[Kli+23] Arthur Klippenstein, Christoph Weskamp, Florian Laux, Florian Neuhaus, Karlson Pfannschmidt, Melissa Bülling, and Stephan Kassanke. "A Prototype to Support Business Model Innovation Through Crowdsourcing and Artificial Intelligence." In: *Wirtschaftsinformatik 2023 Proceedings*. WI 2023 (Paderborn, Germany, Sept. 18, 2023). Ed. by Daniel Beverungen, Christiane Lehrer, and Matthias Trier. 47. Sept. 8, 2023 (cited on page 223).

[Koo51] Tjalling C Koopmans. *Analysis of Production as an Efficient Combination of Activities*. Wiley, 1951 (cited on page 157).

[KGd18] Anna Korba, Alexandre Garcia, and Florence d'Alché-Buc. "A Structured Prediction Approach for Label Ranking." In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS 2018 (Montréal, Canada). Red Hook, NY, USA: Curran Associates Inc., Dec. 3, 2018, pp. 9008–9018 (cited on pages 75, 76).

[Koy+15] Oluwasanmi Koyejo, Pradeep Ravikumar, Nagarajan Natarajan, and Inderjit S Dhillon. "Consistent Multilabel Classification." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS 2015 (Montreal, Canada). Cambridge, MA, USA: MIT Press, July 12, 2015, pp. 3321–3329. doi: 10.5555/2969442.2969610 (cited on pages 88, 114, 207).

[Kra67] David H Krantz. "Rational Distance Functions for Multidimensional Scaling." In: *J. Math. Psychol.* 4 (2 June 1967), pp. 226–245. doi: 10/b26g8v (cited on pages 2, 29).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. NIPS 2012 (Lake Tahoe, Nevada, USA, Dec. 3, 2012). Vol. 25. Curran Associates, Inc., 2012 (cited on page 62).

[KLP75] H T Kung, F Luccio, and F P Preparata. "On Finding the Maxima of a Set of Vectors." In: *J. ACM* 22 (4 Oct. 1, 1975), pp. 469–476. doi: 10.1145/321906.321910 (cited on page 165).

[Lai+13] Hanjiang Lai, Yan Pan, Cong Liu, Liang Lin, and Jie Wu. "Sparse Learning-to-Rank via an Efficient Primal-Dual Algorithm." In: *IEEE Transactions on Computers* 62 (6 2013), pp. 1221–1233. doi: 10.1109/TC.2012.62 (cited on page 78).

[Lap+14] Léa Laporte, Rémi Flamary, Stéphane Canu, Sébastien Déjean, and Josiane Mothe. "Nonconvex Regularizations for Feature Selection in Ranking With Sparse SVM." In: *IEEE Transactions on Neural Networks and Learning Systems* 25 (6 2014), pp. 1118–1130. doi: 10.1109/TNNLS.2013.2286696 (cited on page 78).

[Lec89] Yann Lecun. "Generalization and Network Design Strategies." In: *Connectionism in Perspective*. Ed. by R Pfeifer, Z Schreter, F Fogelman, and L Steels. Zurich, Switzerland: Elsevier, 1989 (cited on page 63).

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning." In: *Nature* 521 (7553 May 28, 2015), pp. 436–444. doi: 10.1038/nature14539 (cited on page 56).

[LCB10] Yann LeCun, Corinna Cortes, and Christopher J C Burges. *The MNIST Database of Handwritten Digits*. 2010. url: http://yann.lecun.com/exdb/mnist/ (visited on 06/21/2024) (cited on page 209).

[Lew95] David D Lewis. "Evaluating and Optimizing Autonomous Text Classification Systems." In: *Proceedings of the 18th International Conference on Research and Development in Information Retrieval*. SIGIR 1995 (Seattle, WA, USA, July 9, 1995). ACM Press, 1995, pp. 246–254 (cited on page 207).

[Li+18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the Loss Landscape of Neural Nets." In: *Advances in Neural Information Processing Systems*. NIPS 2018 (Montréal, Canada, Dec. 2, 2018). Ed. by S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett. Vol. 31. Curran Associates, Inc., 2018 (cited on page 137).

[LY20] Miqing Li and Xin Yao. "Quality Evaluation of Solution Sets in Multiobjective Optimisation: A Survey." In: *ACM Comput. Surv.* 52 (2 Mar. 31, 2020), pp. 1–38. doi: 10.1145/3300148 (cited on page 141).

[LBW07] Ping Li, Christopher J C Burges, and Qiang Wu. "McRank: Learning to Rank Using Multiple Classification and Gradient Boosting." In: *Proceedings of the 20th International Conference on Neural Information Processing Systems*. NIPS 2007 (Vancouver, BC, Canada, Dec. 3, 2007). Red Hook, NY, USA: Curran Associates Inc., Dec. 3, 2007, pp. 897–904. doi: 10.5555/2981562.2981675 (cited on page 95).

[Liu+23] Jian Liu, Zhiye Guo, Tianqi Wu, Raj S Roy, Chen Chen, and Jianlin Cheng. "Improving AlphaFold2-Based Protein Tertiary Structure Prediction with MULTICOM in CASP15." In: *Commun. Chem.* 6 (1 Sept. 7, 2023), p. 188. doi: 10.1038/s42004-023-00991-6 (cited on page 109).

[Liu11] Tie-Yan Liu. *Learning to Rank for Information Retrieval*. 1st ed. Springer Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-14267-3 (cited on pages 92, 93, 101).

[Lu+17] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. "The Expressive Power of Neural Networks: A View from the Width." In: *Advances in Neural Information Processing Systems*. NIPS 2017 (Long Beach, California, USA, Dec. 4, 2017). Ed. by I Guyon, U Von Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett. Vol. 30. Curran Associates, Inc., 2017 (cited on page 57).

[Luc58] R Duncan Luce. "A Probabilistic Theory of Utility." In: *Econometrica* 26 (2 Apr. 1958), p. 193. doi: 10.2307/1907587 (cited on page 29).

[Luc59] R Duncan Luce. *Individual Choice Behavior*. Oxford, England: John Wiley, 1959 (cited on pages 1, 9, 19, 24–27, 103, 109).

[LS65] R Duncan Luce and Patrick Suppes. "Preference, Utility, and Subjective Probability." In: *Handbook of Mathematical Psychology*. Ed. by R Duncan Luce, Robert R Bush, and Eugene Galanter. Vol. 3. New York: Wiley, 1965. Chap. 19, pp. 249–410 (cited on pages 20–23).

[LL17] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS 2017 (Long Beach, California, USA, Dec. 7, 2017). Red Hook, NY, USA: Curran Associates Inc., Dec. 4, 2017, pp. 4768–4777. doi: 10.5555/3295222.3295230 (cited on page 152).

[MB13] Matouš Macháček and Ondřej Bojar. "Results of the WMT13 Metrics Shared Task." In: *Proceedings of the 8th Workshop on Statistical Machine Translation*. WMT13 (Sofia, Bulgaria, Aug. 8, 2013). 2013, pp. 45–51 (cited on page 91).

[MB14] Matouš Macháček and Ondřej Bojar. "Results of the WMT14 Metrics Shared Task." In: *Proceedings of the 9th Workshop on Statistical Machine Translation*. WMT 14 (Baltimore, USA, June 26, 2014). 2014, pp. 293–301. doi: 10.3115/v1/W14-3336 (cited on page 91).

[Mac92a] David J C MacKay. "A Practical Bayesian Framework for Backpropagation Networks." In: *Neural Comput.* 4 (3 May 1992), pp. 448–472. doi: 10.1162/neco.1992.4.3.448 (cited on page 53).

[Mac92b] David J C MacKay. "Bayesian Methods for Adaptive Models." PhD thesis. California Institute of Technology, 1992 (cited on page 53).

[MDA15] Dougal Maclaurin, David Duvenaud, and Ryan Adams. "Gradient-Based Hyperparameter Optimization through Reversible Learning." In: *Proceedings of the 32nd International Conference on Machine Learning*. ICML 2015 (Lille, France, July 6, 2015). PMLR, June 1, 2015, pp. 2113–2122 (cited on page 50).

[MMW15] Sebastián Maldonado, Ricardo Montoya, and Richard Weber. "Advanced Conjoint Analysis Using Feature Selection via Support Vector Machines." In: *Eur. J. Oper. Res.* 241 (2 2015), pp. 564–574. doi: 10.1016/j.ejor.2014.09.051 (cited on page 134).

[Mal57] C L Mallows. "Non-Null Ranking Models. I." In: *Biometrika* 44 (1/2 1957), pp. 114–130 (cited on pages 75, 85, 116).

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, July 7, 2008. doi: 10.1017/CBO9780511809071 (cited on page 91).

[Man66] Nathan Mantel. "Models for Complex Contingency Tables and Polychotomous Dosage Response Curves." In: *Biometrics* 22 (1 1966), pp. 83–95 (cited on page 115).

[MM07] Paola Manzini and Marco Mariotti. "Sequentially Rationalizable Choice." In: *Am. Econ. Rev.* 97 (5 2007), pp. 1824–1839 (cited on page 171).

[MT90] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, England: John Wiley & Sons, Aug. 22, 1990. 308 pp. (cited on page 110).

[MN07] Yusufcan Masatlioglu and Daisuke Nakajima. *A Theory of Choice by Elimination*. Research rep. University of Michigan, May 23, 2007, p. 34 (cited on pages 27, 171).

[Mat60] Bertil Matérn. *Spatial Variation*. Springer New York, 1960. 0 pp. doi: 10.1007/978-1-4615-7892-5 (cited on page 53).

[McC89] Donna Katzman McClish. "Analyzing a Portion of the ROC Curve." In: *Med. Decis. Making* 9 (3 1989), pp. 190–195 (cited on page 89).

[MFL20] R Thomas McCoy, Robert Frank, and Tal Linzen. "Does Syntax Need to Grow on Trees? Sources of Hierarchical Inductive Bias in Sequence-to-Sequence Networks." In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 125–140. doi: 10.1162/tacl_a_00304 (cited on page 60).

[McF74] Daniel McFadden. "Conditional Logit Analysis of Qualitative Choice Behavior." In: *Frontiers in Econometrics*. Economic Theory and Mathematical Economics. New York: Academic Press, 1974, pp. 105–142 (cited on pages 2, 33, 34, 134).

[McF76] Daniel McFadden. "The Revealed Preferences of a Government Bureaucracy: Empirical Evidence." In: *Bell J. Econ.* 7 (1 1976), p. 55. doi: 10.2307/3003190 (cited on page 33).

[McF77] Daniel McFadden. *Modelling the Choice of Residential Location*. Research rep. 477. Cowles Foundation for Research in Economics, Yale University, 1977 (cited on page 35).

[McF81] Daniel McFadden. "Econometric Models of Probabilistic Choice." In: *Structural Analysis of Discrete Data with Econometric Applications*. Ed. by C F Manski and Daniel McFadden. Cambridge, MA, USA: MIT Press, 1981, pp. 198–272 (cited on page 34).

[McF01] Daniel McFadden. "Economic Choices." In: *Am. Econ. Rev.* 91 (3 2001), pp. 351–378 (cited on pages 33, 36).

[MTT77] Daniel McFadden, Kenneth Train, and William B Tye. "An Application of Diagnostic Tests for the Independence from Irrelevant Alternatives Property of the Multinomial Logit Model." In: *Forecasting Passenger and Freight Travel*. Transportation Research Board, 1977 (cited on page 112).

[Mea92] A Mead. "Review of the Development of Multidimensional Scaling Methods." In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 41 (1 1992), pp. 27–39 (cited on page 162).

[MB10] Nicolai Meinshausen and Peter Bühlmann. "Stability Selection." In: *J. R. Stat. Soc. Series B Stat. Methodol.* 72 (4 Sept. 1, 2010), pp. 417–473. doi: 10.1111/j.1467-9868.2010.00740.x (cited on page 79).

[Men+13] Aditya Menon, Harikrishna Narasimhan, Shivani Agarwal, and Sanjay Chawla. "On the Statistical Consistency of Algorithms for Binary Classification under Class Imbalance." In: *Proceedings of the 30th International Conference on Machine Learning*. ICML 2013 (Atlanta, Georgia, USA, June 17, 2013). Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, 2013, pp. 603–611 (cited on pages 89, 174).

[MT17] Cristina Mollica and Luca Tardella. "Bayesian Plackett-Luce Mixture Models for Partially Ranked Data." In: *Psychometrika* 82 (2 June 2017), pp. 442–458. doi: 10.1007/s11336-016-9530-0 (cited on page 85).

[Mon13] Pierre Rémond de Montmort. *Essay d'analyse sur les jeux de hazard*. 2nd ed. Paris: Jacques Quillau, 1713 (cited on pages 1, 19).

[Moo+10] Taesup Moon, Alex Smola, Yi Chang, and Zhaohui Zheng. "IntervalRank: Isotonic Regression with Listwise and Pairwise Constraints." In: *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. WSDM 2010 (New York, NY, USA, Feb. 4, 2010). New York, NY, USA: ACM, Feb. 4, 2010. doi: 10.1145/1718487.1718507 (cited on pages 78, 81).

[MD11] Robert Moore and John DeNero. "L1 and L2 Regularization for Multiclass Hinge Loss Models." In: *Proceedings of the Symposium on Machine Learning in Speech and Language Processing*. MLSLP 2011 (Bellevue, WA, USA, June 27, 2011). 2011 (cited on page 136).

[Mos18] Ivan Moscati. *Measuring Utility: From the Marginal Revolution to Behavioral Economics*. Oxford Studies in History of Economics. New York, NY: Oxford University Press, Dec. 20, 2018. 352 pp. doi: 10.1093/oso/9780199372768.001.0001 (cited on pages 20, 22).

[MA17] Alejandro Mottini and Rodrigo Acuna-Agost. "Deep Choice Model Using Pointer Networks for Airline Itinerary Prediction." In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD 2017 (Halifax, NS, Canada, Aug. 13, 2017). ACM, 2017, pp. 1575–1583. doi: 10.1145/3097983.3098005 (cited on page 83).

[Mul09] Stanley A Mulaik. *Foundations of Factor Analysis*. 2nd ed. Chapman and Hall/CRC, 2009. 548 pp. doi: 10.1201/b15851 (cited on page 162).

[Nal04] Ramesh Nallapati. "Discriminative Models for Information Retrieval." In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2004 (Sheffield, UK, July 25, 2004). New York, NY, USA: ACM, July 25, 2004. doi: 10.1145/1008992.1009006 (cited on page 94).

[Nam+14] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. "Large-Scale Multi-Label Text Classification — Revisiting Neural Networks." In: *Machine Learning and Knowledge Discovery in Databases*. ECML PKDD 2014 (Nancy, France, Sept. 15, 2014). Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 437–452. doi: 10.1007/978-3-662-44851-9_28 (cited on page 115).

[NBS22] Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. "A Review on Weight Initialization Strategies for Neural Networks." In: *Artif. Intell. Rev.* 55 (1 Jan. 2022), pp. 291–322. doi: 10.1007/s10462-021-10033-z (cited on page 58).

[NZ22] Samer Nashed and Shlomo Zilberstein. "A Survey of Opponent Modeling in Adversarial Domains." In: *J. Artif. Intell. Res.* 73 (Jan. 14, 2022), pp. 277–327. doi: 10.1613/jair.1.12889 (cited on page 114).

[Nea96] Radford M Neal. *Bayesian Learning for Neural Networks*. 1996th ed. Lecture notes in statistics. New York, NY: Springer, Aug. 9, 1996. 204 pp. doi: 10.1007/978-1-4612-0745-0 (cited on page 53).

[Nes83] Yurii Nesterov. "A Method of Solving a Convex Programming Problem with Convergence Rate $\mathcal{O}(1/k^2)$." In: *Soviet Mathematics Doklady*. Vol. 27. 1983, pp. 372–376 (cited on pages 59, 176, 206).

[Nog+19] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. "Multi-Stage Document Ranking with BERT." Oct. 31, 2019. eprint: 1910.14424 (cs.IR) (cited on page 101).

[Ore62] O Ore. *Theory of Graphs*. Vol. 38. Colloquium Publications. Providence, Rhode Island: American Mathematical Society, Dec. 31, 1962 (cited on page 167).

[Orh09] A Yeşim Orhun. "Optimal Product Line Design When Consumers Exhibit Choice Set-Dependent Preferences." In: *Marketing Science* 28 (5 2009), pp. 868–886 (cited on pages 37, 38).

[OO14] Takayuki Osogami and Makoto Otsuka. "Restricted Boltzmann Machines Modeling Human Choice." In: *Advances in Neural Information Processing Systems*. NIPS 2014 (Montreal, Quebec, Canada, Dec. 8, 2014). Ed. by Z Ghahramani, M Welling, C Cortes, N Lawrence, and K Q Weinberger. Vol. 27. Curran Associates, Inc., 2014, pp. 73–81 (cited on page 83).

[OO16] Makoto Otsuka and Takayuki Osogami. "A Deep Choice Model." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI 2016 (Phoenix, Arizona USA, Feb. 2016). 30. AAAI Press, 2016. doi: 10.1609/aaai.v30i1.10059 (cited on page 83).

[Ouy+22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. "Training Language Models to Follow Instructions with Human Feedback." In: *Advances in Neural Information Processing Systems*. NeurIPS 2022 (New Orleans, United States, 2022). Ed. by S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, and A Oh. Vol. 35. Dec. 6, 2022 (cited on pages 3, 73, 83, 85).

[OS21] Ali I Ozkes and M Remzi Sanver. "Anonymous, Neutral, and Resolute Social Choice Revisited." In: *Soc. Choice Welfare* 57 (1 July 1, 2021), pp. 97–113. doi: 10.1007/s00355-020-01308-5 (cited on page 113).

[Pah+10] Tapio Pahikkala, Antti Airola, Pekka Naula, and Tapio Salakoski. "Greedy RankRLS: A Linear Time Algorithm for Learning Sparse Ranking Models." In: *Proceedings of SIGIR Workshop on Feature Generation and Selection for Information Retrieval*. SIGIR 2010 (Geneva, Switzerland, July 23, 2010). Ed. by Evgeniy Gabrilovich, Alex Smola, and Naftali Tishby. New York, NY: Association of Computing Machinery, 2010, pp. 11–18 (cited on page 78).

[Pah+07] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jorma Boberg, and Tapio Salakoski. "Learning to Rank with Pairwise Regularized Least-Squares." In: *Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*. SIGIR 2007 (Amsterdam, Netherlands, July 27, 2007). Ed. by Thorsten Joachims, Hang Li, Tie-Yan Liu, and Chengxiang Zhai. 2007 (cited on page 78).

[Pah+09] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jouni Järvinen, and Jorma Boberg. "An Efficient Algorithm for Learning to Rank from Preference Graphs." In: *Mach. Learn.* 75 (1 Apr. 1, 2009), pp. 129–165. doi: 10.1007/s10994-008-5097-z (cited on page 78).

[Pan+17] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. "DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval." In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM 2017 (Singapore, Nov. 6, 2017). Ed. by Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J Shane Culpepper, Eric Lo, Joyce C Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S Tseng, and Chenliang Li. ACM, 2017, pp. 257–266. doi: 10.1145/3132847.3132914 (cited on page 78).

[Par06] Vilfredo Pareto. *Manuale di economia politica, con una introduzione ulla scienza sociale*. Milan, Italy, 1906 (cited on pages 1, 9, 19, 20, 22, 157).

[PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the Difficulty of Training Recurrent Neural Networks." In: *Proceedings of the 30th International Conference on Machine Learning*. ICML 2013 (Atlanta, Georgia, USA, June 17, 2013). Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research. Atlanta, Georgia, USA: PMLR, 2013, pp. 1310–1318 (cited on page 66).

[Pas+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems*. NeurIPS 2019 (Vancouver, Canada, Dec. 8, 2019). Ed. by H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, and R Garnett. Vol. 32. Curran Associates, Inc., 2019 (cited on page 56).

[Ped+11] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. "Scikit-learn: Machine Learning in Python." In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830 (cited on page 207).

[Pei+19] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. "Personalized Re-Ranking for Recommendation." In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys 2019 (Copenhagen, Denmark, Sept. 16, 2019). New York, NY, USA: ACM, Sept. 10, 2019. doi: 10.1145/3298689.3347000 (cited on page 78).

[Pet12] Jonathan C Pettibone. "Testing the Effect of Time Pressure on Asymmetric Dominance and Compromise Decoys in Choice." In: *Judgm. Decis. Mak.* 7 (4 July 2012), pp. 513–521. doi: 10.1017/s1930297500002849 (cited on pages 32, 122).

[Pfa+22] Karlson Pfannschmidt, Pritha Gupta, Björn Haddenhorst, and Eyke Hüllermeier. "Learning Context-Dependent Choice Functions." In: *Int. J. Approx. Reason.* 140 (2022), pp. 116–155. doi: 10/gn9kh3 (cited on pages 83, 223).

[PGH18] Karlson Pfannschmidt, Pritha Gupta, and Eyke Hüllermeier. "Deep Architectures for Learning Context-Dependent Ranking Functions." Mar. 15, 2018. eprint: 1803.05796 (stat.ML) (cited on page 223).

[PHA22] Karlson Pfannschmidt, Evan Hubinger, and Gupta Aksh. *kiudee/bayes-skopt: bayes-skopt 0.10.8 (02 April 2022)*. Comp. software. Version 0.10.8. Apr. 2, 2022. doi: 10.5281/zenodo.6408435 (cited on page 175).

[PH20] Karlson Pfannschmidt and Eyke Hüllermeier. "Learning Choice Functions via Pareto-Embeddings." In: *KI 2020: Advances in Artificial Intelligence*. KI 2020 (2020). Ed. by Ute Schmid, Franziska Klügl, and Diedrich Wolter. Lecture Notes in Computer Science. Springer International Publishing, 2020 (cited on pages 83, 163, 223).

[PH22] Karlson Pfannschmidt and Eyke Hüllermeier. "A Characterization of Choice Functions Representable by Pareto-Embedding of Alternatives." In: *DA2PL 2022 Proceedings*. DA2PL 2022 (Compiègne, France, Nov. 17, 2022). Nov. 17, 2022 (cited on page 223).

[Pfa+16] Karlson Pfannschmidt, Eyke Hüllermeier, Susanne Held, and Reto Neiger. "Evaluating Tests in Medical Diagnosis: Combining Machine Learning with Game-Theoretical Concepts." In: IPMU 2016 (Eindhoven, The Netherlands, June 20, 2016). Ed. by Joao Paulo Carvalho, Marie-Jeanne Lesot, Uzay Kaymak, Susana Vieira, Bernadette Bouchon-Meunier, and Ronald R Yager. Springer International Publishing, June 20, 2016 (cited on page 223).

[Phe+15] Andrew S Phelps, David M Naeger, Jesse L Courtier, Jack W Lambert, Peter A Marcovici, Javier E Villanueva-Meyer, and John D MacKenzie. "Pairwise Comparison Versus Likert Scale for Biomedical Image Assessment." In: *AJR Am. J. Roentgenol.* 204 (1 Jan. 2015), pp. 8–14. doi: 10.2214/AJR.14.13022 (cited on page 71).

[Pis05] David Pisinger. "Where Are the Hard Knapsack Problems?" In: *Comput. Oper. Res.* 32 (9 Sept. 1, 2005), pp. 2271–2284. doi: 10.1016/j.cor.2004.03.002 (cited on page 110).

[Pla75] R L Plackett. "The Analysis of Permutations." In: *J. R. Stat. Soc. Ser. C Appl. Stat.* 24 (2 1975), pp. 193–202. doi: 10.2307/2346567 (cited on pages 75, 103, 116).

[PLE71] A Pnueli, A Lempel, and S Even. "Transitive Orientation of Graphs and Identification of Permutation Graphs." In: *Canad. J. Math.* 23 (1 1971), pp. 160–175. doi: 10/cj5mw8 (cited on page 168).

[Pow11] David Martin Powers. "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation." In: *Journal of Machine Learning Technologies* 2 (1 2011), pp. 37–63 (cited on page 89).

[Qi16] Shaofang Qi. "A Characterization of the N-Agent Pareto Dominance Relation." In: *Soc. Choice Welfare* 46 (3 Mar. 1, 2016), pp. 695–706. doi: 10.1007/s00355-015-0934-z (cited on page 169).

[QL13] Tao Qin and Tie-Yan Liu. "Introducing LETOR 4.0 Datasets." June 9, 2013. eprint: 1306.2597 (cs.IR) (cited on pages 138, 142, 143, 214, 215).

[QLL10] Tao Qin, Tie-Yan Liu, and Hang Li. "A General Approximation Framework for Direct Optimization of Information Retrieval Measures." In: *Inf. Retr. Boston.* 13 (4 Aug. 2010), pp. 375–397. doi: 10.1007/s10791-009-9124-x (cited on page 102).

[Qin+08] Tao Qin, Tie-Yan Liu, Xu-Dong Zhang, De-Sheng Wang, Wen-Ying Xiong, and Hang Li. "Learning to Rank Relational Objects and its Application to Web Search." In: *Proceedings of the 17th International Conference on World Wide Web*. WWW 2008 (Beijing China, Apr. 21, 2008). New York, NY, USA: ACM, Apr. 21, 2008. doi: 10.1145/1367497.1367553 (cited on pages 77, 78).

[Qin+07] Tao Qin, Xu-Dong Zhang, De-Sheng Wang, Tie-Yan Liu, Wei Lai, and Hang Li. "Ranking with Multiple Hyperplanes." In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2007 (Amsterdam, The Netherlands, July 23, 2007). New York, NY, USA: ACM, 2007, pp. 279–286. doi: 10.1145/1277741.1277791 (cited on pages 77, 78).

[Raf+23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model." May 29, 2023. eprint: 2305.18290 (cs.LG) (cited on pages 9, 73, 83, 85).

[Rak04] Alain Rakotomamonjy. "Optimizing Area Under ROC Curve with SVMs." In: *Proceedings of the 1st International Workshop of ROC Analysis in Artificial Intelligence*. ROCAI 2004 (Valencia, Spain, Aug. 22, 2004). Ed. by José Hernández-Orallo, César Ferri, Nicolas Lachiche, and Peter A Flach. 2004, pp. 71–80 (cited on page 78).

[Rak12] Alain Rakotomamonjy. "Sparse Support Vector Infinite Push." In: *Proceedings of the 29th International Conference on Machine Learning*. ICML 2012 (Edinburgh, Scotland, UK, June 26, 2012). 2012 (cited on pages 78, 79).

[RW06] Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006. 248 pp. (cited on pages 52, 54).

[REB71] Paul R Rassam, Raymond H Ellis, and John C Bennett. "The n-Dimensional Logit Model: Development and Application." In: *Choice of Travel Mode and Considerations in Travel Forecasting*. Vol. 369. Highway Research Record. Washington, D.C., USA: Highway Research Board, 1971, pp. 135–147 (cited on page 33).

[RSS87] Srinivasan Ratneshwar, Allan D Shocker, and David W Stewart. "Toward Understanding the Attraction Effect: The Implications of Product Stimulus Meaningfulness and Familiarity." In: *J. Consum. Res.* 13 (4 1987), pp. 520–533 (cited on pages 2, 31).

[RSP17] Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos. "Equivariance Through Parameter-Sharing." In: *Proceedings of the 34th International Conference on Machine Learning*. ICML 2017 (Sydney, Australia, Aug. 6, 2017). PMLR, July 17, 2017, pp. 2892–2901 (cited on pages 67, 125).

[Reb+15] Cláudio Rebelo de Sá, Carla Rebelo, Carlos Soares, and Arno Knobbe. "Distance-Based Decision Tree Algorithms for Label Ranking." In: *Progress in Artificial Intelligence*. Springer International Publishing, 2015, pp. 525–534. doi: 10.1007/978-3-319-23485-4_52 (cited on page 75).

[RS05] Jason D M Rennie and Nathan Srebro. "Loss Functions for Preference Levels: Regression with Discrete Ordered Labels." In: *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*. IJCAI 2005 (Edinburgh, Scotland, July 2005). Ed. by Ronen Brafman and Ulrich Junker. Menlo Park, CA: AAAI Press, July 2005 (cited on page 95).

[Rib+12] Geraldina Ribeiro, Wouter Duivesteijn, Carlos Soares, and Arno Knobbe. "Multilayer Perceptron for Label Ranking." In: *Proceedings of the 22nd International Conference on Artificial Neural Networks*. ICANN 2012 (Lausanne, Switzerland, Sept. 11, 2012). Springer Berlin Heidelberg, 2012, pp. 25–32. doi: 10.1007/978-3-642-33266-1_4 (cited on pages 75, 76).

[Rig+11] Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. "SortNet: Learning to Rank by a Neural Preference Function." In: *IEEE Trans. Neural Netw.* 22 (9 2011), pp. 1368–1380 (cited on pages 99, 128).

[Rob32] Lionel Robbins. *An Essay on the Nature and Significance of Economic Science*. 1st ed. London: Macmillan, 1932 (cited on page 22).

[Rob+95] Stephen Robertson, S Walker, S Jones, M M Hancock-Beaulieu, and M Gatford. "Okapi at TREC-3." In: *Overview of the Third Text REtrieval Conference*. TREC-3 (Gaithersburg, Maryland, USA, Nov. 2, 1994). Gaithersburg, MD: NIST, Jan. 1995, pp. 109–126 (cited on page 101).

[ROS20] Nir Rosenfeld, Kojin Oshiba, and Yaron Singer. "Predicting Choice with Set-Dependent Aggregation." In: *Proceedings of the International Conference on Machine Learning*. ICML 2020 (Virtual, July 13, 2020). Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8220–8229 (cited on pages 41, 83, 125, 134, 189).

[Rud09] Cynthia Rudin. "The P-Norm Push: A Simple Convex Ranking Algorithm That Concentrates at the Top of the List." In: *J. Mach. Learn. Res.* 10 (Dec. 2009), pp. 2233–2271 (cited on page 79).

[RS09] Cynthia Rudin and Robert E Schapire. "Margin-Based Ranking and an Equivalence Between AdaBoost and RankBoost." In: *J. Mach. Learn. Res.* 2193 (Oct. 2009), p. 2232 (cited on page 78).

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning Representations by Back-Propagating Errors." In: *Nature* 323 (6088 Oct. 1986), pp. 533–536. doi: 10.1038/323533a0 (cited on pages 59, 99).

[RN20] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th ed. Pearson, 2020 (cited on page 20).

[Sá+18] Cláudio Rebelo de Sá, Wouter Duivesteijn, Paulo Azevedo, Alípio Mário Jorge, Carlos Soares, and Arno Knobbe. "Discovering a Taste for the Unusual: Exceptional Models for Preference Mining." In: *Mach. Learn.* 107 (11 Nov. 9, 2018), pp. 1775–1807. doi: 10.1007/s10994-018-5743-z (cited on page 85).

[Sá+11] Cláudio Rebelo de Sá, Carlos Soares, Alípio Mário Jorge, Paulo Azevedo, and Joaquim Costa. "Mining Association Rules for Label Ranking." In: *Advances in Knowledge Discovery and Data Mining*. PAKDD 2021 (Virtual, May 11, 2011). Springer Berlin Heidelberg, 2011, pp. 432–443. doi: 10.1007/978-3-642-20847-8_36 (cited on pages 75, 76).

[SWF16] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. "Probabilistic Programming in Python using PyMC3." In: *PeerJ Comput. Sci.* 2 (Apr. 2016), e55 (cited on page 207).

[Sam38] P A Samuelson. "A Note on the Pure Theory of Consumer's Behaviour." In: *Economica* 5 (17 1938), pp. 61–71 (cited on page 165).

[Sat75] Mark Allen Satterthwaite. "Strategy-Proofness and Arrow's Conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions." In: *J. Econ. Theory* 10 (2 Apr. 1975), pp. 187–217. doi: 10.1016/0022-0531(75)90050-2 (cited on page 113).

[SH18] Dirk Schäfer and Eyke Hüllermeier. "Dyad Ranking Using Plackett–Luce Models Based on Joint Feature Representations." In: *Mach. Learn.* 107 (5 May 10, 2018), pp. 903–941. doi: 10.1007/s10994-017-5694-9 (cited on page 81).

[Sch02] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. 2003rd ed. Algorithms and Combinatorics. Berlin, Germany: Springer, Dec. 10, 2002. 1879 pp. (cited on page 109).

[Scu10] D Sculley. "Combined Regression and Ranking." In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. SIGKDD 2010 (Washington DC USA, July 25, 2010). New York, NY, USA: ACM, July 25, 2010. doi: 10.1145/1835804.1835928 (cited on pages 78, 81).

[SPU19] Arjun Seshadri, Alex Peysakhovich, and Johan Ugander. "Discovering Context Effects from Raw Choice Data." In: *Proceedings of the 36th International Conference on Machine Learning*. ICML 2019 (Long Beach, California, USA, June 9, 2019). Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 9, 2019, pp. 5660–5669 (cited on pages 118, 123).

[SL02] Amnon Shashua and Anat Levin. "Ranking with Large Margin Principle: Two Approaches." In: *Proceedings of the 15th International Conference on Neural Information Processing Systems*. NIPS 2002 (Vancouver, BC, Canada, Dec. 9, 2002). Cambridge, MA, USA: MIT Press, Jan. 1, 2002, pp. 961–968. doi: 10.5555/2968618.2968738 (cited on page 95).

[Sim57] Herbert Alexander Simon. *Models of Man: Social and Rational; Mathematical Essays on Rational Human Behavior in Society Setting*. Wiley, 1957 (cited on page 20).

[Sim89] Itamar Simonson. "Choice Based on Reasons: The Case of Attraction and Compromise Effects." In: *J. Consum. Res.* 16 (2 1989), pp. 158–174 (cited on pages 2, 31).

[ST92] Itamar Simonson and Amos Tversky. "Choice in Context: Tradeoff Contrast and Extremeness Aversion." In: *J. Mark. Res.* 29 (3 1992), pp. 281–295. doi: 10.2307/3172740 (cited on page 31).

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015 (cited on page 64).

[Smi07] Gene Smith. *Tagging: People-Powered Metadata for the Social Web*. New Riders, 2007 (cited on page 212).

[Son+16] Yang Song, Alexander G Schwing, Richard S Zemel, and Raquel Urtasun. "Training Deep Neural Networks via Direct Loss Minimization." In: *Proceedings of the 33rd International Conference on Machine Learning*. ICML 2016 (New York, NY, USA, June 19, 2016). Vol. 48. JMLR.org, June 19, 2016, pp. 2169–2177. doi: 10.5555/3045390.3045619 (cited on pages 78, 80).

[Spe04] C Spearman. "The Proof and Measurement of Association Between Two Things." In: *Am. J. Psychol.* 15 (1 Jan. 1904), p. 72. doi: 10.2307/1412159 (cited on pages 90, 91).

[Spr+16] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. "Bayesian Optimization with Robust Bayesian Neural Networks." In: *Advances in Neural Information Processing Systems*. NIPS 2016 (Barcelona, Spain, Dec. 5, 2016). Ed. by D Lee, M Sugiyama, U Luxburg, I Guyon, and R Garnett. Vol. 29. Curran Associates, Inc., 2016 (cited on page 53).

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *J. Mach. Learn. Res.* 15 (56 2014), pp. 1929–1958 (cited on page 62).

[Ste20] Stefan Steinerberger. "Dynamically Defined Sequences with Small Discrepancy." In: *Mon. Hefte Math.* 191 (3 Mar. 1, 2020), pp. 639–655. doi: 10.1007/s00605-019-01360-z (cited on page 175).

[Sti50] George J Stigler. "The Development of Utility Theory. II." In: *J. Polit. Econ.* 58 (5 Oct. 1950), pp. 373–396. doi: 10/cfsgp8 (cited on page 22).

[Sto69] Peter R Stopher. "A Probability Model of Travel Mode Choice for the Work Journey." In: *Travel Factors and Travel Models*. Highway Research Record 274. Washington, D.C., USA: Highway Research Board, 1969, pp. 57–65 (cited on page 33).

[Stu53] A Stuart. "The Estimation and Comparison of Strengths of Association in Contingency Tables." In: *Biometrika* 40 (1/2 June 1953), p. 105. doi: 10.2307/2333101 (cited on page 91).

[TL19] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." In: *Proceedings of the 36th International Conference on Machine Learning*. ICML 2019 (Long Beach, CA, USA, June 9, 2019). Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6105–6114 (cited on page 69).

[TN18] Luke Taylor and Geoff Nitschke. "Improving Deep Learning with Generic Data Augmentation." In: *Proceedings of the IEEE Symposium Series on Computational Intelligence*. SSCI 2018 (Bengaluru, India, Nov. 18, 2018). Nov. 2018, pp. 1542–1547. doi: 10.1109/SSCI.2018.8628742 (cited on page 62).

[Tay+08] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. "SoftRank: Optimizing Non-Smooth Rank Metrics." In: *Proceedings of the International Conference on Web Search and Web Data Mining*. WSDM 2008 (Palo Alto, California, USA, Feb. 11, 2008). New York, New York, USA: ACM Press, 2008. doi: 10.1145/1341531.1341544 (cited on page 101).

[Tes88] Gerald Tesauro. "Connectionist Learning of Expert Preferences by Comparison Training." In: *Advances in Neural Information Processing Systems*. NIPS 1988 (Denver, Colorado). Ed. by David S Touretzky. 1. Morgan-Kaufmann, 1988, pp. 99–106 (cited on pages 78, 80, 98, 134, 175).

[The69] Henri Theil. "A Multinomial Extension of the Linear Logit Model." In: *Int. Econ. Rev.* 10 (3 1969), pp. 251–259 (cited on pages 33, 115).

[The70] Henri Theil. "On the Estimation of Relationships Involving Qualitative Variables." In: *Am. J. Sociol.* 76 (1 July 1970), pp. 103–154. doi: 10.1086/224909 (cited on page 33).

[TC21] Veronika Thost and Jie Chen. "Directed Acyclic Graph Neural Networks." In: *Proceedings of the International Conference on Learning Representations*. ICLR 2021 (Virtual, May 3, 2021). 2021 (cited on page 60).

[Thu27] L L Thurstone. "A Law of Comparative Judgment." In: *Psychol. Rev.* 34 (4 1927), pp. 273–286 (cited on pages 27, 33, 114).

[Tia+11] Yingjie Tian, Yong Shi, Xiaojun Chen, and Wenjing Chen. "AUC Maximizing Support Vector Machines with Feature Selection." In: *Procedia Computer Science*. ICCS 2011 (Singapore, June 1, 2011). Vol. 4. Proceedings of the International Conference on Computational Science, ICCS 2011. 2011, pp. 1691–1698. doi: 10.1016/j.procs.2011.04.183 (cited on page 78).

[Tia+21] Louis C Tiao, Aaron Klein, Matthias W Seeger, Edwin V Bonilla, Cedric Archambeau, and Fabio Ramos. "BORE: Bayesian Optimization by Density-Ratio Estimation." In: *Proceedings of the 38th International Conference on Machine Learning*. ICML 2021 (Virtual, July 18, 2021). PMLR, July 1, 2021 (cited on page 52).

[Tra09] Kenneth E Train. *Discrete Choice Methods with Simulation*. 2nd ed. New York: Cambridge University Press, Sept. 2009 (cited on pages 34, 134).

[TRE07] TREC. *TREC 2007 Million Query Track*. 2007. url: https://trec.nist.gov/data/million.query07.html (visited on 06/21/2024) (cited on page 142).

[TRE08] TREC. *TREC 2008 Million Query Track*. 2008. url: https://trec.nist.gov/data/million.query08.html (visited on 06/21/2024) (cited on page 142).

[Tru22] Jennifer S Trueblood. "Theories of Context Effects in Multialternative, Multiattribute Choice." In: *Curr. Dir. Psychol. Sci.* 31 (5 Oct. 16, 2022), pp. 428–435. doi: 10.1177/09637214221109587 (cited on page 122).

[Tsa+07] Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. "FRank: A Ranking Method with Fidelity Loss." In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2007 (Amsterdam, The Netherlands, July 23, 2007). New York, NY, USA: ACM, July 23, 2007. doi: 10.1145/1277741.1277808 (cited on pages 78, 79).

[TK81] A Tversky and D Kahneman. "The Framing of Decisions and the Psychology of Choice." In: *Science* 211 (4481 Jan. 30, 1981), pp. 453–458. doi: 10.1126/science.7455683 (cited on page 32).

[Tve72] Amos Tversky. "Elimination by Aspects: A Theory of Choice." In: *Psychol. Rev.* 79 (4 1972), p. 281 (cited on page 134).

[TE69] Amos Tversky and J Edward Russo. "Substitutability and Similarity in Binary Choices." In: *J. Math. Psychol.* 6 (1 Feb. 1969), pp. 1–12. doi: 10/bsb4nb (cited on pages 2, 29, 30).

[TK91] Amos Tversky and Daniel Kahneman. "Loss Aversion in Riskless Choice: A Reference-Dependent Model." In: *Q. J. Econ.* 106 (4 Nov. 1991), pp. 1039–1061. doi: 10.2307/2937956 (cited on page 37).

[VPB03] Mark Van der Laan, Katherine Pollard, and Jennifer Bryan. "A New Partitioning Around Medoids Algorithm." In: *J. Stat. Comput. Simul.* 73 (8 2003), pp. 575–584 (cited on page 139).

[VC71] V N Vapnik and A Ya Chervonenkis. "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities." In: *Theory Probab. Appl.* 16 (2 Jan. 1971), pp. 264–280. doi: 10.1137/1116025 (cited on page 44).

[Vel+08] Adriano A Veloso, Humberto M Almeida, Marcos A Gonçalves, and Wagner Meira Jr. "Learning to Rank at Query-Time Using Association Rules." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2008 (Singapore, July 20, 2008). New York, NY, USA: ACM, July 20, 2008. doi: 10.1145/1390334.1390381 (cited on page 95).

[VG10] Shankar Vembu and Thomas Gärtner. "Label Ranking Algorithms: A Survey." In: *Preference Learning*. 2010, pp. 45–64 (cited on pages 60, 74).

[Ver38] Pierre-François Verhulst. "Notice sur la loi que la population suit dans son accroissement." In: *Correspondance mathématique et Physique* 10 (1838). Ed. by A Quetelet, pp. 113–120 (cited on page 33).

[VSR11] Jesse Vig, Shilad Sen, and John Riedl. "Navigating the Tag Genome." In: *Proceedings of the 16th International Conference on Intelligent User Interfaces*. IUI 2011 (Palo Alto, CA, USA, Feb. 13, 2011). 2011, pp. 93–102. doi: 10.1145/1943403.1943418 (cited on page 142).

[VSR12] Jesse Vig, Shilad Sen, and John Riedl. "The Tag Genome: Encoding Community Knowledge to Support Novel Interaction." In: *ACM Trans. Interact. Intell. Syst.* 2 (3 2012), p. 13 (cited on pages 138, 142, 212, 213).

[Vit+18] Valeria Vitelli, Øystein Sørensen, Marta Crispino, Arnoldo Frigessi, and Elja Arjas. "Probabilistic Preference Learning with the Mallows Rank Model." In: *J. Mach. Learn. Res.* 18 (158 2018), pp. 1–49 (cited on page 85).

[VC20] Robin Vogel and Stéphan Clémençon. "A Multiclass Classification Approach to Label Ranking." In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. AISTATS 2020 (Palermo, Italy, Aug. 2020). Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1421–1430 (cited on page 74).

[VZ09] Maksims N Volkovs and Richard S Zemel. "BoltzRank: Learning to Maximize Expected Ranking Gain." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML 2009 (Montréal, Canada, June 14, 2009). New York, NY, USA: ACM, June 14, 2009. doi: 10.1145/1553374.1553513 (cited on page 104).

[VM47] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947 (cited on pages 1, 9, 19, 20, 22–24, 27, 109).

[Vov97] Peter Vovsha. "Application of Cross-Nested Logit Model to Mode Choice in Tel Aviv, Israel, Metropolitan Area." In: *Transp. Res. Rec.* 1607 (1 Jan. 1, 1997), pp. 6–15. doi: 10.3141/1607-02 (cited on page 35).

[Wae+14] Willem Waegeman, Krzysztof Dembczynski, Arkadiusz Jachnik, Weiwei Cheng, and Eyke Hüllermeier. "On the Bayes-Optimality of F-Measure Maximizers." In: *J. Mach. Learn. Res.* 15 (1 2014), pp. 3333–3388 (cited on pages 87, 88, 207).

[WP97] Eric Walter and Luc Pronzato. *Identification of Parametric Models: From Experimental Data*. Springer, Jan. 14, 1997. 440 pp. (cited on page 119).

[Wan+13] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. "A Theoretical Analysis of NDCG Type Ranking Measures." In: *Proceedings of the Conference on Learning Theory*. COLT 2013 (Princeton, NJ, USA, June 12, 2013). PMLR, June 13, 2013, pp. 25–54 (cited on page 91).

[WJ17] Zi Wang and Stefanie Jegelka. "Max-Value Entropy Search for Efficient Bayesian Optimization." In: *Proceedings of the 34th International Conference on Machine Learning*. ICML 2017 (Sydney, Australia, Aug. 6, 2017). Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3627–3635 (cited on page 175).

[WHV22] Douglas H Wedell, William M Hayes, and Mansi Verma. "Context Effects on Choice Under Cognitive Load." In: *Psychon. Bull. Rev.* 29 (5 Oct. 26, 2022), pp. 1986–1996. doi: 10.3758/s13423-022-02113-0 (cited on page 32).

[WK01] Chieh-Hua Wen and Frank S Koppelman. "The Generalized Nested Logit Model." In: *Trans. Res. Part B: Methodol.* 35 (7 2001), pp. 627–641 (cited on pages 35, 134).

[Wer19] Tino Werner. "Gradient-Free Gradient Boosting." PhD thesis. Oldenburg: Carl von Ossietzky University of Oldenburg, 2019 (cited on pages 78, 79).

[Wer22] Tino Werner. "A Review on Instance Ranking Problems in Statistical Learning." In: *Mach. Learn.* 111 (2 Feb. 1, 2022), pp. 415–463. doi: 10.1007/s10994-021-06122-3 (cited on pages 76, 77).

[Wil77] H C W L Williams. "On the Formation of Travel Demand Models and Economic Evaluation Measures of User Benefit." In: *Environment and Planning A: Economy and Space* 9 (3 1977), pp. 285–344 (cited on page 134).

[Wil+16] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. "Deep Kernel Learning." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. AISTATS 2016 (Cadiz, Spain, May 9, 2016). PMLR, May 2, 2016, pp. 370–378 (cited on page 53).

[Wir+17] Christian Wirth, R Akrour, G Neumann, and Johannes Fürnkranz. "A Survey of Preference-Based Reinforcement Learning Methods." In: *J. Mach. Learn. Res.* (2017) (cited on page 84).

[Wu+21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. "A Comprehensive Survey on Graph Neural Networks." In: *IEEE Transactions on Neural Networks and Learning Systems* 32 (1 Jan. 2021), pp. 4–24. doi: 10.1109/TNNLS.2020.2978386 (cited on page 68).

[Xia+08] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. "Listwise Approach to Learning to Rank: Theory and Algorithm." In: *Proceedings of the 25th International Conference on Machine Learning*. ICML 2008 (Helsinki, Finland, July 5, 2008). New York, NY, USA: ACM Press, 2008. doi: 10.1145/1390156.1390306 (cited on page 103).

[Xio+17] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. "End-to-End Neural Ad-Hoc Ranking with Kernel Pooling." In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2017 (Shinjuku, Tokyo, Japan, Aug. 7, 2017). New York, NY, USA: ACM, Aug. 7, 2017. doi: 10.1145/3077136.3080809 (cited on page 78).

[XL07] Jun Xu and Hang Li. "AdaRank: A Boosting Algorithm for Information Retrieval." In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2007 (Amsterdam, The Netherlands, July 23, 2007). New York, NY, USA: ACM, July 23, 2007. doi: 10.1145/1277741.1277809 (cited on page 102).

[Xu+08] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. "Directly Optimizing Evaluation Measures in Learning to Rank." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2008 (Singapore, July 20, 2008). New York, NY, USA: ACM, July 20, 2008. doi: 10.1145/1390334.1390355 (cited on page 102).

[Xu+19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *Proceedings of the 7th International Conference on Learning Representations*. ICLR 2019 (New Orleans, LA, USA, May 6, 2019). OpenReview.net, 2019 (cited on pages 68, 69).

[XZC23] Peng Xu, Xiatian Zhu, and David A Clifton. "Multimodal Learning with Transformers: A Survey." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45 (10 Oct. 2023), pp. 12113–12132. doi: 10.1109/TPAMI.2023.3275156 (cited on page 192).

[Yan+22] Nianzu Yang, Huaijin Wu, Kaipeng Zeng, Yang Li, and Junchi Yan. "Molecule Generation for Drug Design: A Graph Learning Perspective." Feb. 18, 2022. doi: 10.48550/arXiv.2202.09212. eprint: 2202.09212 (cs.LG) (cited on page 67).

[Yan82] Mihalis Yannakakis. "The Complexity of the Partial Order Dimension Problem." In: *SIAM j. algebr. discrete methods* 3 (3 Sept. 1982), pp. 351–358. doi: 10.1137/0603036 (cited on pages 167, 169).

[Yao+21] Liuyi Yao, Zhixuan Chu, Sheng Li, Yaliang Li, Jing Gao, and Aidong Zhang. "A Survey on Causal Inference." In: *ACM Trans. Knowl. Discov. Data* 15 (5 Oct. 31, 2021), pp. 1–46. doi: 10.1145/3444944 (cited on page 152).

[Ye+12] Nan Ye, Kian Ming Adam Chai, Wee Sun Lee, and Hai Leong Chieu. "Optimizing F-Measure: A Tale of Two Approaches." In: *Proceedings of the 29th International Conference on Machine Learning*. ICML 2012 (Edinburgh, Scotland, UK, June 26, 2012). 2012 (cited on page 207).

[Yeh+07] Jen-Yuan Yeh, Jung-Yi Lin, Hao-Ren Ke, and Wei-Pang Yang. "Learning to Rank for Information Retrieval Using Genetic Programming." In: *Proceedings of the LR4IR Workshop on Learning to Rank for Information Retrieval*. SIGIR 2007 (Amsterdam, The Netherlands, July 27, 2007). Ed. by Thorsten Joachims, Hang Li, Tie-Yan Liu, and Chengxiang Zhai. Brighton, England: ACM, July 27, 2007 (cited on page 102).

[YR10] Emine Yilmaz and Stephen Robertson. "On the Choice of Effectiveness Measures for Learning to Rank." In: *Inf. Retr. Boston.* 13 (3 June 2010), pp. 271–290. doi: 10.1007/s10791-009-9116-x (cited on page 101).

[You50] W J Youden. "Index for Rating Diagnostic Tests." In: *Cancer* 3 (1 Jan. 1, 1950), pp. 32–35. doi: 10.1002/1097-0142(1950)3:1<32::aid-cncr2820030106>3.0.co;2-3 (cited on page 89).

[Yue+07] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. "A Support Vector Method for Optimizing Average Precision." In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2007 (Amsterdam, The Netherlands, July 23, 2007). New York, NY, USA: ACM, July 23, 2007. doi: 10.1145/1277741.1277790 (cited on page 102).

[YJ09] Yisong Yue and Thorsten Joachims. "Interactively Optimizing Information Retrieval Systems as a Dueling Bandits Problem." In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML 2009 (Montréal, Quebec, Canada, June 14, 2009). New York, NY, USA: ACM, June 14, 2009. doi: 10.1145/1553374.1553527 (cited on page 84).

[YJ11] Yisong Yue and Thorsten Joachims. "Beat the Mean Bandit." In: *Proceedings of the 28th International Conference on Machine Learning*. ICML 2011 (Bellevue, Washington, USA, June 28, 2011). Omnipress, 2011, pp. 241–248 (cited on page 26).

[Zah+17] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R Salakhutdinov, and Alexander J Smola. "Deep Sets." In: *Advances in Neural Information Processing Systems*. NIPS 2017 (Long Beach, CA, USA, Dec. 4, 2017). Vol. 30. Curran Associates, Inc., 2017, pp. 3394–3404 (cited on pages 66, 124–126).

[Zha+06] Hongyuan Zha, Zhaohui Zheng, Haoying Fu, and Gordon Sun. "Incorporating Query Difference for Learning Retrieval Functions in World Wide Web Search." In: *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*. CIKM 2006 (Arlington, Virginia, USA, Nov. 6, 2006). New York, NY, USA: ACM Press, 2006. doi: 10.1145/1183614.1183660 (cited on pages 78, 81).

[ZZ06] Min-Ling Zhang and Zhi-Hua Zhou. "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization." In: *IEEE Trans. Knowl. Data Eng.* 18 (10 Oct. 2006), pp. 1338–1351. doi: 10.1109/tkde.2006.162 (cited on page 115).

[ZC18] Muhan Zhang and Yixin Chen. "Link Prediction Based on Graph Neural Networks." In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS 2018 (Montréal, Canada, Dec. 3, 2018). Red Hook, NY, USA: Curran Associates Inc., Dec. 3, 2018, pp. 5171–5181. doi: 10.5555/3327345.3327423 (cited on page 67).

[ZC05] Qiaoping Zhang and Isabelle Couloigner. "A New and Efficient K-Medoid Algorithm for Spatial Clustering." In: *Proceedings of Computational Science and Its Applications*. ICCSA 2005 (Singapore, May 9, 2005). Springer-Verlag, 2005, pp. 181–189 (cited on page 139).

[ZY15] Yongli Zhang and Yuhong Yang. "Cross-Validation for Selecting a Model Selection Procedure." In: *J. Econom.* 187 (1 July 1, 2015), pp. 95–112. doi: 10.1016/j.jeconom.2015.02.006 (cited on page 46).

[ZWZ19] Yuan Zhang, Dong Wang, and Yan Zhang. "Neural IR Meets Graph Embedding: A Ranking Model for Product Search." In: *Proceedings of the World Wide Web Conference*. WWW 2019 (San Francisco, CA, USA, May 13, 2019). New York, NY, USA: ACM, May 13, 2019. doi: 10.1145/3308558.3313468 (cited on pages 78, 80).

[Zha+24] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. "Dense Text Retrieval Based on Pretrained Language Models: A Survey." In: *ACM Trans. Inf. Syst.* 42 (4 Feb. 9, 2024), pp. 1–60. doi: 10.1145/3637870 (cited on page 193).

[ZZS08] Zhaohui Zheng, Hongyuan Zha, and Gordon Sun. "Query-level Learning to Rank Using Isotonic Regression." In: *Proceedings of the 46th Annual Allerton Conference on Communication, Control, and Computing*. Allerton Conference on Communication, Control, and Computing 2008 (Monticello, IL, USA, Sept. 23, 2008). IEEE, Sept. 2008, pp. 1108–1115. doi: 10.1109/allerton.2008.4797684 (cited on pages 78, 79).

[Zhe+07] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. "A General Boosting Method and its Application to Learning Ranking Functions for Web Search." In: *Advances in Neural Information Processing Systems*. NIPS 2007 (Vancouver, BC, Canada, Dec. 3, 2007). Ed. by J Platt, D Koller, Y Singer, and S Roweis. Vol. 20. Curran Associates, Inc., 2007 (cited on pages 78, 79).

[ZKF21] Allan Zhou, Tom Knowles, and Chelsea Finn. "Meta-Learning Symmetries by Reparameterization." In: *Proceedings of the 9th International Conference on Learning Representations*. ICLR 2021 (Virtual, May 3, 2021). OpenReview.net, 2021 (cited on page 67).

[ZC88] Zhou and Chellappa. "Computation of Optical Flow Using a Neural Network." In: *Proceedings of the International Conference on Neural Networks*. ICNN 1988 (San Diego, CA, USA, July 24, 1988). July 1988, pp. 71–78. doi: 10.1109/ICNN.1988.23914 (cited on page 64).

[Zho+08] Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. "Learning to Rank with Ties." In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR 2008 (Singapore, July 20, 2008). New York, NY, USA: ACM, July 20, 2008. doi: 10.1145/1390334.1390382 (cited on pages 78, 79).

[Zho+14] Yangming Zhou, Yangguang Liu, Jiangang Yang, Xiaoqi He, and Liangliang Liu. "A Taxonomy of Label Ranking Algorithms." In: *J. Comput.* 9 (3 Mar. 1, 2014). doi: 10.4304/jcp.9.3.557-565 (cited on page 74).

[Zie+19] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. "Fine-Tuning Language Models from Human Preferences." Sept. 18, 2019. eprint: 1909.08593 (cs.CL) (cited on page 73).

[ZDT00] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results." In: *Evol. Comput.* 8 (2 2000), pp. 173–195. doi: 10.1162/106365600568202 (cited on pages 141, 173, 177).

[Zoe+08] Onno Zoeter, Mike Taylor, Ed Snelson, John Guiver, Nick Craswell, and Martin Szummer. "A Decision Theoretic Framework for Ranking using Implicit Feedback." In: *Proceedings of the SIGIR Workshop on Learning to Rank for Information Retrieval*. SIGIR 2008 (Singapore, July 20, 2008). July 2008 (cited on page 101).