Learning Context-Dependent Choice Functions

Preprint, compiled October 22, 2021

Karlson Pfannschmidt [1*], **Pritha Gupta** [1†], **Björn Haddenhorst** [1‡], **and Eyke Hüllermeier** [2§]

[1]Paderborn University, Warburger Straße 100, Paderborn, Germany
[2]LMU Munich, Akademiestr. 7, Munich, Germany

Abstract

Choice functions accept a set of alternatives as input and produce a preferred subset of these alternatives as output. We study the problem of learning such functions under conditions of *context-dependence* of preferences, which means that the preference in favor of a certain choice alternative may depend on what other options are also available. In spite of its practical relevance, this kind of context-dependence has received little attention in preference learning so far. We propose a suitable model based on context-dependent (latent) utility functions, thereby reducing the problem to the task of learning such utility functions. Practically, this comes with a number of challenges. For example, the set of alternatives provided as input to a choice function can be of any size, and the output of the function should not depend on the order in which the alternatives are presented. To meet these requirements, we propose two general approaches based on two representations of context-dependent utility functions, as well as instantiations in the form of appropriate end-to-end trainable neural network architectures. Moreover, to demonstrate the performance of both networks, we present extensive empirical evaluations on both synthetic and real-world datasets.

***K*eywords** preference learning · choice functions · context-dependence · neural networks

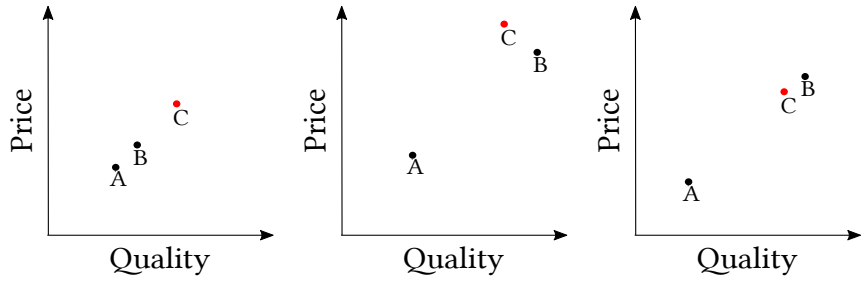
1 Introduction

The notion of *preference* plays a central role in various scientific disciplines, such as economics, psychology, and more recently also computer science and artificial intelligence [19]. In these fields, mathematical formalisms have been developed for modelling and reasoning about preferences, and for analyzing data that originates from observed or revealed preferences. In this regard, *choice* observations are of specific interest, in which a subset of "good" alternatives is selected from a set of available candidates. In particular, starting with the seminal work by Arrow [6], *choice functions* have been analyzed as a key concept of a formal theory of choice and preference. The study of pairwise preferences even goes back to work by Fechner [36], who considered the varying perception of different stimuli.

In machine learning, preferences are at the core of *preference learning*, which has received increasing attention in recent years [40]. Roughly speaking, the goal in preference learning is to learn (predictive) preference models from preference data. Somewhat surprisingly, and in spite of a close connection between ranking and choice, the problem of learning *subset* choice functions has received very little attention so far, with only a few notable exceptions [10, 109]. In this paper, we therefore address the problem of learning choice functions, which express preferences in terms of subsets (or equivalently, bipartitions) of Q. From a machine learning point of view, the problem of learning choice functions comes with a number of challenges. For example, while algorithms for supervised learning normally assume inputs in the form of feature vectors of fixed length, the inputs in our setting are neither vectors nor of fixed size. Instead, a choice function is supposed to accept inputs in the form of sets Q of any size, and to return a subset (choice) of the elements as output. In case a set Q is represented by an ordered list of its elements, a choice function thus has to be invariant with respect to permutations of its input.

Not less interestingly, and in fact the key motivation of this paper, choice functions could be *context-dependent*, in the sense that the preference in favor of an alternative may depend on what other options are available. Context-dependence of this kind has been observed, for example, in marketing studies [26, 13], and has been investigated systematically

*kiudee@mail.upb.de
†prithag@mail.upb.de
‡bjoernha@mail.upb.de
§eyke@ifi.lmu.de



(a) Compromise (b) Attraction (c) Similarity

Figure 1: Context effects identified in the literature [94].

in fields like economics and psychology. More specifically, three major context effects have been identified in the literature, the compromise effect [103], the attraction effect [52], and the similarity effect [113]:

- The compromise effect states that the relative utility of an object increases by adding an extreme option that makes it a compromise in the set of alternatives [94]. For instance, consider the set of objects $\{A, B\}$ in Figure 1a. The ordering of these objects depends on how much the consumer is weighing the quality of the product in relation to its price. If price is the main constraint, then the preference order will be $A \succ B$. But as soon as another extreme option C becomes available, object B may be considered more favorable, because it represents a compromise between the three alternatives. Thus, the preference relation between A and B might get inverted and turned into $B \succ A$.
- Figure 1b illustrates the attraction effect. Here, if we add another object C to the set of objects $\{A, B\}$, where C is slightly dominated by B, the relative utility share for object B increases with respect to A. The major psychological reason is that consumers have a strong preference for dominating products [52]. Thus, the preference relation between A and B may again be influenced.
- The similarity or substitution effect is another phenomenon, according to which the presence of similar objects tends to reduce the overall probability of an object to be chosen, as it will divide the loyalty of potential consumers [52]. In Figure 1c, B and C are two similar objects. Consumers who prefer high quality will be divided amongst the two objects, resulting in a decrease of the relative utility share of object B. Again, this may lead to turning a preference $B \succ A$ into $A \succ B$, at least on an aggregate (population) level, if preferences are defined on the basis of choice probabilities.

Context-dependence as explained above has received only limited consideration in the machine learning literature until recently [22, 85, 10, 101, 95, 15, 63].

Additionally, the context effects discussed so far focus on effects that have been observed for humans, but ignore that the space of (subset) choice functions and thus the number of possible applications is much larger. Many algorithmic problems can be framed as a choice problem, e. g., in the Knapsack problem one is tasked in choosing a set of maximal utility while obeying capacity constraints. Computing the medoid of a set of points (i. e., the point with minimal distance to each other point) is a singleton choice problem. It is clear that these problems cannot be solved by considering each choice alternative individually, but the complete choice context needs to be incorporated. In practice, there are many abstract choice problems similar to these, e. g., portfolio selection [72], algorithm selection [91, 14] and team selection [119] just to name a few. All these problems have in common, that the context-dependence naturally arises because the output depends jointly on all objects in the set and not because a decision maker behaves rationally or irrationally.

Motivated by its practical relevance, we formalize the problem of learning context-dependent choice functions. To this end, we provide a formal definition of such functions and propose a data-generating process consisting of two stages: First, choice alternatives are scored in terms of latent utility degrees, and then, a choice set is determined on the basis of these scores (Section 3). Based on this model, we propose two representations of the latent (context-dependent) utility, called First Evaluate Then Aggregate (FETA) and First Aggregate Then Evaluate (FATE), which have appealing properties from a learning point of view (Section 4), as well as realizations of these models in terms of neural network architectures (Section 5). Thanks to these architectures, called FETA-NET and FATE-NET [85], we are able to learn subset choices on sets of objects in an end-to-end trainable manner. To demonstrate the performance of both networks, we present extensive empirical evaluations on both synthetic and real-world choice datasets (Section 6). Additional information and supplementary material is provided in an appendix, to which we will refer occasionally.

2 Related Literature

The problem of how to model preferences in general has been extensively studied from different viewpoints in the past. From an axiomatic/normative perspective, one posits which properties have to hold for preferences to be considered "rational," and studies consequences of these properties. Luce's choice axiom was introduced in 1959 by Luce and requires that the preference between two items does not depend on the presence or absence of any other choice alternative, a property commonly referred to as independence of irrelevant alternatives (IIA). The set of objects from which a particular preference is observed is also called the *context* [77, 20, 94], and thus preferences obeying IIA are also called context-independent [60]. In the same year, Debreu [27, pp. 56f] proved the ordinal representation theorem, which shows that preferences can be represented by a continuous utility function, if certain conditions including transitivity are assumed to hold. A related line of research was concerned with the concept of *revealed preferences*, for which most axioms can be reduced to some notion of transitivity [98, 50, 100].

On the other side of the spectrum, observational studies in economics and psychology were more concerned with how humans actually behave, and studied how the observed behavior deviates from IIA [114, 113, 51, 52, 103, 82, 104, 102, 115, 30, 83, 99]. It consistently was observed that choice behavior depended on the specific collection of alternatives available, the *context* of the choice. Rooderkerk, Van Heerde, and Bijmolt [94] and Rieskamp, Busemeyer, and Mellers [92] provide an extensive overview of the different context effects which were identified over the years and which we already showcased in the introduction. This motivated researchers to come up with methods able to model these violations. Classical random utility models (RUMs), like the multinomial logit (MNL) model, are not able to take these effects into account. Therefore, extensions of RUMs were proposed, which are able to capture the compromise and attraction effect [115, 61, 80], the similarity effect [113, 56] or all of the above [94]. One important line of research focuses on the assumption that the decision maker chooses based on multiple utility functions (so called "multiple selfs", or "multi-self" for short), which are suitably aggregated. This setting has been studied in economics [73, 55, 61, 39, 71, 45] and psychology [113, 102, 115]. Continuing this line of research, Ambrus and Rozen [5] show that by utilizing a collection of context-independent utility functions, combined with a suitable aggregation, one is able to model arbitrary choice functions. That is, choice behavior across multiple sets can be modelled even though it might violate context-independence.

While traditional research on preferences, as discussed above, is mostly of a normative, prescriptive or descriptive nature, the advent of machine learning triggered a shift towards "predictive" models. Rosenfeld, Oshiba, and Singer [95] build on ideas of the multi-self literature and propose to learn set-dependent weights and embeddings, which are then linearly combined to arrive at an aggregated score for each object. Benson, Kumar, and Tomkins [10] consider the problem of learning preferences in the form of subsets of objects. To this end, they extend the classical multinomial logit model to account for violations of context-independence. Higher-order interactions between objects are added specifically for those subsets that cause a violation. The set of objects for which choices or choice sets are observed is assumed to be fixed. Therefore, the approach cannot be used for arbitrary task sets, where it can happen that an object is only observed once. Our approach to decompose a context-dependent utility function into an aggregation across smaller sub-contexts has been a recent, promising direction in studying choices [85, 101], and will be the focus of this paper.

Decomposition approaches have also been employed in the related field of "learning to rank". Ai et al. [2] employ a context-independent model to pre-sort the objects, while a recurrent neural network is used in a subsequent step to fine-tune the ranking. The FATE approach, introduced in the context of choice by Pfannschmidt, Gupta, and Hüllermeier [85], obviates the need to pre-sort the objects, by directly embedding each object to produce a representation for each set of objects (aggregation), which is then used as the context to produce the final ranking (evaluation). The authors also introduce an algorithm where this order is swapped, called FETA, in which each object is scored in the context of another object first, and only then the scores are aggregated to produce a final ranking. Ai et al. [3] later consider a similar decomposition, where higher order interactions are approximated by employing sampling.

3 A Probabilistic Model of Choice

We start by establishing the necessary notation (refer to Appendix A for an overview). Throughout this paper, $[\![A]\!]$ is defined to be 1 if A is a true statement, and 0 otherwise. We will denote by $\mathcal{X} \subset \mathbb{R}^d$ a set of reference objects serving as choice alternatives, which, for simplicity, we assume to be finite (albeit of arbitrary size), if not explicitly stated otherwise. An object or item $\boldsymbol{x} \in \mathcal{X}$ is represented by a vector of features $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathcal{X}$. A non-empty subset $\mathcal{Q}$ of $2^{\mathcal{X}} \setminus \{\emptyset\}$ is called a *choice task space* if $\emptyset \notin \mathcal{Q} \neq \emptyset$ and any $Q \in \mathcal{Q}$ is called a *choice task*. A *choice* for $Q \in \mathcal{Q}$ is a non-empty subset of Q and the set $\mathcal{C} \coloneqq \bigcup_{Q \in \mathcal{Q}} 2^Q \setminus \{\emptyset\}$ of choices for any $Q \in \mathcal{Q}$ is called the *choice space*.

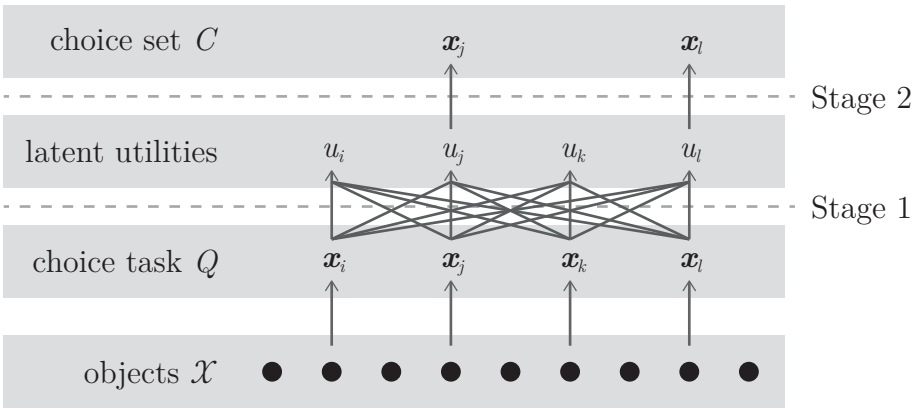


Figure 2: Overview of the data-generating process: First, a task Q is produced (with probability $p(Q)$) by sampling from $\mathcal{X}$. The objects in Q are assigned latent utility degrees, and the choice set is finally constructed on the basis of these scores.

We say that a function $c\colon \mathcal{Q} \longrightarrow C$ is a *(subset) choice function (for $\mathcal{Q}$)* if $c(Q) \subseteq Q$ is fulfilled for any $Q \in \mathcal{Q}$, and in case $|c(Q)| = 1$ holds for any $Q \in \mathcal{Q}$, c is called a *singleton choice function (for $\mathcal{Q}$)*. A typical example for a real-world singleton choice function is when a user enters a query in a search engine and receives a list of results (Q) of which they pick one and click on. Subset choice functions usually occur, when a diverse set of objects is sought, e. g., a search engine decides on a set of the most relevant, but diverse, results to display to the user.

As common in machine learning, the input-output dependency of interest, in our case between tasks and choices, is not assumed to be deterministic. Instead, we assume a probabilistic dependence, which is captured by a (conditional) probability distribution $p(\cdot \mid Q)$ on the non-empty subsets of Q for every $Q \in \mathcal{Q}$. Here, $p(C \mid Q)$ is interpreted as the probability to observe the choice C given the task Q. For the sake of convenience, we suppose w.l.o.g. $p(\cdot \mid Q)$ to be extended to C via $p(C \mid Q) \coloneqq 0$ for any $C \in C \setminus 2^Q$. Moreover, we write for short $p(\boldsymbol{x} \mid Q)$ for $p(\{\boldsymbol{x}\} \mid Q)$. In case $p(Q)$ is the latent probability that $Q \in \mathcal{Q}$ is given as task, the whole data-generating process is modelled by the joint distribution

$$p(Q, C) \coloneqq p(Q) \cdot p(C \mid Q) \tag{1}$$

on $\mathcal{Q} \times C$.

We call the choice probabilities *context-independent* if

$$\frac{p(C \mid Q)}{p(C' \mid Q)} = \frac{p(C \mid Q')}{p(C' \mid Q')}$$

is fulfilled for every $Q, Q' \in \mathcal{Q}$ and any $C, C' \in C$ with $C, C' \subseteq Q \cap Q'$. Conversely, we say that a system of choice distributions is *context-dependent*, if this equality is violated on at least one pair of $Q, Q' \in \mathcal{Q}$. This definition extends in a straight-forward and consistent way the notion of independence of irrelevant alternatives (IIA) introduced by Arrow [6], which was originally only defined for the case of singleton choice, in which C consists of elements of size one only. We choose to use the more general term of context-(in)dependence, for the simple reason that the notion of "irrelevant" alternatives is rather tailored to the analysis of human choices but less meaningful in our more general setting of arbitrary choice functions.

As an example, consider the knapsack problem, where the goal is to select a set of objects which maximize a certain utility, while obeying capacity constraints. It is clear that the decision on which object to include in the choice set needs to incorporate the complete choice task context, and that one is not able to ascertain the relative choice probability of two alternatives while ignoring all others. As already explained in the introduction, context-independence is often violated in practice. This motivates the development of context-dependent learning methods.

Utility-Based Choices We propose to model choices as the result of a two-stage process (cf. Figure 2 for an overview), grounding them on the notion of *utility*: In the first stage, each object in a given task $Q \in \mathcal{Q}$ is assigned a real-valued utility score. Then in the second stage, choices are generated based on these scores.

Utility theory has a long history in economics [120, 25, 73]. Originally introduced as a way to measure the satisfaction achieved by a certain alternative [11], it is nowadays common in decision theory to consider utility more as an abstract value that ought to be maximized by any rational decision maker [120, 96]. This is formalized by means of a *generalized*

utility function (for $\mathcal{Q}$)

$$U\colon \{(\boldsymbol{x}, Q) : \boldsymbol{x} \in Q \in \mathcal{Q}\} \longrightarrow \mathbb{R}, \quad (\boldsymbol{x}, Q) \mapsto U(\boldsymbol{x}, Q), \tag{2}$$

which allows for modelling the utility of an object as a function of both, properties of the object itself as well as properties of other choice alternatives in Q, which constitute the context in which $\boldsymbol{x}$ is considered: $U(\boldsymbol{x}, Q)$ expresses a degree of utility of $\boldsymbol{x}$ in the context Q, i. e., given the availability of other choice alternatives $\boldsymbol{x}' \in Q \setminus \{\boldsymbol{x}\}$. The score $U(\boldsymbol{x}, Q)$ is supposed to capture an abstract notion of utility, which in turn reflects the propensity of $\boldsymbol{x}$ to be chosen in any task Q.

We call a utility function *context-independent* in case $U(\boldsymbol{x}, Q) = U(\boldsymbol{x}, Q')$ holds for any $Q, Q' \in \mathcal{Q}$ with $\boldsymbol{x} \in Q \cap Q'$ and *context-dependent* otherwise. Via abbreviating $U(\boldsymbol{x}) \coloneqq U(\boldsymbol{x}, Q)$ for some arbitrary $Q \in \mathcal{Q}$ with $\boldsymbol{x} \in Q$, any context-independent utility function may be thought of as a function $U : \mathcal{X} \longrightarrow \mathbb{R}$.

Moving on to the second stage, based on a utility function U, one may define in a deterministic manner for $Q \in \mathcal{Q}$ the corresponding *singleton choice* as

$$C_{\text{singleton}}(U, Q) \coloneqq \arg\max_{\boldsymbol{x} \in Q} U(\boldsymbol{x}, Q) \tag{3}$$

and for $t \in \mathbb{R}$ the *subset choice (with threshold t)* as

$$C^t_{\text{subset}}(U, Q) \coloneqq \{\boldsymbol{x} \in Q : U(\boldsymbol{x}, Q) \geq t\}. \tag{4}$$

Clearly, $C_{\text{singleton}}(U, \cdot)$ and $C^t_{\text{subset}}(U, \cdot)$ are in fact choice functions and in case $\boldsymbol{x} \mapsto U(\boldsymbol{x}, Q)$ is injective (i. e., there are no ties), for any $Q \in \mathcal{Q}$, the former one is a singleton choice function. There is an interesting connection to social choice theory, where a social choice rule is employed to select an outcome out of a set of possible outcomes in order to maximize some notion of utility for a population of individuals with possibly varying utility functions. The injectivity of such a social choice rule is called *resoluteness* and it is an important property considered in social choice theory, where it also plays a role in several impossibility results [59, 81]. The singleton choice is a special case of the more general top-k choice, where the goal is to select the k best objects. It differs from subset choice in so far that the size of the choice sets is always fixed, whereas in subset choice it can vary. The top-k choice setting has strong connections to the ranking setting, which we will discuss below.

Further note that using thresholding to convert a set of scores into a partition is a standard approach in multi-label classification [64] and multi-criteria sorting [4].

In the probabilistic setting, the utility function U may serve to model probabilistic choices $p(\cdot \mid Q)$, $Q \in \mathcal{Q}$, on C by using the utility scores as the corresponding parameters of the distributions. Certainly, there are various ways in which this idea could be realized:

Singleton choice In the case of *singleton choice*, a natural assumption is the multinomial logit (MNL) model, in which for any $Q \in \mathcal{Q}$ and $\boldsymbol{x} \in Q$,

$$p_{\text{MNL}}(\boldsymbol{x} \mid Q) \coloneqq \frac{\exp\left(U(\boldsymbol{x}, Q)\right)}{\sum_{\boldsymbol{x}' \in Q} \exp(U(\boldsymbol{x}', Q))}. \tag{5}$$

and $p(C \mid Q) = 0$ for any $C \in 2^Q$ of size ≥ 2 [12, 46, 24, 70, 108]. Note here that these choice probabilities are context-independent, if U is context-independent. An important special case is the Bradley-Terry-Luce model [16], which only considers pairwise comparisons (i. e., $|Q| = 2$ for all $Q \in \mathcal{Q}$).

Subset choice For the choice of arbitrary subsets (not limited to singleton sets), a simple model is obtained by treating the inclusion or exclusion of each object $\boldsymbol{x}$ in a task Q as independent given the utilities. This results in the distributions $p(\cdot \mid Q)$ given by

$$p(C \mid Q) \coloneqq \gamma(U, Q) \prod_{\boldsymbol{x} \in Q} \frac{\exp([\![\boldsymbol{x} \in C]\!] U(\boldsymbol{x}, Q))}{1 + \exp(U(\boldsymbol{x}, Q))} \tag{6}$$

for any non-empty $C \in 2^Q$ and $Q \in \mathcal{Q}$, where $\gamma(U, Q)$ is a constant such that $\sum_{C \in 2^Q \setminus \{\emptyset\}} p(C \mid Q) = 1$ holds. If U is context-independent, the quantity

$$\frac{p(C \mid Q)}{p(C' \mid Q)} = \prod_{\boldsymbol{x} \in Q} \frac{\exp([\![\boldsymbol{x} \in C]\!] U(\boldsymbol{x}))}{\exp([\![\boldsymbol{x} \in C']\!] U(\boldsymbol{x}))} = \prod_{\boldsymbol{x} \in C \cup C'} \frac{\exp([\![\boldsymbol{x} \in C]\!] U(\boldsymbol{x}))}{\exp([\![\boldsymbol{x} \in C']\!] U(\boldsymbol{x}))}$$

does not depend on Q, and thus the choice probabilities $p(C \mid Q)$ are context-independent as well.

Choices based on rankings Yet another type of model is obtained by assuming that, based on the latent utilities $U(\boldsymbol{x}, Q)$, $\boldsymbol{x} \in Q$, a ranking π on Q is sampled first and then turned into a choice set via a (possibly probabilistic)

procedure $g\colon \pi \mapsto g(\pi) \in 2^Q$ afterwards. The probability $p(C \mid Q)$ is then simply the probability that this procedure results in the output C, i. e.,

$$p(C \mid Q) \coloneqq \sum\nolimits_{\pi} p(\pi)\; p(g(\pi) = C) \;\;, \tag{7}$$

where the sum is taken over all possible rankings π over Q. An approach of that kind might be appealing, because probability distributions on rankings have been studied quite thoroughly in the literature. Important families of ranking distributions include distance-based ranking models [37], of which the Mallows model [69] is a popular instance, and multistage ranking models [38], most prominently represented by the Plackett-Luce distribution [86]. An important special case for g is *top-k choice*, where the first k objects are chosen deterministically (i. e., $g(\pi) = \{\pi^{-1}(1), \ldots, \pi^{-1}(k)\} \subseteq Q$ holds with probability 1 for any ranking $\pi\colon Q \longrightarrow \mathbb{N}$). This can be generalized, for example, by assuming that the size k is not fixed but random. An even more general model has recently been proposed by Fahandar, Hüllermeier, and Couso [34], where choices are not necessarily restricted to top-k sets.

In this paper, we are mainly interested in tackling the problem of learning context-dependent choice functions from training data. The performance of a particular hypothesis, i. e., a choice function $c\colon \mathcal{Q} \longrightarrow C$, is measured by an appropriate loss function (see Section 4). In Section 6.2 we go into more detail on how to derive suitable loss functions from (5) and (6). After having introduced suitable models for utility-based choices, we now turn to the problem of representing context-dependent choice functions.

4 Learning Context-Dependent Choice Functions

Our main interest in this paper is to tackle choice from a machine learning perspective. More specifically, we seek to induce a predictive choice function $c\colon \mathcal{Q} \longrightarrow C$ from training data $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N \subset \mathcal{Q} \times C$ in the form of exemplary tasks Q_i together with observed choices $C_i \in 2^{Q_i}$. The performance of such a function is measured in terms of its expected loss (risk)

$$R(c) \coloneqq \int_{\mathcal{Q}\times C} L(C, c(Q))\, dp(Q, C)\,,$$

where $L : C \times C \longrightarrow \mathbb{R}$ is a loss function (cf. Section 6.2 for an overview of the loss functions we consider), and p the probability measure associated with the distribution (1), i. e., the underlying data-generating process modelling the probability of observing tasks Q together with choices C. The Bayes predictor c^* assigns each task Q the respective loss minimizer

$$c^*\colon Q \mapsto \arg\min_{\hat{C}\in C} \int_C L(C, \hat{C})\, dp(C \mid Q)\;.$$

Since $p(Q, C)$ is usually unknown, one therefore opts to minimize the *empirical risk*

$$R_{\text{emp}}(c) \coloneqq \frac{1}{N} \sum_{i=1}^{N} L(C_i, c(Q_i)) \tag{8}$$

on the given data $\mathcal{D}$ instead.

Assuming the data to be generated according to one of (3)–(7) (known to the learner) and by means of an (unknown) latent utility function (2), this loss minimization problem essentially comes down to learning the generalized utility function (2). This function, while allowing one to model context-dependence, causes several practical problems, mainly because its second argument, Q, is a *set* of *variable* size.

Many machine learning models such as neural networks or support vector machines require data to be given in the form of a feature vector $\boldsymbol{x} \in \mathbb{R}^m$. Hence, in order to apply such a model for learning a utility function $U\colon \{(\boldsymbol{x}, Q) : \boldsymbol{x} \in Q \in \mathcal{Q}\} \longrightarrow \mathbb{R}$, we have to fix an injective feature transformation $\Psi\colon \mathcal{Q} \longrightarrow \mathbb{R}^m$.

We choose to represent $Q = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\} \in \mathcal{Q} \subset \mathbb{R}^d$ by the vector $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k) \in \mathbb{R}^{kd}$. Of course, this does only define a valid transformation Ψ in case $|Q|$ is the same for each $Q \in \mathcal{Q}$. Assuming this to be the case, we may consider a utility function $U\colon \{(\boldsymbol{x}, Q) : \boldsymbol{x} \in Q \in \mathcal{Q}\} \longrightarrow \mathbb{R}$ as a function $\mathbb{R}^{(k+1)d} \longrightarrow \mathbb{R}$. Noticing that $Q = \{\boldsymbol{x}_{\sigma(i)} : i \in [k]\}$ holds for any bijection $\sigma\colon [k] \longrightarrow [k]$, this function should necessarily be *permutation-invariant* or *symmetric* in the sense that

$$U(\boldsymbol{x}, (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)) = U(\boldsymbol{x}, (\boldsymbol{x}_{\sigma(1)}, \ldots, \boldsymbol{x}_{\sigma(k)})) \tag{9}$$

for each permutation σ on $[k]$ [106].

The utility choice models proposed below will enforce this property and are also capable of dealing with tasks of different sizes. More specifically, we present two general decompositions, which are able to approximate a generalized

latent utility function (2). Section 4.1 describes FETA, which decomposes (2) into first- and second-order (or, more generally, higher order) utility functions and aggregates the corresponding scores into an overall utility score. The FATE approach (Section 4.2), on the other hand, first computes an embedding of the complete object context Q in a space of fixed dimensionality, and evaluates the utility of each object in that space. The former could be advantageous for datasets, of which the choice task contexts can be expressed through *local* interactions, while the latter is useful, if the set of objects as a whole can be summarized by suitable *global* properties (e. g., choosing that element of a set, which is closest to the centroid of all elements in this set).

4.1 First Evaluate Then Aggregate

Recall that the overall objective is to model the context-dependent utility function (2), i. e., the utility of each object should not only depend on object attributes, but also on the choice task Q. One way of handling the problem of rating objects in contexts of variable size is to decompose a context into sub-contexts of a fixed size k [85, 101]. More specifically, the idea is to learn *sub-utility functions* $U_0, \ldots, U_K$ of the form $U_0 : \mathcal{X} \longrightarrow \mathbb{R}$ and

$$U_k \colon D_k \longrightarrow \mathbb{R}\,, \quad D_k \coloneqq \{(\boldsymbol{x}, A) : \boldsymbol{x} \in \mathcal{X} \text{ and } A \subseteq \mathcal{X} \setminus \{\boldsymbol{x}\} \text{ with } |A| = k\}$$

for $1 \leq k \leq K \leq |Q|$, and represent the original function (2) as an aggregation

$$U(\boldsymbol{x}, Q) \coloneqq U_0(\boldsymbol{x}) + \sum_{k=1}^{K} \overline{U}_k(\boldsymbol{x}, Q), \tag{10}$$

where $\overline{U}_k(\boldsymbol{x}, Q)$ is the average over the values $U_k(\boldsymbol{x}, Q')$ for subsets Q' of $Q \setminus \{\boldsymbol{x}\}$ consisting of k distinct elements, i. e., formally

$$\overline{U}_k(\boldsymbol{x}, Q) = \frac{1}{\binom{|Q|}{k} - \binom{|Q|-1}{k-1}} \sum_{Q' \subseteq Q \setminus \{\boldsymbol{x}\} \colon |Q'|=k} U_k(\boldsymbol{x}, Q').$$

Note, that the sum is taken w.r.t. to all k-sized subsets Q' of $Q \setminus \{\boldsymbol{x}\}$, potentially including some in $2^{\mathcal{X}} \setminus \mathcal{Q}$. Here, $U_k(\boldsymbol{x}, Q)$ may be thought of as a measure to which extent an item $\boldsymbol{x}$ is preferred to the elements of Q, and $\overline{U}_k(\boldsymbol{x}, Q)$ as an indicator of how much $\boldsymbol{x}$ is on average preferred to k distinct elements from $Q \setminus \{\boldsymbol{x}\}$. We refer to this approach as First Evaluate Then Aggregate (FETA), because an alternative is first evaluated in each sub-context, and these evaluations are then aggregated. Accordingly, we call U defined in (10) the *FETA utility function with sub-utility functions* $U_0, \ldots, U_K$ and denote it by $U_{\text{FETA}}^{U_0,\ldots,U_K}$.

Batsell and Polking [7] propose a related expansion in the context of market share modelling. Seshadri, Peysakhovich, and Ugander [101] call it an instantiation of the *universal logit model*, since it can be seen as a generalization of the multinomial logit model (5), when conditioning on the task Q.

Roughly speaking, the motivation behind the above decomposition is that dependencies and interaction effects between objects should only occur up to a certain order $K + 1$, or at least can be limited to this order without losing too much information. To see what we mean by "order" in this context, observe that the first order model ($K = 0$) reduces to $U_0(\boldsymbol{x})$ and thus only models the inherent utility of each object. A second order model ($K = 1$) then introduces pairwise terms. This is an assumption that is commonly made in the literature on aggregation functions [44]. The reason why the utilities are averaged for a fixed k, but summed across different k, is to give each order equal weight. This prevents the utility from being dominated by higher-order interactions. Furthermore, it allows the sub-utility functions to output scores in roughly the same scale, which is advantageous when the model is applied to choice tasks Q of varying size.

Given the models of context-dependent choices as outlined above, the learning problem essentially comes down to learning the utility function (10) of order $K + 1$. From this function, one can then derive the utility function (2), which in turn allows for deriving predictions of choices via the choice functions discussed before.

In this paper, we realize (10) for the special case $K = 1$, which can be seen as a second-order approximation of a context-dependent utility function. Thus, we propose the representation of a choice function c based on a latent sub-utility function $U_0 : \mathcal{X} \longrightarrow \mathbb{R}$ and a pairwise function $U_1 : D_1 \longrightarrow \mathbb{R}$. In this way, the FETA utility function with sub-utility functions U_0, U_1 may be written as

$$U(\boldsymbol{x}, Q) = U_0(\boldsymbol{x}) + \frac{1}{|Q| - 1} \sum\nolimits_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} U_1(\boldsymbol{x}, \{\boldsymbol{y}\})\,. \tag{11}$$

The value $U_0(\boldsymbol{x})$ can be seen as a kind of inherent, context-independent utility of $\boldsymbol{x}$, whereas the scores $U_1(\boldsymbol{x}, \{\boldsymbol{y}\})$, $\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}$, serve as "corrections" of this utility in the context of the task Q. Seshadri, Peysakhovich, and Ugander

Example 4.1 (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 $\boldsymbol{d}$.

[101] propose a similar approximation, but instead of averaging the task context, the authors simply sum up all utilities and impose sum-to-zero constraints to guarantee identifiability.

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 4.2 *Suppose $|\mathcal{X}| \geq 4$ and $\mathcal{Q}$ to be such that for any distinct $\boldsymbol{x}, \boldsymbol{y}, z \in \mathcal{X}$ there is some $Q \in \mathcal{Q}$ with $\{\boldsymbol{x}, \boldsymbol{y}\} \subseteq Q \not\ni z$. Let $U_0, \tilde{U}_0 \colon \mathcal{X} \longrightarrow \mathbb{R}$ and $U_1, \tilde{U}_1 \colon D_1 \longrightarrow \mathbb{R}$ 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*

$$\forall \boldsymbol{x} \in \mathcal{X}, \forall \boldsymbol{y} \in \mathcal{X} \setminus \{\boldsymbol{x}\} \colon \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) = U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \tilde{U}_0(\boldsymbol{x}) + U_0(\boldsymbol{x}).$$

Proof. $\Leftarrow$ is clear. For proving the remaining implication $\Rightarrow$, suppose that $U_{\text{FETA}}^{(U_0,U_1)} = U_{\text{FETA}}^{(\tilde{U}_0,\tilde{U}_1)}$.

Claim 4.2.1 *For any distinct $\boldsymbol{x}, \boldsymbol{y}, z \in \mathcal{X}$ we have*

$$U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - U_1(\boldsymbol{x}, \{z\}) = \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \tilde{U}_1(\boldsymbol{x}, \{z\}).$$

Proof. For arbitrary $Q, Q' \subseteq \mathcal{X}$ with $|Q| = |Q'|$, $\{\boldsymbol{x}, \boldsymbol{y}\} \subseteq Q \not\ni z$ and $\{\boldsymbol{x}, z\} \subseteq Q' \not\ni \boldsymbol{y}$ we have

$$U_{\text{FETA}}^{(U_0,U_1)}(\boldsymbol{x}, Q) - U_{\text{FETA}}^{(U_0,U_1)}(\boldsymbol{x}, Q') = \frac{1}{|Q| - 1} \left(U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - U_1(\boldsymbol{x}, \{z\})\right).$$

Since this holds for arbitrary (U_0, U_1) (and thus also for $(\tilde{U}_0, \tilde{U}_1)$), Claim 4.2.1 follows. □

Now, let $\boldsymbol{x}_0 \in \mathcal{X}$ be fixed for the moment and define $b \colon \mathcal{X} \longrightarrow \mathbb{R}$ via

$$b(\boldsymbol{x}) \coloneqq \begin{cases} \tilde{U}_0(\boldsymbol{x}) - U_0(\boldsymbol{x}), & \text{if } \boldsymbol{x} = \boldsymbol{x}_0, \\ U_1(\boldsymbol{x}, \{\boldsymbol{x}_0\}) - \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{x}_0\}), & \text{if } \boldsymbol{x} \neq \boldsymbol{x}_0. \end{cases}$$

According to Claim 4.2.1 we have for any distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \setminus \{\boldsymbol{x}_0\}$ the identity

$$\tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) = U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \left(U_1(\boldsymbol{x}, \{\boldsymbol{x}_0\}) - \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{x}_0\})\right) = U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - b(\boldsymbol{x}).$$

Moreover, the definition of b assures that $\tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{x}_0\}) = U_1(\boldsymbol{x}, \{\boldsymbol{x}_0\}) - b(\boldsymbol{x})$ holds for any $\boldsymbol{x} \neq \boldsymbol{x}_0$, i. e., b already fulfills

$$\forall \boldsymbol{x} \in \mathcal{X} \setminus \{\boldsymbol{x}_0\} \colon \forall \boldsymbol{y} \in \mathcal{X} \setminus \{\boldsymbol{x}\} \colon \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{y}\}) = U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - b(\boldsymbol{x}). \tag{12}$$

For $\boldsymbol{x} \in \mathcal{X} \setminus \{\boldsymbol{x}_0\}$ we may choose a query set $Q \subseteq \mathcal{X} \setminus \{\boldsymbol{x}_0\}$ and then (12) assures us

$$\begin{aligned} \tilde{U}_0(\boldsymbol{x}) - U_0(\boldsymbol{x}) &= U_{\text{FETA}}^{(\tilde{U}_0,\tilde{U}_1)}(\boldsymbol{x}, Q) - U_{\text{FETA}}^{(U_0,U_1)}(\boldsymbol{x}, Q) + \frac{1}{|Q| - 1} \sum\nolimits_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} \left(U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{y}\})\right) \\ &= \frac{1}{|Q| - 1} \sum\nolimits_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} \left(U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) - \tilde{U}_1(\boldsymbol{x}, \{\boldsymbol{y}\})\right) \\ &= \frac{1}{|Q| - 1} \sum\nolimits_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} b(\boldsymbol{x}) \\ &= b(\boldsymbol{x}). \end{aligned}$$

Since $\tilde{U}_0(\boldsymbol{x}_0) = U_0(\boldsymbol{x}_0) + b(\boldsymbol{x}_0)$ holds by definition of b, we thus have shown

$$\forall \boldsymbol{x} \in \mathcal{X}\colon b(\boldsymbol{x}) = \tilde{U}_0(\boldsymbol{x}) - U_0(\boldsymbol{x}). \tag{13}$$

With regard to (12) it remains to show

$$\forall \boldsymbol{y} \in \mathcal{X} \setminus \{\boldsymbol{x}_0\}\colon \tilde{U}_1(\boldsymbol{x}_0, \{\boldsymbol{y}\}) = U_1(\boldsymbol{x}_0, \boldsymbol{y}) - b(\boldsymbol{x}_0). \tag{14}$$

For this, note that the same argumentation as before with $\boldsymbol{x}_0$ replaced by some arbitrary $\boldsymbol{x}_1 \in \mathcal{X} \setminus \{\boldsymbol{x}_0\}$ shows us that b also fulfills (12) with $\boldsymbol{x}_0$ replaced by $\boldsymbol{x}_1$. In particular, (14) holds. Combining (12), (13) and (14) completes the proof. □

Corollary 4.3 *Suppose $\mathcal{X}$ and $\mathcal{Q}$ are as in Proposition 4.2 and let $U_0\colon \mathcal{X} \longrightarrow \mathbb{R}$ be fixed. Then, the mapping $U_1 \mapsto U_{\text{FETA}}^{(U_0,U_1)}$ is injective.*

Another interesting theoretical question concerns the expressiveness of the FETA decomposition: Which predictors $c\colon \mathcal{Q} \longrightarrow \mathcal{C}$ can be represented by FETA? The following result shows that the decomposition into pairwise utilities (11) is indeed a restriction, in the sense that it does not allow for representing the entire class of predictors in case $|\mathcal{X}| \geq 7$.

Proposition 4.4 *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 $U_0\colon \mathcal{X} \longrightarrow \mathbb{R}$, $U_1\colon D_1 \longrightarrow \mathbb{R}$ and $t \in \mathbb{R}$ such that the choice function $c\colon \mathcal{Q} \longrightarrow \mathcal{C}$ defined either via $c(\cdot) \coloneqq C_{\text{singleton}}(U_{\text{FETA}}^{U_0,U_1}, \cdot)$ or via $c(\cdot) \coloneqq C_{\text{subset}}^{t}(U_{\text{FETA}}^{U_0,U_1}, \cdot)$ fulfills*

$$c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{x}\}) = \{\boldsymbol{a}\}, \qquad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{y}\}) = \{\boldsymbol{a}\}, \qquad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}, \boldsymbol{y}\}) = \{\boldsymbol{b}\}, \tag{15}$$

$$c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{x}'\}) = \{\boldsymbol{b}\}, \qquad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{y}'\}) = \{\boldsymbol{b}\}, \qquad c(\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}', \boldsymbol{y}'\}) = \{\boldsymbol{a}\}. \tag{16}$$

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 U_0, U_1 and $t \in \mathbb{R}$ such that c defined either via $c(\cdot) \coloneqq C_{\text{singleton}}(U_{\text{FETA}}^{U_0,U_1}, \cdot)$ or via $c(\cdot) \coloneqq C_{\text{subset}}^{t}(U_{\text{FETA}}^{U_0,U_1}, \cdot)$ fulfills both (15) and (16). With the convenient abbreviations $u_{\boldsymbol{r}} \coloneqq U_0(\boldsymbol{r})$ and $u_{\boldsymbol{r},\boldsymbol{s}} \coloneqq U_1(\boldsymbol{r}, \{\boldsymbol{s}\})$, the following constraints for (11) immediately follow from (15):

$$\begin{aligned}
u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{c}} + u_{\boldsymbol{a},\boldsymbol{x}}\right) &> u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{c}} + u_{\boldsymbol{b},\boldsymbol{x}}\right), \\
u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{c}} + u_{\boldsymbol{a},\boldsymbol{y}}\right) &> u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{c}} + u_{\boldsymbol{b},\boldsymbol{y}}\right), \\
u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{x}} + u_{\boldsymbol{b},\boldsymbol{y}}\right) &> u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{x}} + u_{\boldsymbol{a},\boldsymbol{y}}\right).
\end{aligned}$$

Summing up the first two inequalities and then applying the third one yields

$$\begin{aligned}
&2u_{\boldsymbol{a}} + \frac{1}{3}\left(2u_{\boldsymbol{a},\boldsymbol{b}} + 2u_{\boldsymbol{a},\boldsymbol{c}} + u_{\boldsymbol{a},\boldsymbol{x}} + u_{\boldsymbol{a},\boldsymbol{y}}\right) \\
&> u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + u_{\boldsymbol{b},\boldsymbol{x}} + u_{\boldsymbol{b},\boldsymbol{y}}\right) + u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + 2u_{\boldsymbol{b},\boldsymbol{c}}\right) \\
&> u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + u_{\boldsymbol{a},\boldsymbol{x}} + u_{\boldsymbol{a},\boldsymbol{y}}\right) + u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + 2u_{\boldsymbol{b},\boldsymbol{c}}\right),
\end{aligned}$$

from which we obtain via subtracting common terms

$$u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + 2u_{\boldsymbol{a},\boldsymbol{c}}\right) > u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + 2u_{\boldsymbol{b},\boldsymbol{c}}\right). \tag{17}$$

Exactly the same argumentation (with the roles of $\boldsymbol{a}$ and $\boldsymbol{b}$ interchanged and $\boldsymbol{x}$ resp. $\boldsymbol{y}$ replaced by $\boldsymbol{x}'$ resp. $\boldsymbol{y}'$) lets us infer from (16)

$$u_{\boldsymbol{b}} + \frac{1}{3}\left(u_{\boldsymbol{b},\boldsymbol{a}} + 2u_{\boldsymbol{b},\boldsymbol{c}}\right) > u_{\boldsymbol{a}} + \frac{1}{3}\left(u_{\boldsymbol{a},\boldsymbol{b}} + 2u_{\boldsymbol{a},\boldsymbol{c}}\right),$$

which contradicts (17). This completes the proof. □

Note that a limited expressivity should not necessarily be seen as a negative property. In particular, from a machine learning perspective, an overly excessive expressivity (or *capacity* of the underlying hypothesis space) is connected with the practical problem of poor generalization due to overfitting, i. e., being overly expressive may prevent the learner from identifying the right model. In any case, we expect FETA to work well for all choice functions that (approximately) decompose into a pairwise relation between objects. Naturally, this leads to the question whether it is possible to incorporate more of the set-based context without ultimately increasing computational complexity. This question motivated our next decomposition.

4.2 First Aggregate Then Evaluate

To deal with the problem of task contexts of variable size, our previous approach was to decompose the context into sub-contexts of a fixed size, evaluate an object $\boldsymbol{x}$ in each of the sub-contexts, and then aggregate these evaluations into an overall assessment. An alternative to this FETA strategy, and in a sense contrariwise approach, consists of first aggregating the task into a representation of fixed size, and then evaluating the object $\boldsymbol{x}$ in the presence of this task representative.

More specifically, the FATE approach requires a mapping ϕ from $\mathcal{X}$ to some m-dimensional embedding space $\mathcal{Z} \subseteq \mathbb{R}^m$ as well as a context-dependent *sub-utility function* $U' \colon \mathcal{X} \times \mathcal{Z} \longrightarrow \mathbb{R}$. To evaluate an object $\boldsymbol{x}$ in a choice task $Q \in \mathcal{Q}$, the FATE strategy first computes $\frac{1}{|Q|} \sum_{\boldsymbol{y} \in Q} \phi(\boldsymbol{y})$ as representative for the task and then evaluates it via U' as

$$U(\boldsymbol{x}, Q) \coloneqq U'\left(\boldsymbol{x}, \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{y} \in Q} \phi(\boldsymbol{y})\right). \tag{18}$$

We call this U the *FATE utility function with sub-utility function U' and transformation ϕ* and denote it by $U_{\text{FATE}}^{U',\phi}$.

Example 4.5 (FATE: Context-dependence) Similar as in Example 4.1, suppose $\mathcal{X}$ to consist of four elements $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$, let $\mathcal{Z} \coloneqq \mathbb{R}$ and $\phi \colon \mathcal{X} \longrightarrow \mathcal{Z}$ and $U' \colon \mathcal{Z} \longrightarrow \mathbb{R}$ be such that

$\mathcal{X}$	$\phi(\cdot)$	$U'(\cdot, 2)$	$U'(\cdot, 3)$
$\boldsymbol{a}$	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 $U'(\cdot, z)$ 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 $U(\boldsymbol{a}, Q_1) = U'(\boldsymbol{a}, 2) = 0.5 > -0.1 = U'(\boldsymbol{b}, 2) = U(\boldsymbol{b}, Q_1)$ and at the same time $U(\boldsymbol{a}, Q_2) = U'(\boldsymbol{a}, 3) < U'(\boldsymbol{b}, 3) = U(\boldsymbol{b}, Q_2)$, i. e., the preference between $\boldsymbol{a}$ and $\boldsymbol{b}$ changes depending on whether the third item in the set is $\boldsymbol{c}$ or $\boldsymbol{d}$.

This approach is related to recent advances on dealing with set-valued inputs in neural networks [126, 90, 8], where a permutation-equivariant network directly maps from sets of objects to scores. Rosenfeld, Oshiba, and Singer [95] propose to learn set-dependent aggregation functions with an inductive bias towards principles from behavioral choice theory. They note that general models like Deep Sets [126], which try to approximate set functions using a permutation-invariant neural network, are overly general, because they have a high *violation capacity*, i. e., the flexibility of the model to change its choices, when objects are removed from the choice task. The FATE approach on the other hand first condenses the task context into a representative 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.

Without further assumptions on ϕ and U', this model is able to express any possible choice function c on $\mathcal{Q}$, as we show in the following. The proof of the upcoming result is similar to the proof of Theorem 2 by Zaheer et al. [126].

Proposition 4.6 *Suppose $\mathcal{X}$ to be countable and $\mathcal{Q} \subseteq \{Q \subseteq \mathcal{X} : |Q| < \infty\}$. There exists a parametrization $\phi \colon \mathcal{X} \longrightarrow \mathbb{R}$ with the following property:*

(i) For any singleton choice function c on $\mathcal{Q}$, there is a utility function $U_c' \colon \mathcal{X} \times \mathbb{R} \longrightarrow \mathbb{R}$ such that $C_{\text{singleton}}(U_{\text{FATE}}^{U_c',\phi}, Q) = c(Q)$ holds for any $Q \in \mathcal{Q}$.

(ii) For any subset choice function c on $\mathcal{Q}$ there exists a utility function $U_c' \colon \mathcal{X} \times \mathbb{R} \longrightarrow \mathbb{R}$ with $C_{\text{subset}}^{1/2}(U_{\text{FATE}}^{U_c',\phi}, Q) = c(Q)$ for any $Q \in \mathcal{Q}$.

Proof. Since $\mathcal{X}$ is countable, there exists an injective function $\delta \colon \mathcal{X} \longrightarrow \mathbb{N}$. 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

$$\Phi \colon \mathcal{Q} \longrightarrow \mathbb{R}, \quad Q \mapsto \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{x} \in Q} \phi(\boldsymbol{x})$$

is injective. For this, let $Q, Q' \in \mathcal{Q}$ with $\Phi(Q) = \Phi(Q')$. Then,

$$\frac{|Q'|}{|Q|} = \frac{\ln\left(\prod_{\boldsymbol{x}\in Q'} p_{\delta(\boldsymbol{x})}\right)}{\ln\left(\prod_{\boldsymbol{x}\in Q} p_{\delta(\boldsymbol{x})}\right)} = \log_b(a)$$

holds for the integers $a \coloneqq \prod_{\boldsymbol{x}\in Q'} p_{\delta(\boldsymbol{x})}$ and $b \coloneqq \prod_{\boldsymbol{x}\in Q} p_{\delta(\boldsymbol{x})}$, 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 $\mathcal{Q}$ to be fixed. Since Φ from above is injective, there exists a mapping $\Psi\colon \mathbb{R} \longrightarrow \mathcal{Q}$ such that $\Psi(\frac{1}{|Q|}\sum_{\boldsymbol{x}\in Q}\phi(\boldsymbol{x})) = Q$ 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

$$U'_c(\boldsymbol{x}, z) \coloneqq [\![\boldsymbol{x} \in c(\Psi(z))]\!].$$

□

Although this expressivity is desirable in general, it comes at a cost. The FATE model $U_{\text{FATE}}^{(\phi,U')}$ as such is *not identifiable*: For example, suppose $U'\colon \mathcal{X}\times\mathcal{Z} \longrightarrow \mathbb{R}$ is of the form $U'(\boldsymbol{x}, z) \coloneqq f(\boldsymbol{x}) + \|z\|_2$ for some function $f\colon \mathcal{X} \longrightarrow \mathbb{R}$, where $\|\cdot\|_2$ denotes the standard euclidean norm in $\mathbb{R}^d \supseteq \mathcal{Z}$. For arbitrary $\phi_1\colon \mathcal{X} \longrightarrow \mathcal{Z}$, we obtain with $\phi_2 \coloneqq -\phi_1$ that

$$U_{\text{FATE}}^{(\phi_1,U')}(\boldsymbol{x}, Q) - U_{\text{FATE}}^{(\phi_2,U')}(\boldsymbol{x}, Q) = \left\|\frac{1}{|Q|}\sum\nolimits_{\boldsymbol{y}\in Q}\phi_1(\boldsymbol{x})\right\|_2 - \left\|-\frac{1}{|Q|}\sum\nolimits_{\boldsymbol{y}\in Q}\phi_1(\boldsymbol{x})\right\|_2 = 0$$

for any $Q \in \mathcal{Q}, \boldsymbol{x} \in Q \subseteq \mathcal{X}$, i. e., $U_{\text{FATE}}^{(\phi_1,U')} = U_{\text{FATE}}^{(\phi_2,U')}$ holds.

4.3 Linear Sub-Utility Functions

A related question concerns the expressivity of the FATE and FETA approaches, when the underlying sub-utility functions and transformations are linear functions. In case ϕ and U' are chosen as linear functions in the sense that $\phi(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ and $U'(\boldsymbol{x}, z) = \boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t z$ for any $(\boldsymbol{x}, z) \in \mathcal{X}\times\mathcal{Z}$ and some $\boldsymbol{A} \in \mathbb{R}^{d\times m}$, $\boldsymbol{c} \in \mathbb{R}^d$ and $\boldsymbol{d} \in \mathbb{R}^m$, (18) takes the form

$$U_{\text{FATE}}^{U',\phi}(\boldsymbol{x}, Q) = \boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t\left(\frac{1}{|Q|}\sum\nolimits_{\boldsymbol{y}\in Q}\boldsymbol{A}\boldsymbol{y}\right).$$

As the second summand therein does not depend on $\boldsymbol{x}$, for any $Q \in \mathcal{Q}$, the singleton choice $C_{\text{singleton}}(U_{\text{FATE}}^{U',\phi}, Q)$ is the same as that corresponding to the linear utility function $\boldsymbol{x} \mapsto \boldsymbol{c}^t\boldsymbol{x}$ and thus independent of the context Q. Consequently, at least one of U' and ϕ has to be non-linear in order to model context-dependent choices.

In contrast to this, for the case of FETA, linearity of the sub-utility functions does not imply context-independence of the model: If U_0 and U_1 are linear in the sense that $U_0(\boldsymbol{x}) = \boldsymbol{b}^t\boldsymbol{x}$ and $U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) = \boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t\boldsymbol{y}$ for any distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ and some weight vectors $\boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^d$, the FETA utility function with sub-utility functions U_0, U_1 is given as

$$\begin{aligned} U(\boldsymbol{x}, Q) &= \boldsymbol{b}^t\boldsymbol{x} + \frac{1}{|Q|-1}\sum\nolimits_{\boldsymbol{y}\in Q\setminus\{\boldsymbol{x}\}}\left(\boldsymbol{c}^t\boldsymbol{x} + \boldsymbol{d}^t\boldsymbol{y}\right) \\ &= (\boldsymbol{b}+\boldsymbol{c})^t\boldsymbol{x} + \frac{1}{|Q|-1}\sum\nolimits_{\boldsymbol{y}\in Q\setminus\{\boldsymbol{x}\}}\boldsymbol{d}^t\boldsymbol{y} \end{aligned}$$

for any $\boldsymbol{x} \in Q \in \mathcal{Q}$. As the second summand therein depends not only on Q but also on $\boldsymbol{x}$, U can in general **not** be represented as a linear function.

5 Implementation Using Neural Networks

Having defined the decomposition strategies FETA and FATE in the preceding section, we are still missing an algorithm, which can actually learn the utility functions involved. In this section, we propose realizations of the FETA and FATE approaches in terms of neural network architectures FETA-Net and FATE-Net, respectively. Our design goals for both neural networks are twofold. First, they should be end-to-end trainable using (stochastic) gradient descent, such that they can be used as part of a larger neural network architecture. To this end, we ensure that the outputs of the networks are differentiable almost everywhere with respect to the weights. Similarly, 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.

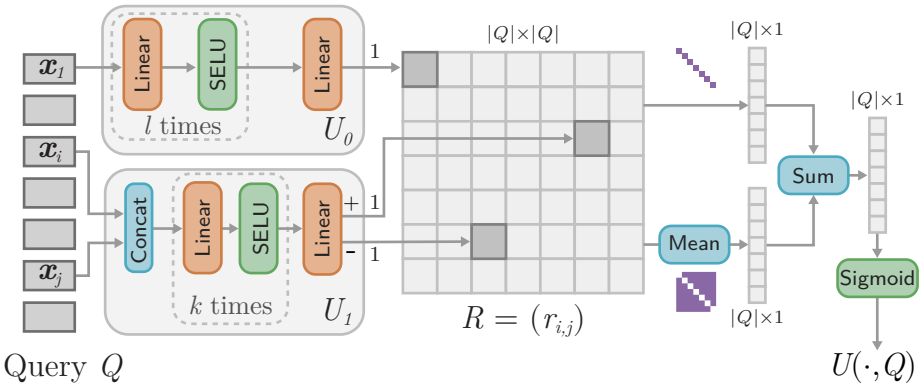


Figure 3: The FETA-NET architecture implementing the FETA approach. Layers with trainable weights are shown in orange, while operations without trainable weights are drawn in blue and non-linearities are depicted in green.

5.1 FETA-NET *Architecture*

We will now describe our first neural network architecture FETA-NET and its training. Recall from Section 4.1 that we seek to predict utility scores $U(\boldsymbol{x}_i, Q)$ of the form (11) for every object $\boldsymbol{x}_i \in Q$. What we need to learn, therefore, is the functions U_0 and U_1. In FETA-NET, we do so by means of a deep neural network architecture (shown in Figure 3). The network is trained in a set of data $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^N$, where each Q_i is a choice task and $C_i \in 2^{Q_i} \setminus \{\emptyset\}$ the choice set observed for that task.

The main component is the neural network tasked with learning the pairwise utility function U_1 (depicted in blue). It receives the feature vectors of two objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and outputs a score for $\boldsymbol{x}_i$ in the presence of object $\boldsymbol{x}_j$. To build up the complete matrix $R = (r_{i,j})$ would require iterating over all pairs of objects in Q. This is why we choose to adopt the CmpNN approach by Rigutini et al. [93] for the pairwise scoring function, i. e., instead of one output neuron we utilize two U_1^+ and U_1^-. Weight sharing ensures that $U_1^+(\boldsymbol{x}_i, \boldsymbol{x}_j) = U_1^-(\boldsymbol{x}_j, \boldsymbol{x}_i)$ and $U_1^-(\boldsymbol{x}_i, \boldsymbol{x}_j) = U_1^+(\boldsymbol{x}_j, \boldsymbol{x}_i)$ holds. For the diagonal, we evaluate a separate network $U_0(\boldsymbol{x}_i)$, which learns a latent utility component for each object (corresponding to the case $k = 0$ in (10)). With that it suffices to iterate over all combinations of objects once, and to construct the matrix R as follows:

$$r_{i,j} = \begin{cases} U_1^+(\boldsymbol{x}_i, \boldsymbol{x}_j) & \text{if } i < j \\ U_1^-(\boldsymbol{x}_i, \boldsymbol{x}_j) & \text{if } i > j \\ U_0(\boldsymbol{x}_i) & \text{otherwise} \end{cases} \tag{19}$$

Then, each row of the relation R is averaged to obtain a score $U(\boldsymbol{x}_i, Q) = r_{i,i} + \frac{1}{|Q|-1} \sum_{1 \le j \ne i \le |Q|} r_{i,j}$ for each object $\boldsymbol{x}_i \in Q$. Therefore, the network U_1 is a mapping $\mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^2$ and U_0 a mapping $\mathbb{R}^d \longrightarrow \mathbb{R}$ which can be instantiated by any neural network architectures suitable for the given objects. For our experiments later on, we shall use deep, densely connected networks. We treat the number of layers and units as hyperparameters and optimize them jointly with all the other hyperparameters.

The complete training algorithm for FETA-NET is shown in Algorithm 1, which is an instantiation of stochastic gradient descent. We will denote the weight vectors of the networks U_0 and U_1 by θ_0 and θ_1, respectively. In the beginning, these weight vectors are suitably initialized in order to avoid exploding/vanishing gradients [42, 48]. In each epoch, the algorithm shuffles the given dataset and constructs mini-batches $\mathcal{B}_1, \ldots, \mathcal{B}_T$ with $\mathcal{B}_i \subset \mathcal{D}$ for all $i \in [T]$. In lines 10 to 18, the pairwise relation is constructed as described above. The utilities $\boldsymbol{u} = (u_1, \ldots u_{|Q|})$ for the objects inside the task Q are computed in line 19 by summing the pairwise relation $r_{i,j}$ across the columns of the matrix. Finally, the loss is computed in line 20 and added to the cumulative loss for the batch. The weight vectors θ_0 and θ_1 are updated using backpropagation in lines 22–23.

It is easy to see, that the training runtime complexity per epoch (including backpropagation) of FETA-NET is $O\left(Ndq^2\right)$, where N denotes the number of instances, d is the number of features per object, and $q := \max_{(Q,Y)\in\mathcal{D}} |Q|$ is an upper bound on the number of objects in each choice task. For a new task Q, the prediction time is in $O\left(d|Q|^2\right)$.

Algorithm 1 FETA-NET training algorithm

Require:
Dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^{N}$ with $Q_i \subset \mathbb{R}^d$
Pairwise network $U_1 \colon \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^2$, parametrized by θ_1
Diagonal network $U_0 \colon \mathbb{R}^d \longrightarrow \mathbb{R}$, 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 $L \colon \mathcal{C} \times \bigcup_{k\in\mathbb{N}} \mathbb{R}^k \longrightarrow \mathbb{R}$

procedure TRAIN-FETA-NET($\mathcal{D}, U_0, U_1, b, E, \boldsymbol{\eta}, L$)
  Initialize random weight vectors θ_0, θ_1
  for Epoch $ep \in [E]$ **do**
    $\mathcal{D} \leftarrow$ SHUFFLE($\mathcal{D}$)
    $T \leftarrow \left\lceil \frac{N}{b} \right\rceil$
    Construct mini-batches $\mathcal{B}_1, \dots, \mathcal{B}_T$
    for Iteration $t \in [T]$ **do**
      $\ell_t \leftarrow 0$
      for all $(Q, C) \in \mathcal{B}_t$ **do**
        for $1 \le i \le j \le |Q|$ **do**
          if $i < j$ **then**
            $r^{\text{tmp}} \leftarrow U_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$
            $r_{i,j} \leftarrow r_0^{\text{tmp}}, \quad r_{j,i} \leftarrow r_1^{\text{tmp}}$
          else
            $r_{i,i} \leftarrow U_0(\boldsymbol{x}_i)$
        $\boldsymbol{u} \leftarrow \left(r_{i,i} + \frac{1}{|Q|-1} \sum_{1\le j\ne i\le |Q|} r_{i,j}\right)_{i=1}^{|Q|}$
        $\ell_t \leftarrow \ell_t + L(C, \boldsymbol{u})$
      $\theta_0 \leftarrow \theta_0 - \frac{\eta_{ep\cdot T+t}}{|\mathcal{B}_t|} \nabla_{\theta_0} \ell_t$
      $\theta_1 \leftarrow \theta_1 - \frac{\eta_{ep\cdot T+t}}{|\mathcal{B}_t|} \nabla_{\theta_1} \ell_t$

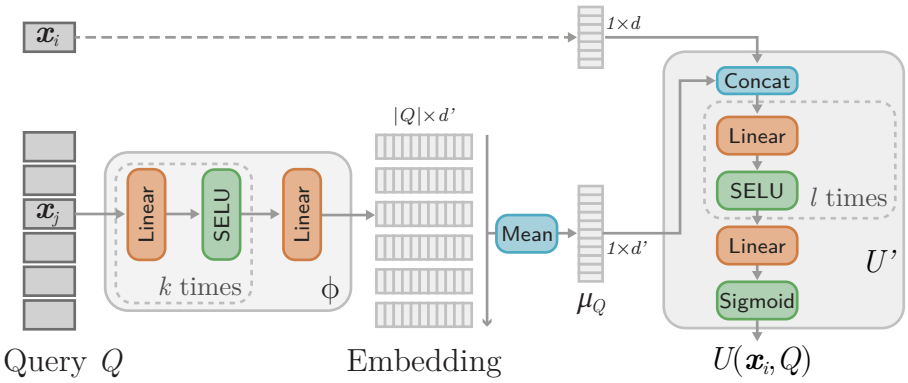


Figure 4: The FATE-NET architecture implementing the FATE approach. Here we show the score head for object $\boldsymbol{x}_i$. Layers with trainable weights are shown in orange, while operations without trainable weights are drawn in blue and non-linearities are depicted in green.

5.2 FATE-NET *Architecture*

The second architecture we propose is called FATE-NET, and the structure for predicting the score for one object is depicted in Figure 4. Inputs are the n objects of the task $Q = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ (shown in green). Each object is independently passed through a deep, densely connected embedding layer (shown in blue). The embedding layer approximates the function ϕ in (18) and is a map $\mathbb{R}^d \longrightarrow \mathbb{R}^{d'}$. Note that we employ weight sharing, i. e., the same embedding is used for each object. Then, the representative μ_Q for the task Q is computed by averaging the representations of each object. To calculate the score $U(\boldsymbol{x}_i, \mu_Q)$ for an object $\boldsymbol{x}_i$, the feature vector is concatenated with μ_Q to form the input to the final joint neural network layers (here depicted in orange). Again, weight sharing is used to learn only one scoring network.

Algorithm 2 FATE-NET training algorithm

Require:
Dataset $\mathcal{D} = \{(Q_i, C_i)\}_{i=1}^{N}$ with $Q_i \subset \mathbb{R}^d$
Embedding network $\phi\colon \mathbb{R}^d \longrightarrow \mathbb{R}^{d'}$, parametrized by θ_ϕ
Utility network $U\colon \mathbb{R}^d \times \mathbb{R}^{d'} \longrightarrow \mathbb{R}$, 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 $L\colon C \times \bigcup_{k\in\mathbb{N}} \mathbb{R}^k \longrightarrow \mathbb{R}$

1: **procedure** TRAIN-FATE-NET($\mathcal{D}, U_0, U_1, b, E, \boldsymbol{\eta}, L$)
2: Initialize random weight vectors θ_ϕ, θ_U
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: $\mu_Q \leftarrow \frac{1}{|Q|} \sum_{\boldsymbol{x}\in Q} \phi(\boldsymbol{x})$
11: $\boldsymbol{u} \leftarrow (U(\boldsymbol{x}, \mu_Q))_{\boldsymbol{x}\in Q}$
12: $\ell_t \leftarrow \ell_t + L(C, \boldsymbol{u})$
13: $\theta_U \leftarrow \theta_U - \frac{\eta_{ep\cdot T+t}}{|\mathcal{B}_t|} \nabla_{\theta_U} \ell_t$
14: $\theta_\phi \leftarrow \theta_\phi - \frac{\eta_{ep\cdot T+t}}{|\mathcal{B}_t|} \nabla_{\theta_\phi} \ell_t$

For both neural networks, we treat the number of layers, units and embedding dimensions as hyperparameters, which are to be optimized.

The detailed training algorithm is shown in Algorithm 2. As mentioned before for FETA-NET, it is an instantiation of stochastic gradient descent. We will denote the weight vectors of the networks U and ϕ by θ_U and θ_ϕ, respectively. The initialization of the weight vectors and the construction of the mini-batches (lines 2–6) is again the same as for FETA-NET. In line 10, the representative object μ_Q is constructed by first mapping each object to the embedding space using ϕ_{θ_ϕ}, and then computing the centroid of the embedded points. The embedding network can be any network that receives an object and returns a d'-dimensional real-valued vector, and should be adapted to the data at hand. The utility scores $\boldsymbol{u}$ are then computed by evaluating each object $\boldsymbol{x} \in Q$ in conjunction with the representative point μ_Q (see line 11). The cumulative loss for the mini batch is updated in line 12. The weight vectors θ_ϕ and θ_U are updated by calculating the gradient of the loss using backpropagation and scaling it by an appropriate learning rate (lines 14–13).

The training runtime complexity per epoch of FATE-NET (including backpropagation) is $O\left(Ndq^2\right)$, where N denotes the number of choice tasks, d is the number of features per object, and q is an upper bound on the number of objects in each task. For a new choice task Q, the prediction can be done in only $O(d|Q|)$ time (i. e., *linear* in the number of objects). This is due to the fact that μ_Q only needs to be computed once.

6 EMPIRICAL EVALUATION

The main goal of our empirical evaluation is to find out for which kind of problems FATE-NET and FETA-NET work well. Moreover, we wish to compare these approaches with existing methods for ranking and choice. In particular, the following questions will be addressed:

- Are the decompositions FATE and FETA suitable for learning context-dependent choice functions?
- How important is (i) the complexity/expressiveness of the underlying model class and (ii) its ability to model context-dependent choice functions, and how do these two factors interact? For example, are deep neural networks (i. e. FATE-NET and FETA-NET) really needed, or would a simpler (e. g. linear) model also suffice? Can the additional complexity/expressiveness compensate for the inability to model context-dependent choice functions?
- To what extent is our approach able to generalize over the task size? For example, is it possible to produce accurate predictions on tasks of a specific size, even if that size has never occurred in the training data?

For the first two questions, we evaluate the approaches on a variety of general choice and singleton choice problems. We also introduce the variant FETA-LINEAR, which learns the FETA decomposition using only linear functions, to ascertain whether it is able to account for some of the context-effects present in the data.

In addition, we evaluate the performance of different logit models used in economics: multinomial logit (MNL) [75], nested logit (NL) [123], generalized nested logit (GNL) [122] and mixed logit (ML) [110]. The first logit model is the MNL model (referred as GENLINEARMODEL for subset choice task), which assumes that the choice between two objects does not depend on other objects in the set [67]. The NL and GNL belong to the generalized extreme value (GEV) class of models that learn correlations amongst the objects in the given set, which implicitly accounts for some of the context effects, but mainly the similarity effect [9, 113]. GEV models allocate the objects in the given task Q into different sets called nests and learn correlations between the objects inside each nest [122, 110]. These nests are disjoint in case of NL [123]. GNL is the most general model of this class, which allows the fractional allocation of each object in Q to each nest and it learns the correlation between them [122]. ML estimates the choice probability as a mixture of multiple logits [76, 125].

Another model which was proposed for solving the task of singleton choice is the PAIRWISESVM, which makes use of induced pairwise preferences to fit a linear model [32, 68].

As a recent context-dependent baseline model, we implement the set-dependent aggregation (SDA) approach by Rosenfeld, Oshiba, and Singer [95]. We also implement the RANKNET model as an additional context-independent baseline, which learns a non-linear utility for each object by converting them to pairwise preferences [107, 18]. Due to a lack of algorithms specifically designed for the subset choice problem, we employ the same thresholding of the utilities described in (4) we use for our approaches. The threshold is tuned on a small validation set for all approaches, using the F_1-score as target loss (see Appendix C for details).

All in all, we compare to both deep neural networks and linear models, so that we have baselines of varying representative power, which helps to contextualize the performance of our approaches on each dataset. Finally, to answer the third question, we train the different models on a fixed task size and predict on queries of deviating size.

6.1 Setup

All experiments are implemented in Python, and the code and the dataset generators are publicly available[5]. To properly compare all models in a fair and unbiased way, we make sure to optimize the hyperparameters of each model by employing Bayesian optimization in a nested validation loop (we use the Gaussian process based implementation in scikit-optimize [49]) The final out-of-sample estimates are then computed using another outer cross-validation loop with the best hyperparameters found in each fold. The loss functions and the datasets considered throughout our empirical evaluation are introduced in the following two subsections, respectively (see Appendix C for more details).

The experiments were run on a compute cluster with a mix of NVIDIA GTX 1080 Ti and RTX 2080 Ti GPUs (on average 15-20) and Intel Xeon E5-2670 processors. One job consisting of one outer split with complete hyperparameter optimization on the validation set took on average 8 hours. The training of FATE-NET and FETA-NET on average (across datasets) required 11 hours. Combined, all experiments took roughly 11 400 GPU hours and 6000 CPU hours.

6.2 Loss Functions

As explained in Section 4, our goal during learning is to minimize a suitable target loss $L\colon \mathcal{C} \times \mathcal{C} \longrightarrow \mathbb{R}$. This is usually the loss one is interested in minimizing, e. g., the F_1-measure in our case. Since these losses are usually not differentiable, they cannot readily be used in a gradient descent algorithm. Therefore, during training we opt to minimize surrogate losses which are differentiable almost everywhere instead. In this section, we will first introduce the target losses we consider (cf. Section 6.2.1). We then derive surrogate losses based on the probabilistic choice models introduced in Section 3 and based on practical considerations (cf. Section 6.2.2).

6.2.1 Target Loss Functions

The canonical loss function, which we focus on in the singleton choice setting, is the *categorical 0/1-loss*

$$L_{0/1}(C, C') \coloneqq [\![C \neq C']\!], \tag{20}$$

i. e., in case the ground-truth choice C is $\{\boldsymbol{x}\}$, each false prediction $C' \neq \{\boldsymbol{x}\}$ is penalized with a loss of 1. In addition, we will call the quantity $1 - L_{0/1}(C, C')$ the *categorical accuracy*. Moving from singleton to subset choice, where C and C' can now be choice sets of arbitrary size, the same loss function (20) can still be used. To signify that it is used in subset

[5] https://github.com/kiudee/cs-ranking

choice, we will call it the *subset* 0/1*-loss*. Targeting the subset 0/1-loss is problematic, especially whenever a task Q contains many objects, since already one incorrectly predicted object results in the whole prediction being declared incorrect. One could instead opt to consider the average of the item-wise 0/1-loss, which is called the Hamming loss in the setting of multi-label classification [54]. However, this loss exhibits some properties that could be questioned in the context of choice. In particular, the non-prediction of a selected item (false negative) is penalized in the same way as the prediction of a non-selected item (false positive), although positives and negatives might be highly imbalanced.

A more suitable measure, which is widely used in classification, is the F_1*-measure* defined as

$$F_1(C, C') := \frac{2 \sum_{\boldsymbol{x} \in \mathcal{X}} [\![\boldsymbol{x} \in C \cap C']\!]}{\sum_{\boldsymbol{x} \in \mathcal{X}} [\![\boldsymbol{x} \in C]\!] + \sum_{\boldsymbol{x} \in \mathcal{X}} [\![\boldsymbol{x} \in C']\!]} \tag{21}$$

for any $C, C' \in \mathcal{C}$. This measure takes values in $[0, 1]$ and large values indicate conformity between C and C', whence an appropriate loss can be defined as [6]

$$L_{F_1}(C, C') := 1 - F_1(C, C').$$

In spite of the existence of other measures that specifically aim at correctly predicting positives, such as the informedness [88, 87], we will mostly focus on L_{F_1} as the target loss, because it is well known and commonly used as a performance metric. That means that we will use it as the validation loss for the Bayesian hyperparameter optimization we run for every learner. Additional evaluation measures we report are described in Appendix B.

6.2.2 *Surrogate Losses*

The probabilistic setting for choice that we introduced in Section 3 suggests a natural approach to learning and prediction:

- First, a learner is trained using the log-likelihood of the probabilistic model as a loss function. This loss function is not only differentiable, but also calibrated in the sense of being minimized by the true (conditional) probabilities. In other words, a learner trained with this loss is supposed to predict (unbiased) probabilities on the choice space $\mathcal{C}$ (conditioned on the query).
- Thus, given a query for which a prediction is sought, a probability distribution on the choice space $\mathcal{C}$ can be obtained as a prediction, which in turn allows for minimizing any target loss in expectation.

More specifically, let $U(\cdot, Q)$ denote the latent utility scores $U(\boldsymbol{x}, Q)$, $\boldsymbol{x} \in Q$, predicted by a learner on a query $Q \in \mathcal{Q}$. In a singleton choice scenario, where the data is supposed to be generated according to choice probabilities $p^{\tilde{U}}_{\mathrm{MNL}}(\boldsymbol{x} \mid Q) = p_{\mathrm{MNL}}(\boldsymbol{x} \mid Q)$ of the form (5) for some unknown ground-truth $\tilde{U}$, one may define the corresponding categorical cross-entropy loss gained when observing $C = \{\boldsymbol{x}\} \in \mathcal{C}$

$$\begin{aligned} L_{\mathrm{CE}}(\{\boldsymbol{x}\}, U(\cdot, Q)) &:= -\log\left(p^{U}_{\mathrm{MNL}}(\boldsymbol{x} \mid Q)\right) \\ &= \log\Big(\sum\nolimits_{\boldsymbol{y} \in Q} \exp(U(\boldsymbol{y}, Q))\Big) - U(\boldsymbol{x}, Q)\,. \end{aligned} \tag{22}$$

This expression is minimized in case $\boldsymbol{x} = \arg\max_{\boldsymbol{y} \in Q} U(\boldsymbol{y}, Q)$.

If dealing with subset choice data that is presumably sampled according to the choice probability distribution $p^U(C \mid Q) = p(C \mid Q)$ from (6), it is natural to measure prediction $C \in 2^Q \setminus \{\emptyset\}$ by means of the corresponding binary cross-entropy loss

$$\begin{aligned} L_{\mathrm{BE}}(C, U(\cdot, Q)) &:= -\log\left(p^{U}(C \mid Q)\right) \\ &= \sum\nolimits_{\boldsymbol{y} \in Q} \log\left(1 + \exp(U(\boldsymbol{y}, Q))\right) - [\![\boldsymbol{y} \in C]\!] U(\boldsymbol{y}, Q). \end{aligned} \tag{23}$$

In spite of the theoretical justification of the logistic losses discussed above, we found that "hinge-variants" of the respective 0/1-losses may sometimes lead to more stable results. More specifically, for the singleton choice setting *categorical hinge loss* defined via

$$L_{\mathrm{CH}}(\{\boldsymbol{x}\}, U(\cdot, Q)) := \max\Big(1 + \max\nolimits_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} U(\boldsymbol{y}, Q) - U(\boldsymbol{x}, Q), 0\Big), \tag{24}$$

for any $\boldsymbol{x} \in Q \in \mathcal{Q}$, is inspired by the hinge loss used in multi-class classification [29, 78] and can be used instead of (22).

Finally, for training FATE-NET and FETA-NET in the experiments below, we use the *binary cross-entropy* loss for the subset choice setting and the *categorical hinge* loss for the singleton choice setting, since these turned out to work well in preliminary experiments. In addition, an L_2-regularization term for the magnitude of the weights is added and optimized as part of the loss during training.

[6] Later on, we will nevertheless report the F_1-measure itself, which is common practice in machine learning.

Table 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]

Convexity of the Surrogate Losses An important consideration for the surrogate losses to be used during training is whether they are convex with respect to the utility scores $U(\boldsymbol{x}, Q)$. All three losses introduced above are indeed convex. To see this for L_{CE}, notice that (22) can equivalently be written as $\log(\sum_{\boldsymbol{y} \in Q} \exp(U(\boldsymbol{y}, Q) - U(\boldsymbol{x}, Q)))$. The inner difference of utilities is linear and therefore convex. The outer function is also known as *LogSumExp* and is defined via $\mathrm{LSE}(\boldsymbol{x}) \coloneqq \log(\sum_{j \in [m]} \exp(x_j))$. It is convex and since it is also strictly decreasing in each argument, the composition (22) is convex as well.

As for the binary cross-entropy L_{BE}, note that the inner function $s \colon \mathbb{R} \longrightarrow \mathbb{R}$, $s(x) \coloneqq \log(1 + \exp(x))$ of (23) is smooth with strictly positive first and second derivatives and hence convex and non-decreasing. Similarly, $\tilde{s}(x) \coloneqq s(x) - x$ is convex and strictly decreasing on $\mathbb{R}$. Hence, we can conclude that (23) is convex.

Finally, the categorical hinge (24) contains the function $h \colon \mathbb{R}^m \longrightarrow \mathbb{R}$, $\boldsymbol{x} \mapsto \log(\sum_{j \in [m]} \exp(x_j - x_i))$, which is convex as the logarithm of a maximum of convex functions. Since $s \colon \mathbb{R} \longrightarrow \mathbb{R}$, $x \mapsto \max(1 + x, 0)$ is convex and non-decreasing, $s \circ h$ and therefore (24) is convex as well.

The FETA model further decomposes $U(\boldsymbol{x}, Q)$ into an aggregation of sub-utility functions U_0 and U_1. It is therefore interesting to ask whether the surrogate losses are also convex with respect to the sub-utility values $U_0(\boldsymbol{x})$, $U_1(\boldsymbol{x}, \{\boldsymbol{y}\})$. We can answer this question in the affirmative, since the FETA utility values are positively weighted sums of these sub-utility scores.

However, the overall learning problem depends on the parameter θ of the realization of U_{FETA} and U_{FATE} and the corresponding loss function can possibly still be non-convex w.r.t. θ (as this is the case with the neural networks employed here). That means in practice we lose the guarantee of stochastic gradient descent to find a global optimum, but with careful tuning of the optimization process one can still expect to find reasonable solutions.

6.3 Datasets

We now introduce the learning problems used for the empirical comparison as follows:

(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.

(d) Different choice problems defined on the well-known MNIST dataset.

(e) Similarity/dissimilarity-based movie selection using the MovieLens Tag Genome dataset [118].

(f) The LEarning TO Rank (LETOR) MQ2007 and MQ2008 datasets [89] consisting of query-document pairs, with the goal to select 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 [33].

(h) The Sushi dataset, where the task is to choose the most preferred sushi from a set of 10 options provided to a user.

See Table 1 for an overview of the datasets and their properties. In the following sections, we will describe the different datasets, their motivation, and if applicable, how they are generated.

6.3.1 The Medoid Problem

The motivation for this problem is the general idea of learning to choose a most representative element from a set. More concretely, the medoid of a set is the object with the smallest cumulative dissimilarity to all other objects of the set[7]. It is commonly used as a representative element, especially for structured objects such as graphs, 2-D trajectories, images, etc. [116, 127].

Formally, we are interested in learning the choice function $c_{\text{medoid}} \colon \mathcal{Q} \longrightarrow C$ given as

$$c_{\text{medoid}}(Q) \coloneqq \arg\min_{\boldsymbol{x} \in Q} \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{y} \in Q} \|\boldsymbol{x} - \boldsymbol{y}\|,$$

where we write here and throughout the remainder of this paper $\|\cdot\|$ for the standard euclidean norm defined as $\|z\| = \sqrt{z^t z}$. The singleton choice produced by this procedure incorporates all pairwise distances among the objects, which makes it a good context-dependent learning problem to investigate. In particular, c_{medoid} is sensitive to changes of the elements in the task. With $U_0(\boldsymbol{x}) \coloneqq 0$ and $U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \coloneqq -\|\boldsymbol{x} - \boldsymbol{y}\|$ we clearly have

$$\begin{aligned} c_{\text{medoid}}(Q) &= \arg\min_{\boldsymbol{x} \in Q} \frac{1}{|Q| - 1} \sum_{\boldsymbol{y} \in Q} \|\boldsymbol{x} - \boldsymbol{y}\| \\ &= \arg\max_{\boldsymbol{x} \in Q} U_0(\boldsymbol{x}) + \frac{1}{|Q| - 1} \sum_{\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}} U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \end{aligned}$$

and thus $U_{\text{FETA}}^{U_0, U_1}$ is able to exactly model c_{medoid}.

In contrast to this, for the FATE approach, it is not immediately obvious if and how it is capable of modelling c_{medoid} exactly. However, the choices $\mathcal{Z} \coloneqq \mathcal{X}$, $\phi \coloneqq \mathrm{id}_{\mathcal{X}}$ and $U'(\boldsymbol{x}, z) \coloneqq -\|\boldsymbol{x} - z\|$ yield

$$U_{\text{FATE}}^{U', \phi}(\boldsymbol{x}, Q) = -\|\boldsymbol{x} - \text{centroid}(Q)\|$$

with $\text{centroid}(Q) \coloneqq \frac{1}{|Q|} \sum_{\boldsymbol{y} \in Q} \boldsymbol{y}$ being the centroid of Q. Thus, the item $\boldsymbol{x} \in Q$, which is closest to $\text{centroid}(Q)$, i. e., $\arg\max_{\boldsymbol{x} \in Q} U_{\text{FATE}}^{U', \phi}(\boldsymbol{x}, Q)$, is likely to coincide with the medoid of Q. As we construct our synthetic medoid dataset by sampling Q according to the uniform distribution ν on $\{A \subseteq [0,1]^d : |A| = r\}$ for some predefined $r \in \mathbb{N}$, there is with $U_{\text{FATE}}^{U', \phi}$ a FATE-instance, which is expected to have (for the case of singleton choice) an accuracy of at least

$$P_{Q \sim \nu}\left(c_{\text{medoid}}(Q) = \arg\min_{\boldsymbol{x} \in Q} \|\boldsymbol{x} - \text{centroid}(Q)\|\right)$$

on the synthetic medoid dataset. An empirical evaluation revealed that this value is 89.56 % for $r = 10$ and $d = 5$. For the details on this dataset, confer Appendix E.1.

6.3.2 The Pareto-Front Problem

The computation of a Pareto-optimal set of points is an important problem in optimization and various fields of application [41]. 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 $Q \in \mathcal{Q}$ we define the *Pareto-set* or *Pareto-front* of Q as

$$c_{\text{Pareto}}(Q) \coloneqq \{\boldsymbol{x} \in Q : \boldsymbol{x} \text{ is not dominated by any element } \boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}\}.$$

[7] As opposed to the centroid, which is usually not part of the original set.

We wish to investigate the possibility to learn the mapping from sets of points to their respective Pareto-sets. It is clear that the size of the Pareto-sets is not constant, which makes it a good candidate for a general subset choice problem. With the choices $U_0(\boldsymbol{x}) \coloneqq 0$ and $U_1(\boldsymbol{x}, \{\boldsymbol{y}\}) \coloneqq -[\![\boldsymbol{y} \succ \boldsymbol{x}]\!]$ we have

$$U_{\text{FETA}}^{U_0,U_1}(\boldsymbol{x}, Q) = -\sum\nolimits_{\boldsymbol{y}\in Q} [\![\boldsymbol{y} \succ \boldsymbol{x}]\!] \in \begin{cases} (-\infty, -1], & \text{if } \boldsymbol{x} \notin c_{\text{Pareto}}(Q), \\ \{0\}, & \text{otherwise.} \end{cases}.$$

Hence, $c_{\text{Pareto}}(Q) = \arg\max_{\boldsymbol{x}\in Q} U_{\text{FETA}}^{(U_1)}(\boldsymbol{x}, Q)$ holds trivially for each $Q \in \mathcal{Q}$, i. e., the Pareto problem is exactly solvable via the FETA approach. We created our corresponding synthetic dataset by generating a set of points uniformly at random in $\mathbb{R}^2$ and $\mathbb{R}^5$ to construct a choice task Q, and the ground-truth is the Pareto-set of Q containing only the non-dominated objects. In order to perform the experiments, we generate sets of 30 random points in $\mathbb{R}^2$ and $\mathbb{R}^5$, and determine the choices as described in detail in Appendix E.2.

6.3.3 *Hypervolume*

A related but much harder problem is the computation of hypervolume contributions of objects on a Pareto front. The hypervolume $\lambda_{\text{HypVol}}(Q)$ 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

$$\begin{aligned} \lambda_{\text{HypVol}}(Q) &\coloneqq \lambda\left(\bigcup\nolimits_{\boldsymbol{x}\in c_{\text{Pareto}}(Q)} [0, x_1] \times \cdots \times [0, x_d]\right) \\ &= \lambda\left(\bigcup\nolimits_{\boldsymbol{x}\in Q} [0, x_1] \times \cdots \times [0, x_d]\right) \end{aligned}$$

where λ denotes the Lebesgue measure of $\mathbb{R}^d$. In the context of multi-objective evolutionary algorithms (MOEAs), one usually computes the contributions $\lambda_{\text{HypVol}}(Q) - \lambda_{\text{HypVol}}(Q \setminus \{\boldsymbol{x}\})$ of each point $\boldsymbol{x} \in Q$ to the overall hypervolume $\lambda_{\text{HypVol}}(Q)$, i. e., the reduction in hypervolume caused by removing one object from the set. We consider the problem of learning the corresponding Hypervolume choice function $c_{\text{HypVol}} \colon \mathcal{Q} \longrightarrow C$, which picks that element $\boldsymbol{x} \in Q$ with the smallest contribution to the overall hypervolume, i. e.,

$$\begin{aligned} c_{\text{HypVol}}(Q) &\coloneqq \arg\max_{\boldsymbol{x}\in Q} \lambda_{\text{HypVol}}(Q) - \lambda_{\text{HypVol}}(Q \setminus \{\boldsymbol{x}\}) \\ &= \arg\min_{\boldsymbol{x}\in Q} \lambda_{\text{HypVol}}(Q \setminus \{\boldsymbol{x}\}). \end{aligned}$$

As shown by Bringmann and Friedrich [17, Theorem 1], it is #P-hard to calculate $c_{\text{HypVol}}(Q)$. Here, we generate sets of 10 random points in $\mathbb{R}^2$ and determine the singleton choice.

6.3.4 *MNIST Number Problems*

The original goal of the Modified National Institute of Standards and Technology (MNIST) dataset was to facilitate the comparison between different handwritten digits classifiers [65]. It consists of 70 000 28×28 grayscale images. We use the dataset to create challenging choice problems, both singleton and general subset choice. To level the playing field between all the approaches, we first train a convolutional neural network (CNN) on 10 000 instances and use it to extract high level features for the remaining 60 000 images (see Appendix E.3 for more details). To convert this dataset to a choice problem, we randomly sample sets of 10 numbers and choose based on the following procedures:

1. **Mode**: For the Mode dataset, we choose the numbers that occur most often in the choice task Q. For example, given a set of numbers {1, 1, 2, 4, 4, 5, 5, 6, 6, 6}, we choose all instances with value equal to the mode value 6. For the singleton choice task, we only output one of the numbers (the representation of which has the least angle to a predefined vector).
2. **Unique**: Here, we choose all numbers that occur only once in the set of sampled label values. For example, given a set of numbers {1, 1, 2, 3, 4, 4, 5, 5, 6, 6}, we choose the numbers {2, 3}. For the singleton choice problem, we ensure that exactly one of the digits is unique.

6.3.5 *MovieLens Tag Genome*

The MovieLens Tag Genome dataset consists of a large collection of movies and community curated tags [118]. For each movie, the relevance of every tag is provided on a continuous scale in [0, 1]. Thus, the complete relevance vector of a movie can be regarded as that movies' "genome."

We consider the problem of choosing the most similar/dissimilar movie from a set of movies, where one movie is regarded as the reference to which the others are compared. We define this reference movie to be the medoid of the movies in a given set. To compute similarities in tag relevance space, we use the weighted cosine similarity as proposed by Vig, Sen, and Riedl [117].

6.3.6 LETOR

LETOR is a collection of benchmark datasets for different learning-to-rank problems [89]. The Gov2 web page collection, consisting of roughly 25 M pages, is the corpus and the query sets of the Million Query track of the TREC 2007 and 2008 [111, 112] are used to create 8 datasets. Each query-document pair is defined by a vector consisting of 46 features. We use the supervised ranking datasets MQ2007 and MQ2008 to create the choice dataset. We treat all documents with a relevance score of 1 and 2 as the chosen objects. Since all queries include multiple documents with relevance scores 1 and 2, we cannot extract singleton choices from this dataset. The listwise ranking datasets MQ2007-list and MQ2008-list contain real-valued scores of the documents in the underlying permutations, and hence facilitate the singleton choice for each query (details of the exact procedure can be found in Appendix F.1).

6.3.7 Expedia

The Expedia dataset was released on the Kaggle website as a competition in 2016 [33]. It consists of 399 344 lists of hotels, each resulting from a search query of a user. For each hotel, there are 45 features and a relevance score, indicating how relevant the hotel is to the provided query. A score of 0 means that it was not relevant, a score of 1 indicates that the user clicked on it, and a 2 implies that the hotel was booked. It is straightforward to construct choice datasets: for singleton choice the goal is simply to predict the booked hotel, whereas for subset choice we required the learners to output the complete set of hotels that were at least clicked on (see Appendix F.2 for more details).

6.3.8 SUSHI

SUSHI[8] is a dataset created by Kamishima [57] specifically for the task of *object ranking*. The authors considered 100 sushis and asked users to rank them according to their preference. The dataset consists of two sets of 5000 rankings. Each ranking consists of 10 sushis, which were ranked by users in a survey. For the first set, the authors asked the users to rank the top-10 most popular sushis. In the second set, users were shown random sets of 10 sushis instead. Each sushi is described by 7 object features. Additional user features are available, but not used in our experiments. For our experiments, we merge both datasets into a single one containing 10 000 instances. We use it as a singleton choice dataset by choosing the most preferred sushi as the singleton choice for the given task set Q (details of the exact procedure can be found in Appendix F.3).

6.4 Results and Discussion

In this section, we provide the results obtained by evaluating different subset choice and singleton choice models on the datasets. To be concise, we only show plots for the target losses here and list the complete set of results in Tables 9 to 11 in Appendix G. It is illuminating to compare the performance of FATE-NET and FETA-NET to the context-independent neural network RANKNET. This provides a rough indicator for how important being able to model context-dependence is.

6.4.1 Singleton Choice

We will start by discussing the results for the singleton choice models (cf. Figure 5), where the bars depict the mean value of the categorical accuracy (26) across the cross-validation folds, with black lines depicting the standard deviation.

The first observation is that FATE-NET and FETA-NET significantly outperform all other baselines on the tasks for which it was clear that the underlying choice function is context-dependent (i. e., Hypervolume, Medoid and the MNIST datasets). The SDA network, which is also a context-dependent model, achieves competitive results on the Medoid and the MNIST datasets. The linear FETA variant FETA-LINEAR non-linear neural network RANKNET perform comparably to the other baseline approaches. This suggests that a combination of non-linearity and the ability to model context-dependence is really necessary to improve on these tasks. One notable exception is the Medoid dataset, for which RANKNET and FETA-LINEAR manage to outperform the other baselines by a large margin.

[8] This dataset can be downloaded from `http://www.kamishima.net/sushi/`

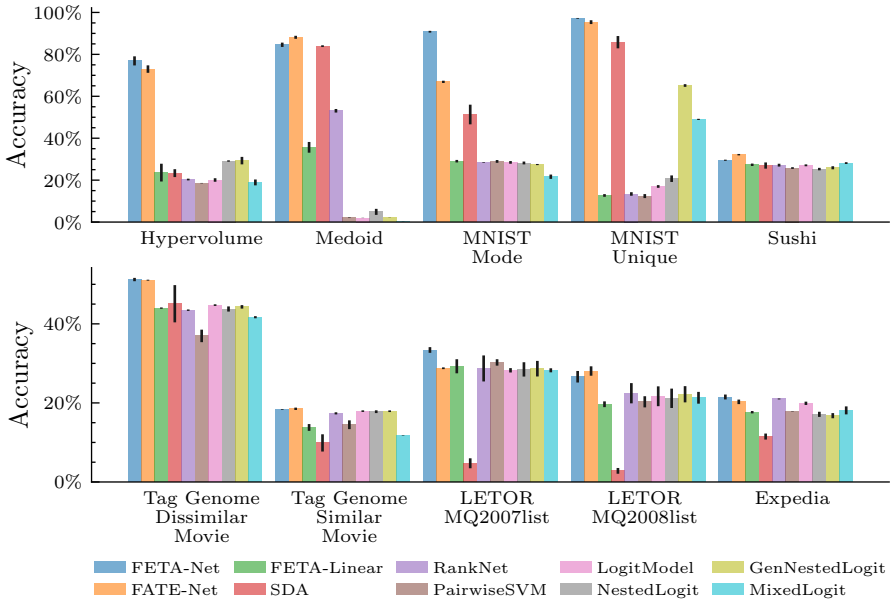


Figure 5: Categorical accuracies and standard deviations (vertical bars) of the singleton choice models on different singleton choice tasks (measured across 5 outer cross-validation folds).

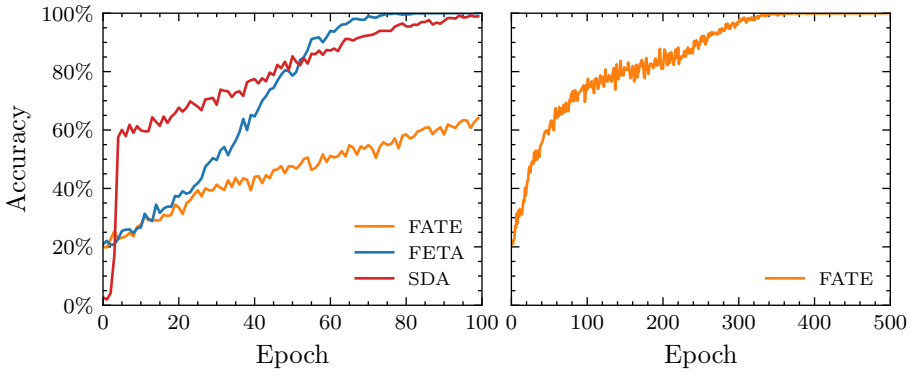


Figure 6: Result of the infinite data experiment for FATE-NET, FETA-NET and SDA on the synthetic unique problem. For the left plot the neural networks were calibrated to have roughly equal numbers of parameters. The right plot shows the repetition of the experiment where FATE-NET received a higher epoch and parameter budget.

For the MNIST-Unique problem, FATE-NET and FETA-NET achieve an accuracy of more than 90 % and SDA is competitive with over 80 %. Additionally, the GNL and ML models are also able to perform better than the other baselines. It is easy to see that the dataset exhibits the similarity context effect proposed by Huber and Puto [52], i. e., adding multiple instances of the same digit to the choice task reduces the choice probability of all equal digits to 0. As is apparent, the GNL and ML model are able to account for it and score better than chance.

Since FATE-NET, FETA-NET and SDA were able to achieve close to 100 % accuracy on the MNIST-Unique problem, we performed an additional experiment where we generated instances completely synthetically. Each number $i \in \{0, \dots, 9\}$ we represent by the corresponding standard unit vector $\boldsymbol{e}_i$, which is 1 in the i-th position and is 0 everywhere else. Apart from that, the task remains the same. We calibrate each network to have roughly the same number of parameters (2870 for FATE-NET, 2849 for FETA-NET and 2850 for SDA) and the remaining hyperparameters were equal for all networks. We then trained them on a stream of newly generated batches with 1024 instances, each of which with 10 objects until convergence. The resulting convergence behavior is shown in Figure 6. Both FETA-NET and SDA are able to converge to 100 % out-of-sample categorical accuracy within 100 epochs, while FATE-NET only achieves slightly over 60 % and more epochs alone were not able to let it learn the target function without error. We therefore repeated the experiment for FATE-NET with a higher epoch and parameter budget. With 5985 parameters, FATE-NET is now able to perfectly

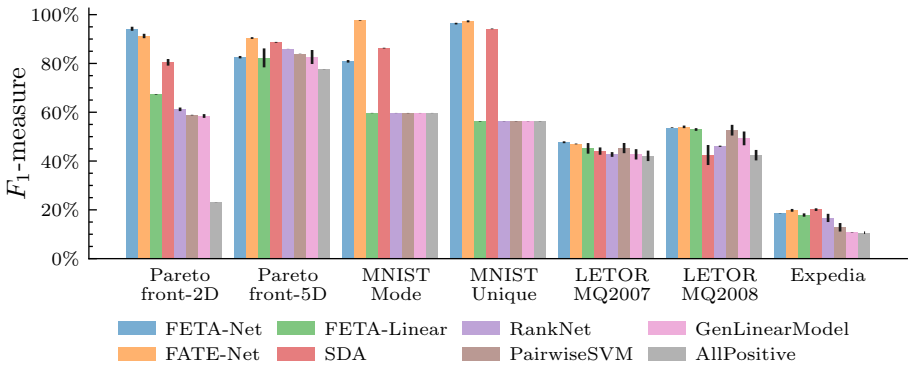


Figure 7: Average F_1-measure and standard deviation (vertical bars) of the subset-choice models on the tasks different choice tasks (measured across 5 outer cross-validation folds).

learn the fully synthetic unique problem within 400 epochs. On the one hand, this shows that from a representational perspective, all three models are able to learn this particular target choice function perfectly. FATE-NET appears to be less parameter- and data-efficient though, which could indicate that evaluating the utilities in the context of the set embedding is not well suited to represent these kinds of problems. The behavior of all three networks was consistent across repetitions of the experiment.

On the real-world datasets (i. e. Sushi, Movielens Tag Genome, LETOR and Expedia) the performance of FATE-NET and FETA-NET is closer to the ones achieved by the remaining baselines. Although they still obtain slightly higher accuracy on average, the margin is not as pronounced. Surprisingly, the SDA achieved the worst accuracy on LETOR and Expedia. We suspect that this results from the models being trained only on a fixed choice task size in our experiments, while they are evaluated on choice tasks of varying size during test time. Since SDA learns a set-dependent aggregation function, it could be that this does not generalize well to the larger choice tasks present in the real-world datasets.

6.4.2 Subset Choice

We evaluate the subset choice models in terms of their F_1-measure (21) and report the results in Figure 7. To see if the models are able to learn anything, we also show the performance of the baseline that always predicts positive. The general pattern is confirmed: FATE-NET, FETA-NET and SDA surpass the other baselines on the datasets Pareto-front 2D, MNIST Mode, and MNIST Unique, while being competitive for the real-world datasets LETOR and Expedia. For the MNIST tasks Unique and Mode, the first observation is that all linear and/or context-independent baseline approaches fail to learn anything on these datasets, since they all achieve the same F_1-measure as the all-positive baseline. Thus, it is clear that these tasks can only be solved by models that are both context-dependent and non-linear.

For the Pareto problem, it can be observed that the context-dependent models FETA-NET, FATE-NET, and SDA outperform all benchmark choice models on the 2D version. On the 5D version of the dataset, however, the performance of all approaches reach a comparable level. This indicates that solving the task of selecting the Pareto-front becomes less context-dependent in higher dimensions, since the distance of a point from the center becomes more and more informative. At the same time, more points are on the Pareto-front overall, which is apparent from the high F_1-measure of the AllPositive baseline.

As before, the results are more homogeneous on the real-world datasets Expedia and LETOR MQ2007/MQ2008. FATE-NET and FETA-NET are still outperforming all the benchmarks. This suggests that the ability to model context-dependence in the data is slightly more important for these datasets than learning a non-linear utility function. SDA achieves the best result on the Expedia dataset, which when compared to the bad performance on the singleton choice variant of the dataset suggests that the thresholding of the utilities is robust to the model output changing with varying choice task sizes.

Overall, the results demonstrate that FATE-NET and FETA-NET are able to improve on the context-independent baselines by a large margin on tasks which are strongly context-dependent and show competitive results when compared to SDA. The improvement is due to both the task-sensitivity of these models and the ability to model non-linear utility functions. For the real-world datasets, the improvements are smaller, suggesting that context-effects are either less pronounced or that the context-effects in real-world data cannot fully be captured yet.

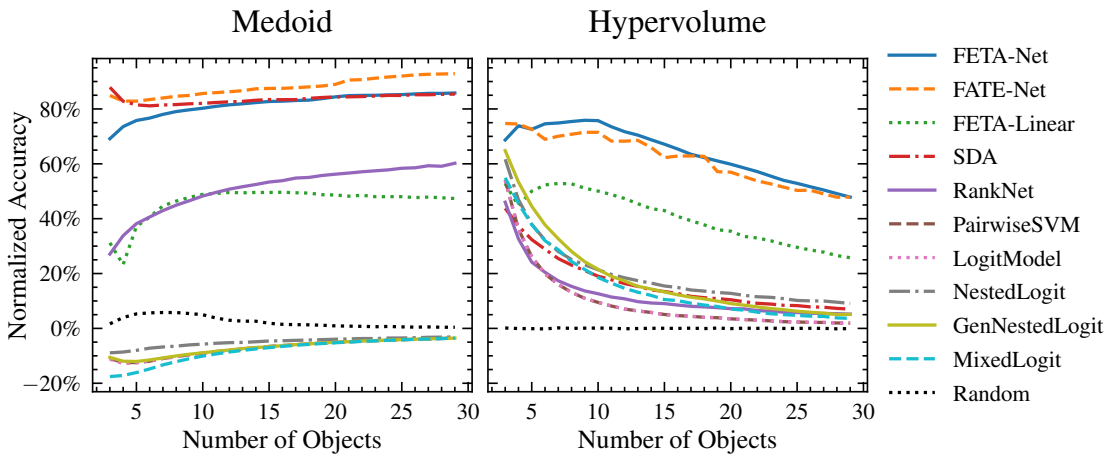


Figure 8: Normalized Accuracy of the singleton choice models (SCMs) trained on queries of size 10, then predicting on queries of a varying size.

6.4.3 Generalization Across Task Sizes

We conduct additional experiments to gauge the generalization capability of the learned models to unseen task sizes (refer to Appendix D for more details). We show the results for the datasets Medoid and Hypervolume, because, as will be seen, they exhibit some interesting properties. We specifically compare the performance on the singleton choice datasets (Figure 8). We train the models on a fixed task size and then test them on sets containing between 3 and 21 objects. Note that for singleton choice, the accuracy is not comparable across differing task sizes. We instead report the *normalized accuracy* (see Appendix B.2), which fixes this issue and guarantees that random guessing achieves exactly 0.

Overall, the models manage to generalize quite well to task sizes for which they were not trained. The exact generalization behavior depends on the dataset, though. Considering the Medoid dataset, we can observe that the models FETA-NET, FETA-LINEAR and RANKNET even improve in performance with larger task sizes. This is plausible, since the more points fill the space, the more the problem can be solved by a context-independent model, which assigns the highest score to objects in the center. For the singleton choice version of Hypervolume, on the other hand, the performance of all models drops with an increasing numbers of objects, suggesting it becomes much harder to identify the object that contributes the most to the overall hypervolume. This is especially visible for the baselines, which, even though they were trained on 10 objects, achieve their best performance on 3 objects. FETA-NET, FATE-NET, and FETA-LINEAR stand out here, since their performance decays much slower. All in all, we conclude that our networks FETA-NET and FATE-NET are able to generalize very well to unseen task sizes, with FETA-NET additionally benefiting if the task becomes less context-dependent with larger task sizes.

7 CONCLUSION AND FUTURE WORK

In this paper, we tackle the problem of choice from a machine learning perspective. More specifically, we propose a framework for learning context-dependent choice functions, which, on the basis of choice behavior observed in the past, allow for predicting the choice of objects in new situations. This is essentially accomplished by learning generalized (latent) scoring (utility) functions, which are supposed to control the choice behavior.

Violations of context-independence are common in human choice behavior. Therefore, accounting for the various context effects they can exhibit can be seen as an important problem. Still, we consider the space of interesting non-trivial choice functions to be vastly larger, and the goal is to have general purpose models that can adapt to a wide variety of (yet unknown) context effects.

To this end, we propose two principled decompositions: The FETA decomposition is a first-order approximation to a more general utility decomposition. It considers each object in *local* sub-contexts, the contributions of which are averaged. The FATE approach, on the other side, first transfers each object into an embedding space and computes a representative of the choice task by averaging these embedded points. The utility of each object is then evaluated with the representative as *global* context. Both approaches are complementary and have differing inductive biases. In spite of this, both show promising predictive performance.

While the FETA and FATE decompositions are general and in a sense quite natural approaches to model context-dependent choice functions, a promising direction is the investigation of application-specific models with more focused inductive biases. An example is the SDA approach, which applies principles from behavioral choice theory and also tries to take the risk-aversion of humans into account [95].

While the most influential context effects for human choices have been studied, gaining a deeper understanding of the rich mathematical structure of general choice problems is an important future endeavor.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support provided by the European Regional Development Fund (ERDF) and the valuable feedback provided by the industry partners of the Smart-GM research project – EFRE-0801915.

Funded by the Deutsche Forschungsgemeinschaft (DFG – German Research Foundation) – 317046553.

This work is part of the Collaborative Research Center "On-the-Fly Computing" at Paderborn University, which is supported by the German Research Foundation (DFG). Experiments were performed on resources provided by the Paderborn Center for Parallel Computing.

REFERENCES

[1] Charu C Aggarwal and Philip S Yu. "Outlier detection for high dimensional data". In: *ACM Sigmod Record*. Vol. 30. 2. ACM. 2001, pp. 37–46.

[2] Qingyao Ai et al. "Learning a Deep Listwise Context Model for Ranking Refinement". In: *SIGIR*. ACM, 2018, pp. 135–144.

[3] Qingyao Ai et al. "Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks". In: *ICTIR*. ACM, 2019, pp. 85–92.

[4] Pavel Anselmo Alvarez, Alessio Ishizaka, and Luis Martínez. "Multiple-criteria decision-making sorting methods: A survey". In: *Expert Systems with Applications* 183 (2021), p. 115368.

[5] Attila Ambrus and Kareen Rozen. "Rationalising Choice with Multi-Self Models". In: *The Economic Journal* 125.585 (Mar. 2014), pp. 1136–1156.

[6] Kenneth J Arrow. *Social Choice and Individual Values*. John Wiley & Sons, 1951.

[7] Richard R. Batsell and John C. Polking. "A New Class of Market Share Models". In: *Marketing Science* 4.3 (1985), pp. 177–198.

[8] Peter W. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *CoRR* abs/1806.01261 (2018).

[9] Moshe Ben-Akiva and Steven R Lerman. *Discrete Choice Analysis: Theory and Application to Travel Demand*. Vol. 9. MIT Press, 1985.

[10] 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 '18. New York, NY, USA: ACM, 2018, pp. 37–45.

[11] Jeremy Bentham. *An Introduction to the Principles of Morals and Legislation*. London: T. Payne, and Son, 1789.

[12] Joseph Berkson. "Application of the Logistic Function to Bio-Assay". In: *Journal of the American Statistical Association* 39.227 (1944), pp. 357–365.

[13] James R Bettman, Mary Frances Luce, and John W Payne. "Constructive Consumer Choice Processes". In: *Journal of Consumer Research* 25.3 (1998), pp. 187–217.

[14] Bernd Bischl et al. "ASlib: A benchmark library for algorithm selection". In: *Artificial Intelligence* 237 (2016), pp. 41–58.

[15] Amanda Bower and Laura Balzano. "Preference Modeling with Context-Dependent Salient Features". In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1067–1077.

[16] 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.

[17] Karl Bringmann and Tobias Friedrich.
"Approximating the volume of unions and intersections of high-dimensional geometric objects".
In: *Computational Geometry* 43.6 (2010), pp. 601–610.

[18] Christopher J. C. Burges et al. "Learning to Rank using Gradient Descent". In: *ICML*. Vol. 119.
ACM International Conference Proceeding Series. ACM, 2005, pp. 89–96.

[19] C. Domshlak et al. "Preferences in AI: An overview".
In: *Artificial Intelligence* 175.7–8 (2011), pp. 1037–1052.

[20] Dipankar Chakravarti and John G. Lynch Jr.
"A Framework For Exploring Context Effects on Consumer Judgment and Choice".
In: *NA - Advances in Consumer Research* 10 (1983), pp. 289–297.

[21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A survey".
In: *ACM Computing Surveys (CSUR)* 41.3 (2009), p. 15.

[22] Shuo Chen and Thorsten Joachims. "Modeling Intransitivity in Matchup and Comparison Data". In: *WSDM*.
ACM, 2016, pp. 227–236.

[23] François Chollet et al. *Keras*. https://keras.io. 2017.

[24] 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.

[25] Gerard Debreu. "Representation of a preference ordering by a numerical function".
In: *Decision Processes* 3 (1954), pp. 159–165.

[26] Gerard Debreu. "Review of R. D. Luce, Individual Choice Behavior: A Theoretical Analysis".
In: *American Economic Review* 50.1 (1960), pp. 186–188.

[27] Gerard Debreu. *Theory of Value: An Axiomatic Analysis of Economic Equilibrium*.
Yale University Press, 1959.

[28] James Diamond and William Evans. "The Correction for Guessing".
In: *Review of Educational Research* 43.2 (1973), pp. 181–191.

[29] Ürün Dogan, Tobias Glasmachers, and Christian Igel.
"A Unified View on Multi-class Support Vector Classification".
In: *Journal of Machine Learning Research* 17.45 (2016), pp. 1–32.

[30] John R. Doyle et al. "The robustness of the asymmetrically dominated effect: Buying frames, phantom
alternatives, and in-store purchases". In: *Psychology & Marketing* 16.3 (1999), pp. 225–243.

[31] John Duchi, Elad Hazan, and Yoram Singer.
"Adaptive Subgradient Methods for Online Learning and Stochastic Optimization".
In: *Journal of Machine Learning Research* 12 (2011), pp. 2121–2159.

[32] Theodoros Evgeniou, Constantinos Boussios, and Giorgos Zacharia.
"Generalized Robust Conjoint Estimation". In: *Marketing Science* 24.3 (2005), pp. 415–429.

[33] Expedia. *Expedia Hotel Recommendations*.
https://www.kaggle.com/c/expedia-hotel-recommendations/overview. Accessed: 2021-03-28.
June 2016.

[34] M. Ahmadi Fahandar, E. Hüllermeier, and I. Couso.
"Statistical Inference for Incomplete Ranking Data: The Case of Rank-Dependent Coarsening". In: *ICML*.
Vol. 70. PMLR, 2017, pp. 1078–1087.

[35] Tom Fawcett. "An Introduction to ROC Analysis".
In: *Pattern Recognition Letters* 27.8 (June 2006), pp. 861–874.

[36] Gustav Theodor Fechner. *Elemente der Psychophysik*. Vol. 2. Breitkopf u. Härtel, 1860.

[37] M. A. Fligner and J. S. Verducci. "Distance based Ranking Models".
In: *Journal of the Royal Statistical Society. Series B (Methodological)* 48.3 (1986), pp. 359–369.

[38] Michael A. Fligner and Joseph S. Verducci. "Multistage Ranking Models".
In: *Journal of the American Statistical Association* 83.403 (1988), pp. 892–901.

[39] Drew Fudenberg and David K. Levine. "A Dual-Self Model of Impulse Control".
In: *American Economic Review* 96.5 (Dec. 2006), pp. 1449–1476.

[40] Johannes Fürnkranz and Eyke Hüllermeier, eds. *Preference Learning*. Springer, 2010.

[41] Marc Geilen et al. "An Algebra of Pareto Points". In: *Fundamenta Informaticae* 78.1 (2007), pp. 35–74.

[42] Xavier Glorot and Yoshua Bengio.
"Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS*. Vol. 9.
JMLR Proceedings. JMLR, 2010, pp. 249–256.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[44] M. Grabisch et al. *Aggregation Functions*. Cambridge University Press, 2009.

[45] Jerry Green and Daniel Hojman. *Choice, Rationality and Welfare Measurement*. KSG Faculty Research Working Paper Series RWP07-054. Rochester, NY: Harvard University, Nov. 1, 2007.

[46] John Gurland, Ilbok Lee, and Paul A. Dahm. "Polychotomous Quantal Response in Biological Assay". In: *Biometrics* 16.3 (1960), pp. 382–398.

[47] F Maxwell Harper and Joseph A Konstan. "The MovieLens Datasets: History and Context". In: *ACM Trans. Interact. Intell. Syst.* 5.4 (2015).

[48] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *ICCV*. IEEE Computer Society, 2015, pp. 1026–1034.

[49] Tim Head et al. *scikit-optimize/scikit-optimize: v0.5.2*. Version v0.5.2. Mar. 2018.

[50] H. S. Houthakker. "Revealed Preference and the Utility Function". In: *Economica* 17.66 (1950), pp. 159–174.

[51] Joel Huber, John W. Payne, and Christopher Puto. "Adding Asymmetrically Dominated Alternatives: Violations of Regularity and the Similarity Hypothesis". In: *Journal of Consumer Research* 9.1 (1982), pp. 90–98.

[52] Joel Huber and Christopher Puto. "Market Boundaries and Product Choice: Illustrating Attraction and Substitution Effects". In: *Journal of Consumer Research* 10.1 (1983), pp. 31–44.

[53] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML*. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR, 2015, pp. 448–456.

[54] K. Dembczynski et al. "On Label Dependence and Loss Minimization in Multi-Label Classification". In: *Machine Learning* 88.1–2 (2012), pp. 5–45.

[55] Gil Kalai, Ariel Rubinstein, and Ran Spiegler. "Rationalizing Choice Functions by Multiple Rationales". In: *Econometrica* 70.6 (2002), pp. 2481–2488.

[56] Wagner A. Kamakura and Rajendra K. Srivastava. "Predicting Choice Shares under Conditions of Brand Interdependence". In: *Journal of Marketing Research* 21.4 (1984), pp. 420–434.

[57] Toshihiro Kamishima. "Nantonac Collaborative Filtering: Recommendation Based on Order Responses". In: *KDD*. ACM, 2003, pp. 583–588.

[58] Toshihiro Kamishima, Hideto Kazawa, and Shotaro Akaho. "A Survey and Empirical Comparison of Object Ranking Methods". In: *Preference Learning*. Springer, 2010, pp. 181–201.

[59] Jerry S. Kelly and Maxwell Hall. "Impossibility results with resoluteness". In: *Economics Letters* 34.1 (1990), pp. 15–19.

[60] Mark Kelman, Yuval Rottenstreich, and Amos Tversky. "Context-Dependence in Legal Decision Making". In: *The Journal of Legal Studies* 25.2 (1996), pp. 287–318.

[61] Ran Kivetz, Oded Netzer, and V. Srinivasan. "Alternative Models for Capturing the Compromise Effect". In: *Journal of Marketing Research* 41.3 (2004), pp. 237–257.

[62] Günter Klambauer et al. "Self-Normalizing Neural Networks". In: *NIPS*. Curran Associates Inc., 2017, pp. 972–981.

[63] Jon M. Kleinberg, Sendhil Mullainathan, and Johan Ugander. "Comparison-based Choices". In: *EC*. ACM, 2017, pp. 127–144.

[64] Oluwasanmi Koyejo et al. "Consistent Multilabel Classification". In: *NIPS*. MIT Press, 2015, pp. 3321–3329.

[65] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. *The MNIST database of handwritten digits*. accessed: 2021-03-29. 2010.

[66] David D. Lewis. "Evaluating and Optimizing Autonomous Text Classification Systems". In: *SIGIR*. ACM Press, 1995, pp. 246–254.

[67] R. Duncan Luce. *Individual Choice Behavior.* Oxford, England: John Wiley, 1959.

[68] Sebastián Maldonado, Ricardo Montoya, and Richard Weber. "Advanced conjoint analysis using feature selection via support vector machines". In: *European Journal of Operational Research* 241.2 (2015), pp. 564–574.

[69] C. L. Mallows. "Non-Null Ranking Models. I". In: *Biometrika* 44.1/2 (1957), pp. 114–130.

[70] Nathan Mantel. "Models for Complex Contingency Tables and Polychotomous Dosage Response Curves". In: *Biometrics* 22.1 (1966), pp. 83–95.

[71] Paola Manzini and Marco Mariotti. "Sequentially Rationalizable Choice". In: *American Economic Review* 97.5 (2007), pp. 1824–1839.

[72] Harry Markowitz. "Portfolio Selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91.

[73] Kenneth O. May. "Intransitivity, Utility, and the Aggregation of Preference Patterns". In: *Econometrica* 22.1 (1954), pp. 1–13.

[74] Donna Katzman McClish. "Analyzing a Portion of the ROC Curve". In: *Medical Decision Making* 9.3 (1989), pp. 190–195.

[75] Daniel McFadden. "Conditional Logit Analysis of Qualitative Choice Behavior". In: *Frontiers in Econometrics*. Academic Press, 1974, pp. 105–142.

[76] Daniel McFadden and Kenneth Train. "Mixed MNL models for discrete response". In: *Journal of Applied Econometrics* 15.5 (2000), pp. 447–470.

[77] Barbara A. Mellers and Michael H. Birnbaum. "Contextual effects in social judgment". In: *Journal of Experimental Social Psychology* 19.2 (1983), pp. 157–171.

[78] Robert Moore and John DeNero. "L1 and L2 Regularization for Multiclass Hinge Loss Models". In: *MLSLP*. 2011, pp. 1–5.

[79] Yurii Nesterov. "A method of solving a convex programming problem with convergence rate $O(1/k^2)$". In: *Soviet Mathematics Doklady*. Vol. 27. 2. 1983, pp. 372–376.

[80] A. Yeşim Orhun. "Optimal Product Line Design When Consumers Exhibit Choice Set-Dependent Preferences". In: *Marketing Science* 28.5 (2009), pp. 868–886.

[81] Ali I. Ozkes and M. Remzi Sanver. "Anonymous, neutral, and resolute social choice revisited". In: *Social Choice and Welfare* 57.1 (July 2021), pp. 97–113.

[82] John W Payne, James R Bettman, and Eric J Johnson. "Behavioral Decision Research: A Constructive Processing Perspective". In: *Annual Review of Psychology* 43.1 (1992), pp. 87–131.

[83] John W Payne et al. "Measuring Constructed Preferences: Towards a Building Code". In: *Elicitation of Preferences*. Springer, 1999, pp. 243–275.

[84] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[85] Karlson Pfannschmidt, Pritha Gupta, and Eyke Hüllermeier. "Deep Architectures for Learning Context-dependent Ranking Functions". In: *CoRR* abs/1803.05796 (2018).

[86] R. L. Plackett. "The Analysis of Permutations". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24.2 (1975), pp. 193–202.

[87] 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.

[88] David Martin Powers. "Recall & Precision versus The Bookmaker". In: *ICCS*. University of New South Wales, 2003, pp. 529–534.

[89] Tao Qin and Tie-Yan Liu. "Introducing LETOR 4.0 Datasets". In: *CoRR* abs/1306.2597 (2013).

[90] Siamak Ravanbakhsh, Jeff G. Schneider, and Barnabás Póczos. "Equivariance Through Parameter-Sharing". In: *ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 2892–2901.

[91] John R. Rice. "The Algorithm Selection Problem". In: *Advances in Computers*. Vol. 15. Advances in Computers. Elsevier, 1976, pp. 65–118.

[92] Jörg Rieskamp, Jerome R. Busemeyer, and Barbara A. Mellers. "Extending the Bounds of Rationality: Evidence and Theories of Preferential Choice". In: *Journal of Economic Literature* 44.3 (Sept. 2006), pp. 631–661.

[93] Leonardo Rigutini et al. "SortNet: Learning to Rank by a Neural Preference Function". In: *IEEE Trans. Neural Networks* 22.9 (2011), pp. 1368–1380.

[94] Robert P Rooderkerk, Harald J Van Heerde, and Tammo HA Bijmolt. "Incorporating Context Effects into a Choice Model". In: *Journal of Marketing Research* 48.4 (2011), pp. 767–780.

[95] Nir Rosenfeld, Kojin Oshiba, and Yaron Singer. "Predicting Choice with Set-Dependent Aggregation". In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 8220–8229.

[96] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th. Pearson, 2020.

[97] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. "Probabilistic programming in Python using PyMC3". In: *PeerJ Computer Science* 2 (Apr. 2016), e55.

[98] P. A. Samuelson. "A Note on the Pure Theory of Consumer's Behaviour". In: *Economica* 5.17 (1938), pp. 61–71.

[99] Constantine Sedikides, Dan Ariely, and Nils Olsen. "Contextual and Procedural Determinants of Partner Selection: Of Asymmetric Dominance and Prominence". In: *Social Cognition* 17.2 (1999), pp. 118–139.

[100] Amartya K. Sen. "Choice Functions and Revealed Preference". In: *The Review of Economic Studies* 38.3 (1971), pp. 307–317.

[101] Arjun Seshadri, Alex Peysakhovich, and Johan Ugander. "Discovering Context Effects from Raw Choice Data". In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5660–5669.

[102] Eldar Shafir, Itamar Simonson, and Amos Tversky. "Reason-based choice". In: *Cognition* 49.1 (1993), pp. 11–36.

[103] Itamar Simonson. "Choice Based on Reasons: The Case of Attraction and Compromise Effects". In: *Journal of Consumer Research* 16.2 (1989), pp. 158–174.

[104] Itamar Simonson and Amos Tversky. "Choice in Context: Tradeoff Contrast and Extremeness Aversion". In: *Journal of Marketing Research* 29.3 (1992), pp. 281–295.

[105] Gene Smith. *Tagging: People-Powered Metadata for the Social Web*. New Riders, 2007.

[106] Richard P. Stanley. *Enumerative Combinatorics*. 2nd. Vol. 1. Cambridge University Press, 2011.

[107] Gerald Tesauro. "Connectionist Learning of Expert Preferences by Comparison Training". In: *NIPS*. Morgan Kaufmann Publishers Inc., 1989, pp. 99–106.

[108] Henri Theil. "A Multinomial Extension of the Linear Logit Model". In: *International Economic Review* 10.3 (1969), pp. 251–259.

[109] Kiran Tomlinson and Austin Benson. "Choice Set Optimization Under Discrete Choice Models of Group Decisions". In: *ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 9514–9525.

[110] Kenneth E Train. *Discrete Choice Methods with Simulation*. 2nd. Cambridge University Press, 2009.

[111] TREC. *TREC 2007 Million Query Track*. Accessed: 2021-03-29. 2007.

[112] TREC. *TREC 2008 Million Query Track*. Accessed: 2021-03-29. 2008.

[113] Amos Tversky. "Elimination by Aspects: A Theory of Choice". In: *Psychological Review* 79.4 (1972), p. 281.

[114] Amos Tversky. "Intransitivity of Preferences". In: *Psychological Review* 76.1 (1969), p. 31.

[115] Amos Tversky and Itamar Simonson. "Context-dependent Preferences". In: *Management Science* 39.10 (1993), pp. 1179–1189.

[116] Mark Van der Laan, Katherine Pollard, and Jennifer Bryan. "A new partitioning around medoids algorithm". In: *Journal of Statistical Computation and Simulation* 73.8 (2003), pp. 575–584.

[117] Jesse Vig, Shilad Sen, and John Riedl. "Navigating the Tag Genome". In: *IUI*. ACM. 2011, pp. 93–102.

[118] 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.

[119] Milan Vojnovic and Se-Young Yun. *On the Team Selection Problem*. Tech. rep. MSR-TR-2016-7. Microsoft Research, Feb. 2016.

[120] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. 1st. Princeton University Press, 1944.

[121] Willem Waegeman et al. "On the Bayes-Optimality of F-Measure Maximizers". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3333–3388.

[122] Chieh-Hua Wen and Frank S Koppelman. "The generalized nested logit model". In: *Transportation Research Part B* 35.7 (2001), pp. 627–641.

[123] 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.

[124] Nan Ye et al. "Optimizing F-measure: A Tale of Two Approaches". In: *ICML*. icml.cc / Omnipress, 2012, pp. 1555–1562.

[125] L. Yu and B. Sun. "Four types of typical discrete Choice Models: Which are you using?" In: *Proceedings of 2012 IEEE International Conference on Service Operations and Logistics, and Informatics*. 2012, pp. 298–301.

[126] Manzil Zaheer et al. "Deep Sets". In: *NIPS*. Vol. 30. Curran Associates, Inc., 2017, pp. 3394–3404.

[127] Qiaoping Zhang and Isabelle Couloigner. "A New and Efficient K-Medoid Algorithm for Spatial Clustering". In: *ICCSA*. ICCSA'05. Singapore: Springer-Verlag, 2005, pp. 181–189.

A NOTATION

Table 2: Notation used throughout the paper.

Symbol	Meaning
$[n]$	$\{1, 2, \ldots, n\}$
$[\![A]\!]$	1 if A is a true statement and 0 otherwise
$\boldsymbol{I}_d$	the unit matrix of size $d \times d$
$\mathcal{X}$	set of reference objects
$\boldsymbol{x}$	object, choice alternative
$\mathcal{Q}$	choice task space
$\mathcal{C}$	choice space
$\mathcal{Z}$	embedding space
Q	task
C	choice set, i. e., an element of $\mathcal{C}$
c	choice function $\mathcal{Q} \longrightarrow \mathcal{C}$
π	ranking
P	probability measure
p	probability distribution (mass function)
L	loss function on $\mathcal{C}$
U	utility function
$C_{\text{singleton}}(U, Q)$	singleton choice from Q according to U; formally defined as $\arg\max_{\boldsymbol{x} \in Q} U(\boldsymbol{x})$
$C^U_{\text{subset}}(t, Q)$	subset choice from Q according to U and t; formally defined as $\{\boldsymbol{x} \in Q : U(\boldsymbol{x}, Q) \geq t\}$
D_k	domain of the sub-utility function U_k of FETA; formally defined as $\{(\boldsymbol{x}, A) : \boldsymbol{x} \in \mathcal{X} \text{ and } A \subseteq \mathcal{X} \setminus \{\boldsymbol{x}\} \text{ with } \lvert A\rvert = k\}$
$U^{U_0, U_1}_{\text{FETA}}$	FETA utility function with sub-utility functions U_0 and U_1
ϕ	embedding function
$U^{U', \phi}_{\text{FATE}}$	FATE utility function with sub-utility function U' and transformation ϕ
$\lVert\cdot\rVert$	standard euclidean norm in $\mathbb{R}^n$, i. e., $\lVert\boldsymbol{x}\rVert = \sqrt{x_1^2 + \cdots + x_n^2}$
θ	parameters of a model
$\mathcal{D}$	Dataset

B EVALUATION MEASURES

Besides the target losses introduced in Section 6.2, we evaluate the trained models using additional evaluation measures. These should give a more complete picture of the performance of the different models. The results including the additional measures can be found in Appendix G.

B.1 Singleton Choice

To define the evaluation measures in the singleton choice setting, suppose in the following a choice task space $\mathcal{Q} \subset 2^{\mathcal{X}}$, a utility function U for $\mathcal{Q}$ as well as $Q \in \mathcal{Q}$ and $\boldsymbol{x} \in Q$ to be arbitrary but fixed.

Top-k Categorical Accuracy The top-k categorical accuracy is defined as the fraction of times in which the set of objects in the top k positions, according to the predicted scores, contains the ground-truth chosen object [23, 9]. Formally, writing $Q = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{|Q|}\}$ with $U(\boldsymbol{y}_1, Q) \geq \cdots \geq U(\boldsymbol{y}_{|Q|}, Q)$, we have

$$\mathrm{m}_{\text{top-}k}(U, Q, \{\boldsymbol{x}\}) \coloneqq [\![\boldsymbol{x} \in \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k\}]\!] \ . \tag{25}$$

Categorical Accuracy The categorical accuracy is defined as the fraction of times in which the object with the largest score is the same as that ground-truth singleton choice, i. e.,

$$\mathrm{m}_{\text{CA}}(U, Q, \{\boldsymbol{x}\}) = [\![\boldsymbol{x} \in \arg\max_{\boldsymbol{y} \in Q} U(\boldsymbol{y}, Q)]\!] \ . \tag{26}$$

The categorical accuracy is the most common measure used for the evaluation of SCMs and commonly referred to as *hit-rate* [9]. It is evident that $\mathrm{m}_{\text{CA}}(U, Q, \{\boldsymbol{x}\}) = \mathrm{m}_{\text{top-1}}(U, Q, \{\boldsymbol{x}\})$ holds, provided $\arg\max_{\boldsymbol{y} \in Q} U(\boldsymbol{y}, Q)$ is a singleton set.

Normalized Accuracy The measures defined above are not a reasonable estimate when observing the performance of an SCM on the choice tasks of different sizes $|Q|$, since the task becomes harder as the choice task size increases. The hardness of the task should be adjusted with respect to the accuracy that random guessing can achieve, which is defined as the probability of choosing the correct singleton choice from the choice task Q. Assuming each object to be chosen with the same probability, the probability for choosing a fixed object is $\frac{1}{|Q|}$. These considerations motivate the definition of the *normalized accuracy* as follows:

$$\mathrm{m}_{\mathrm{CANorm}}(U, Q, \{\boldsymbol{x}\}) \coloneqq \frac{\mathrm{m}_{\mathrm{CA}}(U, Q, \{\boldsymbol{x}\}) - \frac{1}{|Q|}}{1 - \frac{1}{|Q|}} \quad . \tag{27}$$

Note that this measure takes values in $[-\frac{1}{|Q|-1}, 1]$. The minimum value of $-\frac{1}{|Q|-1}$ is achieved when the algorithm performs with an accuracy of 0, i. e., it is worse than random guessing, and the maximum value of 1 when the learner always predicts correctly. A value of 0 indicates that the learner performs similar to random guessing. This measure was derived using the "correction for guessing" formulation [28].

B.2 Subset Choice

For the subset choice setting, we introduce accuracy measures in terms of a choice task Q and two corresponding choices $C, \widehat{C} \subseteq Q$ for Q. Here, C may be thought of as the ground-truth choice for Q and $\widehat{C}$ as a prediction made by a learner. In contrast to the singleton choice setting, these measures do not depend on a utility function. For the sake of convenience, we suppose Q, C and $\widehat{C}$ to be arbitrary but fixed in the following. To prepare some of the measures, let us formally define the quantities *true positives* ($\widehat{TP}$), *true negatives* ($\widehat{TN}$), *false positives* ($\widehat{FP}$) and *false negatives* ($\widehat{FN}$) via

$$\begin{aligned}
\widehat{TP}(Q, C, \widehat{C}) &\coloneqq \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{x} \in Q} [\![\boldsymbol{x} \in C, \boldsymbol{x} \in \widehat{C}]\!], \\
\widehat{TN}(Q, C, \widehat{C}) &\coloneqq \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{x} \in Q} [\![\boldsymbol{x} \notin C, \boldsymbol{x} \notin \widehat{C}]\!], \\
\widehat{FP}(Q, C, \widehat{C}) &\coloneqq \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{x} \in Q} [\![\boldsymbol{x} \notin C, \boldsymbol{x} \in \widehat{C}]\!], \\
\widehat{FN}(Q, C, \widehat{C}) &\coloneqq \frac{1}{|Q|} \sum\nolimits_{\boldsymbol{x} \in Q} [\![\boldsymbol{x} \in C, \boldsymbol{x} \notin \widehat{C}]\!],
\end{aligned}$$

respectively. These quantities are similar to those used to define the confusion matrix in the case of binary classification [64].

Subset $0/1$ Accuracy The *Subset* $0/1$ *Accuracy* measures the number of times the ground-truth choice set C and the predicted choice set $\widehat{C}$ are exactly the same. This measure is used to measure how often the algorithms predictions match the complete choice set. Formally, it is defined as

$$\mathrm{m}_{\mathrm{SUBSET}}(Q, C, \widehat{C}) \coloneqq [\![C = \widehat{C}]\!].$$

Recall Recall is defined as the proportion of real positive cases that are correctly predicted positive [87]. In the field of information retrieval, it is the fraction of the relevant documents that are successfully retrieved. For our choice setting this can be defined as the fraction of objects from the ground-truth choice set C which chosen successfully or are present in the predicted choice set $\widehat{C}$, i. e., formally as

$$\mathrm{m}_{\mathrm{RE}}(Q, C, \widehat{C}) \coloneqq \frac{\widehat{TP}(Q, C, \widehat{C})}{\widehat{TP}(Q, C, \widehat{C}) + \widehat{FN}(Q, C, \widehat{C})}$$

Precision *Precision* denotes the proportion of predicted positive labels that are correct [87]. For the choice setting, this can be defined as the fraction of objects from the predicted choice set $\widehat{C}$ that are actually chosen by the decision maker or that are present in the ground-truth choice set C. Formally, it is defined as:

$$\mathrm{m}_{\mathrm{PR}}(Q, C, \widehat{C}) \coloneqq \frac{\widehat{TP}(Q, C, \widehat{C})}{\widehat{TP}(Q, C, \widehat{C}) + \widehat{FP}(Q, C, \widehat{C})}$$

F_1-Measure The *F_1-measure* is defined as the harmonic mean of precision and recall:

$$\mathrm{m}_{F_1}(Q, C, \widehat{C}) := \frac{2\,\mathrm{m}_{\mathrm{PR}}(Q, C, \widehat{C})\,\mathrm{m}_{\mathrm{RE}}(Q, C, \widehat{C})}{\mathrm{m}_{\mathrm{PR}}(Q, C, \widehat{C}) + \mathrm{m}_{\mathrm{RE}}(Q, C, \widehat{C})}$$

It can also be expressed in form of the confusion matrix quantities as follows [64]:

$$\mathrm{m}_{F_1}(Q, C, \widehat{C}) = \frac{2\widehat{TP}(Q, C, \widehat{C})}{2\widehat{TP}(Q, C, \widehat{C}) + \widehat{FN}(Q, C, \widehat{C}) + \widehat{FP}(Q, C, \widehat{C})}$$

Informedness The *informedness* is a measure proposed by Powers [88, 87], which is, in contrast to the F_1-measure, unbiased with respect to the population prevalence of positives. It specifies the probability that the learner makes an informed prediction if compared to chance and is formally defined as

$$\mathrm{m}_{\mathrm{Inf}}(Q, C, \widehat{C}) := \frac{\widehat{TP}(Q, C, \widehat{C})}{\widehat{TP}(Q, C, \widehat{C}) + \widehat{FN}(Q, C, \widehat{C})} + \frac{\widehat{TN}(Q, C, \widehat{C})}{\widehat{TN}(Q, C, \widehat{C}) + \widehat{FP}(Q, C, \widehat{C})} - 1$$

A very desirable property of this measure is that it is exactly 0 in case the learner is guessing or is constant.

AUC-ROC The *AUC-ROC* is a performance measure, which estimates the capacity of a classification model to distinguish between two classes [35, 74]. It computes the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one [74]. It is estimated by computing the area under the ROC-curve, which is created by plotting the true positive rate $\mathrm{m}_{\mathrm{TPR}}$ against the false positive rate $\mathrm{m}_{\mathrm{FPR}}$, where

$$\mathrm{m}_{\mathrm{TPR}}(Q, C, \widehat{C}) := \frac{\widehat{TP}(Q, C, \widehat{C})}{\widehat{TP}(Q, C, \widehat{C}) + \widehat{FN}(Q, C, \widehat{C})},$$

$$\mathrm{m}_{\mathrm{FPR}}(Q, C, \widehat{C}) := \frac{\widehat{FP}(Q, C, \widehat{C})}{\widehat{TN}(Q, C, \widehat{C}) + \widehat{FP}(Q, C, \widehat{C})}\ .$$

A very desirable property of this measure is that it exactly 0.5 in case the learner is guessing.

C Additional Experimental Details

In this section, we will now list all experimental details which were excluded from the main paper for conciseness reasons. First, we explain the process of nested cross-validation using the hyperparameter optimization in detail. Then we explain different hyperparameters which were tuned for different models and which parameters were kept fixed. Lastly, we explain the design generalization experiment.

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 9. 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*. First, we configure the given the learner M with the default parameters p_d described in the next section. Then we generate 5 sets of training $\mathcal{D}_k$ and test dataset $\mathcal{D}_{Tk}$ $\forall k \in [5]$ and the process which is used to generate a train-test set for k is described in Table 1.

Hyperparameter Optimization The training set $\mathcal{D}_k$ is used to first identify the best hyperparameters using 3-fold stratified cross-validation, and then to train the final learner for out-of-sample evaluation. The hyperparameter optimizer picks hyperparameters from the ranges in Table 3 (p_i) for the i^{th} iteration. In the inner loop $1 \leq j \leq 3$, we split the full training dataset $\mathcal{D}_k$ into train set (D_{kj} 90 % of $\mathcal{D}_k$) and validation dataset (V_{kj} 10 % of $\mathcal{D}_k$) using the stratified shuffle split. For the given hyperparameters p_i, we train the model on the train set (D_{kj}) and evaluate on the validation dataset V_{kj} using the target loss function. We use the $1 - F_1$-measure for general subset choice and the 1-categorical accuracy for singleton choice as the target loss to evaluate the hyperparameter configuration. We calculate the mean loss $\ell_i = \mathrm{mean}(l_1, l_2, l_3)$ for the given hyperparameters p_i. The optimization loop is run for 100 iterations to validate 100 sets of hyperparameters, in order to acquire the optimal parameters p_b for the given learning model.

Out-of-Sample Evaluation Finally, after optimization, we configure the learners M using the best found hyperparameters p_b and the remaining default parameters p_d. Then, we train the model M on the complete training dataset $\mathcal{D}_k$ and evaluate on the test dataset $\mathcal{D}_{Tk}$ using different evaluation measures m defined in Appendix B. To obtain a good estimate of the mean performance and an estimate for the standard deviation, we repeat this procedure 5 times using outer cross-validation. For each fold $k \in [K], K = 5$, we get the evaluated value a_k and calculate the mean and the standard deviation of the performance measure m.

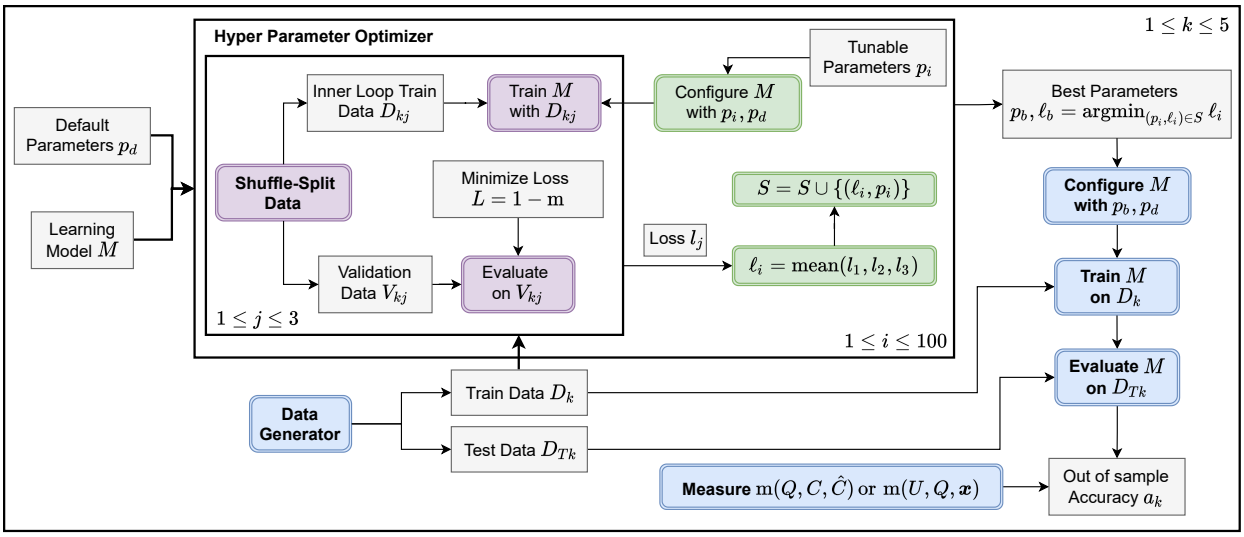


Figure 9: Overview of the complete evaluation pipeline.

Table 3: 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]

Hyperparameters & Inference We will now describe the specific hyperparameters we optimize and which ranges of values we consider (see Table 3 for an overview). For probabilistic models, we also describe how the inference is done. For all neural network models, we make use of the following techniques:

- We use either rectified linear units (ReLU) non-linearities in conjunction with batch normalization (BN) [53] or self-normalizing linear units (SELU) non-linearities [62] for each hidden layer.
- Regularization: L_2 penalties are applied and the corresponding regularization strength is tuned.
- Optimizer: stochastic gradient descent (SGD) with Nesterov momentum [79].
- A step-decay function is used for the learning rate annealing schedule. The decay factor is tuned [31].

The step-decay function drops the learning rate by a factor after a certain number epochs [31]. Formally, it is defined as:

$$lr = lr_0 \cdot d_r^{\left\lfloor \frac{e}{e_{\text{drop}}} \right\rfloor},$$

where lr_0 is the initial learning rate, $0 < d_r < 1$ is the rate with which the learning rate should be reduced, 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 epochs the neural networks are trained for to 1000.

The hyperparameters of each algorithm were tuned using the package scikit-optimize [49]. Apart from the number of hidden layers and units, we also tune the learning rate of the stochastic gradient descent optimizer, regularization strength and batch size (fraction of training examples used for estimating the gradient in one iteration). We also tune the drop-rate d_r and epoch-drop e_{drop} for the step-decay function used by the Stochastic gradient descent optimizer by the neural networks. For PairwiseSVM, we tune the value of the penalty parameter C of the error term, and another is *tol* (*tol* in scikit-learn) which is the tolerance for the stopping criteria of the optimization algorithm [84]. All of the different GEV models are implemented in PyMC3 a library for facilitating Markov Chain Monte Carlo estimation of the posterior distribution [97]. An overview of all the hyperparameters and their admissible ranges is shown in Table 3.

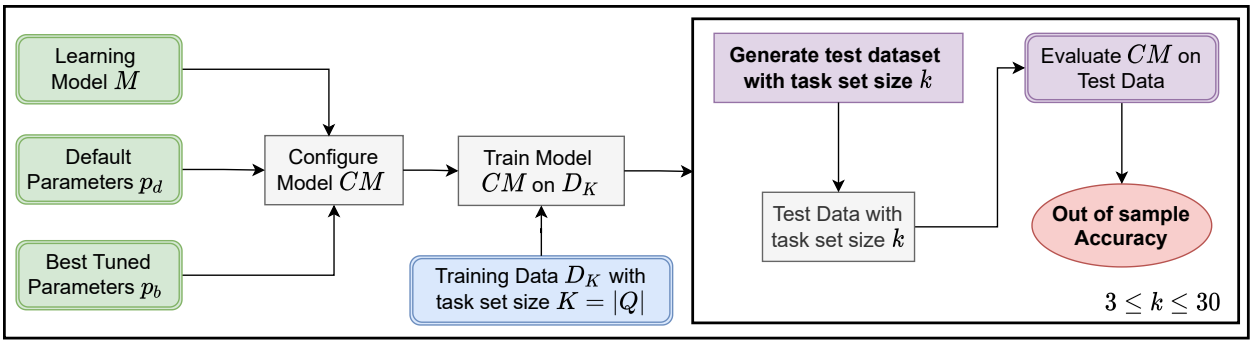


Figure 10: Design of the generalization experiments.

Table 4: 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

Threshold Tuning In order to set the threshold for the subset choice models (4), we tune the threshold for all models on a small validation set. Obviously, an optimal value for t will depend on the underlying target loss function. Our main target loss is the (micro-averaged) F_1-measure (21), which balances precision and recall of the predictions [66, 124, 121]. Koyejo et al. [64] show 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 to the multi-label classification setting is the absence of a fixed set of labels. Instead, we have a dynamically changing set of objects. Thus, it only makes sense to consider micro-averaged performance metrics.

D Design of the Generalization Experiment

The second experimental setup is designed to gauge the generalization capability of the learning models by measuring the accuracy obtained by a trained model on unseen task set sizes. To this end, we vary the task set sizes from 3 to 30 as shown in Figure 10.

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 CM on different test datasets $\mathcal{D}_k$ containing the task sets of sizes in S ($|Q| = k \in S$) as described in Table 4.

E Synthetic Datasets

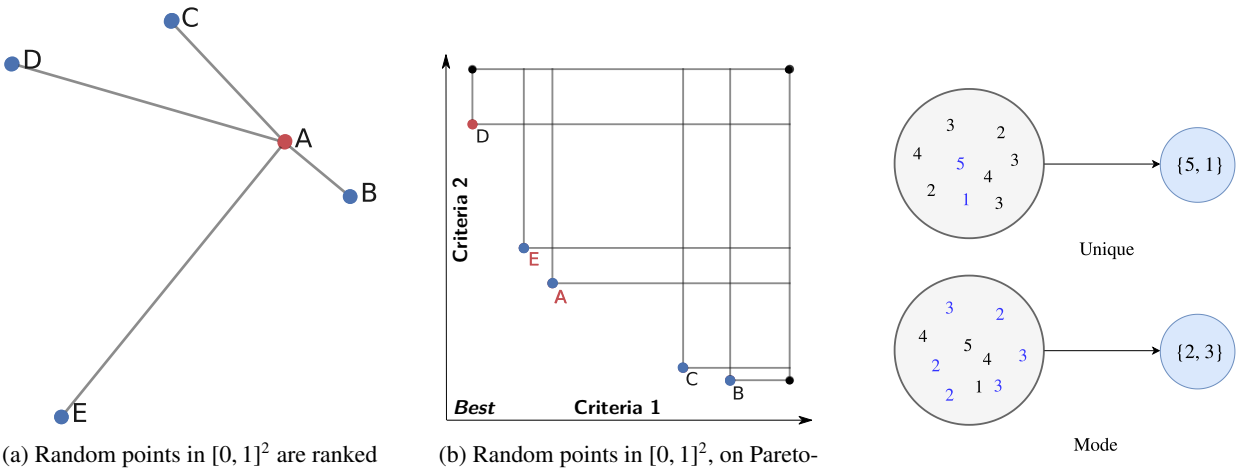
In this section, we will formally describe the process of generating the datasets for the experimental evaluation. In the case of synthetic datasets, this entails the complete process by which the objects and queries are generated.

E.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, cf. the illustration of a choice set Q of size 5 and its medoid in Figure 11a. As it depends on its distance to any other point from Q, the medoid of Q 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

$$\left\{ Q \subset [0,1]^d : |Q| = n \text{ and } |c_{\text{medoid}}(Q)| = 1 \right\}$$



(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 thier contributions

(c) Example for unique and mode function on MNIST digits

Figure 11: Examples for synthetic datasets

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 $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ uniformly at random from $[0,1]^d$ until $Q = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ has size n and a unique medoid. Regarding that this condition is already fulfilled with probability 1 after sampling $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ only once, this method is efficient.

E.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 $\boldsymbol{x} \in Q$ which are not dominated by any $\boldsymbol{y} \in Q \setminus \{\boldsymbol{x}\}$, wherein $\boldsymbol{x}$ was said to dominate $\boldsymbol{y}$ if $\forall i \in [d]\colon x_i \leq y_i$ and $\exists i \in [d]\colon x_i < y_i$. Figure 11b 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 $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_n$ i.i.d. uniformly at random from $\{\boldsymbol{x} \in \mathbb{R}^d : \|x\| \leq 1\}$
2. Draw i.i.d. samples $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_n$ from $N(\mathbf{0}, \boldsymbol{I}_d)$, the standard Gaussian distribution on $\mathbb{R}^d$, and define $\boldsymbol{x}_i := \boldsymbol{\mu}_i + \boldsymbol{\xi}_i$ for each $i \in [n]$.
3. Choose $Q := \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ and $C := c_{\text{Pareto}}(Q)$.

Hypervolume In Section 6.3.3 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, cf. Section 6.3.3 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 11b, 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 if we remove the point D from the choice set, it increases the contribution of the point E for 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

$$|Q| = n, \ \forall \boldsymbol{x} \in Q\colon (\|\boldsymbol{x}\| = 1 \text{ and } \forall i \in [d]\colon x_i \leq 0) \text{ and } |c_{\text{HypVol}}(Q)| = 1,$$

and then defining $C_i := c_{\text{HypVol}}(Q_i)$ afterwards. Similarly, as in the construction of the Medoid data set, sampling can be done via the acception-rejection method.

E.3 MNIST Number Problems

In this section, we will describe the process of generating different semisynthetic datasets using the MNIST dataset [65].

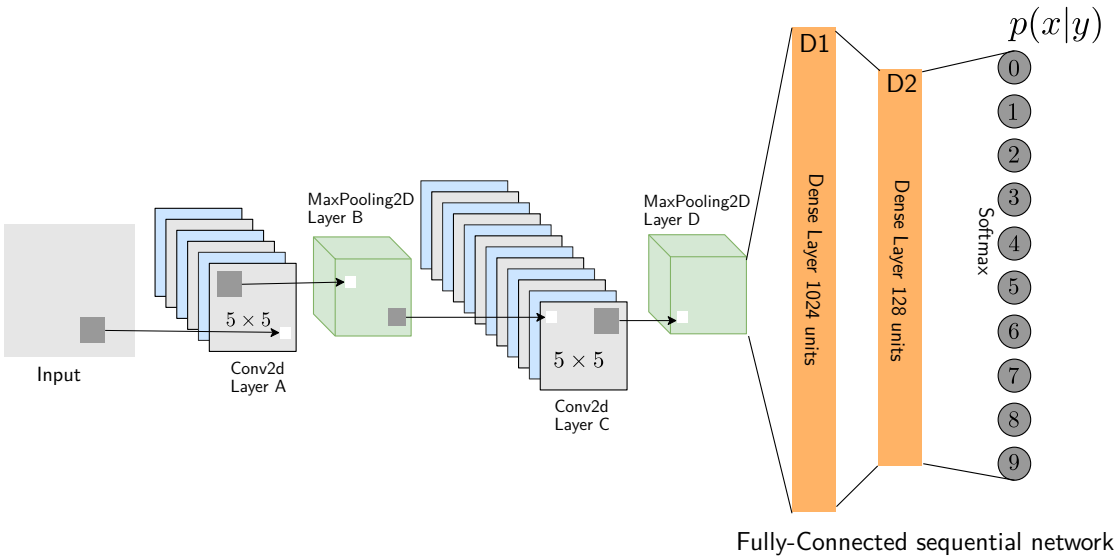


Figure 12: CNN-For converting MNIST images to high-level features

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 12 [43, 53]. The 2-D convolutional layer is of kernel-size 5×5 using rectified linear units (ReLU) non-linear activation function and l-2 regularization and 2-D max-pooling 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 [43]. 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 [43]. 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 which 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 used to generate choices depend on all other objects in the given task set Q, thus making the datasets highly context-dependent.

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 11c). 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 $(\boldsymbol{x}_{i,1}, l_1), \ldots, (\boldsymbol{x}_{i,n}, l_n)$ from $\mathcal{D}_M$, let $\boldsymbol{l}_i \coloneqq (l_1, \ldots, l_n)$ and $Q_i \coloneqq \{\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,n}\}$
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 $\boldsymbol{k} \coloneqq \{k_0, \ldots, k_9\}$ 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).

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 11c). 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 $(\boldsymbol{x}_{i,1}, l_1), \ldots, (\boldsymbol{x}_{i,n}, l_n)$ uniformly at random from $\mathcal{D}_M$, abbreviate $\boldsymbol{l}_i \coloneqq \{l_1, \ldots, l_n)$ and let $Q_i \coloneqq \{\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,n}\}$.
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 $\boldsymbol{k} \coloneqq \{k_0, \ldots, k_9\}$ 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.,
$$C_i \coloneqq \left\{\arg\max_{\boldsymbol{x} \in C_i} \cos^{-1} \frac{\boldsymbol{x} \cdot \boldsymbol{w}}{\|\boldsymbol{x}\|\|\boldsymbol{w}\|}\right\}$$

E.4 Tag Genome Dataset

The GroupLens Research group released many datasets collected from the MovieLens website[9] for research in the field of recommender systems [47]. As of August 2017, the full dataset collected from this website consists of 26 000 000 ratings and 750 000 tags applied to 45 000 movies by 270 000 users [47]. One of the datasets is the Tag Genome dataset[10], which provides real-valued features to characterize the movies [117].

Tags are meta-data in the form of keywords, which help to describe an object (such as movie, music, books). In recent years tagging has gained popularity due to the growth of social networking websites and web search engines [105]. 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 [118]. 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 13). Currently, this dataset consists of around 12 million relevance scores across 1128 tags applied on 10 993 movies.

Framework According to Vig, Sen, and Riedl [117] 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 13).

[9] https://movielens.org/
[10] This dataset is available on https://grouplens.org/datasets/movielens/

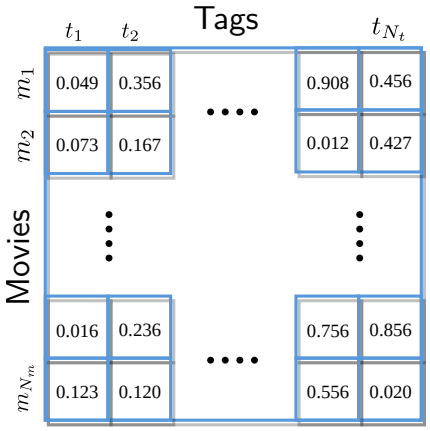


Figure 13: Structure of the Tag Genome dataset.

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., 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 in [117] to measure the similarity between two movies. The weight vector $\boldsymbol{w}$ is defined in such a way 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 [117]. A log-transform is applied to both values to bring them closer to the normal distribution. The weighted cosine similarity between two movies his defined as:

$$\text{sim}_{\text{ws}}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{w}) = \frac{\sum_{k=1}^{N_t} w_k x_{ik} x_{jk}}{\sqrt{(\sum_{k=1}^{N_t} w_k x_{ik}^2)} \cdot \sqrt{(\sum_{k=1}^{N_t} w_k x_{jk}^2)}} \quad , \tag{28}$$

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 [1, 21]. 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 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 $C_1, \dots, C_N$ 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
   $$C_i \coloneqq \left\{\arg\min_{\boldsymbol{x}_{i,j} \in Q \setminus \{\boldsymbol{r}\}} \operatorname{sim}_{\text{ws}}(\boldsymbol{r}, \boldsymbol{x}_{i,j}, \boldsymbol{w})\right\}$$
   (b) For the *Tag Genome Similar Movie* dataset, we select for the task Q_i the singleton choice set
   $$C_i \coloneqq \left\{\arg\max_{\boldsymbol{x}_{i,j} \in Q \setminus \{\boldsymbol{r}\}} \operatorname{sim}_{\text{ws}}(\boldsymbol{r}, \boldsymbol{x}_{i,j}, \boldsymbol{w})\right\},$$
   which consists of the one element from Q_i, that is most similar to $\boldsymbol{r}$.

F Real-World Datasets

Some widely used benchmark-datasets available for solving this task are LETOR and SUSHI [89, 58]. In the following sections, we briefly describe these datasets and the process we use to generate *singleton* and *subset choice* datasets.

F.1 LETOR Datasets

LETOR[11] 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 [89]. 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[12] to create the *singleton choice* dataset.

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 [89]. 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 labelled documents. These datasets consist of 46 features extracted from a query and document constructing an object called *query-document* and each pair is labelled with a relevance score in $\{0, 1, 2\}$, indicating how relevant the document is to the respective query as shown in Figure 14a. A relevance score of 0 means that the document is not relevant, 1 means relevant and 2 means very relevant to the query. For this dataset, the goal of the choice problem 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 $\mathcal{D}_S = \left\{(\tilde{Q}_1, \boldsymbol{l}_1), \dots, (\tilde{Q}_N, \boldsymbol{l}_N)\right\}$, is represented as set of tuples $(\tilde{Q}_i, \boldsymbol{l}_i)$, where $\tilde{Q}_i = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ is the task set ($\boldsymbol{x}_i$ features extracted from *query-document*) and $\boldsymbol{l}_i = (l_1, \dots, l_n)$ 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 $i \in [N]$.

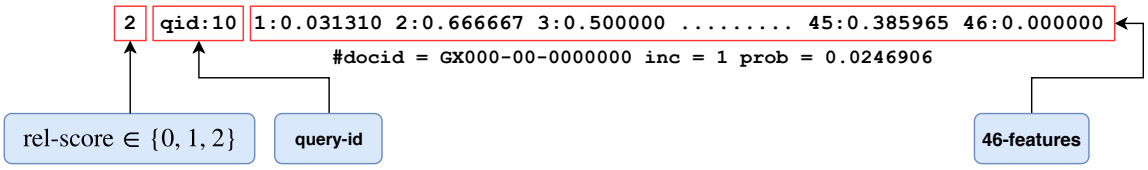
The size of the universal set of objects in the MQ2007 dataset is 59 570, i. e., $|\mathcal{X}| = 59570$ and the MQ2008 dataset is 564, i. e., $|\mathcal{X}| = 12102$. These datasets have been partitioned into 5 parts by Qin and Liu [89], such that $\mathcal{D}_S = \mathcal{D}_{S1} \cup \mathcal{D}_{S2} \cup \mathcal{D}_{S3} \cup \mathcal{D}_{S4} \cup \mathcal{D}_{S5}$. 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 5.

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 $C_i \coloneqq \left\{\boldsymbol{x}_j \in \tilde{Q}_i : l_j \in \{1, 2\}\right\}$ as the corresponding choice set for each instance $(\tilde{Q}_j, \boldsymbol{l}_j) \in \mathcal{D}_S \setminus \mathcal{D}_{Si}$. For training the choice model, we sub-sample 10 objects from each query instance $\tilde{Q}_i$ to construct the task sets. Note, that we still evaluate the models on the corresponding test choice dataset, which consists of all original queries for each fold as described in Table 5.

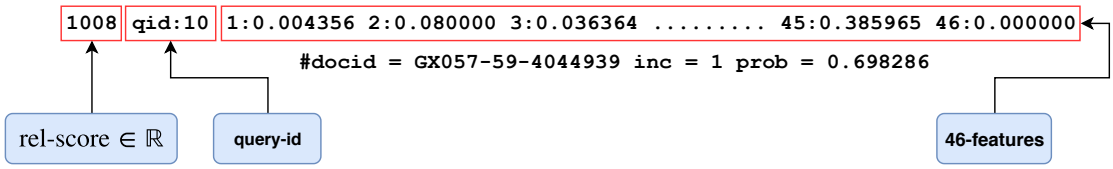
LETOR Listwise Datasets The format of both listwise datasets is the same as the supervised one. There are about 1700 queries in MQ2007-list and about 800 queries in MQ2008-list with each *query-document* pair consisting of 46 features. In this dataset, all the documents for each query are labelled with a real-valued relevance score instead of the multiple level relevance judgments as shown in Figure 14b. The documents on top positions in the ground truth permutation have larger value of the relevance degree.

[11] Version 4.0

[12] These datasets are available on https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/



(a) MQ2007/MQ2008 format



(b) MQ2007-list/MQ2008-list format

Figure 14: LETOR datasets formats [89]

Table 5: 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 $\lvert Q\rvert$	# Instances $\lvert\mathcal{D}_S\rvert$	# Features	# Objects $\lvert Q\rvert$
			1455	46	[6, 147]	564	46	[5, 121]

Structure The dataset consists of a universal set of objects $\boldsymbol{x} \in \mathcal{X}$. Each instance of these datasets $\mathcal{D}_L = \left\{(\tilde{Q}_1, \boldsymbol{l}_1), \ldots, (\tilde{Q}_N, \boldsymbol{l}_N)\right\}$, is represented as a set of tuples $(\tilde{Q}_i, \boldsymbol{l}_i)$, where $\tilde{Q}_i = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is the task set ($\boldsymbol{x}_i$ features extracted from *query-document*) and $\boldsymbol{l}_i = (l_1, \ldots, l_n)$ 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 6: 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 $(\tilde{Q}_j, \boldsymbol{l}_j) \in \mathcal{D}_L \setminus \mathcal{D}_{Li}$. For training the SCM we sub-sample 10 objects from each

Table 7: 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]

query instance $\tilde{Q}_i$ to construct the task sets. Note that 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 5.

F.2 Expedia Hotel Dataset

Expedia released a dataset on the Kaggle website as a competition and for research purposes[13]. 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 position bias of the existing algorithm. 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 labelled with a relevance score of 0, 1 or 2, indicating how relevant the hotel is to the respective query or 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 14. For this dataset, we define the learning target to be the set of relevant hotels (clicked and/or booked). Since for each query, the number of hotels displayed is different, this dataset consists of different task sizes.

Structure The dataset consists of a universal set of objects $\boldsymbol{x} \in \mathcal{X}$. Each instance of the datasets $\mathcal{D}_E = \left\{(\tilde{Q}_1, \boldsymbol{l}_1), \ldots, (\tilde{Q}_N, \boldsymbol{l}_N)\right\}$, is represented as a set of tuples $(\tilde{Q}_i, \boldsymbol{l}_i)$, where $\tilde{Q}_i = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is the task set ($\boldsymbol{x}_i$ features extracted from *hotel*) and $\boldsymbol{l}_i = (l_1, \ldots, l_n)$ 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 which have missing values, and we removed the features which consist of more than 50 % missing values. For the remaining 3 features which have of 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 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 $C_i \coloneqq \{\boldsymbol{x}_j \in \tilde{Q}_i : l_j \in \{1, 2\}\}$ as the corresponding choice set for each instance $(\tilde{Q}_i, \boldsymbol{l}_i) \in \mathcal{D}_E$. The models are trained on the sampled training dataset and corresponding test dataset using 5-fold stratified cross-validation as described in Table 7.

Singleton Choice In order to create the singleton choice dataset, we just 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 $(\tilde{Q}_i, \boldsymbol{l}_i) \in \mathcal{D}_E$.

The models are trained on the sampled training dataset and corresponding test dataset using 5-fold stratified cross-validation as described in Table 7. Note, the instances where the hotel was not booked at all were discarded and only the instances where there was booking were considered.

F.3 SUSHI Dataset

SUSHI[14] 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 consist of 10 most famous sushi and B consists of top 100 sushi famous in Japan. Individuals were asked to provide the preferences in form total

[13] These datasets are available on https://www.kaggle.com/c/expedia-personalized-sort/data

[14] This dataset can be downloaded from http://www.kamishima.net/sushi/

Table 8: 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

order for items in set A, and a real numbered 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.

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 $\mathcal{D}_S = \{(Q_1, \pi_1), \ldots, (Q_N, \pi_N)\}$, are represented as a set of tuples (Q_i, π_i), where $Q_i = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ 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 is a binary feature, which 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 suchi. 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 8 for 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 a 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 $\mathcal{D}_{S\,DC} = \{(Q_1, C_1), \ldots, (Q_N, C_N\}$ with $N = |\mathcal{D}_S|$ instances, such that $|Q_k| = 10$ and $C_k := \{\boldsymbol{x}_{\pi_i(1)}\}$ 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.

G Detailed Experimental Results

The following Tables 9 to 11 contain all experimental results as discussed in Section 6.4 in numeric form for additional evaluation measures.

Table 9: Results for the general subset choice models (mean and standard deviation of different measures, measured across 5 outer cross-validation folds). Best entry for each measure 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 ± 0.000**
	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 ± 0.000
	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 ± 0.000
	GENLINEARMODEL	0.585 ± 0.008	0.044 ± 0.005	0.633 ± 0.013	0.952 ± 0.007
	ALLPOSITIVE	0.232 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
Pareto-front-5D	FETA-NET	0.826 ± 0.005	0.001 ± 0.000	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 ± 0.000	0.006 ± 0.000	0.581 ± 0.006	0.923 ± 0.000
	PAIRWISESVM	0.839 ± 0.000	0.002 ± 0.000	0.491 ± 0.021	0.895 ± 0.000
	GENLINEARMODEL	0.826 ± 0.029	0.002 ± 0.001	0.402 ± 0.225	0.738 ± 0.351
	ALLPOSITIVE	0.775 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
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 ± 0.000
	RANKNET	0.562 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.504 ± 0.001
	PAIRWISESVM	0.562 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.511 ± 0.006
	GENLINEARMODEL	0.562 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.508 ± 0.004
	ALLPOSITIVE	0.562 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
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 ± 0.000	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 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.503 ± 0.002
	PAIRWISESVM	0.597 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.509 ± 0.006
	GENLINEARMODEL	0.597 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.497 ± 0.004
	ALLPOSITIVE	0.597 ± 0.000	0.003 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
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 ± 0.000	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 ± 0.000	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 ± 0.000	0.500 ± 0.000
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	0.000 ± 0.000	0.500 ± 0.000
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 ± 0.000	0.004 ± 0.007	0.503 ± 0.102
	ALLPOSITIVE	0.106 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000

Table 10: Mean and standard deviation of the accuracies on the singleton choice data (measured across 5 outer cross-validation folds). Best entry for each measure marked in bold.

Dataset	SCM	ACCURACY	TOP-3	TOP-5
Medoid	FETA-NET	0.846 ± 0.010	0.994 ± 0.001	**1.000 ± 0.000**
	FATE-NET	**0.881 ± 0.007**	**0.996 ± 0.001**	**1.000 ± 0.000**
	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 ± 0.000
	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 ± 0.000	0.195 ± 0.004	0.500 ± 0.001
	ML	0.003 ± 0.000	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 ± 0.000**
	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 11: Mean and standard deviation of the accuracies on the singleton choice data (measured across 5 outer cross-validation folds). Best entry for each measure 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 ± 0.000	0.405 ± 0.001	0.550 ± 0.000
	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