Learning Choice Functions with Gaussian Processes

Alessio Benavoli[1] **Dario Azzimonti**[2] **Dario Piga**[2]

[1]School of Computer Science and Statistics, Trinity College Dublin, Ireland
[2]Dalle Molle Institute for Artificial Intelligence (IDSIA), USI/SUPSI, Lugano, Switzerland

Abstract

In consumer theory, ranking available objects by means of preference relations yields the most common description of individual choices. However, preference-based models assume that individuals: (1) give their preferences only between pairs of objects; (2) are always able to pick the best preferred object. In many situations, they may be instead choosing out of a set with more than two elements and, because of lack of information and/or incomparability (objects with contradictory characteristics), they may not be able to select a single most preferred object. To address these situations, we need a choice model which allows an individual to express a set-valued choice. Choice functions provide such a mathematical framework. We propose a Gaussian Process model to learn choice functions from choice data. The model assumes a multiple utility representation of a choice function based on the concept of Pareto rationalization, and derives a strategy to learn both the number and the values of these latent multiple utilities. Simulation experiments demonstrate that the proposed model outperforms the state-of-the-art methods.

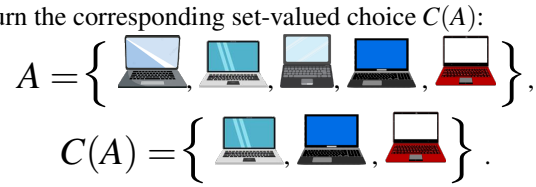
1 INTRODUCTION

We are interested in learning the behavior of an individual (e.g., a consumer), we call her Alice, who is faced with the problem of choosing from among a set of objects, e.g., laptops:

This is an important problem for instance in computational advertising, and for personalisation of products and services. In consumer theory [Kreps et al., 1990], ranking available objects by means of preference relations yields the most common description of individual choices:

Preference-based models assume that Alice is always able to pick the best preferred object from a set of objects. However, in many contexts, Alice is faced with the problem of dealing with several contradictory primitives. For example, if the objects are laptops, Alice has to consider criteria such as: speed, drive-capacity and weight. In other contexts, Alice may not have sufficient knowledge to pick up the best preferred object. For instance, in the laptop case, she may not know if she will use the laptop for expensive computer experiments or for everyday office work.

To deal with incomparability of objects, we need a choice model which allows Alice to express a set-valued choice. *Choice functions* provide such a mathematical framework, as well as allowing individuals to choose out of a set with more than two elements. For any given set of objects A, they return the corresponding set-valued choice $C(A)$:



In the general interpretation of choice functions, the statement that an object in A is rejected (that is, $\notin C(A)$) means that there is at least one object in A that Alice strictly prefers over .[1] On the other hand, any two objects in $C(A)$ are deemed to be incomparable by Alice.

We represent each object by the feature vector $\mathbf{x} \in \mathbb{R}^c$ of its characteristics (e.g., for laptops, speed, weight, etc.) and propose a Gaussian process (GP) model to learn choice functions from choice data $\{(C(A_s), A_s),\ s = 1, \ldots, m\}$.

The uncertainty estimates automatically provided by GPs

[1]Alice is not required to tell us which object(s) in $C(A)$ dominate .

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

can be useful in decision making scenarios where it is important to understand the reliability of the predictions. Moreover, GP models are well adapted to applications such as Bayesian optimization with implicit feedback [Benavoli et al., 2021a, 2023].

Our main contributions are:

We propose a generalisation of the preference learning model by Chu and Ghahramani [2005] to choice functions. This generalisation assumes a multiple utility representation of a choice function based on the concept of Pareto rationalization [Moulin, 1985]: Alice picks the Pareto optimal objects based on the value of latent multiple utilities.

Learning choice functions via a Pareto embedding was originally proposed by Pfannschmidt and Hüllermeier [2020], but using a hinge loss and a neural network based model. We will show that our GP-based approach results into a more accurate and robust model.

The output of a choice function is a choice set and it is invariant with respect to permutations of its elements. We will show that this determines a number of challenges and propose ways to address this issue.

Finally, we propose a method to learn the number of latent multiple utilities via Pareto Smoothed Importance sampling Leave-One-Out (PSIS-LOO) cross-validation [Vehtari et al., 2017]. Exact cross-validation requires re-fitting the model with different training sets. Instead, PSIS-LOO can be computed efficiently using samples from the posterior.

2 BACKGROUND

To begin, let $\mathscr{X}$ represent a set of objects. It is quite typical in applications to think of $\mathscr{X}$ as a subset of $\mathbb{R}^c$, where c is the number of features of the object. The standard way to model Alice (the consumer) is with a preference relation. We present Alice with pairs of objects, $\mathbf{x}_i, \mathbf{x}_j \in \mathscr{X}$, and ask her whether $\mathbf{x}_i$ is better than $\mathbf{x}_j$ (see for instance Kreps et al. [1990], Fürnkranz and Hüllermeier [2010], Domshlak et al. [2011]). If Alice says that $\mathbf{x}_i$ is better than $\mathbf{x}_j$, we write $\mathbf{x}_i \succ \mathbf{x}_j$ and we say that $\mathbf{x}_i$ is strictly preferred to $\mathbf{x}_j$. In this paper, we will assume that no draws are allowed – no two distinct objects are equal – and therefore focus on strict preference relations.

An important result in *preference theory* establishes conditions under which a preference relation can be numerically represented. We refer to a value function that represents preferences as a *utility function*.

Definition 1. *For any preference relation $\succ$ on $\mathscr{X}$, the function $u : \mathscr{X} \to \mathbb{R}$ represents $\succ$ if*

$$\mathbf{x}_i \succ \mathbf{x}_j \quad \textit{iff} \quad u(\mathbf{x}_i) > u(\mathbf{x}_j). \tag{1}$$

We say that u is a utility function for $\succ$.

The relation $\succ$ admits a utility function representation iff[2] it is [Kreps et al., 1990, Ch. 2]:

- *Asymmetric*: if $\mathbf{x}_i \succ \mathbf{x}_j$ then $\neg(\mathbf{x}_j \succ \mathbf{x}_i)$;
- *Negatively transitive*: if $\mathbf{x}_i \succ \mathbf{x}_j$ then for any other element $\mathbf{x}_k \in \mathscr{X}$ either $\mathbf{x}_i \succ \mathbf{x}_k$ or $\mathbf{x}_k \succ \mathbf{x}_j$ or both.

A strict preference relation is said to be *consistent* – Alice is *rational* – when it satisfies the above two properties. It is immediate to verify that any consistent strict preference is also *transitive* and *acyclic* [Kreps et al., 1990, Ch. 2].[3]

Typical preference learning (PL) models can be divided in two categories: (1) those assuming that the preference relation is consistent and aiming to learn the underlying latent utility function, e.g., [Chu and Ghahramani, 2005, Houlsby et al., 2011, Benavoli et al., 2021a]; (2) those solving the problem as an augmented binary classification problem or constrained classification, for example, with Support Vector Machines (SVM), e.g., [Cohen et al., 1997, Herbrich et al., 1998, Aiolli and Sperduti, 2004, Har-Peled et al., 2002, Fiechter and Rogers, 2000].

In the first case, the goal is to learn $u : \mathscr{X} \to \mathbb{R}$ from m preferential observations:

$$\mathscr{D}_m = \{\mathbf{x}_l^{(s)} \succ \mathbf{x}_r^{(s)} : \ s = 1, \ldots, m\},$$

with $\mathbf{x}_l^{(s)} \neq \mathbf{x}_r^{(s)}$, $\mathbf{x}_l^{(s)}, \mathbf{x}_r^{(s)} \in \mathscr{X}$ and the subscripts l, r stay for "left hand side" term and, respectively, "right hand side" term of the inequality. These PL models assume a Gaussian Process (GP) prior on the latent utility u.

These PL models also account for the fact that Alice may make mistakes when stating her preferences and, therefore, violate asymmetry and/or negative transitivity. We therefore assume that the probability of correctly stating $\mathbf{x}_i \succ \mathbf{x}_j$ is a function of the difference $u(\mathbf{x}_i) - u(\mathbf{x}_j)$. This probability can be modelled by the following likelihood:

$$p(\mathbf{x}_i \succ \mathbf{x}_j | u) = \Phi\left(\frac{u(\mathbf{x}_i) - u(\mathbf{x}_j)}{\sigma}\right), \tag{2}$$

where $\Phi(\cdot)$ is the Cumulative Distribution Function (CDF) of the standard Normal distribution and $\sigma > 0$ is a scaling parameter. When $\sigma \to 0$, the CDF converges to an indicator function and (2) reduces to (1). For PL, this likelihood was originally proposed by Chu and Ghahramani [2005] and derived under a Gaussian noise model – see Supp. Mat. 4 for more details about the different interpretations of this likelihood.

A binary relation on $\mathscr{X} \times \mathscr{X}$ can be represented, more in general, through a two-argument function $q : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$

[2] Kreps et al. [1990] proves this result only for a finite $\mathscr{X}$. This result can be extended to infinite sets under some topological assumptions on $\mathscr{X}$ [Debreu, 1954]. In this paper, we assume that the set $\mathscr{X}$ is finite.

[3] Acyclic: if, for any finite number n, $\mathbf{x}_1 \succ \mathbf{x}_2$, $\mathbf{x}_2 \succ \mathbf{x}_3$, $\ldots$, $\mathbf{x}_{n-1} \succ \mathbf{x}_n$ then $\mathbf{x}_n \neq \mathbf{x}_1$.

[Shafer, 1974, Fishburn, 1988]. If $\mathbf{x}_i$ is in relation with $\mathbf{x}_j$ then $q(\mathbf{x}_i, \mathbf{x}_j) > 0$. Since in general $q(\mathbf{x}_i, \mathbf{x}_j) \neq q(\mathbf{x}_j, \mathbf{x}_i)$ we can equivalently write $q(\mathbf{x}_i, \mathbf{x}_j)$ as $q([\mathbf{x}_i, \mathbf{x}_j])$, that is as a function of the vector $[\mathbf{x}_i, \mathbf{x}_j]$. The function q can be interpreted as a "strength of preference", with values of $q([\mathbf{x}_i, \mathbf{x}_j])$ close to zero indicating a difficult decision – Alice cannot distinguish $\mathbf{x}_i, \mathbf{x}_j$. This is a natural generalization of representation results for consistent preferences discussed previously, in which case one can set $q([\mathbf{x}_i, \mathbf{x}_j]) = u(\mathbf{x}_i) - u(\mathbf{x}_j)$ for a utility function u.

Under this representation, PL can be formulated as a classification problem by rewriting $\mathscr{D}_m$ as the dataset (X, Y):

$$X = \begin{bmatrix} \mathbf{x}_l^{(1)} & \mathbf{x}_r^{(1)} \\ \mathbf{x}_l^{(2)} & \mathbf{x}_r^{(2)} \\ \vdots & \vdots \\ \mathbf{x}_l^{(m)} & \mathbf{x}_r^{(m)} \end{bmatrix}, \quad Y = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Indeed, most of the initial PL methods solved the PL problem as an augmented binary classification problem. The resulting classification function is not guaranteed to satisfy asymmetry and negative transitivity in general. However, for kernel-based methods, it is possible to derive classifiers that satisfy one or both of these properties.

Indeed, a GP prior on the latent utility $u(\mathbf{x}) \sim \mathrm{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ induces a GP prior on $q([\mathbf{x}_i, \mathbf{x}_j]) = u(\mathbf{x}_i) - u(\mathbf{x}_j)$ by linearity [Houlsby et al., 2011]: $q([\mathbf{x}_i, \mathbf{x}_j]) \sim \mathrm{GP}(0, k_p([\mathbf{x}_i, \mathbf{x}_j], [\mathbf{x}'_i, \mathbf{x}'_j])$, where

$$k_p([\mathbf{x}_i, \mathbf{x}_j], [\mathbf{x}'_i, \mathbf{x}'_j]) = k(\mathbf{x}_i, \mathbf{x}'_i) - k(\mathbf{x}_i, \mathbf{x}'_j) - k(\mathbf{x}'_i, \mathbf{x}_j) + k(\mathbf{x}_j, \mathbf{x}'_j), \tag{3}$$

which is called *preference kernel*. Functions q sampled from the above GP satisfy asymmetry and negative transitivity, and so do the GP classifier based on it. In the following, we refer to the GP PL-model based on this kernel as *Preferential GP* (PGP).

Pahikkala et al. [2010] instead, using a feature map view, derived a kernel

$$k_a([\mathbf{x}_i, \mathbf{x}_j], [\mathbf{x}'_i, \mathbf{x}'_j]) = k(\mathbf{x}_i, \mathbf{x}'_i) k(\mathbf{x}_j, \mathbf{x}'_j) - k(\mathbf{x}_i, \mathbf{x}'_j) k(\mathbf{x}'_i, \mathbf{x}_j), \tag{4}$$

satisfying asymmetry but not negative transitivity in general. This kernel is known as *intransitive preference kernel*. A PL model which employs a GP prior on q with kernel (4) has been recently proposed by Chau et al. [2022]. In the following, we refer to this model as GPGP.

2.1 CHOICE FUNCTIONS

The PL models discussed in the previous section assume that Alice gives her preferences between pairs of objects. In many situations, she will be instead choosing out of a set with more than two elements. In this more general case, Alice's choices can be formalised through the concept of choice functions. Let $\mathscr{Q}$ denote the set of all finite subsets of $\mathscr{X}$, then [Kreps et al., 1990]:

Definition 2. *A choice function C is a set-valued operator on sets of objects. More precisely, it is a map $C : \mathscr{Q} \to \mathscr{Q}$ such that, for any set of objects $A \in \mathscr{Q}$, the corresponding value of C is a subset $C(A)$ of A.*

It will be assumed throughout this paper that Alice is able to find a choosable object in every set she is presented with, and therefore $C(A) \neq \emptyset$ for all A.

It is convenient to introduce the set of rejected objects, denoted by $R(A)$, and equal to $A \backslash C(A)$. A rejection-function R is a useful tool to explain the behavioral meaning of choice functions. We can think of the decision maker, Alice, as eliminating the objects $\mathbf{x}_i$ in A that she considers to be bad. That is, she thinks that there is at least one object $\mathbf{x}_j \in A$ that is strictly better than $\mathbf{x}_i$. Note that, Alice is not required to tell us which object(s) in $C(A)$ she strictly prefers to $\mathbf{x}_i$

The interpretation of two objects $\{\mathbf{x}_j, \mathbf{x}_k\} \subseteq C(A)$ is that $\mathbf{x}_j$ and $\mathbf{x}_k$ are incomparable for Alice. She cannot reject either of them. Since we have assumed that no two distinct objects are equal, incomparability can arise for two reasons. First, the objects to be compared have multiple utilities for Alice. For example, if the objects are laptops, Alice may consider multiple utilities such as speed and weight. Second, incomparability can arise due to incompleteness [Seidenfeld et al., 2010, Van Camp et al., 2018],[4] which represents simply an absence of knowledge about the underlying utility function. We can model both these cases assuming there are multiple utility functions (due either to incomparability or incompleteness) and then interpret the statement $\{\mathbf{x}_i, \mathbf{x}_j\} \subseteq C(A)$ as "$\mathbf{x}_i$ and $\mathbf{x}_j$ are undominated in A in a Pareto sense".[5] This approach was originally proposed in Pfannschmidt and Hüllermeier [2020] to learn choice functions. The authors devise a differentiable loss function based on two hinge loss terms. Furthermore, they add two additional terms to the loss function: (i) an L^2 regularization term; (ii) a multidimensional scaling (MDS) loss to ensure that objects close to each other in the inputs space $\mathscr{X}$ will also be close in the embedding space $\mathbb{R}^d$. Overall the loss function is the sum of four terms weighted by four non-negative scalar paramet-

[4] Note that, differently from this paper, we consider choices over objects and not over gambles.

[5] Similar to preferences, a choice function needs to satisfy some consistency properties in order to be Pareto rationalisable [Moulin, 1985, Eliaz and Ok, 2006]. However, Pareto rationalisation is not the only way to rationalise choice functions. [Moulin, 1985], referring to previous work, showed that three properties (Chernoff, Expansion, Aizerman) are necessary and sufficient for Pareto rationalisation of choice functions. But one could also define a weaker form of rationality by only requiring a subset of these properties.

ers $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ which sum up to one. These weights are treated as hyperparameters of the learning algorithm. This loss function is then used to learn a (deep) multi-layer perceptron to represent the embedding. We refer to this model as *ChoiceNN*.

In the next section, we instead propose a GP model to learn choice functions from choice data. In Section 4.1, we will show that the GP-based model outperforms *ChoiceNN*.

Finally, it is worth mentioning that PL with more than two comparisons between objects was also considered by Siivola et al. [2021], the so-called batch-preference model. This model considers the case where a subject expresses preferences for a group of objects. However, the batch-preference model in [Siivola et al., 2021] assumes that two objects are always comparable and, therefore, as we will show in Section 3.3, this model assumes a single utility function.

3 METHODOLOGY

For each A, we interpret $C(A)$ as the *undominated set* in the *strong Pareto sense* with $R(A)$ being the set of dominated objects. In other words, we assume that there is a latent vector function $\mathbf{u}(\mathbf{x}) = [u_1(\mathbf{x}), \dots, u_d(\mathbf{x})]^\top$, for some finite dimension d, which embeds the objects $\mathbf{x}$ into a space $\mathbb{R}^d$. The choice set can then be represented through a Pareto set of strongly undominated objects:

$$\neg\left(\min_{i\in\{1,\dots,d\}}(u_i(\mathbf{o}) - u_i(\mathbf{v})) < 0,\ \forall \mathbf{o} \in C(A)\right), \forall \mathbf{v} \in R(A), \tag{5}$$

$$\min_{i\in\{1,\dots,d\}}(u_i(\mathbf{o}) - u_i(\mathbf{v})) < 0,\ \forall \mathbf{o}, \mathbf{v} \in C(A),\ \mathbf{o} \neq \mathbf{v}. \tag{6}$$

Condition (5) means that, for each object $\mathbf{v} \in R(A)$, it is not true ($\neg$ stands for logical negation) that all objects in $C(A)$ are worse than $\mathbf{v}$, i.e. there is at least an object in $C(A)$ which is strictly better than $\mathbf{v}$. It can equivalently be written as $\forall \mathbf{v} \in R(A),\ \exists \mathbf{o} \in C(A)$ such that $\min_{i\in\{1,\dots,d\}}(u_i(\mathbf{o}) - u_i(\mathbf{v})) > 0$.

Condition (6) means that, for each object in $C(A)$, there is no better object in $C(A)$. This requires that the latent functions values of the objects should be consistent with the choice function implied relations.

To account for errors in Alice's choices, we extend the likelihood in (2) to choice functions. Consider the vectors $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t]^\top$ with $\mathbf{x}_i \in \mathscr{X}$, $\mathbf{u}(\mathbf{x}_i) = [u_1(\mathbf{x}_i), u_2(\mathbf{x}_i), \dots, u_d(\mathbf{x}_i)]^\top$ and $\mathbf{u}(X) = [\mathbf{u}(\mathbf{x}_1), \mathbf{u}(\mathbf{x}_2), \dots, \mathbf{u}(\mathbf{x}_t)]^\top$, and the choice dataset

$$\mathscr{D}_m = \{(C(A_s), A_s) :\ \text{for } s = 1, \dots, m\},$$

where $A_s \subset X$ for each s. The likelihood is defined as

$$\begin{aligned} p(\mathscr{D}_m|\mathbf{u}(X)) &= \prod_{k=1}^{m} p(C(A_k), A_k|\mathbf{u}(X)) \\ &= \prod_{k=1}^{m} \prod_{\{\mathbf{o},\mathbf{v}\}\in C_\sharp(A_k)} \left(1 - \prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o}) - u_i(\mathbf{v})}{\sigma}\right)\right. \\ &\qquad \left. - \prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{v}) - u_i(\mathbf{o})}{\sigma}\right)\right) \\ &\prod_{\mathbf{v}\in R(A_k)}\left(1 - \prod_{\mathbf{o}\in C(A_k)}\left(1 - \prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o}) - u_i(\mathbf{v})}{\sigma}\right)\right)\right) \end{aligned} \tag{7}$$

where the notation $\{\mathbf{o}, \mathbf{v}\} \in C_\sharp(A_k)$ means that the pair $\{\mathbf{o}, \mathbf{v}\}$ is an element of $C_\sharp(A_k)$, which denotes the set of all possible 2-combination (without repetition) of the elements of the set $C(A_k)$. We assumed that the choice-statements $k = 1, \dots, m$ are conditionally independent given the utilities $\mathbf{u}(X)$, which is in line with our model based on Pareto rationalisation. The product in the first and second row in (7) is a probabilistic relaxation of (6). Note the negation $1 - \prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o})-u_i(\mathbf{v})}{\sigma}\right) - \prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{v})-u_i(\mathbf{o})}{\sigma}\right)$, which states that $\mathbf{o}$ does not dominate $\mathbf{v}$ and vice versa. The product in the last row in (7) is a probabilistic relaxation of (5). In Supp. Mat. 1, we discuss how to vectorise this likelihood.

Example 1. *To understand the error-model defined by the above likelihood, let us take four objects $\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3, \mathbf{o}_4$ and suppose they have the following utilities:*

$$\mathbf{u}(\mathbf{o}_1) = \begin{bmatrix}0.2\\0\end{bmatrix}, \mathbf{u}(\mathbf{o}_2) = \begin{bmatrix}0.1\\0.2\end{bmatrix}, \mathbf{u}(\mathbf{o}_3) = \begin{bmatrix}-1\\-1\end{bmatrix}, \mathbf{u}(\mathbf{o}_4) = \begin{bmatrix}-0.5\\-0.5\end{bmatrix}.$$

Assume Alice makes the following choices:

$$C(\{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3\}) = \{\mathbf{o}_1, \mathbf{o}_2\},\ \ C(\{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_4\}) = \{\mathbf{o}_1\},$$

making a mistake in the second statement (these choices are not Pareto rational). This might be because the objects $\{\mathbf{o}_1, \mathbf{o}_2\}$ are hard to tell apart since they have very similar utilities. Assuming $\sigma = 1$ we computed the likelihood for the two choices

$$p(\{\mathbf{o}_1, \mathbf{o}_2\}, \{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3\}|\mathbf{u}(X)) \approx 0.48,$$
$$p(\{\mathbf{o}_1\}, \{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_4\}|\mathbf{u}(X)) \approx 0.12.$$

This means that, given the above latent utilities, the probability that Alice jointly makes these choices is $0.48 \cdot 0.12 = 0.057$. For a given $\mathbf{u}(X)$, the probability of error increases with the parameter σ. Indeed, the probability $p(\{\mathbf{o}_1, \mathbf{o}_2\}, \{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3\}|\mathbf{u}(X))p(\{\mathbf{o}_1\}, \{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_4\}|\mathbf{u}(X))$ tends to zero for $\sigma \to 0$, since in this case the likelihood (7) *reduces to* (5)–(6) *and Alice's choice is Pareto irrational.*

Prior: Similarly to GP processes for multiclass classification [Williams and Barber, 1998], we model each latent

utility function in the vector $\mathbf{u}(\mathbf{x}) = [u_1(\mathbf{x}), \dots, u_d(\mathbf{x})]^\top$ as an independent GP:

$$u_i(\mathbf{x}) \sim \mathrm{GP}_i(0, k_i(\mathbf{x}, \mathbf{x}')), \quad i = 1, 2, \dots, d. \tag{8}$$

Each GP is fully specified by its kernel function $k_i(\cdot,\cdot)$, which defines the covariance of the latent function between any two points. The model parameters are the kernel parameters (lengthscales) in $k_i(\cdot,\cdot)$, and the scale parameter σ in the likelihood function. These parameters can be collected into a hyperparameter vector θ.

3.1 POSTERIOR AND PREDICTION

The posterior probability of $\mathbf{u}(X)$ is

$$p(\mathbf{u}(X)|\mathscr{D}_m) = \frac{p(\mathbf{u}(X))}{p(\mathscr{D}_m)} \prod_{k=1}^{m} p(C(A_k), A_k|\mathbf{u}(X)), \tag{9}$$

where the prior over the component of $\mathbf{u}$ is defined in (8), the likelihood is defined in (7) and the probability of the evidence is $p(\mathscr{D}_m) = \int p(\mathscr{D}_m|\mathbf{u}(X))p(\mathbf{u}(X))d\mathbf{u}(X)$. The posterior $p(\mathbf{u}(X)|\mathscr{D}_m)$ is intractable because it is not a GP. Contrarily to the case of binary preferences, the posterior is not a Skew Gaussian Process [Benavoli et al., 2020, 2021b]. Inference on $\mathbf{u}$ could be computed using approximation methods such as (i) the Laplace Approximation (LA) [MacKay, 1996]; (ii) Variational Inference (VI) [Opper and Archambeau, 2009, Hensman et al., 2015].

As discussed in Supp. Mat. 2, LA cannot be applied due to the so-called 'label switching' problem. Therefore, we resort to VI to learn at the same time the kernel hyperparameters θ and a Gaussian approximation of the posterior $p(\mathbf{u}|\mathscr{D}_m)$.[6]

Prediction and Inferences Let $X^* = \{\mathbf{x}_1^*, \dots, \mathbf{x}_p^*\}$ be a set including p test points and $\mathbf{u}(X^*) = [\mathbf{u}(\mathbf{x}_1^*), \dots, \mathbf{u}(\mathbf{x}_m^*)]^\top$. Under the GP prior assumption on u, the conditional predictive distribution $p(\mathbf{u}(X^*)|\mathbf{u}(X))$ is Gaussian and, therefore,

$$p(\mathbf{u}(X^*)|\mathscr{D}_m) = \int p(\mathbf{u}(X^*)|\mathbf{u}(X))p(\mathbf{u}(X)|\mathscr{D}_m)d\mathbf{u}(X) \tag{10}$$

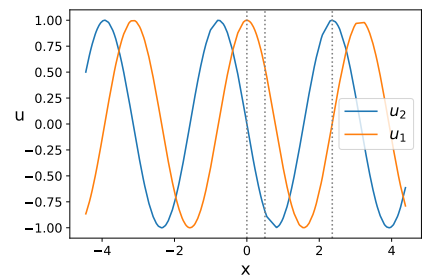
can be easily computed analytically using the VI posterior $p(\mathbf{u}|\mathscr{D}_m)$, which is Gaussian. In choice function learning, we are interested in the inference:

$$P(C(A^*), A^*|\mathscr{D}_m) = \int p(C(A^*), A^*|\mathbf{u}(X^*)) \, p(\mathbf{u}(X^*)|\mathscr{D}_m)d\mathbf{u}(X^*), \tag{11}$$

which returns the posterior probability that the agent chooses the objects $C(A^*)$ from the set of objects A^*. This probability can be easily computed via Monte Carlo sampling from the approximate posterior $p(\mathbf{u}(X^*)|\mathscr{D}_m)$, which is Gaussian.

[6] We implemented our model using automatic-differentiation in Jax [Bradbury et al., 2018]. Details are reported in Supp. Mat. 3.

Example 2. *We illustrate the overall model with an example. We consider the bi-dimensional utility function* $\mathbf{u}(x) = [\cos(2x), -\sin(2x)]$ *with* $x \in \mathbb{R}$.



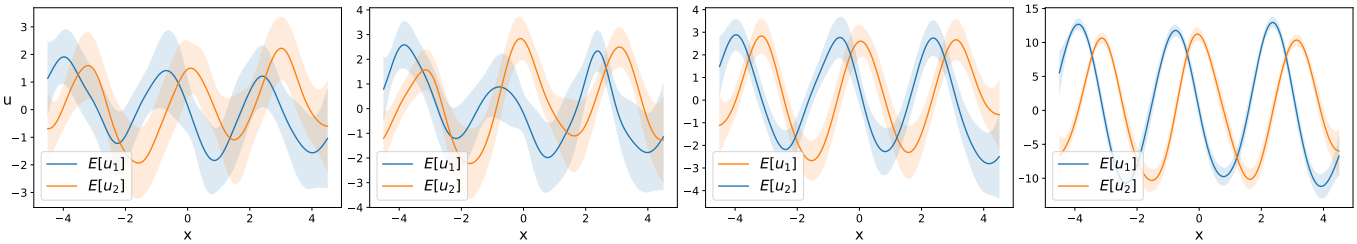
We use $\mathbf{u}$ *to define a choice function. For instance, consider the set of objects* $A_k = \{0, 0.5, 2.36\}$ *(represented by the values of* x *marked by the vertical lines in the above plot), given that* $\mathbf{u}(0) = [1, 0]$, $\mathbf{u}(0.5) = [0.54, -0.84]$, $\mathbf{u}(2.36) = [0, 1]$, *we have that* $C(A_k) = \{0, 2.36\}$ *and* $R(A_k) = A_k \backslash C(A_k) = \{0.5\}$. *In fact, one can notice that* $[1, 0]$ *dominates* $[0.54, -0.84]$ *on both utilities, and* $[1, 0]$ *and* $[0, 1]$ *are incomparable. We sample* 200 *inputs* x_i *at random in* $[-4.5, 4.5]$ *and, using the above approach, we generate*

- $m = 50$ *random subsets* $\{A_k\}_{k=1}^m$ *of the 200 points and compute the corresponding choice pairs* $(C(A_k), A_k)$ *based on* $\mathbf{u}$. *The size of the each choice set is fixed to* $|A_k| = 3$ *and to* $|A_k| = 5$*;*
- $m = 150$ *random subsets* $\{A_k\}_{k=1}^m$ *each one of size* $|A_k| = 3$ *(respectively* $|A_k| = 5$*) and compute the corresponding choice pairs* $(C(A_k), A_k)$ *based on* $\mathbf{u}$*;*

Fixing the latent dimension $d = 2$, *we use these datasets to compute the posterior means and* 95% *credible intervals of the latent functions learned using the model introduced in Section 3.1. The four posterior plots are shown in Figure 1. By comparing the 1st with the 3rd plot and the 2nd with the 4th plot, it can be noticed how the posterior means become more accurate (and the credible interval smaller) as the size of the dataset increases (from m=50 to m=150 choice sets). We can also observe that we obtain better estimates by increasing the dimension of* $|A_k|$.

A key insight from the previous example is that the learned latent utilities only maintain the Pareto dominance among the objects (the x in the example), which is the only thing we observe. Hence, we can never recover the true utilities.

One might ask how the size of choice-set $|A_s|$ should be determined. It is clear from (7) that the computational complexity of calculating the likelihood grows with $|A_s|$. The optimal value should therefore be $|A_s| = 2$. However, having Alice choose objects from a larger set provides more information for a fixed m. In general, the size $|A_s|$ depends on the application, as objects may come in a batch.



$m = 50,\ |A_k| = 3$ $m = 50,\ |A_k| = 5$ $m = 150,\ |A_k| = 3$ $m = 150,\ |A_k| = 5$

Figure 1: Posterior mean and 95% credible intervals of the two latent functions for the four artificial datasets.

Scalability: In terms of the number of latent utilities, the computational complexity for ChoiceGP is similar to that in GP multiclass classification. By exploiting the independence structure of the prior in the VI, we need storing and inverting d kernel matrices with dimension $t \times t$. For large t, there are a number of well established ways to scale up GPs that can be applied to ChoiceGP [Quiñonero-Candela and Rasmussen, 2005, Snelson and Ghahramani, 2006, Titsias, 2009, Hensman et al., 2013] [Hernández-Lobato and Hernández-Lobato, 2016, Bauer et al., 2016, Schuerch et al., 2020, 2023].

3.2 LATENT DIMENSION SELECTION

In the previous sections, we provided a GP-based model to learn choice functions. We refer to this model as *ChoiceGP*$_d$. *ChoiceGP*$_d$ is conditional on the predefined latent dimension d (that is, the dimension of the vector of the latent functions $\mathbf{u}(\mathbf{x}) = [u_1(\mathbf{x}), \ldots, u_d(\mathbf{x})]^\top$). Although, it is sometimes reasonable to assume the number of utility functions defining the choice function is known, it is crucial to derive a statistical method to select d from data.

We propose a forward selection method. We start learning the model *ChoiceGP*$_1$ and we increase the dimension d in a stepwise manner (so learning *ChoiceGP*$_2$,*ChoiceGP*$_3$ and so on) until some model selection criterion is optimised. Criteria like AIC and BIC are inappropriate for the proposed GP-based choice function model, since the nonparametric nature of the model implies that the number of parameters increases with the size of the data (as $d \times m$). We propose to use instead the *Pareto Smoothed Importance Sampling Leave-One-Out* cross-validation (PSIS-LOO, Vehtari et al. [2017]). Exact cross-validation requires re-fitting the model with different training sets. Instead, PSIS-LOO can be computed efficiently using the samples from the posterior.

We define the Bayesian LOO estimate of out-of-sample predictive fit for the model in (9):

$$\varphi = \sum_{k=1}^{m} \ln p(z_k|z_{-k}), \tag{12}$$

where $z_k = (C(A_k), A_k)$, $z_{-k} = \{(C(A_i), A_i)\}_{i=1, i\neq k}^{m}$,

$$p(z_k|z_{-k}) = \int p(z_k|\mathbf{u}(X))p(\mathbf{u}(X)|z_{-k})d\mathbf{u}(X). \tag{13}$$

As derived in Gelfand et al. [1992], we can evaluate (13) using the samples from the full posterior, that is $\mathbf{u}^{(s)}(X) \sim p(\mathbf{u}(X)|\{z_k, z_{-k}\}) = p(\mathbf{u}(X)|\mathscr{D})$ for $s = 1, \ldots, S$.[7] We first define the importance weights:

$$w_k^{(s)} = \frac{1}{p(z_k|\mathbf{u}^{(s)}(X))} \propto \frac{p(\mathbf{u}^{(s)}(X)|z_{-k})}{p(\mathbf{u}^{(s)}(X)|\{z_k, z_{-k}\})}$$

and then approximate (13) as:

$$p(z_k|z_{-k}) \approx \frac{\sum_{s=1}^{S} w_k^{(s)} p(z_k|\mathbf{u}^{(s)}(X))}{\sum_{s=1}^{S} w_k^{(s)}}. \tag{14}$$

It can be noticed that (14) is a function of $p(z_k|\mathbf{u}^{(s)}(X))$ only, which can easily be computed from the posterior samples. Unfortunately, a direct use of (14) induces instability because the importance weights can have high variance. To address this issue, Vehtari et al. [2017] applies a simple smoothing procedure to the importance weights using a Pareto distribution. We provide an example hereafter.

Example 3. *We run the latent-dimension selection procedure on the four datasets in Example 2. The table below reports the PSIS-LOO for different values of the dimension d. It can be observed how the selection procedure always selects the true dimension $d = 2$.*

	$m = 50$		$m = 150$	
d	$\lvert A_k\rvert = 3$	$\lvert A_k\rvert = 5$	$\lvert A_k\rvert = 3$	$\lvert A_k\rvert = 5$
1	*-882*	*-1906*	*-3213*	*-6108*
2	***-34***	***-118***	***-69***	***-84***
3	*-42*	*-134*	*-80*	*-95*
4	*-50*	*-152*	*-91*	*-109*

In Section 4.3, we will show that the proposed PSIS-LOO-based forward procedure also works on real datasets.

[7] We generate these samples from the variational posterior.

3.3 RELATION TO (BATCH-)PREFERENCE

For $d = 1$ (the latent dimension is one), we have that $|C(A_k)| = 1$. This means the subject always selects a single best object. In this case, the likelihood (7), for a given k, simplifies to

$$p((C(A_k), A_k)|\mathbf{u}(X)) = \prod_{\mathbf{o}\in C(A_k)} \prod_{\mathbf{v}\in R(A_k)} \Phi\left(\frac{u(\mathbf{o}) - u(\mathbf{v})}{\sigma}\right), \tag{15}$$

and reduces to the likelihood (2) when $|R(A_k)| = 1$ (that is, in the binary case $|A_k| = 2$). For $|A_k| > 2$, the likelihood (15) is a lower bound of the batch preference likelihood derived in [Siivola et al., 2021, Eq.3]:

$$\int \left(\prod_{\mathbf{v}\in R(A_k)} \Phi\left(\frac{u(\mathbf{o}) + w_k - u(\mathbf{v})}{\sigma}\right)\right) N(w_k; 0, \sigma^2) dw_k, \tag{16}$$

as proven in Supp. Mat. 4. The difference between (15) and (16) comes from two different ways of modelling errors. The likelihood (16) assumes that inconsistencies in Alice's preferences are due to an additive Gaussian noise perturbation of the true utility. The likelihood (15) instead assumes that inconsistencies are due to a limit of discernability, that is the probability of error is inversely proportional to the difference between the utilities of the two objects to be compared (when this difference is zero Alice has 50% chance to select one object or another). Supp. Mat. 4 includes a further discussion about these two likelihoods.

4 EXPERIMENTS

Our experiments aim to compare ChoiceGP with the state-of-the-art methods for choice functions and preference learning. In section 4.1, we compare ChoiceGP against ChoiceNN [Pfannschmidt and Hüllermeier, 2020] on choice data simulated using multi-utility functions taken from benchmark problems used in multi-criteria optimization. In section 4.2, using simulated preferences, we compare *ChoiceGP* with *Preferential GP* (PGP) [Chu and Ghahramani, 2005], *General Preferential GP* (GPGP) [Chau et al., 2022] and GP with data augmentation (PairGP) [Chau et al., 2022]. PairGP solves the PL problem as an augmented binary classification problem. In PairGP, skew-symmetry is further enforced by averaging the model outputs [Chau et al., 2022, Sec. 3.3]. For PGP, GPGP and PairGP, we use the Laplace approximation to compute an approximation of the posterior. Finally in Section 4.2, we compare ChoiceGP, PGP, GPGP and PairGP using real-world datasets. For all methods involving kernels, we use the Gaussian radial basis function (RBF) kernel with automatic relevance determination (ARD):

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_{i=1}^{c} \frac{(x_i - x_i')^2}{2\ell_i^2}\right), \tag{17}$$

where ℓ_i are the lengthscale parameters for each dimension $i = 1, \ldots, c$. The scale-parameter of the kernel is set to one, but we estimate all lengthscale parameters ℓ_i and the scaling parameter σ of the likelihood.

4.1 BENCHMARK OPTIMISATION PROBLEMS

Data generation In this section, we repeat the experiment in [Pfannschmidt and Hüllermeier, 2020, Sec. 4]. Choice data are simulated using multi-utility functions taken from benchmark problems used in multi-criteria optimization: the DTLZ test suite [Deb et al., 2005] and the ZDT test suite [Zitzler et al., 2000], for a total of 10 benchmarks. In all experiments, the dimension of the choice set is $|A| = 10$ and each object $\mathbf{x}_i \in \mathbb{R}^6$ (6 features). A total of 40,960 choice sets are generated. For the DTLZ problems, the number of objective functions is set to 5 (i.e., $d = 5$). As performance, Pfannschmidt and Hüllermeier [2020] used the average *A-mean* (A-mean is the arithmetic mean of the true positive and true negative rate) of 5 repetitions of a Monte Carlo cross validation with a 90/10% split into training and test data. For ChoiceNN, the training instances are further split into 1/9 validation instances and 8/9 training instances in order to optimize the hyperparameters: (a) the loss weights $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ (b) the number of hidden units and layers, using 60 iterations of Bayesian optimization. For both ChoiceNN and ChoiceGP, the latent dimension is equal to the number of objective functions.

Results Figure 2 reports the average A-mean of the two models when predicting choices on held-out data. ChoiceGP significantly outperforms ChoiceNN. We have found that this is due to ChoiceNN not often being able to find latent functions that are consistent with the training data. The disadvantage of a parametric method, like ChoiceNN, is that the latent utility functions depend nonlinearly on the optimisation parameters. Instead, in ChoiceGP, the values of the utility functions at the training data are part of the variational parameters and, therefore, can be more easily optimised. We provide a simple example in Supp. Mat. 5 to illustrate this issue. For this reason, we have not included ChoiceNN in the subsequent experiments.

4.2 INCONSISTENT PREFERENCES

Data generation In this section, we repeat the experiment proposed in Chau et al. [2022, Sec. 4.1]. Consider a data matrix $X \in \mathbb{R}^{n\times c}$. We randomly assign to each row a latent variable $z \in \{1, \ldots, L\}$ and generate a set of utility functions $\{u_{z,z'}\}_{z,z'=1}^{L}$, i.e. a different utility function for each pair z, z' of latent states. We impose the constraint $u_{z,z'} = u_{z',z}$ and assume that $u_{z,z'}(\mathbf{x}) = \sum_{j=1}^{n} \alpha_j^{z,z'} k(\mathbf{x}, \mathbf{x}_j)$, where k is the RBF kernel, eq. (17), and each vector $\alpha_j^{z,z'} \overset{\text{i.i.d.}}{\sim} N(0, I_n)$.

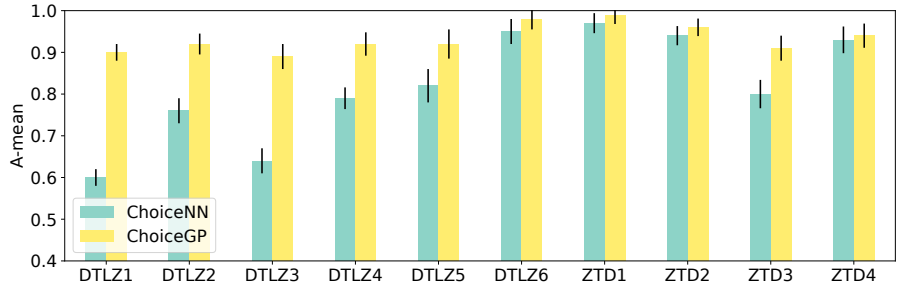
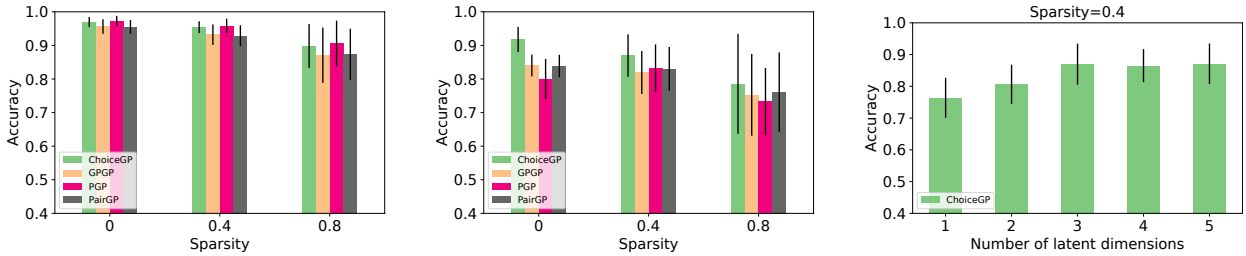


Figure 2: Comparisons ChoiceGP vs. ChoiceNN on 10 multi-criteria optimization problems. A-means are averaged over 5 runs and error bars of 1 standard deviation are provided.



L=1 ($d=1$) L=2 ($d=3$) L=2 ($d=1,2,3,4,5$)

Figure 3: Comparisons of algorithms for simulations at different sparsity and inconsistency level. Accuracy is averaged over 20 runs and error bars of 1 standard deviation are provided

We generate two datasets. In the first dataset, as done in Chau et al. [2022], the comparison between objects $\mathbf{x}_i, \mathbf{x}_j$ is conducted based on the utility selected by their latent states, i.e. $\mathbf{x}_i \succ \mathbf{x}_j$ iff $u_{z_i,z_j}(\mathbf{x}_i) > u_{z_i,z_j}(\mathbf{x}_j)$, leading to a dataset of m pairwise comparisons:

$$\mathscr{D}_m^{(1)} = \{\mathbf{x}_l^{(s)} \succ \mathbf{x}_r^{(s)} : \ s = 1, \dots, m\}.$$

$\mathscr{D}_m^{(1)}$ includes preferences violating negative transitivity. In the second dataset, the comparison between objects $\mathbf{x}_i, \mathbf{x}_j$ is based on Pareto's dominance criterion, i.e. $\mathbf{x}_i$ is chosen from the set $A = \{\mathbf{x}_i, \mathbf{x}_j\}$ iff $u_{z,z'}(\mathbf{x}_i) > u_{z,z'}(\mathbf{x}_j)$ for all z, z'. This naturally originates into a choice dataset:

$$\mathscr{D}_m^{(2)} = \{(C(A_s), A_s) : \ s = 1, \dots, m\},$$

where $A_s = \{\mathbf{x}_l^{(s)}, \mathbf{x}_r^{(s)}\}$ (the same pairs as in $\mathscr{D}_m^{(1)}$) and $C(A_s)$ equal to either $\{\mathbf{x}_l^{(s)}\}$ if $u_{z,z'}(\mathbf{x}_l^{(s)}) > u_{z,z'}(\mathbf{x}_r^{(s)})$ for all z, z' or to $\{\mathbf{x}_l^{(s)}, \mathbf{x}_r^{(s)}\}$ otherwise. Dataset $\mathscr{D}_m^{(2)}$ corresponds to a scenario in which Alice is allowed to express incomplete judgments while, in $\mathscr{D}_m^{(1)}$, Alice is compelled to always choose between two objects.

Results Figure 3 reports the accuracy for the 4 models when predicting preferences on held-out data and when trained on datasets with different sparsity level. To generate training datasets, we set the number of objects to $n = 30$ and their dimension to $c = 5$. We then varied the sparsity level $\gamma \in [0, 1)$ and generated preferences/choices (as described previously) for a random subset of dimension $(1-\gamma)n(n-1)/2$ of the possible pairs of objects.

In the case $L = 1$ ($d = 1$), ChoiceGP and PGP coincide, since there is only one utility function. They outperform both GPGP and PairGP in terms of average accuracy. In the case $L = 2$ ($d = 3$), the average accuracy of ChoiceGP is always higher than the other three methods. The rightmost plot of figure 3 (labelled $L = 2$ ($d = 1,2,3,4,5$)) reports the accuracy for ChoiceGP in the medium sparsity regime as a function of the latent dimension d. It can be observed as the accuracy increases until $d = 3$ (corresponding to the true latent dimension) and then remains stable.

These experiments show that, when inconsistencies in preference assessments are due to multiple conflicting utilities, it is better to allow Alice to express incomparability judgements instead of compelling Alice to always choose a preferred object.

4.3 REAL DATASETS

We now focus on five benchmark datasets – AM, EDM, Jura, Slump, Vehicle – for multi-output regression problems. We use the three output variables as utility functions to generate choice data. For instance, in the Additive Manufacturing (AM) dataset, we consider 6 features (layer height, nozzle temperature, bed temperature, print speed material, fan speed) and we use the three outputs (roughness, tension strength, elongation) to generate choice data. More details on the datasets is provided in Supp. Mat. 6. For each dataset, we set $|A_k| = 2$ and use the three output variables to generate a dense choice set $\{(C(A_k), A_k) : \ k = 1, \dots, m\}$. By using a

Table 1: Average accuracy

	ChoiceGP		PGP	GPGP	PairGP
AM	0.90	maj.	0.84	0.86	0.87
		rand.	0.74	0.73	0.738
EDM	0.88	maj.	0.83	0.80	0.83
		rand.	0.84	0.82	0.82
Jura	0.91	maj.	0.87	0.87	0.87
		rand.	0.84	0.82	0.82
Slump	0.91	maj.	0.93	0.90	0.90
		rand.	0.83	0.79	0.79
Vehicle	0.93	maj.	0.89	0.90	0.90
		rand.	0.80	0.80	0.80

sparsity level equal to 0.4 we randomly generate a training set and use the remaining choice pairs as test set (we repeat this process 5 times generating a total of 25 datasets).

Since PGP, GPGP and PairGP cannot deal with conflicts among preferences (and so with choice data), we consider two common ways[8] to deal with these conflicts: (1) random selection; (2) looking for the alternatives that are favoured by most (but not necessarily all) of the preference criteria. We generate preference data from the above choice datasets as follows. For each $A_k = \{\mathbf{x}_i, \mathbf{x}_j\}$ if $C(A_k) = \{\mathbf{x}_i\}$ then $\mathbf{x}_i \succ \mathbf{x}_j$. Instead, whenever $C(A_k) = \{\mathbf{x}_i, \mathbf{x}_j\}$, we generate preference data in two ways: **random:** coin flip: $\mathbf{x}_i \succ \mathbf{x}_j$ if Heads; $\mathbf{x}_j \succ \mathbf{x}_i$ if Tails; **majority rule:** $\mathbf{x}_i \succ \mathbf{x}_j$ if $\mathbf{x}_i$ is better than $\mathbf{x}_j$ with respect to 2 out of 3 outputs. Table 1 reports the average accuracy.

It can be noted that ChoiceGP overall outperforms PGP, GPGP and PairGP in both the *random* and *majority* rule scenario. The only exception is the dataset *slump*, where PGP has higher accuracy in the *majority* rule scenario. In Supp. Mat. 6, we perform a statistical analysis of the results to show that the difference between the algorithms is practically and statistically significant.

Finally, we run the latent-dimension selection procedure on the five datasets. Table 2 reports the PSIS-LOO for different values of the dimension d in one of the 5 MC repetitions. It can be observed how the selection procedure always selects the true dimension $d = 3$. This happens consistently in 5 out of the 5 repetitions and demonstrates both the accuracy and reliability of the proposed latent dimension selection procedure.

5 CONCLUSIONS

We have developed a Gaussian Process based-method to learn choice functions from choice data via Pareto rationalization. This method extends standard (pairwise) preference learning in two directions, it allows a subject to: (1) choose out of a set with more than two elements; (2) express judgements of incomparability (due to lack of information or to objects having contradictory characteristics). Experimental results on simulations and real-world datasets show the proposed method is both robust and accurate. We also proposed an effective method to learn the number of latent multiple utilities via Pareto Smoothed Importance Sampling Leave-One-Out cross-validation. As future work, we will study the performances of a fully Bayesian method such as reversible jump MCMC to automatically select the latent dimension. We also plan to compare such an heavy approach with the prior predictive matching approach [Silva et al., 2023].

[8]Note that, there are criteria to deal with conflicts which generate consistent preferences, for instance by weighting the utilities. In these cases, PGP and ChoiceGP are the best models.

Table 2: Latent dimension selection, PSIS-LOO values

d	AM	EDM	Jura	Slump	Vehicle
1	-16335	-5646	-11182	-10754	-12692
2	-186	-138	-254	-211	-272
3	**-152**	**-136**	**-194**	**-179**	**-174**
4	-160	-158	-207	-199	-179
5	-170	-173	-223	-234	-205

The framework we proposed may enable more easy-to-elicit, and robust recommendations to users in recommender systems and information retrieval. This model could also be used in Bayesian optimisation with implicit feedback.

The likelihood in Equation (2) assumes that the latent functions are corrupted by a Gaussian noise. We will explore different options for the noise distribution such as a Gumbel distribution and extend it to (7). Finally, we aim to extend this model to handle choice-functions that are not Pareto rationalisable, and thus we will explore weaker forms of rationality. For example, [Pfannschmidt et al., 2022] proposed a generalisation in this direction by considering context-dependent choice functions. We will also investigate choice under uncertainty [Seidenfeld et al., 2010, Van Camp et al., 2018, De Bock and De Cooman, 2019].

Acknowledgements

For the first author, this publication has emanated from research conducted with the financial support of the EU Commission Recovery and Resilience Facility under the Science Foundation Ireland Future Digital Challenge Grant Number 22/NCF/FD/10827. The authors thank Karlson Pfannschmidt for sharing the code for ChoiceNN and Siu Lun Chau for sharing the data generation code for the numerical experiments in the GPGP paper. The authors would also like to thank the Anonymous Reviewers for their fruitful discussion and suggestions.

Software The Python implementation of ChoiceGP is available at https://github.com/benavoli/ChoiceGP

References

Fabio Aiolli and Alessandro Sperduti. Learning preferences for multiclass problems. *Advances in neural information processing systems*, 17, 2004.

Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *Advances in neural information processing systems*, pages 1533–1541, 2016.

Alessio Benavoli, Dario Azzimonti, and Dario Piga. Skew Gaussian Processes for Classification. *Machine Learning*, 109:1877–1902, 2020. doi: 10.1007/s10994-020-05906-3.

Alessio Benavoli, Dario Azzimonti, and Dario Piga. Preferential Bayesian optimisation with Skew Gaussian Processes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, page 1842–1850, New York, NY, USA, 2021a. Association for Computing Machinery. ISBN 9781450383516. doi: 10.1145/3449726.3463128.

Alessio Benavoli, Dario Azzimonti, and Dario Piga. A unified framework for closed-form nonparametric regression, classification, preference and mixed problems with Skew Gaussian Processes. *Machine Learning*, pages 1–39, 2021b. doi: 10.1007/s10994-021-06039-x.

Alessio Benavoli, Dario Azzimonti, and Dario Piga. Bayesian Optimization For Choice Data. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 979-8-4007-0120-7/23/07. doi: 10.1145/3583133.3596324. URL `https://doi.org/10.1145/3583133.3596324`.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Siu Lun Chau, Javier Gonzalez, and Dino Sejdinovic. Learning Inconsistent Preferences with Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics*, pages 2266–2281. PMLR, 2022.

Wei Chu and Zoubin Ghahramani. Preference learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 137–144, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102369.

William W Cohen, Robert E Schapire, and Yoram Singer. Learning to order things. *Advances in neural information processing systems*, 10, 1997.

Jasper De Bock and Gert De Cooman. Interpreting, axiomatising and representing coherent choice functions in terms of desirability. In *International Symposium on Imprecise Probabilities: Theories and Applications*, pages 125–134. PMLR, 2019.

Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary multiobjective optimization*, pages 105–145. Springer, 2005.

Gerard Debreu. Representation of a preference ordering by a numerical function. *Decision processes*, 3:159–165, 1954.

Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. Preferences in AI: An overview. *Artificial Intelligence*, 175(7-8):1037–1052, 2011.

Kfir Eliaz and Efe A Ok. Indifference or indecisiveness? choice-theoretic foundations of incomplete preferences. *Games and economic behavior*, 56(1):61–86, 2006.

Claude-Nicolas Fiechter and Seth Rogers. Learning subjective functions with large margins. In *ICML*, pages 287–294, 2000.

Peter C Fishburn. *Nonlinear preference and utility theory*, volume 5. Wheatsheaf Books, 1988.

Johannes Fürnkranz and Eyke Hüllermeier. *Preference learning*. Springer, 2010.

Alan E Gelfand, Dipak K Dey, and Hong Chang. Model determination using predictive distributions with implementation via sampling-based methods. Technical report, Stanford Univ CA Dept of Statistics, 1992.

Sariel Har-Peled, Dan Roth, and Dav Zimak. Constraint classification: A new approach to multiclass classification. In *International conference on algorithmic learning theory*, pages 365–379. Springer, 2002.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian Processes for Big Data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pages 282–290, Arlington, Virginia, USA, 2013. AUAI Press.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

Ralf Herbrich, Thore Graepel, Peter Bollmann-Sdorra, and Klaus Obermayer. Learning preference relations for information retrieval. In *ICML-98 Workshop: text categorization and machine learning*, pages 80–84, 1998.

Daniel Hernández-Lobato and José Miguel Hernández-Lobato. Scalable Gaussian Process Classification via Expectation Propagation. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 168–176, Cadiz, Spain, 09–11 May 2016. PMLR.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011.

David M Kreps et al. *A course in microeconomic theory*. Princeton university press, 1990.

David JC MacKay. Bayesian methods for backpropagation networks. In *Models of neural networks III*, pages 211–254. Springer, 1996.

Hervé Moulin. Choice functions over a finite set: a summary. *Social Choice and Welfare*, 2(2):147–160, 1985.

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3):786–792, 2009.

Tapio Pahikkala, Willem Waegeman, Evgeni Tsivtsivadze, Tapio Salakoski, and Bernard De Baets. Learning intransitive reciprocal relations with kernel methods. *European Journal of Operational Research*, 206(3):676–685, 2010.

Karlson Pfannschmidt and Eyke Hüllermeier. Learning Choice Functions via Pareto-Embeddings. In *German Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 327–333. Springer, 2020.

Karlson Pfannschmidt, Pritha Gupta, Björn Haddenhorst, and Eyke Hüllermeier. Learning context-dependent choice functions. *International Journal of Approximate Reasoning*, 140:116–155, 2022.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6 (Dec):1939–1959, 2005.

Manuel Schuerch, Dario Azzimonti, Alessio Benavoli, and Marco Zaffalon. Recursive estimation for sparse Gaussian process regression. *Automatica*, 120:109–127, 2020. ISSN 0005-1098. doi: 10.1016/j.automatica.2020.109127.

Manuel Schuerch, Dario Azzimonti, Alessio Benavoli, and Marco Zaffalon. Correlated Product of Experts for Sparse Gaussian Process Regression. *Machine Learning*, 2023. doi: 10.1007/s10994-022-06297-3.

Teddy Seidenfeld, Mark J Schervish, and Joseph B Kadane. Coherent choice functions under uncertainty. *Synthese*, 172(1):157–176, 2010.

Wayne J Shafer. The nontransitive consumer. *Econometrica: Journal of the Econometric Society*, pages 913–919, 1974.

Eero Siivola, Akash Kumar Dhaka, Michael Riis Andersen, Javier González, Pablo García Moreno, and Aki Vehtari. Preferential Batch Bayesian Optimization. In *2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2021. doi: 10.1109/MLSP52302.2021.9596494.

Eliezer de Souza da Silva, Tomasz Kuśmierczyk, Marcelo Hartmann, and Arto Klami. Prior Specification for Bayesian Matrix Factorization via Prior Predictive Matching. *Journal of Machine Learning Research*, 24(67):1–51, 2023. ISSN 1533-7928.

Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David van Dyk and Max Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

Arthur Van Camp, Gert De Cooman, Enrique Miranda, and Erik Quaeghebeur. Coherent choice functions, desirability and indifference. *Fuzzy sets and systems*, 341:1–36, 2018.

Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and computing*, 27(5): 1413–1432, 2017.

Christopher KI Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12): 1342–1351, 1998.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.