Learning Choice Functions with Gaussian Processes (Supplementary Material)

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

1 VECTORISATION OF THE LIKELIHOOD

The product in the second row in main-text-(7) is a probabilistic relaxation of main-text-(6). Since it always involves comparisons between pairs of objects, it can be vectorized as follows:

$$\prod_{k=1}^{m}\prod_{\{\mathbf{o},\mathbf{v}\}\in C_\sharp(A_k)}\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o})-u_i(\mathbf{v})}{\sigma}\right)-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{v})-u_i(\mathbf{o})}{\sigma}\right)\right)=\prod_{\mathbf{a}_i\in\mathscr{A}}\left(1-\Phi_d\left(\frac{\mathbf{a}_i\mathbf{u}(X)}{\sigma}\right)-\Phi_d\left(\frac{-\mathbf{a}_i\mathbf{u}(X)}{\sigma}\right)\right), \tag{1}$$

where there is a vector $\mathbf{a}_k\in\mathbb{R}^{1\times t}$ for each pairs $\{\mathbf{x}_i,\mathbf{x}_j\}\in C(A_k)$ with $\mathbf{x}_i\neq\mathbf{x}_j$. $\mathbf{a}_k$ is a zero vector whose i-th and j-th elements are equal to 1 and respectively, -1, and Φ_d is the CDF of d-dimensional standard multivariate Gaussian distribution.

The product in the last row in main-text-(7) is a probabilistic relaxation of main-text-(5). It cannot be easily vectorized because $\prod_{\mathbf{o}\in C(A_k)}$ has a varying number of terms depending on k. To overcome this issue we assume, as usually done in decision theory (see for instance [Parmigiani and Inoue, 2009, Sec.3.4.2]), the existence a worst object $\omega\in\mathscr{X}$, that is an object such that $u_i(\omega)=-\infty$ for each $i=1,\dots,d$. This allows us to compare any $\mathbf{v}\in C(A_k)$ with the same number of elements (either $\mathbf{o}\in C(A_k)$ or ω).

Assume for instance that $|A_k|=5$, $R(A_k)=\{\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3\}$ and $C(A_k)=\{\mathbf{o}_1,\mathbf{o}_2\}$, then the product in the last row in main-text-(7)

$$\prod_{\mathbf{o}\in C(A_k)}\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o})-u_i(\mathbf{v}_1)}{\sigma}\right)\right)\prod_{\mathbf{o}\in C(A_k)}\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o})-u_i(\mathbf{v}_2)}{\sigma}\right)\right)\prod_{\mathbf{o}\in C(A_k)}\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o})-u_i(\mathbf{v}_1)}{\sigma}\right)\right) \tag{2}$$

For each $\mathbf{v}_j$, we can write each product as

$$\begin{aligned}
&\prod_{\mathbf{o}\in C(A_k)}\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o})-u_i(\mathbf{v}_j)}{\sigma}\right)\right)=\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o}_1)-u_i(\mathbf{v}_j)}{\sigma}\right)\right)\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o}_2)-u_i(\mathbf{v}_j)}{\sigma}\right)\right)\\
&=\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o}_1)-u_i(\mathbf{v}_j)}{\sigma}\right)\right)\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\mathbf{o}_2)-u_i(\mathbf{v}_j)}{\sigma}\right)\right)\left(1-\prod_{i=1}^{d}\Phi\left(\frac{u_i(\omega)-u_i(\mathbf{v}_j)}{\sigma}\right)\right)^3\\
&=\prod_{\mathbf{b}_i\in\mathscr{B}}\left(1-\Phi_d\left(\frac{\mathbf{b}_i\mathbf{u}(\tilde{X})}{\sigma}\right)\right),
\end{aligned} \tag{3}$$

where $\omega\in\mathscr{X}$ such that $u_i(\omega)=-\infty$ for each $i=1,\dots,d$ and $\tilde{X}=[X,\omega]^\top$. $\mathbf{b}_i\in\mathbb{R}^{1\times(t+1)}$ for each compared pairs $\{\mathbf{x}_i,\mathbf{x}_j\}$ with $\mathbf{x}_i\neq\mathbf{x}_j$, is a zero vector whose i-th and j-th elements are equal to 1 and respectively, -1,

2 LABEL SWITCHING PROBLEM

The Laplace Approximation (LA) cannot be applied due to the so-called 'label switching' problem, which is caused by symmetry in the likelihood: any permutation of the labels $i=1,\dots,d$ yields the same likelihood. For this reason, the Hessian

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

of the log-likelihood w.r.t. $\mathbf{u}(X)$ is in general an indefinite matrix and LA is not well-defined.

Consider for instance the case where $d = 2$ and $C(A_1) = \{\mathbf{x}_a, \mathbf{x}_b\}$ is the only choice data we have, the likelihood is

$$L = \left(1 - \prod_{i=1}^{2} \Phi\left(\frac{u_i(\mathbf{x}_a) - u_i(\mathbf{x}_b)}{\sigma}\right) - \prod_{i=1}^{2} \Phi\left(\frac{u_i(\mathbf{x}_b) - u_i(\mathbf{x}_a)}{\sigma}\right)\right)$$

and it is symmetric to the switching of u_1 and u_2.

The Variational Approximation is also affected by this problem, but it is well-defined. It will simply converge to one of the symmetric (to label switching) components of the distribution.

3 VARIATIONAL INFERENCE

We implemented our model using automatic-differentiation in Jax [Bradbury et al., 2018].

For the Variational Approximation (VA), we use the implementation in [Opper and Archambeau, 2009], which has $2t$ parameters with $t = |X|$. In particular, we consider the covariance matrix in [Opper and Archambeau, 2009, Equation (10)]. This means we only need t parameters for the covariance matrix of the VA distribution. This is an approximation, but it allows us reduce the computational load of ChoiceGP, which is composed by d GPs.

Indeed, by exploiting the above parametrisation and the factorised prior main-text-(8), our ChoiceGP model can be implemented efficiently. We need storing and inverting d kernel matrices with dimension $t \times t$.

We initialise the Variational Approximation with MAP estimate and then perform 5000 iterations.

4 INTERPRETATION OF THE PROBIT LIKELIHOOD

There are two ways to interpret the likelihood main-text-(2):

1. *Limit of discernibility:* Alice may make mistakes when comparing two objects $\mathbf{x}_i, \mathbf{x}_j$ whose difference in utility is small (e.g., errors are inversely proportional to the difference between the two utilities $|u(\mathbf{x}_i) - u(\mathbf{x}_j)|$).
2. *Noise:* the observed utility function differs from the true utility function due to disturbances (e.g., $o(\mathbf{x}_i) = u(\mathbf{x}_i) + \text{noise}$).

In this second case, it is well known that

$$p(\mathbf{x}_i \succ \mathbf{x}_j | u) = \Phi\left(\frac{u(\mathbf{x}_i) - u(\mathbf{x}_j)}{\sqrt{2}\sigma}\right) = \int I_{u(\mathbf{x}_i)+w_i-u(\mathbf{x}_j)-w_j>0} N(w_i; 0, \sigma^2) N(w_j; 0, \sigma^2) dw_i dw_j. \tag{4}$$

There is no correct interpretation – it depends on the "error-model" we assume to account for the inconsistencies in the subject's preferences.

For instance, for the computer example in Section 1, assuming that inconsistencies are due to a Gaussian noise model does not make much sense. The features of the computer are observed exactly (without any noise). Instead, it is reasonable to assume that two different computers, which only have slightly different characteristics, are indiscernible for Alice. For this reason, she may state inconsistent preferences when comparing them.

Similarly, there may be cases where the utility is observed through a noisy measurement and, therefore, the second interpretation is more correct in this case.

The issue arises when we compare the same objects multiple times. Assuming that Alice chooses $\mathbf{o}$ and discards the elements in $R(A_k)$, this leads to the following batch-likelihood

$$\int \left(\prod_{\mathbf{v} \in R(A_k)} \Phi\left(\frac{u(\mathbf{o}) + w_k - u(\mathbf{v})}{\sigma}\right)\right) N(w_k; 0, \sigma^2) dw_k, \tag{5}$$

for the case $d = 1$ (single utility) and noise model, which is different from the limit-of-discernibility error-model

$$\prod_{\mathbf{v} \in R(A_k)} \Phi\left(\frac{u(\mathbf{o}) - u(\mathbf{v})}{\sigma}\right). \tag{6}$$

As stated in Proposition 1 (see proof below), (6) is a lower bound for (5). This means that either

- assuming (5) when (6) is the true error-model, or
- assuming (6) when (5) is the true error-model

may lead to a biased posterior. We will further investigate the difference between these two models in future work.

Proposition 1. *The likelihood main-text-*(15) *is a lower bound of the batch-preference likelihood:*

$$\int \left(\prod_{\mathbf{v} \in R(A_k)} \Phi\left(\frac{u(\mathbf{o}) + w_k - u(\mathbf{v})}{\sigma} \right) \right) N(w_k; 0, \sigma^2) dw_k. \tag{7}$$

Proof. We are going to use the following results.

Result: If $\mathbf{v} = [v_1, \ldots, v_d]$ are independent, then for any increasing functions h and g of n variables:

$$E[h(\mathbf{v})g(\mathbf{v})] \geq E[h(\mathbf{v})]E[g(\mathbf{v})].$$

The proof can be found in [Ross, 2013, Sec.9.9]

Consider the likelihood (7)

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

and note that inside the parenthesis we have a product of monotone increasing functions in w_k. Therefore, we can exploit the above result to derive that

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

□

5 CHOICENN VS. CHOICEGP

We illustrate the issue with ChoiceNN considering the 1D utility function $u(x) = \cos(5x) + \exp\left(-\frac{x^2}{8}\right)$ with $x \in [-2.6, 2.6]$ in Figure 1.

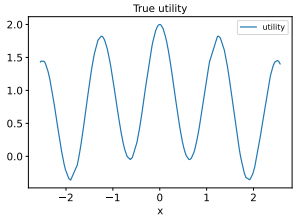


Figure 1: True utility function

We used u to generate choice data. We sampled 150 inputs x_i at random in $[-2.6, 2.6]$. We then generated $m = 500$ random subsets $\{A_k\}_{k=1}^m$ of the 500 points each one of size $|A_k| = 2$ and computed the corresponding choice pairs $(C(A_k), A_k)$ based on u.

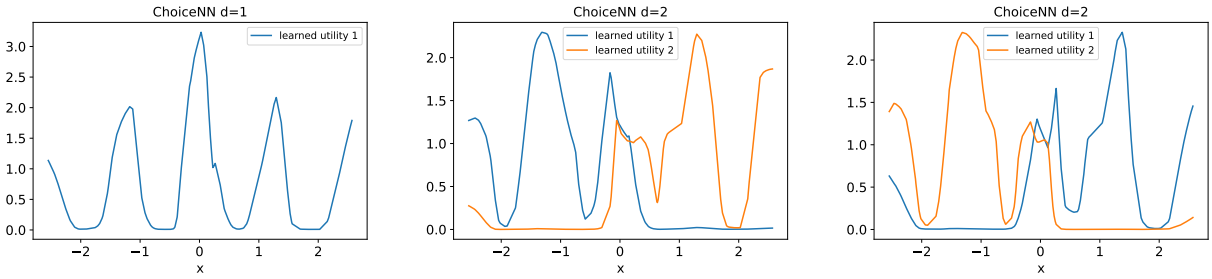
We ran ChoiceNN with fixed latent dimension $d = 1$ The learned utility function is shown in Figure 2 (a), which is reasonably consistent with the true utility.

We then ran ChoiceNN with fixed latent dimension $d = 2$ (the true latent dimension is one) and reported the estimated utility functions, for two different random initialisation of the parameters of the NN, in Figure 2 (b) and, respectively (c). It can be noticed that the model converged to two different local optima. In both cases, the learned utility functions are not Pareto-consistent with the choice data. In other words, the model is not able to find a utility representation of the choice data

and, therefore, it is not able to make correct predictions. We have tried different NN architectures (number of layers and number of nodes) as well as different values of the hyperparameters for ChoiceNN, but the issue remains.

The disadvantage of a nonlinear parametric method, like ChoiceNN, is the fact that the latent utility functions depend nonlinearly on the parameters. Instead, in ChoiceGP, the utility functions (at the training data) are part of the the variational parameters and, therefore, can be more easily optimised to satisfy the Pareto-consistency implied by the choice data.

Figure 3 shows the utilities learned by ChoiceGP with $d = 2$. They coincide. This shows that ChoiceGP is able to easily understand that the true latent dimension is one. Moreover, the learned utility basically coincides with the true utility in Figure 1 (apart from a scaling factor, which cannot be estimated from the data).



(a) (b) (c)

Figure 2: Learned utilities via ChoiceNN

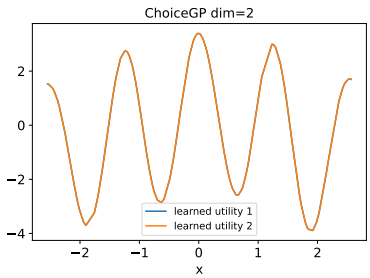


Figure 3: Learned utilities via ChoiceGP

6 REAL-DATASETS

Table 1 displays the characteristics of the considered datasets.

Dataset	#Features	#Outputs
AM	6	3
EDM	4	3
jura	6	3
slump	7	3
vehicle	5	3

Table 1: Characteristics of the datasets.

The first 4 datasets are standard datasets used in multi-target regression. The "vehicle dataset" has been obtained from the Vehicle Safety model[1] using a latin-hypercube design of experiment. We have included the datasets in our repository together with the code to replicate the experiments.

We have implemented GPGP, PGP and PairGP in GPy [GPy, since 2012]. For ChoiceNN, we use the implementation provided by the authors Pfannschmidt and Hüllermeier [2020].

As shown in the average accuracy table in Section 4.3, ChoiceGP has a higher average accuracy than the other models. This claim is also supported by a statistical analysis as we will show hereafter.

We have compared ChoiceGP against PGP, GPGP and PairGP for the majority-rule using the pairwise Bayesian hierarchical hypothesis testing model [Corani et al., 2017]. The test accounts for the correlation between the paired differences of accuracy due to the overlapping training sets built during cross-validation. This test declares two models practically equivalent when the difference of accuracy is less than 0.01 (1%). The interval $[-0.01, 0.01]$ thus defines a region of practical equivalence (rope) for the performance of the models. For instance for the pair (ChoiceGP,PGP), the test returns the posterior samples of the probability vector $[p(ChoiceGP > PGP), p(ChoiceGP \approx PGP), p(ChoiceGP < PGP)]$ and, therefore, this posterior can be visualised in the probability simplex (Figure 4). For all the pairwise comparisons, it can seen that the vast majority of the samples are in the region at the right bottom of the triangle. This confirms that ChoiceGP is practically significantly better than the other three methods. Note that, we have only statistically compared the methods in the majority-rule scenario, because the differences are even larger in the random scenario.

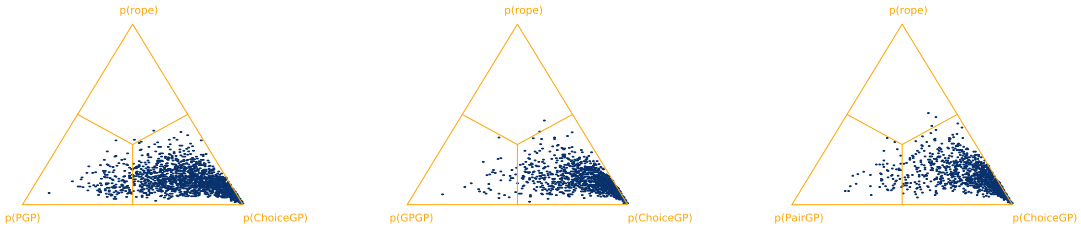


Figure 4: Posterior samples for the pairwise tests ChoiceGP vs. PGP, GPGP and, respectively, PairGP. This confirms that ChoiceGP is practically significantly better than the other three methods.

References

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`.

Giorgio Corani, Alessio Benavoli, Janez Demšar, Francesca Mangili, and Marco Zaffalon. Statistical comparison of classifiers through Bayesian hierarchical modelling. *Machine Learning*, 106:1817–1837, 2017.

GPy. GPy: A Gaussian process framework in Python. `http://github.com/SheffieldML/GPy`, since 2012.

[1] A model that determines the thickness of five reinforced components of a vehicle's frontal frame [Yang et al., 2005]

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

Giovanni Parmigiani and Lurdes Inoue. *Decision theory: Principles and approaches*. John Wiley & Sons, 2009.

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

Sheldon Ross. Chapter 9 - variance reduction techniques. In Sheldon Ross, editor, *Simulation (Fifth Edition)*, pages 153–231. Academic Press, fifth edition edition, 2013. ISBN 978-0-12-415825-2. doi: https://doi.org/10.1016/B978-0-12-415825-2.00009-7.

R. J. Yang, N. Wang, C. H. Tho, J. P. Bobineau, and B. P. Wang. Metamodeling Development for Vehicle Frontal Impact Simulation. *Journal of Mechanical Design*, 127(5):1014–1020, Sep 2005. ISSN 1050-0472, 1528-9001. doi: 10.1115/1.1906264.