Bayesian Optimization Augmented with Actively Elicited Expert Knowledge

Daolang Huang

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 17.5.2022

Supervisor

Prof. Samuel Kaski

Advisors

Dr. Louis Filstroff

M.Sc. Petrus Mikkola

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Daolang Huang

Title Bayesian Optimization Augmented with Actively Elicited Expert Knowledge

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science and Artificial Intelligence (Macadamia) **Code of major** SCI3044

Supervisor Prof. Samuel Kaski

Advisors Dr. Louis Filstroff, M.Sc. Petrus Mikkola

Date 17.5.2022 **Number of pages** 49+10 **Language** English

Abstract

Bayesian optimization (BO) is a well-established method to optimize black-box functions whose direct evaluations are costly. In this thesis, we tackle the problem of incorporating expert knowledge into BO, with the goal of further accelerating the optimization, which has received very little attention so far. We design a multi-task learning architecture for this task, with the goal of jointly eliciting the expert knowledge and minimizing the objective function. In particular, this allows for the expert knowledge to be transferred into the BO task. We introduce a specific architecture based on Siamese neural networks to handle the knowledge elicitation from pairwise queries. Experiments on various benchmark functions with both simulated and actual human experts show that the proposed method significantly speeds up BO even when the expert knowledge is biased compared to the objective function.

Keywords Bayesian optimization, knowledge elicitation, active learning, preference learning, multi-task learning

Preface

First and foremost, I am deeply indebted to my supervisor Prof. Samuel Kaski for his constructive advice and feedback during the thesis process. Also, I want to express my sincere gratitude to my advisors Dr. Louis Filstroff and M.Sc. Petrus Mikkola for their patience and insightful comments. It is a truly great experience to work with them. At last, I am grateful to the members of the Probabilistic machine learning group for their assistance during the experiment and writing stage.

Otaniemi, 17.5.2022

Daolang Huang

Contents

Symbols and abbreviations

Symbols

α	hyper-parameter of combined loss function
α_{AL}	active learning acquisition function
α_{EI}	expected improvement acquisition function
$\boldsymbol{\beta}_f$	output layer weights of $\tilde{f}$
$\boldsymbol{\beta}_g$	output layer weights of $\tilde{g}$
$\mathcal{D}_f$	set of obtained data in BO: $\mathcal{D}_f = \{\mathbf{x}_j, f(\mathbf{x}_j)\}_{j=1}^{J}$
$\mathcal{D}_g$	set of queried preference data: $\mathcal{D}_g = \{(\mathbf{x}_i, \mathbf{x}'_i, \hat{y}_i)\}_{i=1}^{M}$
$\mathcal{D}_{pool}$	pool of unlabeled instance
δ	perturbation function for generating biased knowledge
$\mathbb{E}$	expectation
$\mathbb{E}_x$	expectation with respect to x
f	a black-box objective function
$\tilde{f}$	surrogate model of f
g	latent expert belief towards objective function f
$\tilde{g}$	surrogate model of g
h	binary entropy: $h(p) = -p\log(p) - (1-p)\log(1-p)$
$\mathcal{H}$	differential entropy
I	mutual information
J	BO acquisition budget
k	kernel function
$\mathcal{L}_f$	loss of $\tilde{f}$
$\mathcal{L}_g$	loss of $\tilde{g}$
$\mathcal{L}_j$	combined loss of $\mathcal{L}_f$ and $\mathcal{L}_g$ after jth BO acquisition
ℓ	a length-scale parameter
M	active learning budget
$\mu_{\mathbf{x}}$	predictive mean of $\mathbf{x}$
$\emptyset$	empty set
N	active learning query pool size
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2
p	probability density
q	an approximation to probability density p
$\phi_{\mathbf{w}_h}$	neural network forward pass through $\mathbf{w}_h$
Φ	standard normal cumulative density function
ψ	standard normal probability density function
σ	sigmoid function: $\sigma(x) = \dfrac{1}{1+e^{-x}}$

σ_δ	variance of perturbation function
$s_{\mathbf{x}}^2$	predictive variance of $\mathbf{x}$
$\boldsymbol{\theta}_g$	hyper-parameters of q
t	number of Monte Carlo sampling iterations passed
T	Monte Carlo sampling budget
$\mathbf{w}_f$	neural network weights of $\tilde{f}$: $\mathbf{w}_f = [\mathbf{w}_h, \boldsymbol{\beta}_f]$
$\mathbf{w}_g$	neural network weights of $\tilde{g}$: $\mathbf{w}_g = [\mathbf{w}_h, \boldsymbol{\beta}_g]$
$\mathbf{w}_h$	shared weights in multi-task learning structure
$\mathbf{x}$	a vector-valued input location
$\check{\mathbf{x}}$	the location with largest acquisition function prediction
$\mathbf{x}^\star$	a location attaining the global optimum of f: $\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$
$\mathcal{X}$	domain of objective function
$\hat{y}$	preference label of covariates pair $[\mathbf{x}, \mathbf{x}']$ given by the expert: $\hat{y} \in \{0, 1\}$
y_{best}	current observed minimum value of f
ζ	Siamese network connection function

Abbreviations

BO	Bayesian optimization
GP	Gaussian process
NN	neural network
ANN	artificial neural network
BNN	Bayesian neural network
VI	variance inference
MCMC	Markov Chain Monte Carlo
PBNN	preferential Bayesian neural network
ELBO	evidence lower bound
MFBO	multi-fidelity Bayesian optimization
KL	Kullback-Leibler
MTL	multi-task learning
AL	active learning
BALD	Bayesian active learning by disagreement
PBALD	preferential Bayesian active learning by disagreement
BBB	Bayes by backprop
EI	expected improvement

1 Introduction

1.1 Motivation

Bayesian optimization (BO) [Jones et al., 1998, Brochu et al., 2010] has become a well-established class of methods to optimize black-box functions, with applications in hyper-parameter tuning [Snoek et al., 2012, Swersky et al., 2013], chemistry [Griffiths and Hernández-Lobato, 2017, Hase et al., 2018], and material science [Packwood, 2017, Zhang et al., 2020], to cite only a few. In a nutshell, BO algorithms build a probabilistic surrogate model of the objective function based on its evaluations, such as a Gaussian process [Jones et al., 1998] or a Bayesian neural network (BNN) [Snoek et al., 2015, Springenberg et al., 2016, Kim et al., 2021], and sequentially select optimal new points to evaluate. The selection of the new points is guided by an *acquisition function*, which aims at balancing between exploration and exploitation.

Each domain-specific BO problem naturally has its domain experts with their own knowledge of the problem, i.e., often tacit knowledge about the shape of the objective function f or where the global optimum might lie. Moreover, the cost of asking the expert can be significantly cheaper than the cost of obtaining the value of $f(\mathbf{x})$ in many applications (e.g., when obtaining $f(\mathbf{x})$ corresponds to running a simulation on a supercomputer or conducting a laboratory experiment). However, and surprisingly enough, leveraging that expert knowledge in order to speed up BO has only started to receive attention in the literature very recently. To the best of our knowledge, this problem has only been tackled from the perspective of incorporating expert prior knowledge about the location of the optimum into the BO acquisition function [Li et al., 2020a, Ramachandran et al., 2020, Souza et al., 2021, Hvarfner et al., 2022]. None of these works properly discuss how to obtain such a prior. The reason may be that eliciting knowledge from humans is notoriously challenging. Indeed, humans can be bad at evaluating absolute magnitudes, but on the other hand can be much better at comparing two instances [Millet, 1997, Shah et al., 2014] (e.g., humans can judge which molecule is more stable, but cannot directly give values for the energy levels). This has been utilized for preference learning through pairwise comparisons of items [Chu and Ghahramani, 2005], and has been expanded to an online learning setting [Brochu et al., 2008, González et al., 2017] to find the optimum of the preferences. Even though this approach has recently been shown to work in expert knowledge elicitation [Mikkola et al., 2020], there is still a need for methods to elicit knowledge from the expert with the goal of performing BO for f. Secondly, transferring that knowledge into the BO task also represents a technical challenge.

1.2 Contributions

In this thesis, we propose an expert knowledge-augmented BO method, which solves the aforementioned issues. We formulate the problem as a multi-task learning (MTL) problem [Caruana, 1997], and propose to solve it with a Bayesian neural network-based architecture whose goal is to learn both the objective function f and the expert knowledge. The key insight is to leverage statistical strength across the latent representations of the two tasks, which both are about the same ground truth f, even though both can be imperfect in different ways. We operate by first querying the expert, and then initialize the BO with that knowledge, which leads to a speed-up for the BO. To make this architecture work, we introduce a novel preference learning method for the expert knowledge elicitation, which is based on Siamese neural networks. We call it a preferential Bayesian neural network (PBNN); it not only learns the instance preference relationship, but is also capable of capturing the latent function shape. As working with humans implies a limited number of queries, we use active learning to sequentially ask the most informative queries from the expert. Experiments demonstrate that PBNN leads to better performance of learning preference relationships than existing GP-based approaches with limited numbers of data acquisition steps. More importantly, we show that standard BO optimization can be significantly sped-up when the elicited expert knowledge is transferred to the BO surrogate. For instance, if the simulated expert knowledge is 80% accurate, the BO speedup varies between $1\times$ and $25\times$ across the experimented benchmark functions.

In summary, the main contributions in this thesis are:

- We propose a new preference learning method for knowledge elicitation based on Siamese neural networks; it leads to better performance in capturing latent preference relationships than previous GP-based methods under a limited query budget.
- We study how to effectively transfer the elicited expert knowledge into the BO. To that end, we propose integrating the PBNN into a multi-task learning architecture. By doing so, we can provide the surrogate model of f with a good initialization and then accelerate optimization.
- We demonstrate the effectiveness of our proposed knowledge augmented-BO on different BO benchmark functions with our simulated experts.
- We further carry out demo experiments with actual human experts, which shows promising results as to the applicability of this work to real-world problems.

1.3 Outline

This thesis is organized as follows. Section 2 introduces the necessary background related to this work. Basic ideas of Bayesian optimization, Bayesian neural network, active learning, and multi-task learning will be covered. Section 3 discusses recent works of the literature that aim to elicit the human knowledge, and also the works that try to speed up BO by incorporating human belief, which have the same goal as ours but are tackled in different manners. The details of our proposed PBNN architecture is described in Section 4, as well as the active learning strategy used in the knowledge elicitation stage. In Section 5, we first describe the surrogate model used for f, and then illustrate how to incorporate the elicited expert model into the BO surrogate with the proposed multi-task learning architecture. Experimental results, which consist of preference learning performance comparison between GP and PBNN, and knowledge-augmented BO performance with different levels of simulated experts and real human experts, are shown in Section 6. Finally, Section 7 concludes the thesis and proposes some future work.

2 Background

2.1 Bayesian Optimization

2.1.1 Overview

Let $f : \mathcal{X} \to \mathbb{R}$ be a black-box function with no analytical form defined over some compact space $\mathcal{X}$. The function f lacks easy-to-optimize structure information such as concavity or linearity. Moreover, we cannot apply gradient descent or Newton's method since we are unable to access the first- or second-order information. We assume the only possible thing to do is to evaluate f at some point $\mathbf{x}$, but it may be expensive. The goal of Bayesian optimization (BO) is to find the global optimum $\mathbf{x}^\star$ of f under a limited budget, defined as

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \tag{1}$$

BO is equipped with two major components: a probabilistic surrogate model for f responsible for output prediction and uncertainty estimation, and an acquisition function α used to choose the next point to evaluate. During optimization, we first initialize the surrogate model with the original dataset, then conduct the following operations iteratively:

- Query the next point based on the acquisition function's score across the input domain.
- Augment the existing dataset with the newly added sample.
- Update our surrogate model with the new dataset.

Algorithm 1 Basic BO algorithm

Input: BO acquisition budget J, Initial dataset $\mathcal{D}_f$

Output: Minimum of f

```
1: Initialize $y_{best}$ with the current minimal function value in $\mathcal{D}_f$
2: Initialize a surrogate model $\tilde{f}$ with $\mathcal{D}_f$
3: for $j = 1$ to $J$ do
4:     $\check{\mathbf{x}}_j = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x})$
5:     Evaluate $f(\check{\mathbf{x}}_j)$
6:     $\mathcal{D}_f \leftarrow \mathcal{D}_f \cup (\check{\mathbf{x}}_j, f(\check{\mathbf{x}}_j))$
7:     Update the surrogate model with $\mathcal{D}_f$
8:     $y_{best} \leftarrow \min(y_{best}$, minimum of posterior mean$)$
9: end for
10: return $y_{best}$
```

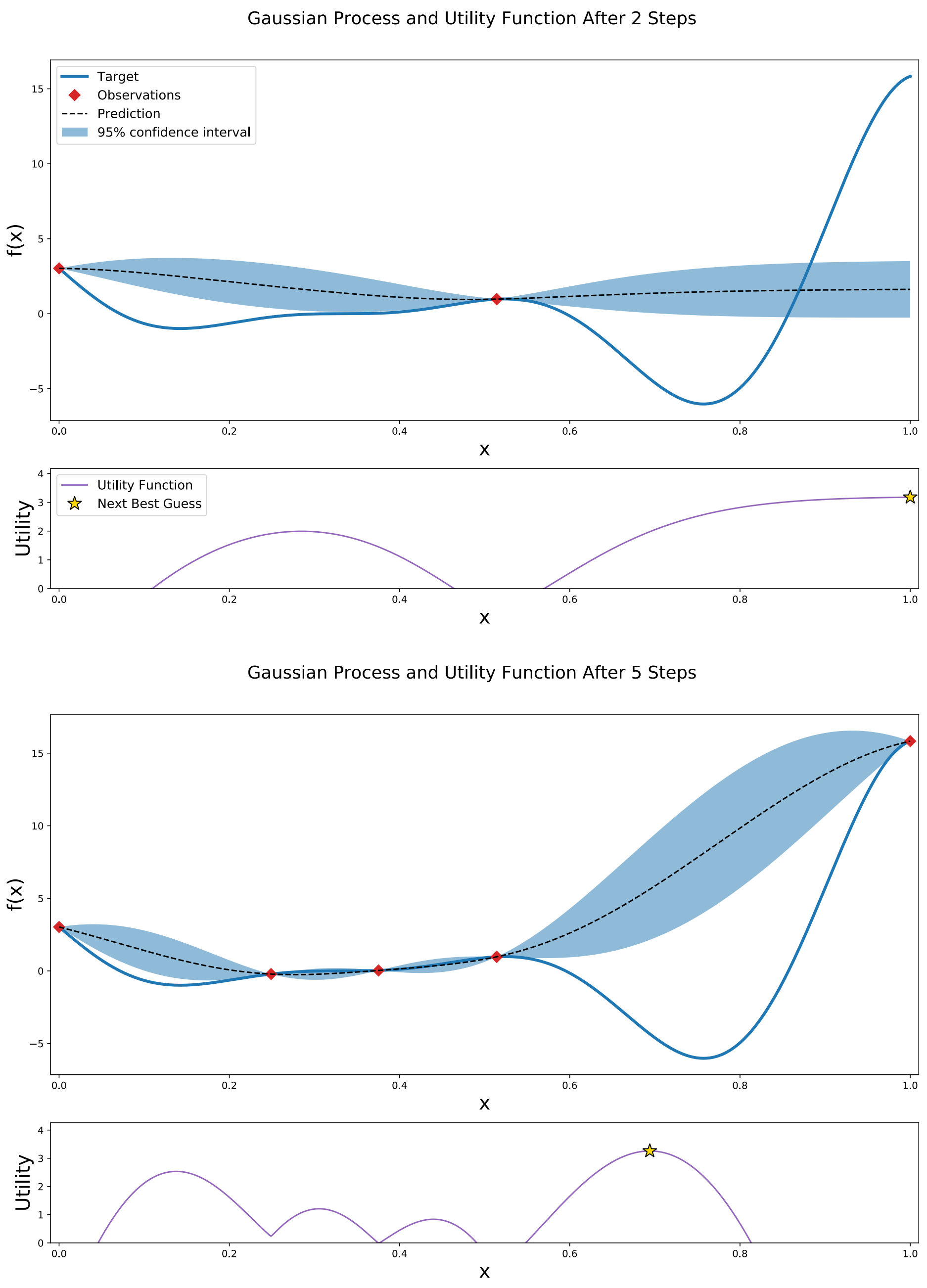

Figure 1: A visualization of how Bayesian optimization works. The target is 1D Forrester function, we use a GP as the surrogate model. The above figure is the posterior prediction distribution with the utility of the acquisition function after 2 steps. The figure below is the result after 5 steps. BO chooses the next query point with the highest utility score of acquisition function.

This standard process is summarized in Algorithm 1. A visualization of BO in Algorithm 1 with a GP [Williams and Rasmussen, 2006] surrogate model and the expected improvement acquisition function (EI) [Jones et al., 1998] after 2 and 5 steps is shown in Figure 1. The solid blue line is the objective function, we use the 1D Forrester function between 0 and 1. The black dotted line is the predicted mean value returned by the Gaussian Process (GP) surrogate model, and the shaded region corresponds to the 95% confidence interval. The bottom panel shows the utility score of the EI function, which we choose the place with maximal value as the next point to query. We will cover more details of the surrogate model and the acquisition function in the subsequent subsections.

2.1.2 Surrogate Model

GP [Williams and Rasmussen, 2006] is the most common surrogate model for BO, as it naturally fits the problem with its flexibility and built-in uncertainty estimation. A GP is a stochastic process that consists of a collection of random variables, where the joint distribution of any finite subset of these is a Gaussian distribution. A GP is fully specified by its mean function m and its covariance function K. The mean function m specifies the expected function value of input $\mathbf{x}$. In BO, m is usually a zero-mean function, i.e., $m(\mathbf{x}) = 0$. K is also called the kernel where $K(\mathbf{x}, \mathbf{x}')$ specifies the statistical relationship between two input points $\mathbf{x}$ and $\mathbf{x}'$. It can be viewed as a method to define the similarity between inputs. There are multiple choices of the kernel, such as squared exponential kernel, linear kernel and periodic kernel. We place a GP prior on the function f, which we write as:

$$f \sim GP(\mathbf{0}, K(\mathbf{x}, \mathbf{x}')). \tag{2}$$

Given a dataset $\mathcal{D}$ which consists of a collection of input points $[\mathbf{x}_1, ..., \mathbf{x}_N]$ and their corresponding function values $f_{1:N} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]$, GP constructs a prior distribution with a zero-mean vector, and a covariance matrix $\mathbf{K}$, where $[\mathbf{K}]_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. When we need to evaluate the predicted function value over a new point $\tilde{\mathbf{x}}$, with the property of multivariate Gaussian distribution, the posterior distribution $P(f(\tilde{\mathbf{x}})|\tilde{\mathbf{x}}, \mathcal{D})$ is also a closed-form Gaussian with:

$$\mu(\tilde{\mathbf{x}}) = \mathbf{k}^T\mathbf{K}^{-1}f_{1:N} \tag{3}$$

$$\sigma^2(\tilde{\mathbf{x}}) = K(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}, \tag{4}$$

where $\mathbf{k} = [K(\tilde{\mathbf{x}}, \mathbf{x}_1), ..., K(\tilde{\mathbf{x}}, \mathbf{x}_N)]$. Therefore, it is convenient to obtain the uncertainty and select the next query over the acquisition function. For more details about the GP, we refer readers to Williams and Rasmussen [2006].

However, GPs scale cubically with the number of available points. It will be impractical when the dataset is large since the computational cost becomes the bottleneck. Moreover, when the data complexity increases, choosing an effective kernel function and its hyperparameters also becomes a tough problem. In Snoek et al. [2015], a BNN (see Section 2.2) is used as a surrogate model to solve the above-mentioned problem, which was the first time BNNs were used as an alternative to GPs to serve as the surrogate model in BO. Kim et al. [2021] used BNNs to incorporate auxiliary information such as symmetries and constraints, which allows the BO to handle more complex tasks.

There are also other surrogate models for BO, such as neural process [Shangguan et al., 2021] or random forest [Jenatton et al., 2017]. Regardless of the type of functional model, as long as it can give reasonable predictions and reliable uncertainty, it can be used as a suitable surrogate model for BO tasks.

2.1.3 Acquisition Functions

The role of the acquisition function is to seek the next informative point to query for the goal of finding the minimizer of f. It has to navigate a trade-off between exploration and exploitation. Exploration aims to discover the most uncertain part where we lack data evidence and are thus not confident about the prediction. Exploitation tries to find the optimal place by staying close to what we currently consider to be a good region for the global minimizer. Typically, we are going to maximize the function α to get the next point, that point might have either a low predicted function value or a large uncertainty.

The most popular acquisition function is expected improvement (EI) [Jones et al., 1998] (see Section 5.3), where it quantifies the improvement over the current optimal function value f^*. There are also a variety of other acquisition functions, we only list some representative ones:

- Probability of improvement (PI) [Kushner, 1964]: in contrast to the EI, which evaluates the magnitude of the improvement, PI only considers the probability of improving the current best estimate. It selects the point with the highest probability of improvement.

- Gaussian process upper confidence bound (GP-UCB) [Srinivas et al., 2009]: tries to balance between exploration and exploitation. In the minimization problem, it has the form $\alpha_{UCB} = \mu(\mathbf{x}) - \beta\sigma(\mathbf{x})$, where β is a trade-off parameter.

- Entropy search (ES) [Hennig and Schuler, 2012], predictive entropy search (PES) [Hernández-Lobato et al., 2014] and max-value entropy search (MES) [Wang and Jegelka, 2017]: these acquisition functions tackle the problem from the information-theoretic view, where they choose the points that minimize the entropy of the predicted distribution.

2.1.4 Multi-Fidelity Bayesian Optimization

A research direction related to our problem is multi-fidelity BO (MFBO). In many realistic problems, in contrast to the traditional setting, MFBO assumes that the function value can be accessed in different fidelities. Different fidelities mean the degree of exactness of the objective function is different, the low fidelity value might be cheaper to evaluate, while the high fidelity value requires simulation with high computational cost. For instance, the simulation of robot action can be realized by expensive actual operation or cheap computer simulation. MFBO tries to reduce the cost by querying the low-value region cheaply with low-fidelity data and captures the optimal value speedily by acquiring high fidelity data on a small promising region. Kandasamy et al. [2016] developed a novel MFBO algorithm called Multi-Fidelity Gaussian Process Upper Confidence Bound (MF-GP-UCB), in which they query the point at each fidelity until a confidence bound exceeds a threshold. As they model each fidelity independently with different GPs, the information between each fidelity is not well-shared. To address the problem, Li et al. [2020b] proposed Deep Neural Network Multi-Fidelity Bayesian Optimization (DNN-MFBO) to establish the connection between different fidelities with a modularized Bayesian neural network, which led to improved performance compared w.r.t. previously proposed methods.

In our problem, the expert can be regarded as a low-fidelity information source since it is costless to query, and the real function values can be seen as the high-fidelity data. The only difference is that we formulate it as a two-step procedure where we query the expert first to get an auxiliary model, then perform standard BO with single fidelity, while MFBO usually conducts the experiment simultaneously.

2.2 Bayesian Neural Network

2.2.1 Definition

A neural network (NN), also known as an artificial neural network (ANN) [LeCun et al., 2015], is a machine learning model inspired by the human brain, which tries to mimic the behavior of synapses that transmit information. It consists of a collection of layers, each layer contains many nodes (artificial neurons). Each node between two conjoint layers has an edge and is assigned a weight parameter.

A standard ANN assigns each parameter a deterministic value, which often tends to be overconfident about its prediction and is prone to overfitting. Bayesian neural network (BNN) is a stochastic artificial neural network that aims to alleviate the above problem with Bayesian inference. Rather than a fixed value, BNN assigns each weight a probability distribution, Figure 2 shows the difference between the classical neural network with BNN. It can be viewed as a special case of ensemble learning, where a BNN is an ensemble of myriad neural networks, the parameters of each network are drawn from a shared probability distribution. By aggregating different sets of parameters, a BNN is able to estimate the epistemic uncertainty of a target by comparing the predicted outputs of multiple sampled parameters. The ability of uncertainty estimation provides us with a new metric to monitor our prediction with the model.

To train a BNN, we first put a prior distribution $P(\mathbf{w})$ over weights $\mathbf{w}$ which represents our prior belief towards the weights, then update the belief based on the training data $\mathcal{D}$ to obtain the posterior distribution of weights:

$$P(\mathbf{w}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathbf{w})P(\mathbf{w})}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\mathbf{w})P(\mathbf{w})}{\int_{\mathbf{w}} P(\mathcal{D}|\mathbf{w})P(\mathbf{w})d\mathbf{w}}. \tag{5}$$

Where $P(\mathcal{D}|\mathbf{w})$ is the likelihood which encodes the aleatoric uncertainty, and $P(\mathcal{D})$ is called evidence or marginal likelihood. Computing the posterior distribution is usually intractable, since the evidence does not have a closed form solution. To address this, there are two available classes of solutions, Markov Chain Monte Carlo (MCMC) and variance inference (VI).

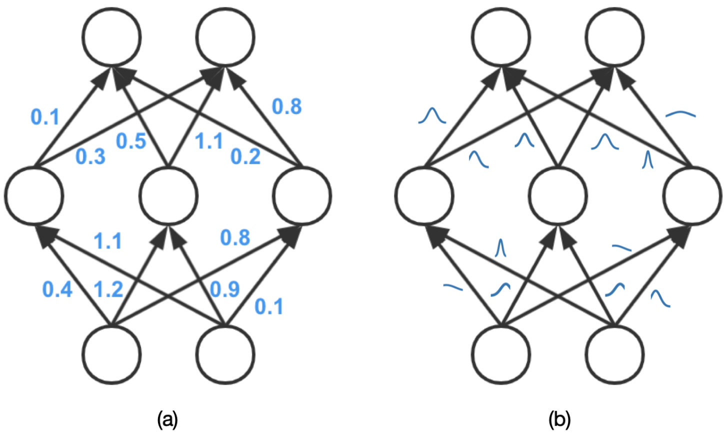

Figure 2: (a) A standard artificial neural network structure with deterministic value for each weight. (b) Bayesian neural network structure, each weight is assigned a distribution.

When making the prediction for the unseen data, the distribution we really care about is posterior predictive distribution $P(\mathbf{y}|\mathbf{x},\mathcal{D})$:

$$\begin{aligned} P(\mathbf{y}|\mathbf{x},\mathcal{D}) &= \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}[P(\mathbf{y}|\mathbf{x},\mathbf{w})] && (6)\\ &= \int_{\mathbf{w}} P(\mathbf{y}|\mathbf{x},\mathbf{w})P(\mathbf{w}|\mathcal{D})d\mathbf{w}. && (7) \end{aligned}$$

Taking the expectation equals making the prediction over all the possible weights; it is a weighted sum of infinite neural networks. In practice, Equation 7 is intractable, so we usually adopt the Monte Carlo approximation:

$$y = \frac{1}{T}\sum_{t=1}^{T} \phi_{\mathbf{w}_t}(\mathbf{x}), \tag{8}$$

where $\mathbf{w}_t \sim P(\mathbf{w}|\mathcal{D})$ and $\phi(\cdot)$ is the forward function of the neural network (a forward pass through all the neural network parameters).

Exploring Bayesian deep learning has multiple advantages. First, it provides a measure of uncertainty for traditional neural networks. This additional metric can help us know more about neural network prediction, as it can prevent the model from being overconfident. Compared with the GPs, another inference model equipped with uncertainty estimation, BNN is more scalable and can be efficiently trained on large datasets. Second, using Bayesian learning, we can interpret some useful tricks in a neural network as explicit prior, e.g., regularization tricks or data augmentation. Therefore, BNN can provide us with a better perspective to explain the theory behind the neural network.

2.2.2 Bayesian Inference Methods

Computing the posterior $P(\mathbf{w}|\mathcal{D})$ is almost always impossible, as it includes the evidence term, which is almost always intractable. There are two existing classes of algorithms that tackle the problem: Markov Chain Monte Carlo (MCMC), which constructs a Markov Chain to sample from the exact posterior distribution; Variational inference (VI), which approximates the posterior distribution with a tractable surrogate distribution. Compared with MCMC, VI gains more attention these days as it is more scalable than MCMC. In this thesis, we will only cover VI. We refer the reader to Jospin et al. [2020] for more details about MCMC methods.

In VI, we are looking for the closest distribution to the true posterior distribution $P(\mathbf{w}|\mathcal{D})$ within a family of distributions $q(\mathbf{w}|\theta)$ parameterized by θ. The optimal value of θ is obtained by minimizing the Kullback-Leibler (KL) divergence between these two distributions:

$$\begin{aligned} KL[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] &= \int q(\mathbf{w}|\theta)\log\frac{q(\mathbf{w}|\theta)}{P(\mathbf{w}|\mathcal{D})}d\mathbf{w} && (9) \\ &= \int q(\mathbf{w}|\theta)\log\frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})}d\mathbf{w} + \log P(\mathcal{D}). && (10) \end{aligned}$$

Since $P(\mathcal{D})$ is a constant w.r.t. $\mathbf{w}$, minimizing the KL divergence $KL[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})]$

equals to maximizing $\mathcal{L}(\theta)$:

$$\mathcal{L}(\theta) = \int q(\mathbf{w}|\theta) \log \frac{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})}{q(\mathbf{w}|\theta)} d\mathbf{w} \tag{11}$$

$$= \int q(\mathbf{w}|\theta) \log \frac{P(\mathbf{w})}{q(\mathbf{w}|\theta)} d\mathbf{w} + \int q(\mathbf{w}|\theta) \log P(\mathcal{D}|\mathbf{w}) d\mathbf{w} \tag{12}$$

$$= -KL[q(\mathbf{w}|\theta)||P(\mathbf{w})] + \mathbb{E}_{q(\mathbf{w}|\theta)}(\log P(\mathcal{D}|\mathbf{w})), \tag{13}$$

which is called the evidence lower bound (ELBO), it consists of two terms, the first part is the data dependent likelihood term under the expectation of the variational distribution on weights, the second part is the complexity cost which depends on the prior distribution of weights. We take ELBO as our objective function to optimize the model. In order to make it applicable to BNNs, Blundell et al. [2015] proposed a practical solution called Bayes by Backprop (BBB), a practical implementation of stochastic variational inference incorporated with a reparameterization trick to make sure back-propagation works normally. It can be seen as a breakthrough in Bayesian deep learning, as it provides a standard training process for the Bayesian neural network.

Recently there have been some new inference approaches designed for deep learning. One remarkable work is MC-Dropout [Gal et al., 2017], which used Dropout as the posterior approximation. Compared with the classical VI methods, MC-Dropout is easier to implement and also leads to faster training, as it quantifies the uncertainty only by turning the Dropout layer on during the test stage. However, in some simple cases that do not require expensive computational costs, BBB usually performs better in uncertainty estimation and generalization ability.

2.3 Active Learning

In the traditional supervised learning setting, we have a training set comprising labeled examples, and we build a model based on this training set. However, it can be expensive to obtain labeled data in a real-life setting, especially in some cases where the object can only be evaluated through simulations or actual experiments. In contrast to the passive setting, *active learning* (AL) [Settles, 2012] sequentially selects the most informative data to label based on a specific criterion and aims to maximize the model accuracy in a data-efficient way. AL approaches can be divided into membership query synthesis [Angluin, 1988], stream-based AL [Dagan and Engelson, 1995] and pool-based AL [Lewis and Gale, 1994]. The current mainstream method is pool-based AL, which chooses the best candidate from a pre-defined set of unlabeled instances. The standard pool-based AL cycle is shown in Figure 3. In machine learning, AL usually is used for classification problems, however the concept straightforwardly extends to regression problems, and there is a more general concept in statistics called "Bayesian experimental design" [Chaloner and Verdinelli, 1995], which aims to design experiments to make the outcomes as informative as possible towards the parameters that we are interested in.

Formally, we have an unlabeled pool $\mathcal{D}_{pool} = \{\mathbf{x}_n\}_{n=1}^{N}$ with N samples, and $\mathcal{D} = \{\mathbf{x}_m, y_m\}_{m=1}^{M}$ is the current labeled training set, our target is to design a query strategy α_{AL} (a.k.a., the acquisition function) to select the data from $\mathcal{D}_{pool}$ to augment $\mathcal{D}$, expecting that $M \ll N$. We wish to use this α_{AL} to choose the most informative samples to reach higher model accuracy in a limited budget setting. Therefore, designing an effective query strategy is the most crucial part in AL problem.

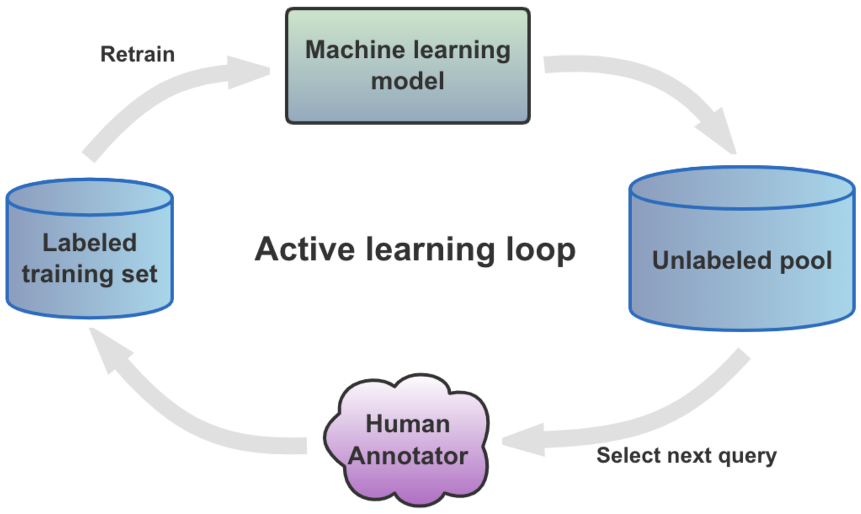

Figure 3: A standard pool-based active learning cycle. We iteratively select the most "interesting" sample to annotate, then augment the existing dataset with newly annotated data to train a better model.

Many different criteria have been proposed, here we are only going to detail the most classical ones. The simplest way to design an acquisition function is to select the most uncertain samples under the current model predictions, which is called uncertainty sampling [Lewis and Catlett, 1994]. The idea is to iteratively label the points that the current model considers the most confusing, then train with these newly added data. In this way, it will reduce the model uncertainty the most. There are various methods to measure the uncertainty based on the actual problem setting. *Least confident sampling* is one of the most straightforward uncertainty sampling methods, it selects the samples whose most confident prediction (highest probability) is actually the smallest one among all the prediction probabilities of unlabeled instances. The drawback of least confident sampling is that it only considers the best prediction while discarding the information in rest labels. Another approach is based on the output margin, which is called *margin sampling*. It chooses a sample with a minimum distance between the highest and second-highest prediction probabilities. It is an improvement of least confident sampling as it takes the second-best labeling into account. Still, when the problem has large categories, margin sampling will lead to information loss. The most common method for uncertainty sampling is the entropy-based method (*entropy sampling*), where it chooses the instance with the largest entropy of the predicted distribution.

In Houlsby et al. [2011], a new criterion called Bayesian Active Learning by Disagreement (BALD) was proposed, where it selects the data points which maximize the mutual information between model predictions and parameters. It is a different interpretation of the expected information gain (EIG) Bayesian experimental design [Lindley, 1956] that computes the entropies in the output space instead of parameter space. In other words, it seeks the next query point for which model is highly uncertain about its predicted label, but each pass through the model tends to produce a similar uncertain result. Kirsch et al. [2019] extended BALD to apply it under batch query setting.

As a methodology that aims to achieve the excellent result under a limited budget, AL is naturally suitable for the interaction with an expert, as human only has limited patience for querying, and we need to find the most informative points for an expert to label.

2.4 Multi-Task Learning

2.4.1 Definition

In contrast to single-task learning, multi-task learning (MTL) [Caruana, 1997] aims to optimize more than one objective in a single learning system. By sharing latent representations between a set of related tasks, it can potentially speed up the learning by integrating the knowledge. Some MTL systems target to optimize several tasks simultaneously and gain improvement on multiple objectives, here we only consider the case where an additional task is introduced to accelerate our main target. We also refer to this method as learning with auxiliary tasks. Since the intermediate representations of different tasks can be shared, we can gain performance improvement in MTL by incorporating specific domain knowledge into another task. In recent years, MTL has been successfully applied to different machine learning fields, from computer vision [Zhang et al., 2014, Dai et al., 2016] to natural language processing [Liu et al., 2019b, 2015]. Recently, multi-modal fusion has gradually gained more attention as it aims to handle multiple data modalities simultaneously to explore the possibility of joint learning. It has a very similar idea with the MTL as they both target to share the latent representation across domains to improve generalization. In traditional deep MTL, the feature vectors are shared in different tasks but within the same modality. However, in multi-task multi-modal learning [Nguyen and Okatani, 2019, Pramanik et al., 2019], the features are distilled with several tasks in different modalities, which is considered an improvement of MTL.

One of the intuitions behind MTL comes from the human cognitive system, where humans tend to utilize the knowledge acquired from previous related tasks when learning a new task. For instance, when a baby learns from babbling to speaking, it constantly learns the abstract pronunciation system and intonation. Once these basic concepts are learned, they can be reused for studying more complex tasks, such as singing and tongue-twisters. In fact, whenever humans want to perceive something new, they always take the prior knowledge they have acquired to speed up their understanding. However, current networks always learn a task from scratch through large training samples and take days of work. MTL tries to solve the problem by introducing a proper inductive bias to eliminate certain improper hypotheses. Auxiliary tasks can provide additional information to let the model prefer hypotheses that favour more than one task, which leads to better generalization.

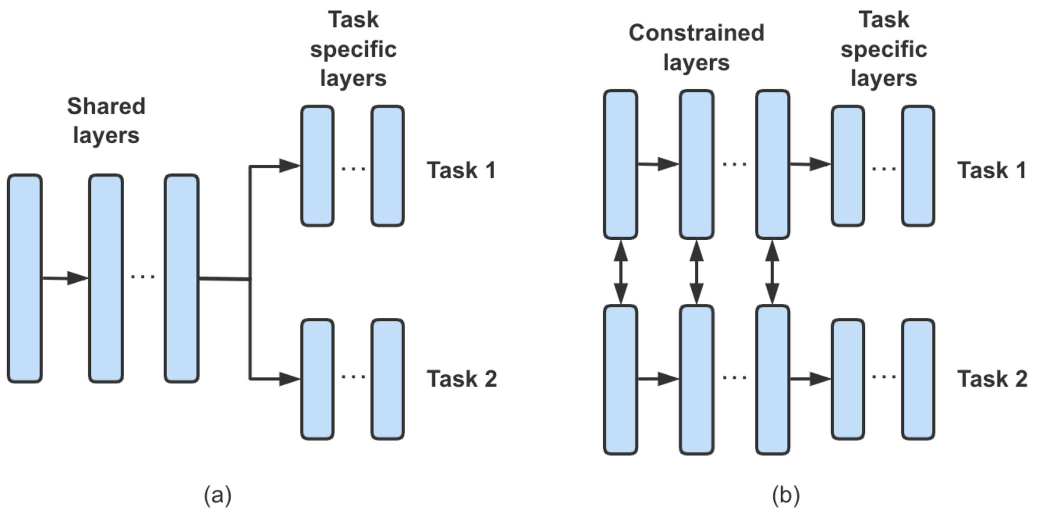

Figure 4: (a) Hard parameter sharing: the parameters in hidden layers are shared among different tasks. (b) Soft parameter sharing: each task has its own stream, but there are some regularization methods to constrain the hidden layers to be more similar between different tasks.

2.4.2 Methods

There are typically two different methods to perform deep MTL, called *soft* or *hard parameter sharing* of hidden layers. In hard parameter sharing, weights in hidden layers are shared among different tasks, while several task-specific output layers are placed after these hidden layers to make predictions for each task. Hard parameter sharing can significantly reduce the overfitting, since it aims to learn a joint representation that captures common features across all the tasks. On the other hand, soft parameter sharing does not require the shared hidden layers, each task has its own learning stream with separate parameters. To establish the connection between tasks, soft parameter sharing put the regularization trick to narrow the distance between parameters. Although there is no explicit parameter sharing scheme, it imposes potential constraints to make parameters more similar between tasks. The difference between these two methods is illustrated in Figure 4.

2.4.3 Loss Weighting

During the optimization, we need to seek a balance among the different loss functions of each task. As we train on several tasks, we must combine the various task-specific loss functions to a unified loss for the model to minimize. A natural question will be how to effectively combine multiple loss functions to make it suitable for the MTL. The most common way is to put a scheduled weight on each loss function to form an aggregated loss, then the problem lies on how to design this schedule properly. The most straightforward way of designing schedule is to declare a static scheme before training stage, it relies on the prior knowledge of human hyper-parameter tuner to provide a suitable scheduler, which is usually sub-optimal. Another way is to adaptively learn this weight thorough training, there are several ways, such as weighting by uncertainty [Kendall et al., 2018], weighting by learning speed [Liu et al., 2019a] and weighting by performance [Guo et al., 2018].

3 Related Work

3.1 Knowledge Elicitation

Knowledge Elicitation is a set of methods that aims to elicit the knowledge of a domain expert. The research mainly focuses on how to efficiently interact with the expert to gather useful information. It has been originally used for constructing prior distribution in Bayesian data analysis [O'Hagan et al., 2006, Mikkola et al., 2021]. The target is to transform the tacit expert knowledge into explicit prior information and further constraint the range of parameters in statistical models. A recent line of works [Soare et al., 2016, Micallef et al., 2017, Afrabandpey et al., 2017, Daee et al., 2017, Sundin et al., 2018] focus on improving prediction by incorporating the expert's feedback into the experimental design. Soare et al. [2016] focused on the "small n large p" regression problem, where the number of samples n is very small compared with the feature dimensionality p. It is impossible to learn a model with good generalization ability in this case. But with the additional expert knowledge, even for the extreme case where $n \rightarrow 1$, it may become realistic to learn a useful predictor. In their work, they assumed the expert knowledge was reliable and accurate, but they had only a limited budget to query. The experiments showed significant improvement with the expert feedback engaged, which opened the door for the following research line. Micallef et al. [2017] presented a novel interactive visualization method to model the domain expert's knowledge of the relevance between features and targets, then taking this elicited information as prior knowledge to obtain a model with higher precision. And showed better performance compared with using randomly chosen features. For the same purpose, Afrabandpey et al. [2017] proposed an expert prior knowledge elicitation method that considered the pairwise similarities between the features. By using an interactive MDS-type scatter-plot, users can provide feedback with their expertise. The feedback then is transformed into a prior distribution of the Bayesian linear regression coefficients. Daee et al. [2017] formulated the knowledge elicitation as a probabilistic inference process. They integrated the expert and regression models into a joint probabilistic model, making it possible to update the uncertainties after each interaction with the expert. The selection process is then treated as an optimal experimental design problem, where the most informative queries are selected by maximizing the expected information gain. Sundin et al. [2018] applied the expert knowledge to the precision medicine field and introduced a new targeted knowledge elicitation approach, which was the extension of the work in Daee et al. [2017].

3.2 User Belief-Enhanced BO

A few very recent works have been proposed to incorporate user belief about the global optimum into BO as prior knowledge for acceleration. Li et al. [2020a] was the first to take advantage of this extra information into account. In their work, a vague probability dense function $\pi(\mathbf{x}^*)$, which encodes the user belief towards the optimal value $\mathbf{x}^*$ was proposed as a tractable form of prior knowledge representation. Then, Thompson sampling was deployed to sample a number of functions from the GP posterior, and the extremes of each function were further obtained. The next point to be queried was then selected by evaluating over $\pi(\mathbf{x}^*)$. However, their method is restricted to only one acquisition function and can only be applied to GP surrogate model.

Ramachandran et al. [2020] developed a method that is agnostic to any acquisition function. Unlike the previous standard BO methods that assign each point an equal probability to be the optimum, they put an expert prior on the model that warped the search space with the probability integral transform. The regions with a higher chance of containing the optimum are expanded, and other regions are shrunk. However, it is still impractical if the prior is misleading, as the searching space is warped on this prior and is hard to recover.

Souza et al. [2021] proposed BOPrO, which combined the expert's prior knowledge with the surrogate model of BO to derive a pseudo-posterior. The configurations were then chosen using the EI acquisition function. Compared with the above methods, it was agnostic to the model used and was capable of recovering from the misleading prior. However, it is restricted to the EI acquisition function and cannot provide convergence guarantees. Hvarfner et al. [2022] improved the above method by introducing πBO, a flexible human belief-augmented BO structure that was not constrained by both the surrogate model and the acquisition function. They implemented it by simply multiplying the acquisition function by the expert's prior distribution, which is straightforward. To deal with the poorly-chosen priors, a decay schedule is arranged to weaken the effect of priors over time.

While all of these methods aim to incorporate expert belief into BO, they neglect the fact that the priors are not coming out of thin air. They all ignore the stage where such a prior has to actually be elicited from a human expert. This elicitation step should be taken into account for human-in-the-loop machine learning methods. In our thesis, an end-to-end approach is proposed, which considers both the elicitation and incorporation stages.

4 Preferential Bayesian Neural Network

In order to elicit the knowledge of the domain expert, we model it as a function, denoted by g, which represents the beliefs of the expert. We can interpret g as a biased version of f. By querying pairwise comparisons from the expert, we build a probabilistic surrogate of g. Note that g corresponds to the *utility function* of the expert, which is a well-studied concept in economics [Rader, 1963].

As motivated in the introduction, it is much easier for humans to compare two items than to give the absolute magnitude of one item (for instance it can be extremely difficult for a material scientist to compute a potential energy of a molecule, but comparing the stability between two molecules is easier). Hence, we assume that the expert cannot directly give the value $g(\mathbf{x})$ for a certain $\mathbf{x}$. Instead, given a pair of covariates $[\mathbf{x}, \mathbf{x}'] \in \mathcal{X} \times \mathcal{X}$, we assume that the expert is able to return a preference label $\hat{y} \in \{0, 1\}$, with value $\hat{y}_i = 1$ if $g(\mathbf{x}) \geq g(\mathbf{x}')$, and $\hat{y}_i = 0$ if $g(\mathbf{x}) < g(\mathbf{x}')$. We will sequentially collect a dataset $\mathcal{D}_g = \{(\mathbf{x}_i, \mathbf{x}'_i, \hat{y}_i)\}_{i=1}^{M}$, which is in turn used to build a probabilistic surrogate of g. Note that based on the ordered data, we will be able to learn about the shape of g, but not about the actual magnitude and scale, meaning that any monotonic transformation of g is an equivalent solution.

We introduce a neural network-based architecture to handle preference learning and approach this problem as binary classification. More precisely, given two inputs $\mathbf{x}$ and $\mathbf{x}'$, we wish to output a value in $[0, 1]$ that corresponds to the probability of $\hat{y}$ equaling 1. A natural solution to this problem would be to expand the input space to $\mathcal{X} \times \mathcal{X}$ by concatenating the covariates pair. Such an architecture is displayed in Figure 5-a. However, by doing so, we would not learn anything about the function g. Instead, we propose to use an architecture based on Siamese networks (Figure 5-b), coined PBNN (preferential Bayesian neural network), which is detailed in the next subsection.

4.1 Preference Learning With Siamese Networks

Network architecture and loss function

A Siamese neural network consists of two parallel, identical sub-networks that share the same set of parameters. Each sub-network takes a distinct input, and the representations produced by each network are then compared using a connection function, which we denote by ζ. They were introduced in the 90s for signature verification [Bromley et al., 1993], and have since become very popular for, e.g., one-shot/few-shot learning [Koch et al., 2015], and object tracking [Bertinetto et al., 2016]. As our knowledge elicitation task amounts to a comparison between two values at a time, the Siamese network architecture naturally fits to our problem.

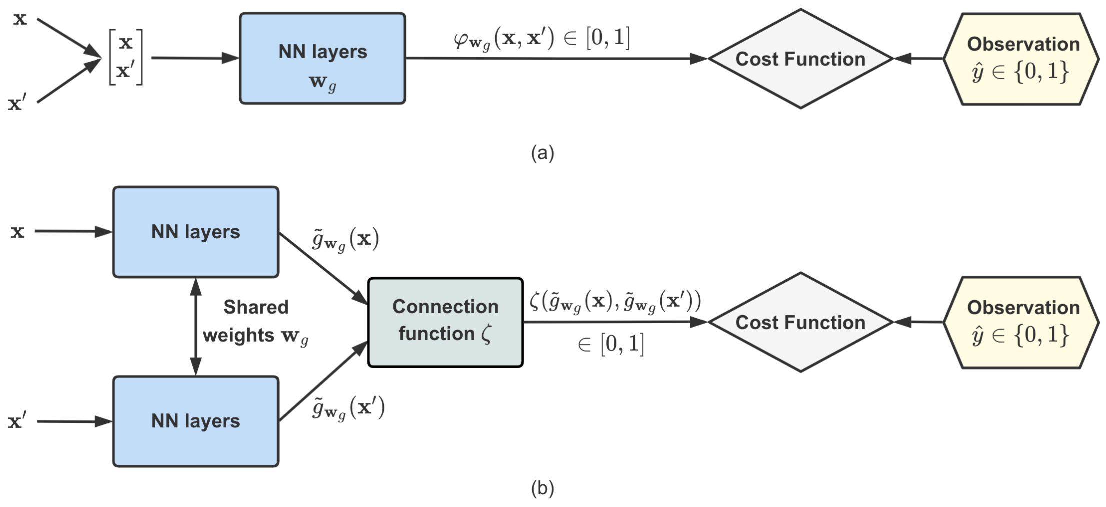
$\mathbf{x}$
$\mathbf{x}'$
$\begin{bmatrix}\mathbf{x}\\ \mathbf{x}'\end{bmatrix}$
NN layers $\mathbf{w}_g$
$\varphi_{\mathbf{w}_g}(\mathbf{x},\mathbf{x}') \in [0,1]$
Cost Function
Observation $\hat{y} \in \{0,1\}$
(a)
$\mathbf{x}$
NN layers
$\tilde{g}_{\mathbf{w}_g}(\mathbf{x})$
Shared weights $\mathbf{w}_g$
Connection function ζ
$\zeta(\tilde{g}_{\mathbf{w}_g}(\mathbf{x}), \tilde{g}_{\mathbf{w}_g}(\mathbf{x}'))$ $\in [0,1]$
Cost Function
Observation $\hat{y} \in \{0,1\}$
$\mathbf{x}'$
NN layers
$\tilde{g}_{\mathbf{w}_g}(\mathbf{x}')$
(b)

Figure 5: (a) A simple neural network architecture to handle preference learning. The neural network outputs the probability of $\hat{y}$ to be 1 given $\mathbf{x}$ and $\mathbf{x}'$, but fails at capturing the shape of the utility function of the expert, g. (b) The proposed architecture, based on a Siamese neural network. It also solves the preference learning problem, but each sub-network outputs a real-valued latent representation that we interpret as $g(\mathbf{x})$. The network is able to learn the shape of g, up to a monotonic transformation.

The proposed PBNN uses that architecture exactly. Let us denote by $\mathbf{w}_g$ the weights shared by the two sub-networks, and let us denote by $\tilde{g}_{\mathbf{w}_g}(\mathbf{x})$ and $\tilde{g}_{\mathbf{w}_g}(\mathbf{x}')$ the representations produced by forwarding $\mathbf{x}$ and $\mathbf{x}'$. PBNN models the probability of $\hat{y}$ to be 1 given two inputs $\mathbf{x}$ and $\mathbf{x}'$ by comparing $\tilde{g}_{\mathbf{w}_g}(\mathbf{x})$ and $\tilde{g}_{\mathbf{w}_g}(\mathbf{x}')$ with the connection function ζ. Contrary to the "concatenation" baseline approach previously described (Figure 5-a), the representations produced by the two sub-networks are real-valued, and we interpret them as the values of the true function g. These two values are further combined to provide a value in $[0, 1]$, i.e., our connection function is naturally chosen as

$$\zeta(\tilde{g}_{\mathbf{w}_g}(\mathbf{x}), \tilde{g}_{\mathbf{w}_g}(\mathbf{x}')) = \sigma(\tilde{g}_{\mathbf{w}_g}(\mathbf{x}) - \tilde{g}_{\mathbf{w}_g}(\mathbf{x}')), \tag{14}$$

where $\sigma(x) = \dfrac{1}{1+e^{-x}}$ is the sigmoid function. We can then train the whole model by minimizing the negative log-likelihood, which is equivalent to using the binary cross-entropy loss. We write

$$\log p(\mathcal{D}_g|\mathbf{w}_g) = \sum_{i=1}^{M} \log p(\hat{y}_i|[\mathbf{x}_i, \mathbf{x}'_i], \mathbf{w}_g) \tag{15}$$

$$= \sum_{i=1}^{M} \Big(\hat{y}_i \log \big(\zeta(\tilde{g}_{\mathbf{w}_g}(\mathbf{x}), \tilde{g}_{\mathbf{w}_g}(\mathbf{x}'))\big) \tag{16}$$

$$+ (1-\hat{y}_i) \log \big(1 - \zeta(\tilde{g}_{\mathbf{w}_g}(\mathbf{x}), \tilde{g}_{\mathbf{w}_g}(\mathbf{x}'))\big)\Big).$$

To sum things up, the Siamese network-based architecture also solves the binary classification problem, but does so by learning an intermediate representation, which we identify as $g(\mathbf{x})$. However, the current architecture only outputs a point estimate for $\tilde{g}_{\mathbf{w}_g}(\mathbf{x})$, which is unsatisfying in our scenario where we wish to characterize uncertainties for active learning. To do so, we resort to Bayesian inference.

Bayesian inference

To characterize the posterior distribution $p(\mathbf{w}_g|\mathcal{D}_g) \propto p(\mathcal{D}_g|\mathbf{w}_g)p(\mathbf{w}_g)$, the network weights $\mathbf{w}_g$ are equipped with a prior distribution $p(\mathbf{w}_g)$. The posterior is then in turn used to compute the predictive posterior distribution of $\tilde{g}_{\mathbf{w}_g}(\mathbf{x})$. We resort to variational inference to characterize the posterior distribution. Variational inference aims at finding the closest approximation in terms of Kullback-Leibler divergence to $p(\mathbf{w}_g|\mathcal{D}_g)$, among a chosen family of distributions parameterized by $\boldsymbol{\theta}_g$. Let us denote this approximation by $q(\mathbf{w}_g|\boldsymbol{\theta}_g)$ (the so-called variational posterior). It can easily be shown that this amounts to minimizing the following expression w.r.t. $\boldsymbol{\theta}_g$:

$$\mathcal{L}_g(\boldsymbol{\theta}_g) = \mathrm{KL}[q(\mathbf{w}_g|\boldsymbol{\theta}_g)||p(\mathbf{w}_g)] \tag{17}$$

$$- \mathbb{E}_{q(\mathbf{w}_g|\boldsymbol{\theta}_g)}[\log p(\mathcal{D}_g|\mathbf{w}_g)],$$

where the term $\log p(\mathcal{D}_g|\mathbf{w}_g)$ is given by Eq. (16). This expression is called the negative ELBO (evidence lower bound). The loss in Eq. (17) and its gradient are intractable, but we use Bayes by backprop (BBB) [Blundell et al., 2015] as our practical implementation. It provides Monte Carlo estimators of the loss and gradients, and ensures that back-propagation works. The minimization is then simply carried out by gradient descent. We refer the reader to the original paper for details.

4.2 Active Data Acquisition

Humans are not passive sources of information, and can only answer to a certain amount of queries before growing tired or impatient. In this limited budget setting, in order to maximize the use of the expert's time, we propose to resort to active learning to learn an accurate model in a sample-efficient way. Active learning methods sequentially select the most informative unlabeled instance (usually from a pool, which we denote by $\mathcal{D}_{pool}$), get the associated label, and retrain the model. Many different informativeness criteria have been proposed in the literature [Settles, 2012]. Here, we propose to use the so-called BALD (Bayesian Active Learning by Disagreement, Houlsby et al. [2011], Gal et al. [2017]), a criterion justified from an information-theoretic perspective. BALD selects the point which maximizes the mutual information between its observation and model parameters. The optimal query $\mathbf{x}^\star$ is such that

$$\begin{aligned}\mathbf{x}^\star &= \underset{x\in\mathcal{D}_{pool}}{\arg\max}\ \mathrm{I}(y;\mathbf{w}|\mathbf{x},\mathcal{D}), &(18)\\ &= \mathcal{H}[y|\mathbf{x},\mathcal{D}] - \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\mathcal{H}[y|\mathbf{x},\mathbf{w}]], &(19)\end{aligned}$$

where $\mathcal{H}$ denotes the differential entropy. Note that this is equivalent to maximizing the expected information gain on the model parameters, a well-known strategy in Bayesian experimental design [Lindley, 1956, MacKay, 1992, Chaloner and Verdinelli, 1995].

Adapting BALD to our setting, we have

$$[\mathbf{x},\mathbf{x}']^\star = \underset{[\mathbf{x},\mathbf{x}']\in\mathcal{D}_{pool}}{\arg\max}\ \mathrm{I}(\hat{y};\mathbf{w}_g|[\mathbf{x},\mathbf{x}'],\mathcal{D}_g). \tag{20}$$

We approximate the mutual information as follows:

$$\begin{aligned}&\mathrm{I}(\hat{y},\mathbf{w}_g|[\mathbf{x},\mathbf{x}'],\mathcal{D}_g)\\ &= \mathcal{H}[\hat{y}|[\mathbf{x},\mathbf{x}'],\mathcal{D}_g] - \mathbb{E}_{p(\mathbf{w}|\mathcal{D}_g)}[\mathcal{H}[\hat{y}|[\mathbf{x},\mathbf{x}'],\mathbf{w}_g]], &(21)\\ &\simeq \mathcal{H}[\hat{y}|[\mathbf{x},\mathbf{x}'],\mathcal{D}_g] - \mathbb{E}_{q(\mathbf{w}_g|\boldsymbol{\theta}_g)}[\mathcal{H}[\hat{y}|[\mathbf{x},\mathbf{x}'],\mathbf{w}_g]], &(22)\\ &\simeq h\left(\frac{1}{T}\sum_{t=1}^{T}\hat{p}_{\mathbf{w}_g^{(t)}}(\mathbf{x},\mathbf{x}')\right) - \frac{1}{T}\sum_{t=1}^{T} h\left(\hat{p}_{\mathbf{w}_g^{(t)}}(\mathbf{x},\mathbf{x}')\right), &(23)\end{aligned}$$

where we have used the notation

$$\hat{p}_{\mathbf{w}_g}(\mathbf{x},\mathbf{x}') = \zeta(\tilde{g}_{\mathbf{w}_g}(\mathbf{x}),\tilde{g}_{\mathbf{w}_g}(\mathbf{x}')) \tag{24}$$

to denote the predicted probability that $\hat{y}=1$ given the pair $[\mathbf{x},\mathbf{x}']$ and parameters $\mathbf{w}_g$ (i.e., the output of the network with parameters $\mathbf{w}_g$), and where $h(p) = -p\log(p) - (1-p)\log(1-p)$ denotes the binary entropy function. The approximation in Eq. (22) comes from swapping the true posterior distribution $p(\mathbf{w}_g|\mathcal{D}_g)$ with the variational posterior $q(\mathbf{w}_g|\boldsymbol{\theta}_g)$, and the approximation in Eq. (23) corresponds to Monte Carlo approximations given that the $\mathbf{w}_g^{(t)}$ are i.i.d. samples from $q(\mathbf{w}_g|\boldsymbol{\theta}_g)$. We will refer to this criterion as PBALD.

5 Expert Knowledge-Augmented Bayesian Optimization

We now tackle the challenge of transferring what was learned in the previous step for the BO task. To that end, we propose to plug the previously described PBNN architecture into a wider multi-task learning (MTL) one. The MTL architecture aims at building probabilistic surrogates for both f and g, by sharing the weights of the hidden layers, and having separate weights for the last layer. Indeed, we leverage the similarity between the functions f and g by sharing some of the latent representations produced by the network. The architecture is detailed in Section 5.2. As we are eliciting the knowledge of the expert in a first step, this will have the effect of providing the surrogate model for f with a good initialization, which in turn will lead to accelerating the task of optimizing f. For this second step, we sequentially update this surrogate by collecting a dataset $\mathcal{D}_f = \{\mathbf{x}_j, f(\mathbf{x}_j)\}_{j=1}^{J}$. Those points are selected using a BO acquisition function, as explained in Section 5.3.

5.1 Surrogate model for f

The probabilistic surrogate we use for f is a Bayesian neural network. Let us denote by $\mathbf{w}_f$ its weights, with prior distribution $p(\mathbf{w}_f)$. We further denote by $\tilde{f}_{\mathbf{w}_f}(\mathbf{x})$ the output obtained by forwarding $\mathbf{x}$. Similarly, we resort to variational inference to characterize the posterior distribution $p(\mathbf{w}_f|\mathcal{D}_f)$, i.e., we aim at minimizing the following expression w.r.t. variational parameters $\boldsymbol{\theta}_f$:

$$\begin{aligned}\mathcal{L}_f(\boldsymbol{\theta}_f) = &\ \mathrm{KL}[q(\mathbf{w}_f|\boldsymbol{\theta}_f)||p(\mathbf{w}_f)] \\ &- \mathbb{E}_{q(\mathbf{w}_f|\boldsymbol{\theta}_f)}[\log p(\mathcal{D}_f|\mathbf{w}_f)],\end{aligned} \tag{25}$$

where $q(\mathbf{w}_f|\boldsymbol{\theta}_f)$ denotes the variational posterior parameterized by $\boldsymbol{\theta}_f$. Note that the log-likelihood term $p(\mathcal{D}_f|\mathbf{w}_f)$ is here Gaussian, which corresponds to the mean squared error.

A straightforward way of transferring what was learned in the first step would be to use the posterior distribution of the weights from the trained PBNN as the prior for $\mathbf{w}_f$. However, since PBNN does not learn the actual scale of f, using it to provide the prior distribution of the weights will not help at all, and may even lead to catastrophic forgetting [French, 1999], where the shape information encoded in the posterior distribution of weights is erased during the training with $\mathcal{D}_f$.

To alleviate this problem, we consider a MTL architecture, with hard parameter sharing among the hidden layers for the surrogates of f and g. In other words, we consider a joint model, whose shared parameters are going to be initialized through the trained PBNN. This is detailed next.

5.2 Multi-task learning

Parameter sharing

We adopt hard parameter sharing among the weights of the hidden layers for the surrogates of f and g. Let us split the weights $\mathbf{w}_g$ of PBNN into $\mathbf{w}_h$ and $\boldsymbol{\beta}_g$, where $\mathbf{w}_h$ are the weights of all hidden layers and $\boldsymbol{\beta}_g$ the weights of the output layer. That is, we can write $\tilde{g}(\mathbf{x}) = \boldsymbol{\beta}_g^T \phi_{\mathbf{w}_h}(\mathbf{x})$, where $\phi_{\mathbf{w}_h}(\mathbf{x})$ represents the feature vector which is produced by forwarding $\mathbf{x}$ through all the hidden layers. The weights $\mathbf{w}_h$ are going to be shared for both surrogates, i.e., we can now write that the BNN surrogate of f is parameterized by $\mathbf{w}_h$ and $\boldsymbol{\beta}_f$, such that $\tilde{f}(\mathbf{x}) = \boldsymbol{\beta}_f^T \phi_{\mathbf{w}_h}(\mathbf{x})$ is the predicted outcome.

As such, the shared representation $\phi_{\mathbf{w}_h}(\mathbf{x})$ will encode common features, such as the shape information that we wish to transfer to the surrogate of f. While expert knowledge may be biased, $\boldsymbol{\beta}_f$ can be interpreted as a calibrator to lead the surrogate of f to its actual scale and also rectify the potentially inaccurate information provided by the expert. By using the joint model, we will need fewer queries for Bayesian optimization, i.e., we save potentially expensive simulation costs.

Combining the losses

The detailed architecture of the MTL system is presented in Figure 6. Now remains the question of combining the loss functions $\mathcal{L}_g$ (Eq. (17)) and $\mathcal{L}_f$ (Eq. (25)). In order to put more emphasis on the actual acquisitions of f over time, we propose a weighted scheme with exponential decay for the $\mathcal{L}_g$. After the j-th BO acquisition, the loss $\mathcal{L}_j$ is such that

$$\mathcal{L}_j = \frac{\alpha^{j-1}}{\alpha^{j-1}+1}\mathcal{L}_g + \frac{1}{\alpha^{j-1}+1}\mathcal{L}_f, \tag{26}$$

with $\alpha < 1$ a hyper-parameter to control the speed of the decay.

5.3 Acquisition function

We adopt the expected improvement (EI) as the acquisition function Jones et al. [1998]. Given $\mu_{\mathbf{x}}$ the predictive mean of BO surrogate model and $s_{\mathbf{x}}^2$ the predictive variance, the EI at point $\mathbf{x}$ can be defined as:

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = s_{\mathbf{x}}[\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \psi(\gamma(\mathbf{x}))], \tag{27}$$

where $\gamma(\mathbf{x}) = (y_{best} - \mu_{\mathbf{x}})/s_{\mathbf{x}}$, y_{best} is the current lowest value of the objective function, and the $\Phi(\cdot)$, $\psi(\cdot)$ are the cumulative distribution function and probability density function of a standard normal random variable. Since Eq. (27) is intractable for a

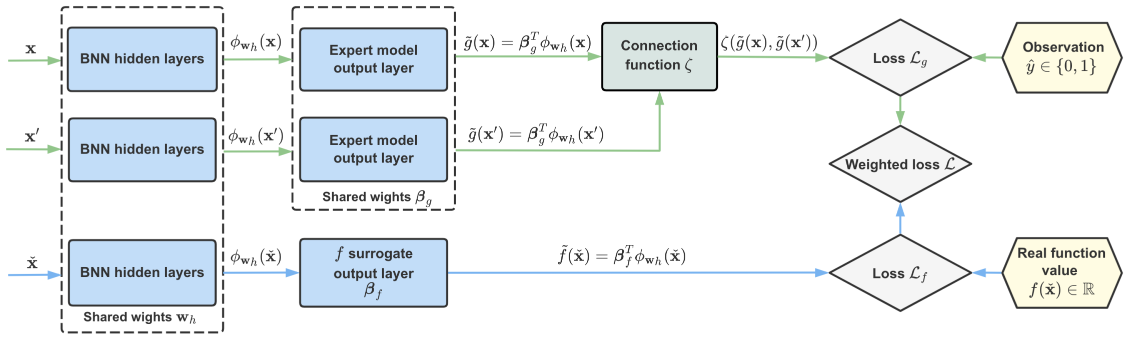
$\mathbf{x}$
$\mathbf{x}'$
$\check{\mathbf{x}}$
BNN hidden layers
BNN hidden layers
BNN hidden layers
Shared wights $\mathbf{w}_h$
$\phi_{\mathbf{w}_h}(\mathbf{x})$
$\phi_{\mathbf{w}_h}(\mathbf{x}')$
$\phi_{\mathbf{w}_h}(\check{\mathbf{x}})$
Expert model output layer
Expert model output layer
Shared wights β_g
f surrogate output layer β_f
$\tilde{g}(\mathbf{x}) = \beta_g^T \phi_{\mathbf{w}_h}(\mathbf{x})$
$\tilde{g}(\mathbf{x}') = \beta_g^T \phi_{\mathbf{w}_h}(\mathbf{x}')$
$\tilde{f}(\check{\mathbf{x}}) = \beta_f^T \phi_{\mathbf{w}_h}(\check{\mathbf{x}})$
Connection function ζ
$\zeta(\tilde{g}(\mathbf{x}), \tilde{g}(\mathbf{x}'))$
Loss $\mathcal{L}_g$
Weighted loss $\mathcal{L}$
Loss $\mathcal{L}_f$
Observation $\hat{y} \in \{0,1\}$
Real function value $f(\check{\mathbf{x}}) \in \mathbb{R}$

Figure 6: Multi-task learning (MTL) architecture, for the tasks of building jointly probabilistic surrogates for both the expert's beliefs g and the function to be optimized f. The green flow corresponds to surrogate of g, used in the knowledge elicitation part, and the blue flow to the surrogate of f, which is used for Bayesian optimization. The first layers (with parameters $\mathbf{w}_h$) are shared for the two surrogates, meaning that we aim at leveraging the similarity between the two functions by sharing some representations. They only differ in the output layer, parameterized by either $\boldsymbol{\beta}_g$ or $\boldsymbol{\beta}_f$. The two losses $\mathcal{L}_g$ and $\mathcal{L}_f$ are combined using a weighted scheme, which will give more and more importance to $\mathcal{L}_f$ as we get evaluations of f.

Algorithm 2 Expert knowledge-augmented BO

Input: Active learning budget M, BO acquisition budget J

Output: Minimum of f

```
 1: // Start Knowledge Elicitation
 2: Initialize the expert model $\tilde{g}$ using PBNN with a random pair, $\mathcal{D}_g = \{(\mathbf{x}_0, \mathbf{x}'_0, \hat{y}_0)\}$.
 3: for $i = 1$ to $M$ do
 4:     $[\mathbf{x}_i, \mathbf{x}'_i] = \underset{[\mathbf{x},\mathbf{x}'] \in \mathcal{D}_{pool}}{\arg\max}\ \mathrm{I}(\hat{y}; \mathbf{w}_g | [\mathbf{x}, \mathbf{x}'], \mathcal{D}_g)$ (Eq. (23))
 5:     Query the expert to obtain $\hat{y}_i$ associated to $[\mathbf{x}_i, \mathbf{x}'_i]$
 6:     $\mathcal{D}_g \leftarrow \mathcal{D}_g \cup (\mathbf{x}_i, \mathbf{x}'_i, \hat{y}_i)$
 7:     Update variational parameters $\boldsymbol{\theta}_g$ by minimizing Eq. (17)
 8: end for
 9: // Start Bayesian Optimization
10: $\mathcal{D}_f = \emptyset$
11: $y_{best} = \infty$
12: for $j = 1$ to $J$ do
13:     $\check{\mathbf{x}}_j = \underset{\mathbf{x} \in \mathcal{X}}{\arg\max}\ \alpha_{\mathrm{EI}}(\mathbf{x})$
14:     Evaluate $f(\check{\mathbf{x}}_j)$
15:     $\mathcal{D}_f \leftarrow \mathcal{D}_f \cup (\check{\mathbf{x}}_j, f(\check{\mathbf{x}}_j))$
16:     Update variational parameters $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_f$ by minimizing the combined loss
        Eq. (26)
17:     $y_{best} \leftarrow \min(y_{best}, f(\underset{\mathbf{x} \in \mathcal{X}}{\arg\min}\ \tilde{f}(\mathbf{x})))$
18: end for
19: return $y_{best}$
```

BNN as there is no analytical form of the output distribution, we use Monte Carlo sampling to obtain the approximate EI [Kim et al., 2021]:

$$\alpha_{\mathrm{EI}}(\mathbf{x}) \approx \frac{1}{T} \sum_{t=1}^{T} \max\left(y_{best} - \tilde{f}^{(t)}(\mathbf{x}), 0 \right), \tag{28}$$

where the $\tilde{f}^{(t)}(\mathbf{x})$ are i.i.d. predictive samples at $\mathbf{x}$.

Our two-step, expert knowledge-augmented BO procedure (knowledge elicitation first, and BO second) is summed up in Algorithm 2.

6 Experiments

6.1 Performance of the PBNN architecture

6.1.1 Toy example: Capturing the shape

We first present a toy example using 1-dimensional benchmark functions to illustrate how the proposed PBNN architecture can learn the shape of the function g through different number of pairwise comparisons. The model is trained by sequentially selecting 10, 20 and 50 pairwise comparisons using the PBALD criterion and getting the associated preference labels. We assume noiseless feedback in this experiment for illustration purposes. Figure 7 displays the comparison between the real function values g and elicited expert model predictions $\tilde{g}$, with the Forrester and Styblinski-Tang functions. From the figure, it can been seen that with 50 pairwise comparisons, the expert model $\tilde{g}$ can capture the ordinal information, i.e., g up to a monotonic constant, as previously explained.

6.1.2 Elicitation performance

We compare the performance of PBNN with the classical GP-based preference learning model by Chu and Ghahramani [2005] on three different datasets. We artificially transform three regression datasets into preference datasets by creating preference labels between all possible pairs of covariates using the target values.

For the GP-based model, we use the squared exponential kernel with the hyper-parameters optimized by maximum marginal likelihood. The PBNN architecture consists of two fully connected layers with width 100 and 10, where each hidden layer is followed by a `tanh` activation function. Optimization is carried out using ADAM [Kingma and Ba, 2014] with learning rate 0.001. We use a Gaussian variational posterior with $\mathcal{N}(0, 0.1)$ as priors for the weights.

The models are initialized using one random pair, and then a total of N_{AL} pairs are sequentially queried by maximizing the BALD criterion [Houlsby et al., 2011] for the GP-based model, and the PBALD criterion for PBNN, respectively. After the active learning phase, the accuracy of the model is assessed by computing a binary accuracy score on a hold-out test set consisting of 1000 pairs. The results on the three datasets, averaged over 20 replications, are presented in Table 2 for $N_{AL} = 50$ and $N_{AL} = 100$. In all scenarios, PBNN achieved better accuracy results w.r.t. the GP-based model.

Lastly, we propose to compare the runtimes of the two methods. The inference of PBNN is typically run GPU-based architectures, however, for a fair comparison, we compare 20 runs of each method on the three datasets using $N_{AL} = 50$ using the

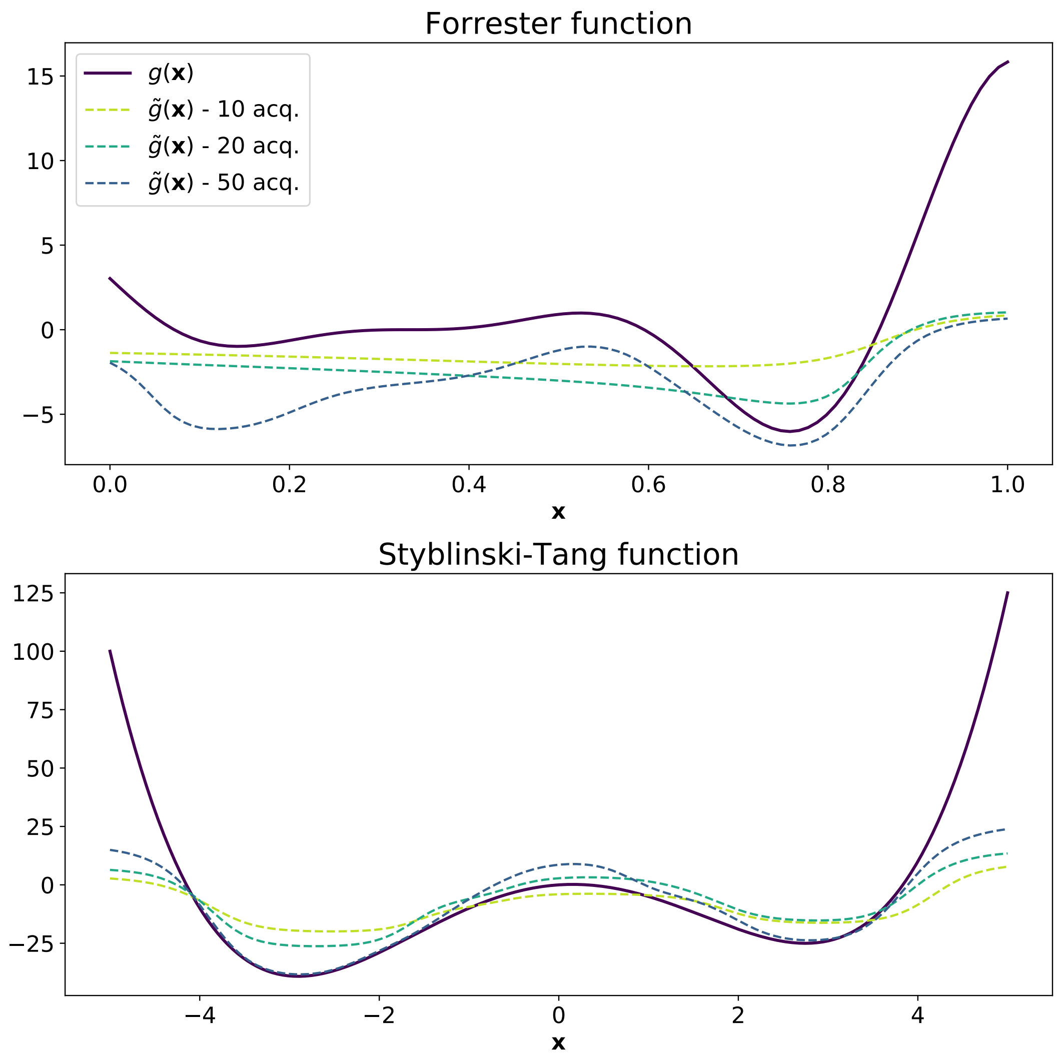

Figure 7: Toy example illustrating how the proposed PBNN architecture can learn the shape of a function using pairwise comparisons. Experiments carried out on the Forrester (top) and Styblinski-Tang (bottom) functions. The thickest, dark blue line represents the true function g, while the dotted lines represent the predicted functions $\tilde{g}$ learned by PBNN using 10, 20 and 50 pairwise comparisons.

same CPU architecture[1]. The results are reported in Table 3. On all three datasets, the proposed PBNN is roughly 20 times faster than the GP-based baseline.

[1]2x20 core Xeon Gold 6248 2.50GHz, 192GB RAM.

Table 2: Accuracy of preference prediction after N_{AL} acquisitions is significantly better than the earlier GP-based method on three datasets. The accuracy score corresponds to the proportion of correct binary preference prediction on a hold-out test set comprising 1000 pairs. The mean and standard deviation of this score are reported over 20 replications.

		Accuracy (%)	
DATASET	N_{AL}	GP	PBNN
Machine CPU (6D)	50	67.29 ± 2.91	$\mathbf{81.07 \pm 3.74}$
	100	69.45 ± 1.82	$\mathbf{84.38 \pm 1.52}$
Boston Housing (13D)	50	67.41 ± 2.35	$\mathbf{83.40 \pm 2.14}$
	100	69.33 ± 2.50	$\mathbf{85.89 \pm 2.52}$
Pyrimidine (27D)	50	70.99 ± 2.35	$\mathbf{79.63 \pm 2.14}$
	100	79.29 ± 2.50	$\mathbf{86.59 \pm 2.52}$

Table 3: Average runtimes (in seconds) after $N_{AL} = 50$ acquisitions on three datasets. The runtime of the proposed PBNN is roughly 20 times faster than the GP-based method. The comparison was carried out with 20 runs on the same CPU architecture. The standard deviation is also reported.

		Runtimes (sec.)	
DATASET	N_{AL}	GP	PBNN
Machine CPU (6D)	50	899 ± 68	$\mathbf{40 \pm 4}$
Boston Housing (13D)	50	981 ± 72	$\mathbf{50 \pm 7}$
Pyrimidine (27D)	50	1003 ± 64	$\mathbf{57 \pm 9}$

6.2 Comparison of the optimization performance on benchmark functions

6.2.1 Experiment on simulated experts

We first study the performance of the proposed knowledge-augmented Bayesian optimization scheme in a simulated setting where we can control the bias of an expert. More precisely, we compare how well the scheme performs w.r.t. to standard Bayesian optimization on several benchmark functions from the literature.

We assume an expert with potentially biased beliefs of the true function f, with the bias expressed as a perturbation function δ:

$$g(x) = f(x) + \delta(x), \tag{29}$$

where δ is a zero-mean Gaussian process draw with kernel $\sigma_\delta^2 k(x, x')$, which encodes the form of the expert's bias. Note that this does not reduce generality, assuming a

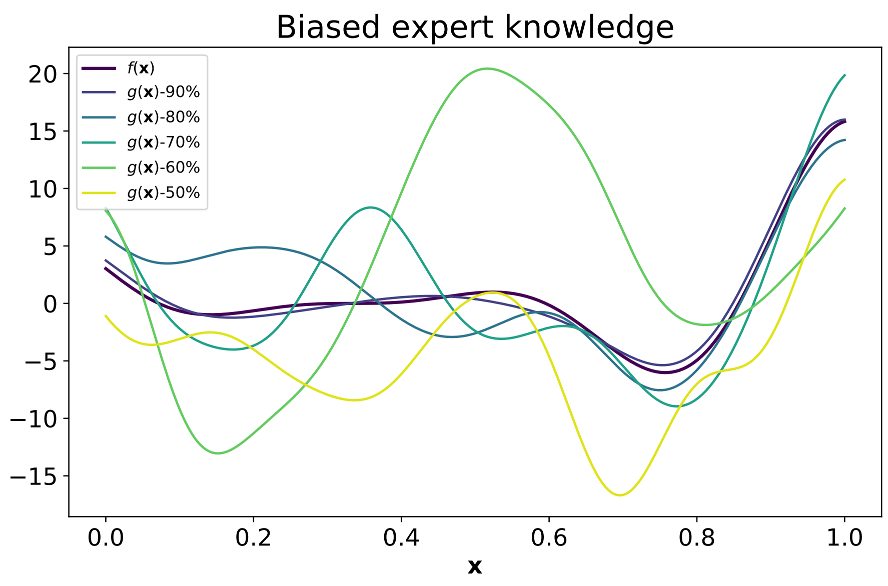

Figure 8: Illustration of the simulated expert's beliefs using the Forrester function. The true function is the thickest, dark blue curve, and the other curves correspond to that function perturbed with a random GP draw with various variances. The variances are chosen such that the accuracy of the expert ranges from 50% to 90 %.

general enough perturbation family. In the experiments reported below, we study the effect of expert's bias on the performance by choosing the SE kernel with lengthscale $\ell = 0.1$ and varying σ_δ^2 so that we obtain five levels of expert knowledge accuracy from 50% up to 90%. Figure 8 illustrates the comparison between the simulated experts with the ground truth objective function with the Forrester function, for the 5 considered accuracy levels.

For the MTL structure, the shared hidden layers have width [100, 30, 15]. Standard BO is run using the BNN surrogate described in Section 5.1, in other words, it is the exact same architecture as the "BO branch" of the MTL, for fair comparison. Experiments were run with $M = 100$, $J = 50$ and $\alpha = 0.95$. The results, detailed in the next paragraph, are averaged over 50 replications of the experiment. The full description of experimental settings is provided in Appendix B.

Figure 9 shows the results on four benchmark functions[2]: "Forrester1D", "Six-hump-camel2D", "Branin2D" and "Levy10D". The results are evaluated by y_{best}, which is the current minimal value of the true objective function predicted by the surrogate of f. We can see that the more accurate the simulated expert is, the more pronounced the acceleration effect. If the expert is reliable enough, we can speed up BO significantly. When the expert does not have any knowledge, i.e. 50%

[2]https://www.sfu.ca/ ssurjano/optimization.html

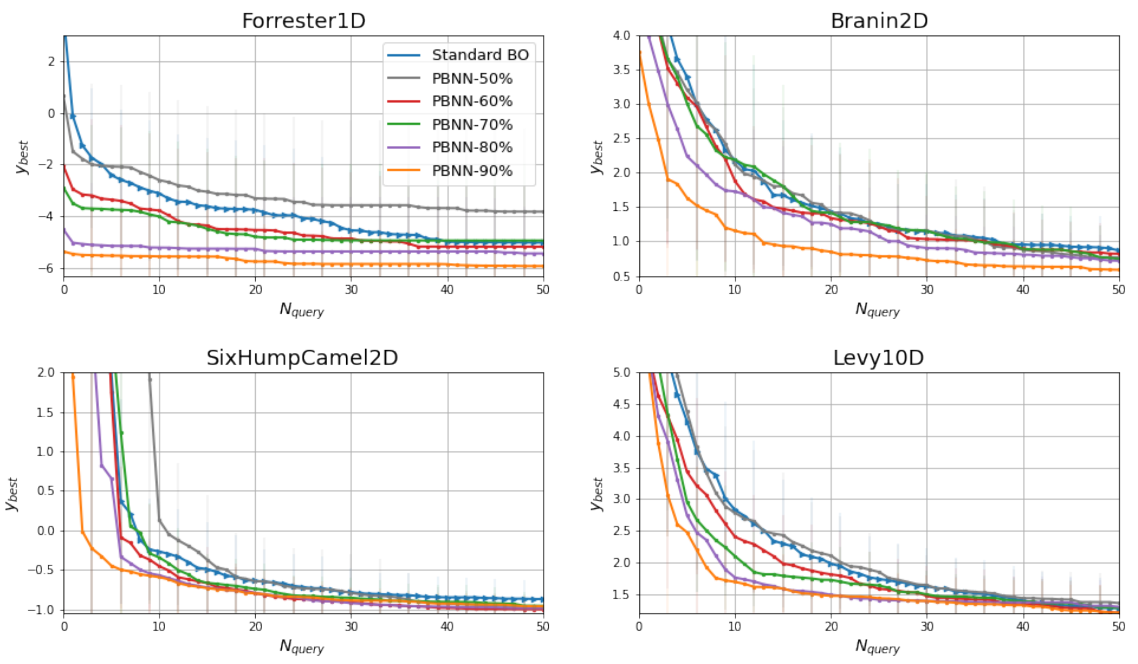

Figure 9: Comparison of the optimization performance of the expert knowledge-augmented BO using 4 benchmark functions w.r.t. standard BO. We simulate 5 experts with different levels of knowledge (denoted `PBNN-xx%`), where the percentage stands for the accuracy of the expert's preferential feedback, i.e., 50% means that the expert is simulated with fully biased knowledge. The knowledge of the simulated expert is elicited using $M = 100$ pairwise comparisons. The standard BO scheme, i.e., without expert knowledge, is denoted `Standard BO`. The results are averaged over 50 simulations.

preference accuracy, this actually leads to performance deterioration w.r.t. standard BO, which meets our expectation. For the all expert accuracy levels $\geq 60\%$, the final round BO performance ($N_{query} = 50$) is at least as competitive as the standard BO. However, the gain is much more striking in the early stages of the BO ($N_{query} \ll 50$). This phenomenon may be due to the challenge of making use of inaccurate expert knowledge as more accurate ground-truth data becomes increasingly available. We also provide additional experiment results with different parameter settings, please refer to Appendix C.

6.2.2 Experiment with actual human experts

We further study the performance of the proposed method in a real-world setting with actual human experts. We conduct a simple user experiment involving their memory abilities. The goal of the experiment is to optimize BO benchmark functions. To induce controlled knowledge about those functions, we let users memorize the shape of the objective function by displaying 2D-plots for a short time. That would provide useful but biased preference information for optimization. For the experiment setup, we restrict the objective functions to 2D benchmark functions as we cannot properly display 3D (or more) functions to users. Based on their memory of the function, the user must then answer a series of questions asked in the preferential form. Finally, BO augmented with expert knowledge is then run with the proposed methodology. We compare this approach with standard BO. The intuition behind this experiment is that users cannot memorize all the details, but still can grasp an understanding of the overall shape of the function, which can expedite the optimization to some extent.

For the details of the experiment, we choose three commonly used 2D benchmark functions: "Six-hump-camel2D", "Three-hump-camel2D" and "Branin2D". We choose these particular functions because they have several local minima, but are still smooth enough so that users can remember the general shape in a short time. We display the plots for 2 minutes after testing it ourselves; we believe it is enough to remember the general shape of the function, but not learn perfect information, thus mimicking expert knowledge on complex problems. We set the number of questions to 25, as we believe it is not too small to effectively build the expert model nor too large to bore the users. For each question, our process selects two points in the definition space, and the question is asked in the format "At which point do you think the value of the function is larger?". We provide the coordinates of the two points to the users and also plot their locations in the coordinate system used for the visualization of the function. Then the user has to make a decision about their preference towards the function values. The questions are determined by the active learning criterion, which is PBALD, detailed in Section 4.2. Other settings remain the same as the experiments of Section 6.2.1. For this experiment, we rely on the existing code of conduct from Aalto University regarding the conduction of user studies. The users are recruited from a student population that have no previous knowledge of the test functions. The experimental data is only used for this thesis, and we will not disclose any private information of the users. To make them quickly understand the content of the entire experiment, we also provide a brief instruction for the users (see Appendix A).

Table 4: Expert accuracy for different functions, data is obtained by dividing the number of questions the user answered correctly by 25. The final averaged accuracy meet our expectation as we believe the "Branin2D" function is the easiest one among the three functions and "Six-hump-camel2D" is the most difficult one.

	Accuracy								
DATASET	Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6	Expert 7	Expert 8	**Expert Average**
Three-hump-camel2D	64%	84%	72%	80%	52%	80%	72%	68%	**71.5%**
Six-hump-camel2D	52%	68%	76%	88%	64%	84%	72%	56%	**70%**
Branin2D	80%	72%	72%	80%	68%	84%	80%	76%	**76.5%**

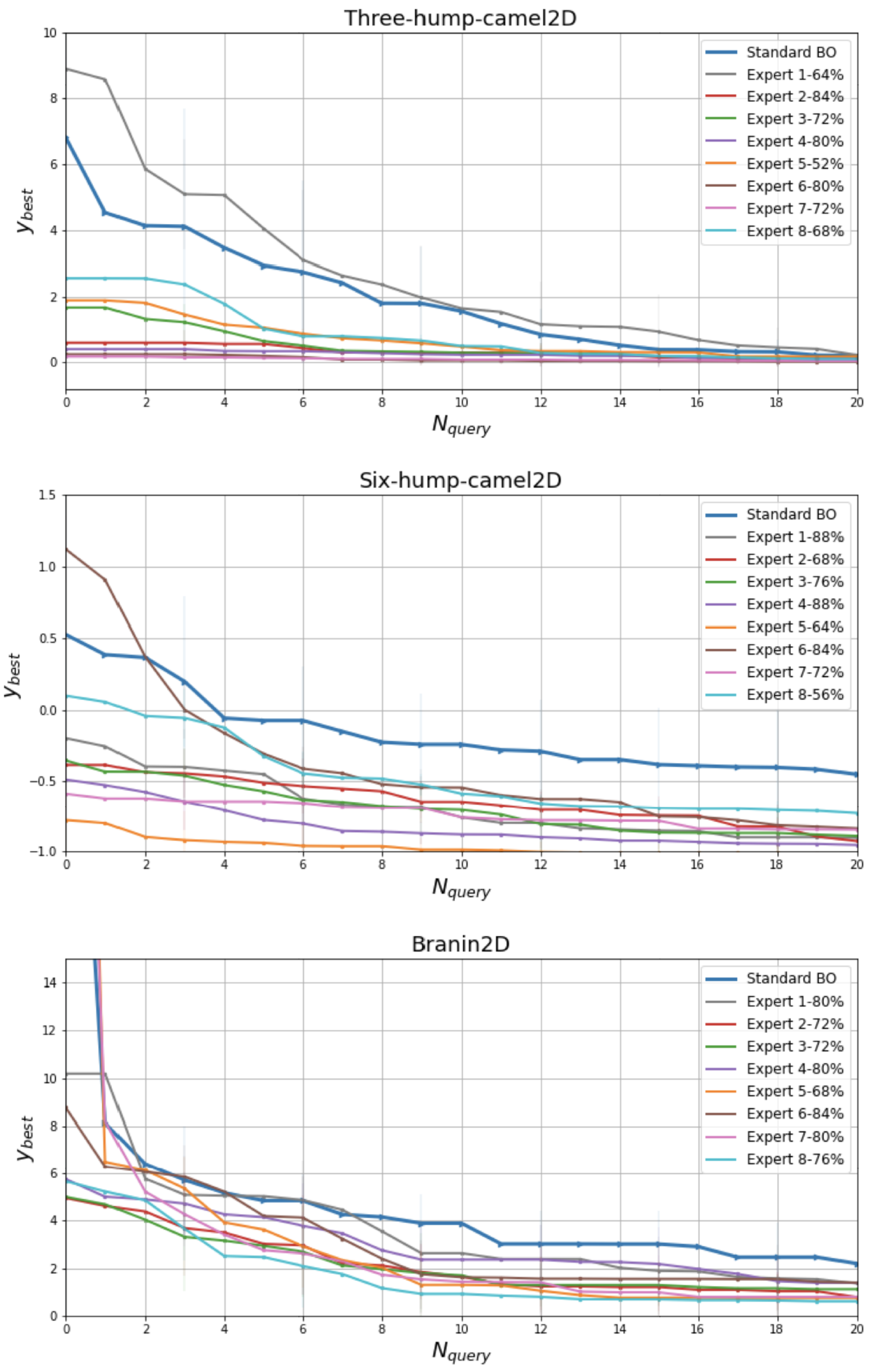

Figure 10: Comparison of the optimization performance of the expert knowledge-augmented BO with real users on 3 2D benchmark functions w.r.t. standard BO. We collect the data from 8 users, where the percentage stands for the accuracy of the expert's preferential feedback. The knowledge of the simulated expert is elicited using M = 25 pairwise comparisons. The results are averaged over 10 simulations.

Table 4 collects the experiment accuracy of 8 different human experts on three objective functions. All the accuracy rates are greater than 50%, and the average accuracy is higher than 70%, which is useful information. Figure 10 shows the comparison between standard BO and our method in terms of optimization performance. Each simulation builds the expert model using PBNN with the same dataset obtained from each user, but with different network initialization. We run 10 simulations to mitigate the randomness. With the help of experts, we can see prominent acceleration compared with standard BO, which again proves the effectiveness of our expert knowledge-augmented BO method. At last, we visualize the objective functions and selected expert models in Figure 11, which shows the expert model is able to capture the general shape information within only limited query times, hence speeding up the optimization by giving the right query direction for BO acquisition function.

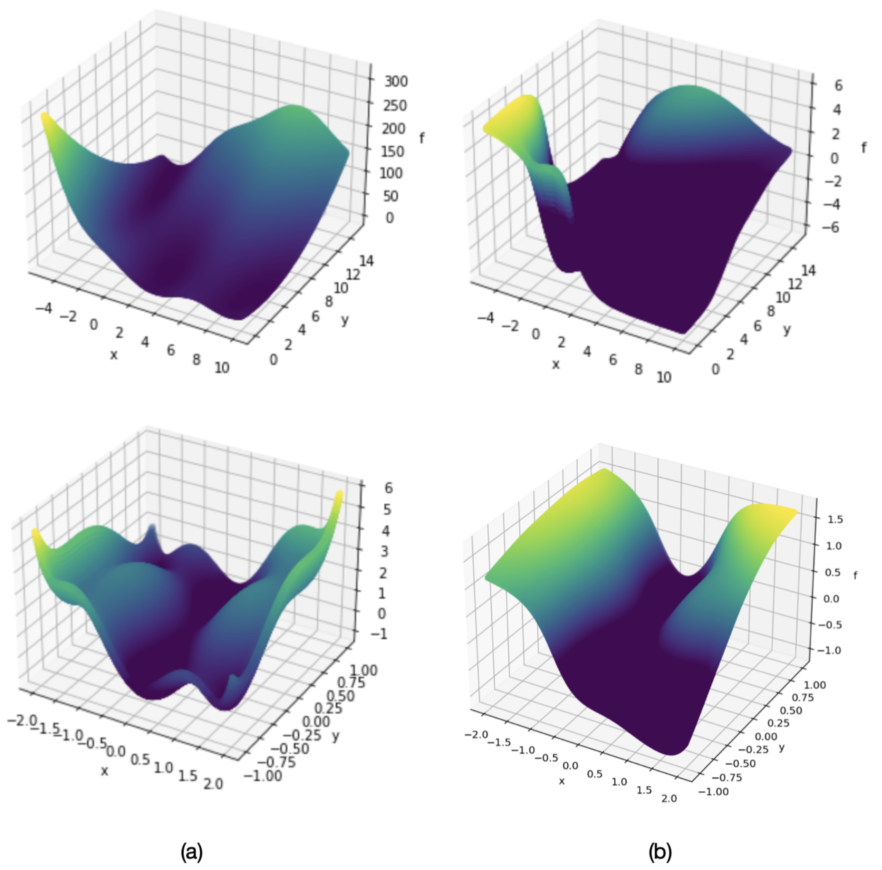

Figure 11: (a) 3D view of objective functions. (b) 3D view of simulated expert models with 25 queries, the user accuracy for the above plot is 80% and 84% for the bottom plot. We can see the expert models successfully capture the approximate shape of the objective functions.

7 Summary

In this thesis, we tackled the incorporation of human expert knowledge into BO with the goal of speeding up the optimization. Our procedure breaks down into two steps. The first is to elicit the expert beliefs by querying them with pairwise comparisons. By doing so, we obtain the approximate shape of the objective function. The second step is to share the expert knowledge with the BO, to provide auxiliary information about the potential location of the optimum.

More precisely, we proposed PBNN, a novel preference learning architecture based on Siamese networks to efficiently elicit the expert knowledge. By sequentially querying the preferences between two objects with active learning, the proposed PBNN is more powerful in capturing the latent preference relationships compared with the former GP-based model on different datasets. To conduct the knowledge transfer, we design a well-aligned multi-task learning structure with a knowledge sharing scheme to combine our expert model with BO surrogate. Experiments on different benchmark functions show that when the expert is trustworthy, we can gain significant benefit from the elicited knowledge and markedly speed up the optimization. For user study, we believe it is promising for real-world problems.

A limitation of this thesis is that we conducted the knowledge elicitation blindly w.r.t. the task of optimizing f. We will aim at proposing task-oriented active learning criterion in future work. Another exciting direction for future work is to directly consider the expert as another source of information, and therefore not resort to the two-step approach considered here. This would bring this work closer to the multi-fidelity BO line of research [Kandasamy et al., 2016, Takeno et al., 2020, Li et al., 2020a]. In that scenario, as humans are not passive sources of information, but rather active planners, we would need to build models able to anticipate behaviours such as steering [Colella et al., 2020].

References

Homayun Afrabandpey, Tomi Peltola, and Samuel Kaski. Interactive prior elicitation of feature similarities for small sample size prediction. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 265–269, 2017.

Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision (ECCV)*, pages 850–865, 2016.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

Eric Brochu, Nando D. Freitas, and Abhijeet Ghosh. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 409–416, 2008.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 1993.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, 1995.

Wei Chu and Zoubin Ghahramani. Preference learning with gaussian processes. In *International Conference on Machine Learning (ICML)*, pages 137–144, 2005.

Fabio Colella, Pedram Daee, Jussi Jokinen, Antti Oulasvirta, and Samuel Kaski. Human strategic steering improves performance of interactive optimization. In *ACM Conference on User Modeling, Adaptation and Personalization*, pages 293–297, 2020.

Pedram Daee, Tomi Peltola, Marta Soare, and Samuel Kaski. Knowledge elicitation via sequential probabilistic inference for high-dimensional prediction. *Machine Learning*, 106(9):1599–1620, 2017.

Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier, 1995.

Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158, 2016.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *International Conference on Machine Learning (ICML)*, pages 1183–1192, 2017.

Javier González, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. Preferential Bayesian Optimization. In *International Conference on Machine Learning (ICML)*, pages 1282–1291, 2017.

Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design. *arXiv preprint arXiv:1709.05501*, 2017.

Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 270–287, 2018.

Florian Hase, Loïc M Roch, Christoph Kreisbeck, and Alán Aspuru-Guzik. Phoenics: a Bayesian optimizer for chemistry. *ACS central science*, 4(9):1134–1145, 2018.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6), 2012.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. *Advances in neural information processing systems*, 27, 2014.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Carl Hvarfner, Danny Stoll, Artur Souza, Luigi Nardi, Marius Lindauer, and Frank Hutter. πBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization. In *International Conference on Learning Representations (ICLR)*, 2022.

Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2017.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998.

Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*, 2020.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

Samuel Kim, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Scalable and flexible deep bayesian optimization with auxiliary information for scientific problems. *arXiv preprint arXiv:2104.11667*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.

Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. 1964.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Antonio Robles-Kelly, and Svetha Venkatesh. Incorporating expert prior knowledge into experimental design via posterior sampling. *arXiv preprint arXiv:2002.11256*, 2020a.

Shibo Li, Wei Xing, Robert Kirby, and Shandian Zhe. Multi-fidelity Bayesian optimization via deep neural networks. 2020b.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

Shengchao Liu, Yingyu Liang, and Anthony Gitter. Loss-balanced task weighting to reduce negative transfer in multi-task learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9977–9978, 2019a.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. 2015.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019b.

David J. C. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604, 1992.

Luana Micallef, Iiris Sundin, Pekka Marttinen, Muhammad Ammad-Ud-Din, Tomi Peltola, Marta Soare, Giulio Jacucci, and Samuel Kaski. Interactive elicitation of knowledge on feature relevance improves predictions in small data sets. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 547–552, 2017.

Petrus Mikkola, Milica Todorović, Jari Järvi, Patrick Rinke, and Samuel Kaski. Projective preferential bayesian optimization. In *International Conference on Machine Learning*, pages 6884–6892, 2020.

Petrus Mikkola, Osvaldo A Martin, Suyog Chandramouli, Marcelo Hartmann, Oriol Abril Pla, Owen Thomas, Henri Pesonen, Jukka Corander, Aki Vehtari, Samuel Kaski, et al. Prior knowledge elicitation: The past, present, and future. *arXiv preprint arXiv:2112.01380*, 2021.

Ido Millet. The effectiveness of alternative preference elicitation methods in the analytic hierarchy process. *Journal of Multi-Criteria Decision Analysis*, 6(1): 41–51, 1997.

Duy-Kien Nguyen and Takayuki Okatani. Multi-task learning of hierarchical vision-language representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10492–10501, 2019.

Anthony O'Hagan, Caitlin E Buck, Alireza Daneshkhah, J Richard Eiser, Paul H Garthwaite, David J Jenkinson, Jeremy E Oakley, and Tim Rakow. Uncertain judgements: eliciting experts' probabilities. 2006.

Daniel Packwood. *Bayesian optimization for materials science.* Springer, 2017.

Subhojeet Pramanik, Priyanka Agrawal, and Aman Hussain. Omninet: A unified architecture for multi-modal multi-task learning. *arXiv preprint arXiv:1907.07804*, 2019.

Trout Rader. The existence of a utility function to represent preferences. *The Review of Economic Studies*, 30(3):229–232, 1963.

Anil Ramachandran, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Incorporating expert prior in bayesian optimisation via space warping. *Knowledge-Based Systems*, 195:105663, 2020.

Burr Settles. *Active Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning Series. Morgan & Claypool, 2012.

Nihar B Shah, Sivaraman Balakrishnan, Joseph Bradley, Abhay Parekh, Kannan Ramchandran, and Martin Wainwright. When is it better to compare than to score? *arXiv preprint arXiv:1406.6618*, 2014.

Zhongkai Shangguan, Lei Lin, Wencheng Wu, and Beilei Xu. Neural process for black-box model optimization under bayesian framework. *arXiv preprint arXiv:2104.02487*, 2021.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing systems (NIPS)*, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning (ICML)*, pages 2171–2180, 2015.

Marta Soare, Muhammad Ammad-Ud-Din, and Samuel Kaski. Regression with n$\to$ 1 by expert knowledge elicitation. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 734–739. IEEE, 2016.

Artur Souza, Luigi Nardi, Leonardo B Oliveira, Kunle Olukotun, Marius Lindauer, and Frank Hutter. Bayesian optimization with a prior for the optimum. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, pages 265–296, 2021.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Iiris Sundin, Tomi Peltola, Luana Micallef, Homayun Afrabandpey, Marta Soare, Muntasir Mamun Majumder, Pedram Daee, Chen He, Baris Serim, Aki Havulinna, et al. Improving genomics-based predictions for precision medicine through active elicitation of expert knowledge. *Bioinformatics*, 34(13):i395–i403, 2018.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search and its parallelization. In *International Conference on Machine Learning*, 2020.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635. PMLR, 2017.

Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Yichi Zhang, Daniel W Apley, and Wei Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1): 1–13, 2020.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer, 2014.

A User Manual

A.1 Introduction

Welcome to the test. During this experiment, you need to try your best to remember the shapes of three different 2-D functions in limited time. After that you need to answer 25 simple questions, by telling which point do you think is larger between a pair of points.

In this test, we rely on the existing code of conduct of Aalto University for conducting user studies in our field. The experimental data is only used for this thesis, and we will not disclose any of your private information.

A.2 Experimental details

Three experiments will be conducted in random order. When each experiment starts, you will be shown a 3-D plot and a 2-D heat map of the function (demo plots are shown in Figure A1), and you can drag the 3-D plot to have a better visualization. You will have 2 minutes to remember the plots, once the time is up, you will no longer be able to view these plots.

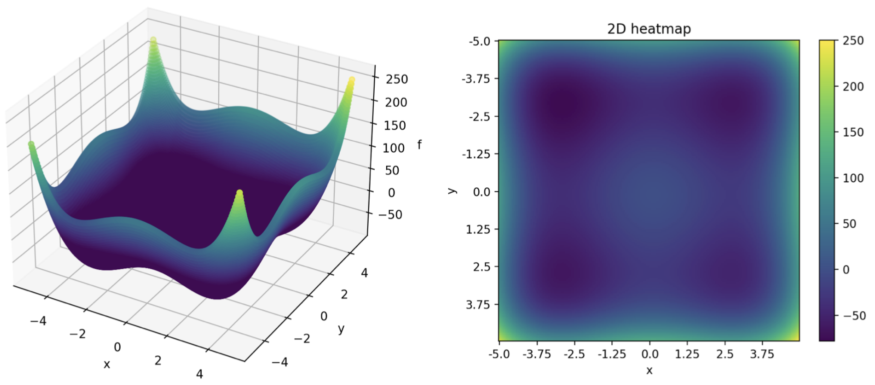

Figure A1: The left plot is the 3-D view of the objective function, and the right one is the 2-D heat map. These plots are only for demonstration, the real objective functions in the experiment will not be shown here.

After that, you will be asked 25 questions. Each question is asked in the format "At which point do you think the value of the function is larger?" And there will be no time limit for you to answer these questions. We will provide the coordinates of the two points to you and also plot their locations in the coordinate system used in the visualization of the function. The demo plot is shown in Figure A2.

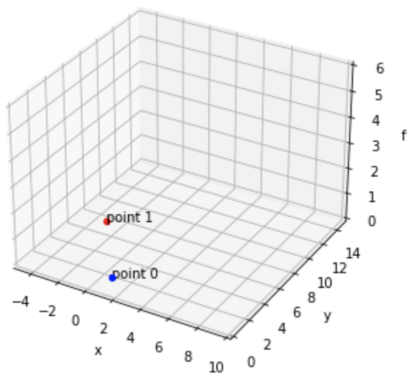

Figure A2: The plot of two points in the question stage.

After answering all the questions, you will directly jump to the next experiment. Upon finish the three experiments, the system will calculate the accuracy of your performance, and you can also view your own user model in 3D plot (see Figure A3 for reference).

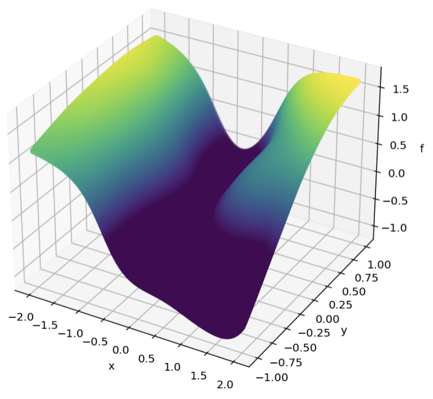

Figure A3: An example of the user model

B Experimental settings

B.1 Elicitation experiment

Datasets

- Machine CPU: A computer hardware dataset. The dimension is 6 and has 209 instances
- Boston housing: This dataset contains information concerning housing in the area of Boston Mass. The dimension is 13 and has 506 cases
- Pyrimidine: A pyrimidine QSAR dataset. The dimension of this dataset is 27 and has 74 instances

The initial training set contains one random pair. The query pool size for active learning is 2000 pairs, and the test set used for evaluating accuracy consists of 1000 pairs. The dataset is shuffled in each epoch.

Hyper-parameters

- Number of active acquisitions in elicitation stage: 50, 100
- Monte Carlo sampling budget in BALD: 100
- Number of simulations: 20

Neural network configurations

- Framework: PyTorch, torchbnn
- Optimizer: ADAM with learning rate = 0.001
- Scheduler: CosineAnnealingLR with $T_{max} = 20$ and $eta_{min} = 0.0001$
- Batch size: 2
- Number of epochs: 20
- Bayesian linear layer: 2 layers with weight prior $\mathcal{N}(0, 0.1)$, width [100, 10]
- Activation function: Tanh

B.2 BO with simulated experts

Benchmark functions

- Forrester1D: A simple one-dimensional test function, with one global minimum, one local minimum and a zero-gradient inflection point. This function is evaluated on $x \in [0, 1]$. The form of this function is:

$$f(x) = (6x - 2)^2 \sin(12x - 4). \tag{B1}$$

- Branin2D: A 2D function with three global minima. We take $a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, r = 6, s = 10$ and $t = \frac{1}{8\pi}$. This function is evaluated on the square $x_1 \in [-5, 10], x_2 \in [0, 15]$. The function form is:

$$f(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)\cos(x_1) + s. \tag{B2}$$

- Six-hump-camel2D: A 2D function with six local minima, two of which are global. This function is evaluated on the square $x_1 \in [-3, 3], x_2 \in [-2, 2]$. The function form is:

$$f(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2. \tag{B3}$$

- Levy10D: A 10D function evaluated on the hypercube $x_i \in [-2, 2]$, for all $i = 1, ..., d$. The function form is:

$$\begin{aligned} f(x) = {} & \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2[1 + 10\sin^2(\pi w_i + 1)] \\ & + (w_d - 1)^2[1 + \sin^2(2\pi w_d)], \end{aligned} \tag{B4}$$

where $w_i = 1 + \frac{x_i - 1}{4}$, for all $i = 1, ..., d$.

The number if initial training pairs for elicitation is 1. The query pool size for active learning is 2000.

Hyper-parameters

- Number of active acquisitions in elicitation stage: 100
- Number of BO acquisition: 50
- Monte Carlo sampling budget in BALD: 100
- Monte Carlo sampling budget in EI: 30
- α in MTL shared weight: 0.95
- Number of simulations: 50

Neural network configurations

- Framework: PyTorch, torchbnn
- Optimizer: ADAM with lr = 0.001 in elicitation stage, lr = 0.01 in BO stage
- Scheduler: CosineAnnealingLR with $T_{max} = 20$ and $eta_{min} = 0.0001$ in elicitation stage, no scheduler in BO.
- Batch size: 10 for preference data, 5 for regression data
- Number of epochs: 100 in elicitation stage, 200 in BO stage
- Bayesian linear layer: 3 shared hidden layers with weight prior $\mathcal{N}(0, 0.1)$, width [100, 30, 15]
- Activation function: Tanh

B.3 BO with actual human experts

Benchmark functions

- Three-hump-camel2D: This function has three local minima and is evaluated on the square $x_1 \in [-2, 2], x_2 \in [-2, 2]$. The form of this function is:

$$f(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2. \tag{B5}$$

- Six-hump-camel2D: A 2D function with six local minima, two of which are global. This function is evaluated on the square $x_1 \in [-2, 2], x_2 \in [-1, 1]$. The function form is:

$$f(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2. \tag{B6}$$

- Branin2D: A 2D function with three global minima. We take $a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, r = 6, s = 10$ and $t = \frac{1}{8\pi}$. This function is evaluated on the square $x_1 \in [-5, 10], x_2 \in [0, 15]$. The function form is:

$$f(x) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t)\cos(x_1) + s. \tag{B7}$$

The number if initial training pairs for elicitation is 1. The query pool size for active learning is 2000.

Hyper-parameters

- Number of active acquisitions in elicitation stage: 25
- Number of BO acquisition: 20
- Monte Carlo sampling budget in BALD: 100
- Monte Carlo sampling budget in EI: 30
- α in MTL shared weight: 0.95
- Number of simulations: 10

Neural network configurations

- Framework: PyTorch, torchbnn
- Optimizer: ADAM with lr = 0.001 in elicitation stage, lr = 0.01 in BO stage
- Scheduler: CosineAnnealingLR with $T_{max} = 20$ and $eta_{min} = 0.0001$ in elicitation stage, no scheduler in BO.
- Batch size: 10 for preference data, 5 for regression data
- Number of epochs: 100 in elicitation stage, 200 in BO stage
- Bayesian linear layer: 3 shared hidden layers with weight prior $\mathcal{N}(0, 0.1)$, width [100, 30, 15]
- Activation function: Tanh

C Additional experiments

We further investigate the performance of our expert-augmented BO with different elicitation budgets. We use the same objective functions as in Section 6.2.1 and simulate four different levels of the experts. We use the same configurations as in the previous experiments, the details can be found in Section B.2.

The results are shown in Figures C1 C2, C3, C4. The overall performance behaves as expected. With a larger elicitation budget, the acceleration of BO is more obvious. We notice that the performance is even worse under a very limited budget than standard BO, i.e., $N_{AL} = 10$. We guess the reason behind this situation is that the insufficient preference training data makes the network prone to overfitting, hence misguiding the training of the surrogate model during the MTL stage. Moreover, in some figures, we can see the performance between $N_{AL} = 50$ and $N_{AL} = 100$ is very close, which implies that there is no need to over-query for the expert under some relatively easy-to-optimize functions since the expert knowledge will then dominate the actual BO regression data and slow down the optimization. In this case, we should consider lowering the value of α (Equation 26).

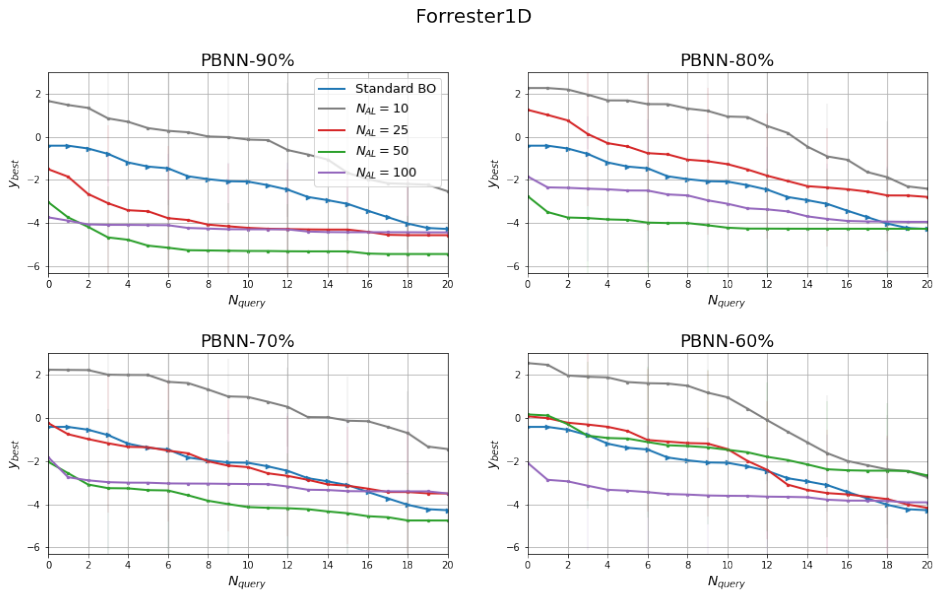

Figure C1: A BO comparison on "Forrester1D" function with different knowledge elicitation budget. We simulate 4 experts with different levels of knowledge, in each subplot we use the same level of the expert. The results are averaged over 20 simulations.

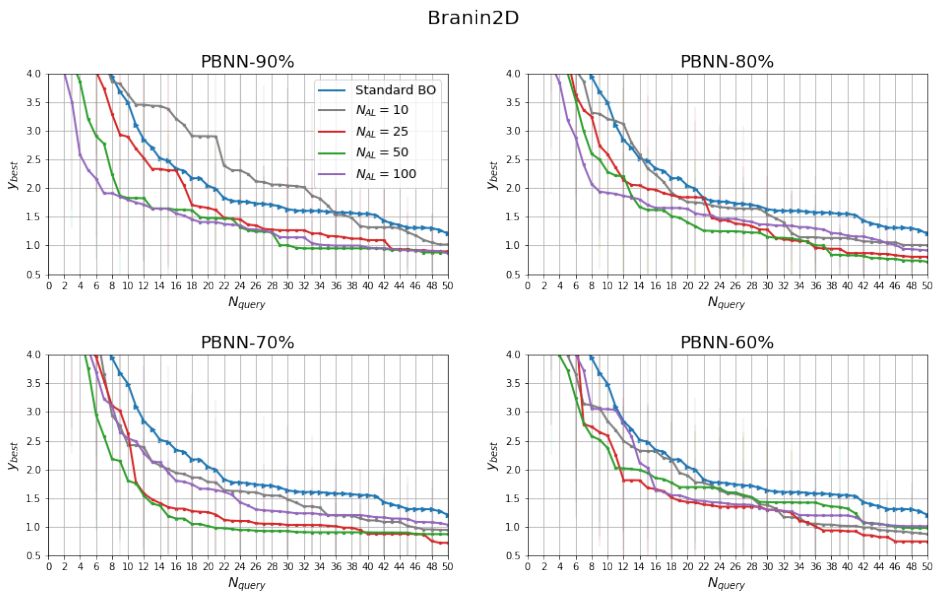

Figure C2: A BO comparison on "Branin2D" function with different knowledge elicitation budget. We simulate 4 experts with different levels of knowledge, in each subplot we use the same level of the expert. The results are averaged over 20 simulations.

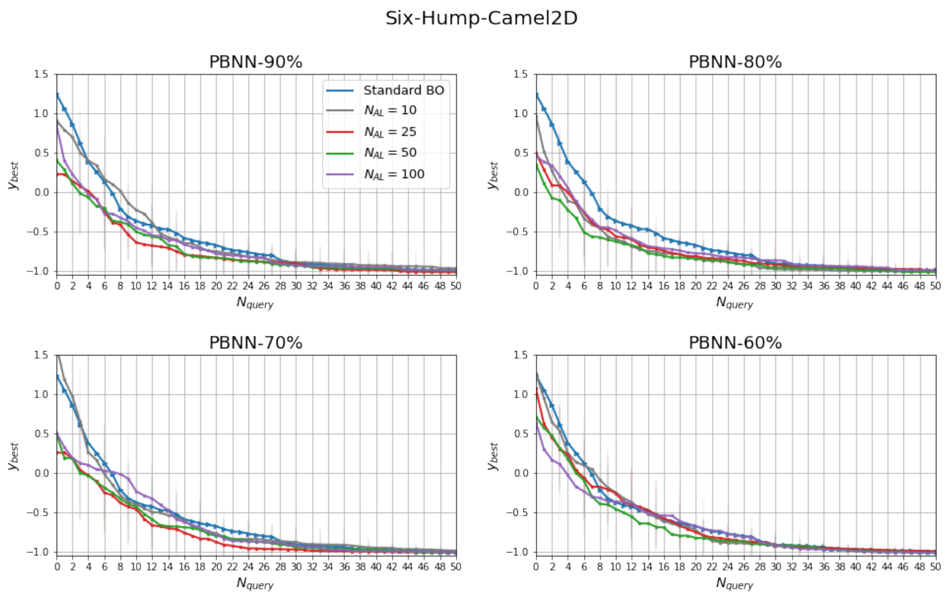

Figure C3: A BO comparison on "Six-Hump-Camel2D" function with different knowledge elicitation budget. We simulate 4 experts with different levels of knowledge, in each subplot we use the same level of the expert. The results are averaged over 20 simulations.

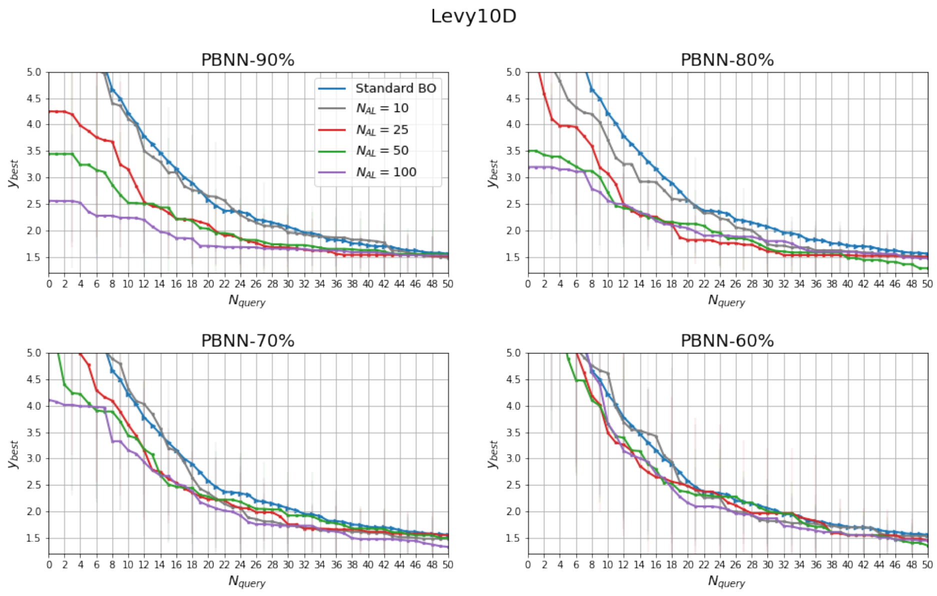

Figure C4: A BO comparison on "Levy10D" function with different knowledge elicitation budget. We simulate 4 experts with different levels of knowledge, in each subplot we use the same level of the expert. The results are averaged over 20 simulations.