HEBO: Heteroscedastic Evolutionary Bayesian Optimisation

Alexander I. Cowen-Rivers*
Huawei Noah's Ark Lab
alexander.cowen.rivers@huawei.com

Wenlong Lyu*
Huawei Noah's Ark Lab
lvwenlong2@huawei.com

Zhi Wang
Huawei Noah's Ark Lab
wangzhi55@huawei.com

Rasul Tutunov
Huawei Noah's Ark Lab
rasul.tutunov@huawei.com

Hao Jianye
Huawei Noah's Ark Lab
haojianye@huawei.com

Jun Wang
Huawei Noah's Ark Lab
University College London
w.j@huawei.com

Haitham Bou Ammar †
Huawei Noah's Ark Lab
University College London
haitham.ammar@huawei.com

Abstract

We introduce HEBO: Heteroscedastic Evolutionary Bayesian Optimisation that won the NeurIPS 2020 black-box optimisation competition. We present non-conventional modifications to the surrogate model and acquisition maximisation process and show such a combination superior against all baselines provided by the `Bayesmark` package. Lastly, we perform an ablation study to highlight the components that contributed to the success of HEBO.

1 Introduction

The NeurIPS 2020 black-box optimisation challenge is an annual competition that evaluates black-box optimisation algorithms on real-world score functions. The contest is constructed from automated machine learning (AutoML) tasks [1] and the Bayesmark package, both of which aim at tuning hyper-parameters of models to improve validation set performance. Given the importance of correctly tuning machine learning algorithms [2], this challenge constitutes a major stepping-stone towards real-world deployment of large-scale models.

In general, the contest can be formulated as a problem of optimising a performance measure (score) for various hyper-parameter configurations of ML algorithms. For instance, these parameters can correspond to learning rates, layer depths and widths, or dropout rates during a neural network training step. The 'score' that we would optimise in this instance would be the validation loss. Among various challenges, the main difficulties in optimising hyper-parameters are; that performance measures are neither explicitly available in closed form, nor do they necessarily adhere to differentiability assumptions, they can typically be very computationally expensive to evaluate, and there is no precise method to perform mixed variable optimisation. Nonetheless, one can still exploit a low number of validation loss evaluations for a better exploration of the hyper-parameter space. This, in turn, transforms the problem into an instance of black-box optimisation:

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}), \tag{1}$$

*Equal contribution.

†Honorary position at University College London.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

with $\boldsymbol{x}$ denoting optimisation variables that correspond to mixed (continuous and discrete) hyper-parameter settings and $f(\boldsymbol{x})$ the score function we wish to minimise.

Bayesian Optimisation (BO) presents a sample-efficient and derivative-free solution for finding a minimum $\boldsymbol{x}^\star$ of black-box objectives such as these encountered in Equation 1. Real-world examples of BBO problems are ubiquitous [3, 4, 5, 6, 7, 8, 9, 10, 11]. BO algorithms are typically equipped with two core components that allow for improved exploration of input points. The first corresponds to a probabilistic surrogate model (generally a Gaussian process) that approximates $f(\boldsymbol{x})$ from historical interactions whilst providing uncertainty estimates needed to guide exploration. The second component equates to an acquisition function that acts as a proxy to the true sequential risk, measuring the utility of gathering new input points by trading off exploration and exploitation.

Lots of BO implementations assume an idealised setting for calculating future black-box evaluations, such as Gaussian noise likelihoods and one optimal acquisition function. While analysing the competition's data, we realised that neither of these two assumptions hold; noise processes are complex and heteroscedastic and different acquisition functions arrived at conflicting results. Furthermore, in case unseen data involved even more compounded stochasticity, as evaluation tasks are hidden to participants, we would prefer to hedge against our losses by equipping the surrogate model with the ability to handle non-stationarities.

Hence, to arrive at a winning solution, we developed a heteroscedastic and evolutionary Bayesian optimisation (HEBO) algorithm that supports: 1) complex noise processes through input warping and output transformations, and 2) multi-objective acquisition functions with evolutionary optimisers. In particular, while input and output non-linear transformations handle non-stationarity and heteroscedasticity respectively, multi-objectivity avoids conflicts by enabling a consensus among various acquisition functions through a Pareto-frontier.

2 Background

We consider a sequential decision approach to the global optimisation of a smooth function $f : \mathcal{X} \rightarrow \mathbb{R}$ over a bounded input domain $\mathcal{X} \subseteq \mathbb{R}^d$. At each decision round, i, we select a collection of q input points $\boldsymbol{x}_{1:q}^{(\text{new})} \in \mathcal{X}^q$ and observe values of the *black-box* function $\boldsymbol{y}_{1:q}^{(\text{new})} = f(\boldsymbol{x}_{1:q}^{(\text{new})})$. We allow the returned value to be either deterministic i.e., $y_i = f(\boldsymbol{x}_i)$ or stochastic with $y_i = f(\boldsymbol{x}_i) + \epsilon_i$, where ϵ_i denotes a bounded-variance random variable. Our goal is to rapidly (in terms of N) approach the maximum $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x})$. Since both $f(\cdot)$ and $\boldsymbol{x}^\star$ are unknown, solvers need to trade off exploitation and exploration during the search process, as shown in Algorithm 1.

To reason about the unknown function, typical Bayesian optimisation algorithms assume smoothness and adopt Bayesian modelling as a principle to carry out inference about the properties of $f(\cdot)$ in light of the observations. Here, one introduces a surrogate model (line 3 of Algorithm 1) to encode beliefs over the smoothness properties and an observation model to describe collected data, $\mathcal{D}_i = \{\boldsymbol{x}_l, y_l\}_{l=1}^{n_i}$, up to the i^{th} round with n_i denoting the total acquired data so far. Using these two components in addition to Bayes rule, we can then compute a posterior $p(f(\cdot)|\mathcal{D}_i)$ to encode all knowledge of $f(\cdot)$ allowing us to account for the location of the maximum. A generic Bayesian optimisation solver finds a batch of q input points, $\boldsymbol{x}_{1:q}^{(\text{new})}$, by maximising an acquisition function that generally trades off exploration versus exploitation as shown in line 4 of the algorithm.

At this stage, $\boldsymbol{x}_{1:q}^{(\text{new})}$ are evaluated by querying the black-box to generate a set of q response variables $\boldsymbol{y}_{1:q}^{(\text{new})} = f(\boldsymbol{x}_{1:q}^{(\text{new})})$ that are then used to augment the dataset; see line 6 of Algorithm 1. The above process repeats for a total of N iterations at which the best performing input in $\mathcal{D}_N$ is returned as the solution to the problem in Equation 1.

2.1 Gaussian Processes as a Surrogate Model

A Gaussian process (GP) offers a flexible and sample-efficient procedure for placing priors over unknown functions [12]. These models are fully specified by a mean function $m(\boldsymbol{x})$ and a covariance function, or kernel, $k(\boldsymbol{x}, \boldsymbol{x}')$ that encodes the smoothness assumptions on $f(\cdot)$. Given any finite collection of inputs $\boldsymbol{x}_{1:n_i}$, the outputs are jointly Gaussian given by:

$$f(\boldsymbol{x}_{1:n_i})|\boldsymbol{\theta} \sim \mathcal{N}\left(m(\boldsymbol{x}_{1:n_i}), \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n_i}, \boldsymbol{x}_{1:n_i})\right),$$

Algorithm 1 Batched Bayesian Optimisation

1: **Inputs:** Total number of outer iterations N, initial randomly-initialised dataset $\mathcal{D}_0 = \{\boldsymbol{x}_l, y_l \equiv f(\boldsymbol{x}_l)\}_{l=1}^{n_0}$, batch size q, acquisition function
2: **for** $i = 0 : N-1$:
3: Fit a surrogate model to the current dataset $\mathcal{D}_i$
4: Find q points $\boldsymbol{x}_{1:q}^{(\text{new})}$ by maximising an acquisition function, trading off exploration and exploitation
5: Evaluate new inputs by querying the black-box to acquire $\boldsymbol{y}_{1:q}^{(\text{new})} = f(\boldsymbol{x}_{1:q}^{(\text{new})})$
6: Update the dataset creating $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{\boldsymbol{x}_l^{(\text{new})}, y_l^{(\text{new})}\}_{l=1}^{q}$
7: **end for**
8: **Output:** Return the best-performing query point from the data $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}\in\mathcal{D}_N} f(\boldsymbol{x})$

where $[m(\boldsymbol{x}_{1:n_i})]_k = m(\boldsymbol{x}_k)$ denotes the mean vector, and $\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n_i}, \boldsymbol{x}_{1:i}) \in \mathbb{R}^{n_i \times n_i}$ the covariance matrix with its $(k,l)^{th}$ entry computed as $[\boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n_i}, \boldsymbol{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}(\boldsymbol{x}_k, \boldsymbol{x}_l)$. Here, $k_{\boldsymbol{\theta}}(\cdot,\cdot)$ represents a parameterised kernel with unknown hyperparameters $\boldsymbol{\theta}$ corresponding to lengthscales or signal amplitudes for example. For ease of presentation following [12], we use a zero-mean prior in our notation here. In terms of the choice of Gaussian process kernel, there are a wide array of options which encode prior modelling assumptions about the latent function. Two of the most commonly-encountered kernels in the Bayesian optimisation literature are the squared exponential (SE) and Matérn(32) kernels.

Given the data $\mathcal{D}_i$, and assuming Gaussian-corrupted observations $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$, we can write the joint distribution over the data and an arbitrary evaluation input $\boldsymbol{x}$ as:

$$\left[\begin{array}{c} \boldsymbol{y}_{1:n_i} \\ f(\boldsymbol{x}) \end{array}\right] \Bigg| \boldsymbol{\theta} \sim \mathcal{N}\left(\left[\begin{array}{c} m(\boldsymbol{x}_{1:n_i}) \\ m(\boldsymbol{x}) \end{array}\right], \left[\begin{array}{cc} \boldsymbol{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2_{\text{noise}}\boldsymbol{I} & \boldsymbol{k}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{x}) \\ \boldsymbol{k}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\boldsymbol{x}) & k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}) \end{array}\right]\right),$$

where $\boldsymbol{K}_{\boldsymbol{\theta}}^{(i)} = \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n_i}, \boldsymbol{x}_{1:n_i})$ and $\boldsymbol{k}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{x}) = \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:n_i}, \boldsymbol{x})$. In the setting of q arbitrary evaluation points, we get $\boldsymbol{f}(\boldsymbol{x}^\star_{1:q})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\boldsymbol{x}^\star_{1:q};\boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\boldsymbol{x}^\star_{1:q};\boldsymbol{\theta}))$ with:

$$\boldsymbol{\mu}_i(\boldsymbol{x}^\star_{1:q};\boldsymbol{\theta}) = \boldsymbol{K}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{x}^\star_{1:q}, \boldsymbol{x}_{1:n_i})(\boldsymbol{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2_{\text{noise}}\boldsymbol{I})^{-1}(\boldsymbol{y}_{1:n_i} - m(\boldsymbol{x}_{1:n_i})) + m(\boldsymbol{x}^\star_{1:q})$$
$$\boldsymbol{\Sigma}_i(\boldsymbol{x}^\star_{1:q};\boldsymbol{\theta}) = \boldsymbol{K}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{x}^\star_{1:q}, \boldsymbol{x}^\star_{1:q}) - \boldsymbol{K}_{\boldsymbol{\theta}}^{(i)}(\boldsymbol{x}^\star_{1:q}, \boldsymbol{x}_{1:n_i})(\boldsymbol{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2_{\text{noise}}\boldsymbol{I})^{-1}\boldsymbol{K}_{\boldsymbol{\theta}}^{\mathsf{T},(i)}(\boldsymbol{x}^\star_{1:q}, \boldsymbol{x}_{1:n_i}).$$

The remaining ingredient needed in a GP pipeline is a process to determine the unknown hyperparameters $\boldsymbol{\theta}$ given a set of observation $\mathcal{D}_i$. In standard GPs [12], $\boldsymbol{\theta}$ are fit by minimising the negative log marginal likelihood (NLML) leading us to the following optimisation problem:

$$\min_{\boldsymbol{\theta},\sigma_{\text{noise}}} \mathcal{J}(\boldsymbol{\theta}, \sigma_{\text{noise}}) = \frac{1}{2}\det\left(\boldsymbol{C}_{\boldsymbol{\theta}}^{(i)}\right) + \frac{1}{2}(\boldsymbol{y}_{1:n_i} - m(\boldsymbol{x}_{1:n_i}))^{\mathsf{T}}\boldsymbol{C}_{\boldsymbol{\theta}}^{(i),-1}(\boldsymbol{y}_{1:n_i} - m(\boldsymbol{x}_{1:n_i})) + \frac{n_i}{2}\log 2\pi, \tag{2}$$

with $\boldsymbol{C}_{\boldsymbol{\theta}}^{(i)} = \boldsymbol{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2_{\text{noise}}\boldsymbol{I}$.

The objective in Equation 2 represents a non-convex optimisation problem making GPs susceptible to local minima. Various off-the-shelf optimisation solvers ranging from first-order [13, 14] to second-order [15, 16] methods have been rigorously studied in the literature.

2.2 Acquisition Functions

Having introduced a distribution over latent black-box functions and specified mechanisms for updating hyperparameters, we now discuss the process by which novel query points are suggested for collection in order to improve the surrogate model's best guess for the global optimiser $\boldsymbol{x}^\star$. In Bayesian optimisation, proposing novel query points is performed through maximising an acquisition function $\alpha_{\text{type}}(\cdot|\mathcal{D}_i)$ that trades off exploration and exploitation by utilising statistics from $p(f(\cdot)|\mathcal{D}_i)$, i.e., $\boldsymbol{x}_{1:q}^{(\text{new})} = \arg\max_{\boldsymbol{x}\in\mathcal{X}^q} \alpha_{\text{type}}(\boldsymbol{x}|\mathcal{D}_i)$. During our participation in the competition, we experimented with the following commonly used batch forms originally introduced in [17]:

1. **Expected Improvement:** One of the most popular acquisition functions is expected improvement [18, 19], which determines new query points by maximising expected gain relative to the function values observed so far:

$$\alpha_{\text{q-EI}}(\boldsymbol{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\boldsymbol{f}(\boldsymbol{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\{\text{ReLU}(\boldsymbol{f}(\boldsymbol{x}_{1:q}) - f(\boldsymbol{x}_i^+)\boldsymbol{1}_q)\}\right], \tag{3}$$

where $\boldsymbol{1}_q$ denotes a q-dimensional vector of ones and as such, the ReLu$(\cdot)$ is to be executed element-wise, and $\boldsymbol{x}_i^+$ is the best performing input in the data so far.

2. **Probability of Improvement:** Another commonly-used acquisition function in Bayesian optimisation is the probability of improvement criterion which measures the probability of acquiring gains in the function value compared to $f(\boldsymbol{x}_i^+)$ [20]. Such a probability is measured through an expected Heaviside step function as follows:

$$\alpha_{\text{q-PI}}(\boldsymbol{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\boldsymbol{f}(\boldsymbol{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\left\{\mathbb{1}\{\boldsymbol{f}(\boldsymbol{x}_{1:q}) - f(\boldsymbol{x}_i^+)\boldsymbol{1}_q\}\right\}\right], \tag{4}$$

where $\mathbb{1}\{\boldsymbol{f}(\boldsymbol{x}_{1:q}) - f(\boldsymbol{x}_i^+)\boldsymbol{1}_q\}$ returns a q-dimensional binary vector with $[\mathbb{1}\{\boldsymbol{f}(\boldsymbol{x}_{1:q}) - f(\boldsymbol{x}_i^+)\}]_j = 1$ if $[\boldsymbol{f}(\boldsymbol{x}_{1:q})]_j \geq [f(\boldsymbol{x}_i^+)\boldsymbol{1}_q]_j$ and zero otherwise for all $j \in \{1, \ldots, q\}$.

3. **Upper Confidence Bound:** In this type of acquisition, the learner trades off the mean and variance of the predictive distribution to gather new query points for function evaluation [21]. To allow parallel computation, the authors in [17] have suggested the following form:

$$\alpha_{\text{q-UCB}}(\boldsymbol{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\boldsymbol{f}(\boldsymbol{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\boldsymbol{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\boldsymbol{\gamma}_i(\boldsymbol{x}_{1:q};\boldsymbol{\theta})|\right\}\right], \tag{5}$$

where $\boldsymbol{\gamma}_i(\boldsymbol{x}_{1:q};\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{x}_{1:q}) - \boldsymbol{\mu}_i(\boldsymbol{x}_{1:q};\boldsymbol{\theta})$.

3 HEBO: Surrogates & Multi-Objective Acquisitions

During our initial attempts, we found that standard off-the-shelve Bayesian optimisation submissions, i.e., ones with simplistic noise models and a single choice of an acquisition function, tended to underperform due to the complexity associated with the competition's data. Tackling these shortcomings, we proposed two adaptations to: 1) handle complex noise processes using input-output non-linear transformations, and 2) enable a multi-objective choice of acquisitions.

3.1 Surrogate Model Design

As noted earlier, we found that a vanilla Gaussian process model faced difficulties handling heteroscedasticity and non-stationarity available in the data. To tackle these problems, we perform two transformations; a power transformation ($\Gamma_{\boldsymbol{\zeta}}(\cdot)$) on the output and Warping ($\Psi_{\boldsymbol{\Theta}}(\cdot)$) on the input, dependent on parameters Θ.

Output: For output objectives, instead of the commonly used linear normalisation, we use the power transformation ($\Gamma_{\boldsymbol{\zeta}}(\cdot)$) to reduce heteroscedasticity. Precisely, we adopt the box-cox transformation[22] if objectives are strictly positive (or strictly negative), otherwise we follow the yeo-johnson's [23] one. For each transformation, the hyper-parameters for stabilising variance and minimising skewness are estimated through maximum likelihood. Note, this is executed **before** minimising the negative log marginal likelihood of the GP.

Input: For input parameters, we utilised an input warped Gaussian process [24] to handle non-stationary functions. Here, we perform a non-linear monotonic transformation by making use of the cumulative distribution function of the Kumaraswamy distribution given by: $\Psi_{\boldsymbol{\Theta}}(\boldsymbol{x};\boldsymbol{a},\boldsymbol{b})_i = 1 - (1 - \boldsymbol{x}_i^{\boldsymbol{a}_i})^{(\boldsymbol{b}_i - 1)}$ with the transformation executed component-wise and $\boldsymbol{\Theta} = \{\boldsymbol{a}, \boldsymbol{b}\}$ denoting the additional parameters introduced.

In executing these transformations, we followed readily available implementations, where the `power_transform` from sklearn [25] and `InputWarpedGP` from the GPy's package [26] were used.

Gaussian Process & Kernels: When selecting a surrogate model needed for line 3 of Algorithm 1, we deemed sample efficiency important and, hence, settled on adopting a Gaussian process with an additive kernel given by:

$$k^{(\text{HEBO})}_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2}(\boldsymbol{x},\boldsymbol{x}') = k^{(\text{Linear})}_{\boldsymbol{\theta}_1}(\boldsymbol{x},\boldsymbol{x}') + k^{(\text{Matérn32})}_{\boldsymbol{\theta}_2}(\boldsymbol{x},\boldsymbol{x}') \;\; \text{with } \boldsymbol{\theta}_1 \text{ and } \boldsymbol{\theta}_2 \text{ are kernel hyper-parameters.}$$

Stochastic mean function: Lastly, we observed from the `Bayesmark` benchmarks that most AutoML tasks were highly noisy. In this case, we found that adding noise to the GP's posterior mean helped aid exploration. To achieve this, we add on an additional likelihood noise-dependent term to the posterior mean, yielding a new mean function given by: $m^{(\text{HEBO})}(\cdot) = m(\cdot) + \boldsymbol{\xi}\sigma^2_{\text{noise}}$.

3.2 Multi-Objective Acquisitions

As mentioned earlier, we found that in certain circumstances, different acquisitions led to conflicting results. To circumvent such a problem, we adopted a multi-objective acquisition as that proposed in [27]. The multi-objective acquisition ensemble algorithm (MACE) searches for a Pareto front across *multiple* acquisitions functions, i.e., the solution that empirically scores higher than any individually tested acquisition. This also enables parallel optimisation as the multi-objective optimisation returns multiple Pareto-optimal recommendations. We utilised an evolutionary optimiser supporting mutation and cross-over – NSGA-II – from the `pymoo` [28] package to solve the following:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \;\; \left(-\alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i), -\alpha_{\text{q-PI}}(\boldsymbol{x}|\mathcal{D}_i), \alpha_{\text{q-UCB}}(\boldsymbol{x}|\mathcal{D}_i)\right) \tag{6}$$

In our implementation, we set $q = 1$ in Equation 6 and perform a logarithm transformation to expected improvement acquisition for faster optimisation. Of course, the introduction of the logarithm can yield numerical instabilities that we control via the following approximation to $\log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i)$:

$$\lim_{\boldsymbol{z}_i\to-\infty} \log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i) = \log \sigma_i(\boldsymbol{x};\boldsymbol{\theta}) - \frac{1}{2}\boldsymbol{z}_i^2 - \log(\boldsymbol{z}_i^2 - 1) - \frac{1}{2}\log(2\pi), \;\; \text{where } \boldsymbol{z}_i = \frac{\tau - \boldsymbol{\mu}_i(\boldsymbol{x};\boldsymbol{\theta})}{\sigma_i(\boldsymbol{x};\boldsymbol{\theta})},$$

With τ is the best function value observed so far, In our implementation, we use the above approximation when $\boldsymbol{z}_i < -6$ and the exact $\log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i)$ otherwise. We compare the proposed logarithmic approximation to the exact evaluation in Figure 2 in the Appendix. It can be seen that our approximation is accurate when $\boldsymbol{z}_i$ is small, but returns $+\infty$ when $z^2 = 1$. Note, a similar approach can be applied for approximating probability improvement.

4 Experiments

In this section, we first detail the summary results of our method compared with all `Bayesmark` baselines. We then discuss a thorough ablation study and the apparent significance of various design choices.

`Bayesmark` **Offline Datasets:** We conducted a comparison of HEBO across all 108 `Bayesmark` Offline datasets and all available baselines provided in the BBO Challenge starter kit. Figure 1b shows that HEBO significantly outperforms others with Pysot [29] and Turbo [30] being the second-and third-best, respectively. Interestingly, we can also see a more widely spread upper-quartile of HEBO vs other baselines[3].

Ablation Study: To ascertain the significance of the components of HEBO, we performed a rigorous ablation study. HEBO was tested with each significant component removed, on all 108 `Bayesmark` Offline Black-box tasks, similarly repeating each task with 20 random seeds. We focused on removing the following components; log approximation, the stochastic mean function, power transformation and input warping. When any component was removed, we observed a reduction in the average normalised score. Thus, we concluded all components to be beneficial. In particular, removing input warping and power transformation showed significant differences.

[3]Note, that the hypotheses from this offline study does not necessarily carry over to online testing that contain differing tasks. Such an analysis, however, provided us with comparative evidence against baselines.

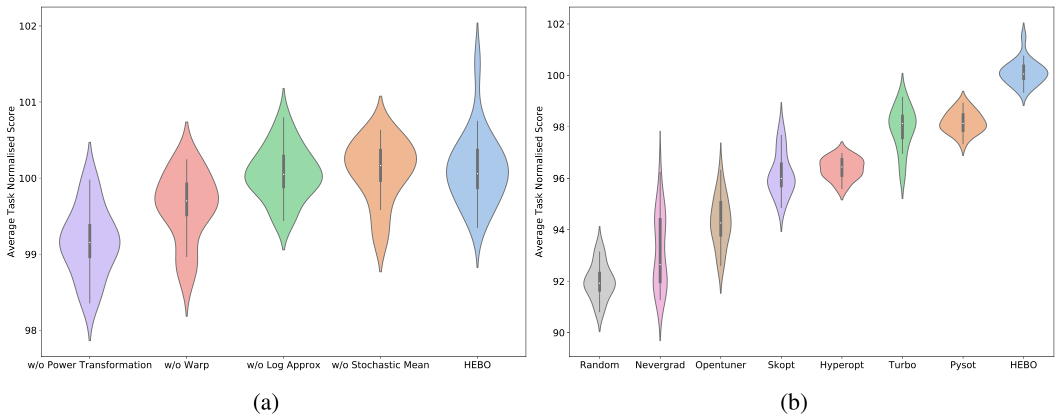


Figure 1: All algorithms in the plots are evaluated on the full `Bayesmark` Offline Dataset of 108 tasks and each task repeated with 20 random seeds. (a) Ablation Study of HEBO on Bayesmark Offline Dataset. We see that when all components are combined, we see a more appealing upper quartile distribution which reaches a much higher maximum scores than other ablations were able to, hinting towards a positively synergistic combination of algorithm components was found. (b) Summary of HEBO vs all available baselines. We see substantial improvements when using HEBO against all baselines, in many cases with HEBO also producing tighter distribution of scores on the lower quartile, and a high distribution on the upper quartile. This distribution of quartiles suggests that worst case we know it should roughly perform near the mean, and best case we could get a result significantly higher than the mean. This, paired with the fact that our mean performance is above satisfactory, give us much confidence in applying this method to a wide-range of BO tasks. Notice, as we significantly outperformed all baselines, similarly we see that Pysot and Turbo significantly outperform all *other* baselines.

	Hyperopt	Opentuner	Random	Nevergrad	Pysot	Skopt	Turbo	HEBO
Mean	96.375	94.317	92.004	93.196	98.177	96.175	97.951	**100.117**
Std	0.407	0.981	0.699	1.457	0.417	0.853	0.847	0.4765
P-value	$\mathbf{7.9e^{-26}}$	$\mathbf{6.9e^{-20}}$	$\mathbf{7.3e^{-31}}$	$\mathbf{3.8e^{-16}}$	$\mathbf{3.8e^{-16}}$	$\mathbf{1.3e^{-17}}$	$\mathbf{5.0e^{-11}}$	-

Table 1: `Bayesmark` Offline Dataset Summary of HEBO vs all available benchmarks on 108 tasks, each task repeated with 20 random seeds. Where P-value is a two-sided test for the null hypothesis that an algorithm has the same expected value as HEBO: p-values are highlight in bold if this null hypothesis is rejected with 95% significance. We found all methods to be significantly worse than HEBO.

In contrast, the removal of the stochastic mean function and log approximation showed insignificant differences. Significance test results from this ablation study are shown in Table 2 in the Appendix. We want to remind the reader again that the hypotheses from this ablation study do not necessarily carry over to the online dataset from the competition, due to the aforementioned change of evaluation domain.

5 Conclusion

We presented HEBO, a Bayesian optimisation method that utilises a novel formulation of warped Gaussian Processes, multi-objective acquisition and a multi-objective evolutionary approach. In union, these adaptations led to a general BO algorithm that won the NeurIPS 2020 Black-box Optimisation Challenge, where the rankings were based on real-world tasks. We performed a thorough ablation study to identify fundamental properties that are strongly correlated with the observed success. In conclusion, we believe our work to help improve the understanding and highlight areas of Bayesian optimisation research that highly impact real-world problems.

Acknowledgements

We want to thank the organisers of the competition, as well as Huawei Noah's Ark for providing all the necessary resources for us to compete in this challenge.

References

[1] Isabelle Guyon, Lisheng Sun-Hosoya, Marc Boullé, Hugo Jair Escalante, Sergio Escalera, Zhengying Liu, Damir Jajetic, Bisakha Ray, Mehreen Saeed, Michéle Sebag, Alexander Statnikov, WeiWei Tu, and Evelyne Viegas. Analysis of the automl challenge series 2015-2018. In *AutoML*, Springer series on Challenges in Machine Learning, 2019.

[2] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[3] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[4] Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pages 3393–3403. PMLR, 2020.

[5] Henry B Moss and Ryan-Rhys Griffiths. Gaussian process molecule property prediction with flowmo. *arXiv preprint arXiv:2010.01118*, 2020.

[6] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2):577–586, 2020.

[7] Zhuhong Zhang, Lei Wang, and Fei Long. Immune optimization approach solving multi-objective chance-constrained programming. *Evolving Systems*, 6(1):41–53, 2015.

[8] Prasant Kumar Mahapatra, Susmita Ganguli, and Amod Kumar. A hybrid particle swarm optimization and artificial immune system algorithm for image enhancement. *Soft Computing*, 19(8):2101–2109, 2015.

[9] Kwang-Seon Yoo and Seog-Young Han. Modified ant colony optimization for topology optimization of geometrically nonlinear structures. *International Journal of Precision Engineering and Manufacturing*, 15(4):679–687, 2014.

[10] Dmitrii V Speranskii. Ant colony optimization algorithms for digital device diagnostics. *Automatic Control and Computer Sciences*, 49(2):82–87, 2015.

[11] Alexander I Cowen-Rivers, Daniel Palenicek, Vincent Moens, Mohammed Abdullah, Aivar Sootla, Jun Wang, and Haitham Ammar. Samba: Safe model-based & active reinforcement learning. *arXiv preprint arXiv:2006.09436*, 2020.

[12] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*, volume 2. MIT press Cambridge, MA, 2006.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[14] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems*, 20:161–168, 2007.

[15] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.

[16] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

[17] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 9884–9895, 2018.

[18] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

[19] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[20] Harold J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86(1):97–106, 03 1964.

[21] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022, 2010.

[22] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964.

[23] In-Kwon Yeo and Richard A Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000.

[24] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pages 1674–1682, 2014.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, 2012.

[27] Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International Conference on Machine Learning*, pages 3306–3314, 2018.

[28] J. Blank and K. Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[29] David Eriksson, David Bindel, and Christine A Shoemaker. pysot and poap: An event-driven asynchronous framework for surrogate optimization. *arXiv preprint arXiv:1908.00420*, 2019.

[30] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 5496–5507, 2019.

A Ablation

Table 2 shows the two-sided independent t-test performed on all components against the complete HEBO algorithm. We observe that the two most significant features were the input-output non-linear transformations. We noticed slight improvements by using the log approximation scheme and having in a stochastic mean function. However, these small improvements were more considerable on the more challenging online dataset.

	HEBO w/o LogApprox	HEBO w/o Stochastic Mean	HEBO w/o Power	HEBO w/o Warp	HEBO
Mean	100.061	100.099	99.158	99.625	**100.117**
Std	0.351	0.408	0.451	0.453	0.476
P-value	0.675	0.894	$\mathbf{1.041e^{-07}}$	**0.002**	-
Min	**99.441**	99.215	98.358	98.680	99.349
25%	**99.884**	99.970	98.963	99.515	99.871
50%	100.051	**100.163**	99.155	99.700	100.057
75%	100.290	100.368	99.375	99.922	**100.369**
Max	100.792	100.629	99.974	100.240	**101.516**

Table 2: Ablation study on HEBO in offline benchmarks, evaluating each component based on normalised score statistics. There are a total of 108 tasks in the offline benchmarks, and we repeat each of these with 20 random seeds. Where P-value is a two sided independent t-test on the hypothesis that the component significantly effects the performance of HEBO. We can see clearly in the offline data that each component offered an increase in mean score. For each ablation study, we perform a t-test with HEBO. The test highlights that the results from using the power transformation and input warped Gaussian Process are significant in their effect on the mean score. However, adding noise to the posterior mean and using the log approximation for numerical stability did not significantly improve HEBO. It is worth noting, however, that we observed a big difference in how algorithms performed offline vs online, so this ablation is not conclusive, but instead offers support to each components claim.

B Log Approximation

In Equation 6, expected improvement is used in the MACE framework. Denote $\boldsymbol{z}_i = \frac{\tau - \boldsymbol{\mu}_i(\boldsymbol{x};\boldsymbol{\theta})}{\sigma_i(\boldsymbol{x};\boldsymbol{\theta})}$, where τ is the best observed function value, $\mu(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x})$ are the posterior mean and variance, expected improvement can be rewritten as:

$$\alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i) = \sigma_i(\boldsymbol{x};\boldsymbol{\theta})(\boldsymbol{z}_i\Phi(\boldsymbol{z}_i) + \phi(\boldsymbol{z}_i)) \;\text{ with }\; \boldsymbol{z}_i = \frac{\tau - \boldsymbol{\mu}_i(\boldsymbol{x};\boldsymbol{\theta})}{\sigma_i(\boldsymbol{x};\boldsymbol{\theta})},$$

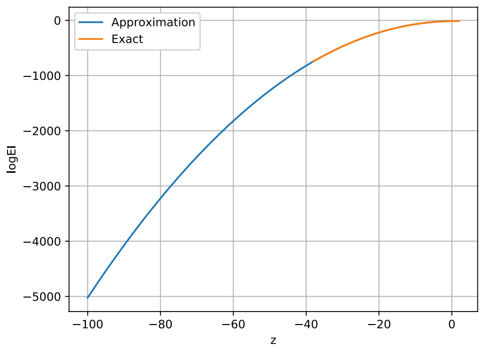
where $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative and density functions of a standard normal distribution $\mathcal{N}(0,1)$.

When optimising expected improvement, sometimes we start from point $\boldsymbol{x}$ with very small $\boldsymbol{z}_i = \frac{\tau - \boldsymbol{\mu}_i(\boldsymbol{x};\boldsymbol{\theta})}{\sigma_i(\boldsymbol{x};\boldsymbol{\theta})}$ (starting point of gradient based optimiser or initial population of evolutionary optimiser), this could happen when $\boldsymbol{x}$ has bad objective value, and is acknowledged by the surrogate model with high confidence. In that case, the expected improvement value (and its gradient) would be very close to zero, making it hard to optimise.

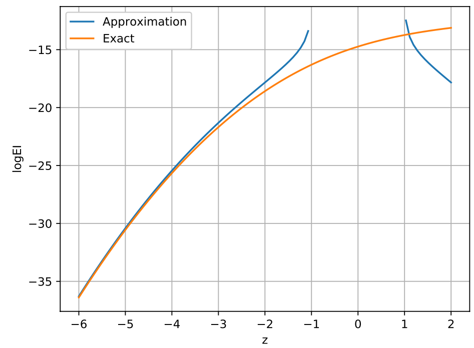
An apparent approach is to instead optimise $\log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i)$. However, we found that sometimes the expected improvement value is so small that evaluating $\log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i)$ returns NaN, leading to numerical instability during the optimisation.

The analytical expression of $\log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i)$ is:

$$\log \alpha_{\text{q-EI}}(\boldsymbol{x}|\mathcal{D}_i) = \log \sigma_i(\boldsymbol{x};\boldsymbol{\theta}) + \log \phi(\boldsymbol{z}_i) + \log\left(1 + \frac{\boldsymbol{z}_i\Phi(\boldsymbol{z}_i)}{\phi(\boldsymbol{z}_i)}\right) \tag{7}$$



(a)



(b)

Figure 2: (a-b) Comparison between the exact $\log \alpha_{\text{q-EI}}(\boldsymbol{x}_{1:q}|\mathcal{D}_i)$ and the proposed logarithm approximation, note that when $z < -40$, exact evaluation returns NaN while the approximation can still be calculated.

In Equation 7, the first two terms are easy to evaluate, but we don't have analytical expression to the last term. With Ĺhospital rule, it's easy to show that:

$$\lim_{\boldsymbol{z}_i \to -\infty} \frac{\boldsymbol{z}_i \Phi(\boldsymbol{z}_i)}{\phi(\boldsymbol{z}_i)} = \lim_{\boldsymbol{z}_i \to -\infty} \frac{\Phi(\boldsymbol{z}_i) + \boldsymbol{z}_i \phi(\boldsymbol{z}_i)}{-\boldsymbol{z}_i \phi(\boldsymbol{z}_i)} = \lim_{\boldsymbol{z}_i \to -\infty} -1 - \frac{\Phi(\boldsymbol{z}_i)}{\boldsymbol{z}_i \phi(\boldsymbol{z}_i)} = -1 - \frac{\phi(\boldsymbol{z}_i)}{\phi(\boldsymbol{z}_i) - \boldsymbol{z}_i^2 \phi(\boldsymbol{z}_i)}$$
$$= -1 + \frac{1}{\boldsymbol{z}_i^2 - 1}.$$

We compare the proposed logarithmic approximation to the exact evaluation in Figure 2. It can be seen that our approximation is accurate when $\boldsymbol{z}_i$ is small, but returns $+\infty$ when $z^2 = 1$.

C Stochastic mean exploration

The motivation for defining a new stochastic mean function was based on preliminary experiments where we first identified important hyper-parameters for a specific task 3 and found that in certain regions of the hyper-parameter space, random exploration appeared the most efficient way to search, as shown in Figure 4. Secondly, we found that adding noise to a function being optimised by a genetic algorithm allows it to switch between random search around a promising region, where the level of random search is dependent on the magnitude of the noise, as shown in Figure 5.

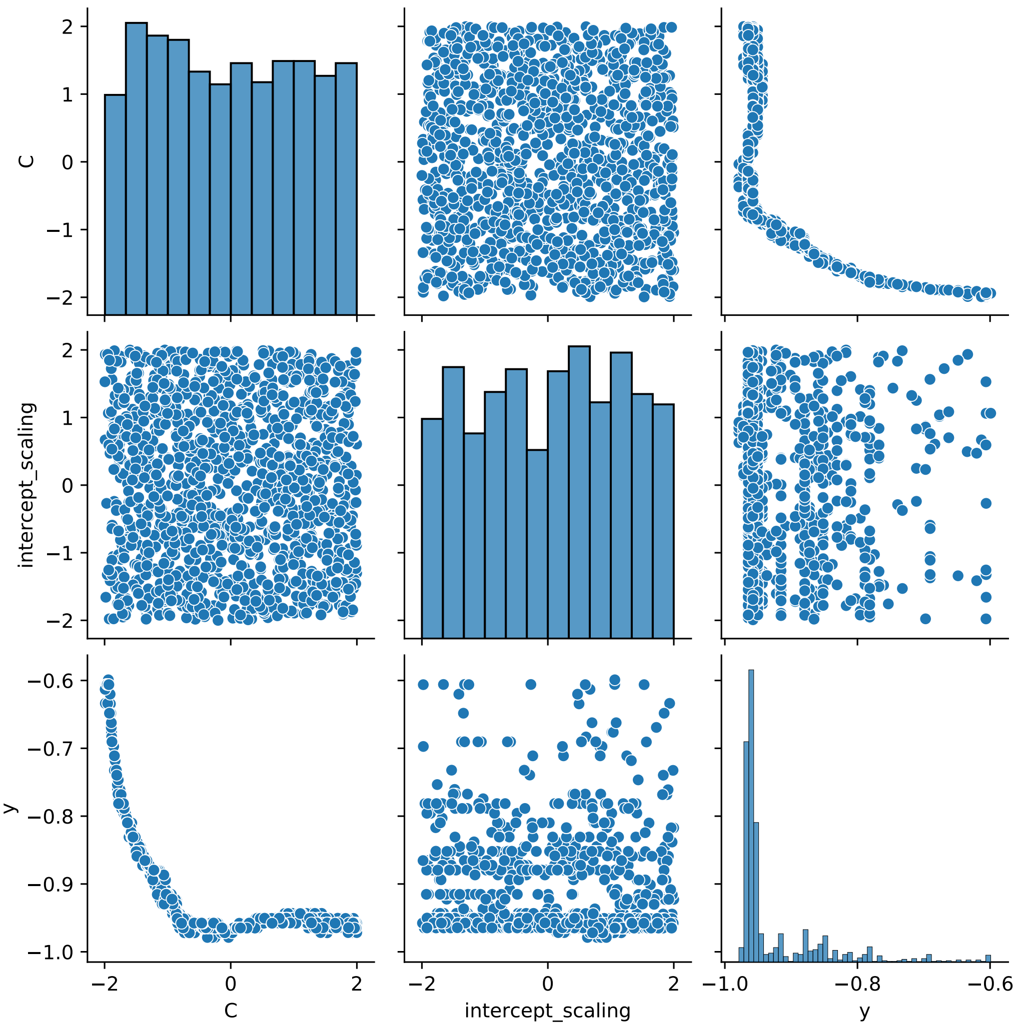
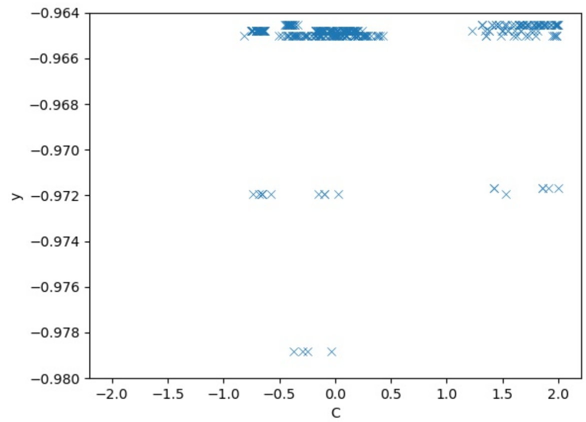
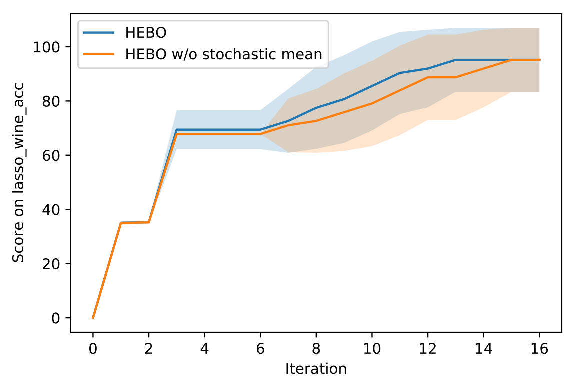


Figure 3: All plots show data from the same `Bayesmark` task. The aim of this task is to tune two hyper-parameters, a regularisation parameter C, and an intercept scaling parameter to maximising the validation accuracy (y) of a lasso classifier on the Wine dataset. (a) Shows each of the hyper-parameters and accuracy correlated against each other. Importantly, we find weak correlation between intercept scaling and accuracy. In contrast, we find strong correlation between C and accuracy.

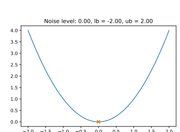


(a)

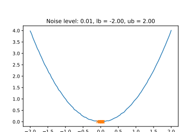


(b)

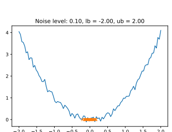
Figure 4: All plots show data from the same `Bayesmark` task. The goal of this task is to tune two hyper-parameters, a regularisation parameter C, and an intercept scaling parameter to maximising the validation accuracy (y) of a lasso classifier on the Wine dataset. (a) We then do a more broader evaluation of many (uniformly sampled) C values and the corresponding validation classification accuracy once the lasso classifier has been trained and notice in a specific region of regularisation values (between $[-1, 2]$), to be no clear correlation with accuracy. Thus, the best method for searching in this region would appear to be random — this motivates us exploring a BO algorithm which can resort to some form of random search in challenging regions. (b) We plot the average score of HEBO w and w/o the stochastic mean term. Note, on this task one achieves a normalised score of 100 if the model achieves 97.9% validation accuracy, 67 if it gets 97.2% validation accuracy and 35 for 96.4% validation accuracy. Hence, we can see the more difficult to identify regularisation hyper-parameters are found faster with the stochastic mean term, than without.



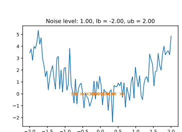
(a)



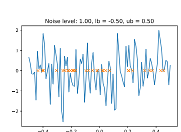
(b)



(c)



(d)



(e)

Figure 5: $f(x) = x^2$ Synthetic black-box function experiment to demonstrate the optimised points (orange crosses) that a genetic algorithm arrives at when varying noise levels are introduced, which simulate the varying likelihood noise values that effect the posterior mean prediction. We can clearly see the correlation of, as noise increases as does the fine spread of optimised. It is this principle that allows HEBO to achieve a coarse-to-fine search