Supplementary Material

BORE: Bayesian Optimization by Density-Ratio Estimation

Louis C. Tiao [1 2] **Aaron Klein** [3] **Matthias Seeger** [3] **Edwin V. Bonilla** [2 1] **Cédric Archambeau** [3] **Fabio Ramos** [1 4]

A. Expected improvement

For completeness, we reproduce the derivations of Bergstra et al. (2011). Recall from eq. 1 (main paper) that the expected improvement (EI) function is defined as the expectation of the improvement utility function $U(\mathbf{x}, y, \tau)$ over the posterior predictive distribution $p(y \mid \mathbf{x}, \mathcal{D}_N)$. Expanding this out, we have

$$\begin{aligned}\alpha(\mathbf{x}; \mathcal{D}_N, \tau) \coloneqq \mathbb{E}_{p(y \mid \mathbf{x}, \mathcal{D}_N)}[U(\mathbf{x}, y, \tau)] &= \int_{-\infty}^{\infty} U(\mathbf{x}, y, \tau) p(y \mid \mathbf{x}, \mathcal{D}_N) \,\mathrm{d}y \\ &= \int_{-\infty}^{\tau} (\tau - y) p(y \mid \mathbf{x}, \mathcal{D}_N) \,\mathrm{d}y \\ &= \frac{1}{p(\mathbf{x} \mid \mathcal{D}_N)} \int_{-\infty}^{\tau} (\tau - y) p(\mathbf{x} \mid y, \mathcal{D}_N) p(y \mid \mathcal{D}_N) \,\mathrm{d}y,\end{aligned}$$

where we have invoked Bayes' rule in the final step above. Next, the denominator evaluates to

$$\begin{aligned}p(\mathbf{x} \mid \mathcal{D}_N) &= \int_{-\infty}^{\infty} p(\mathbf{x} \mid y, \mathcal{D}_N) p(y \mid \mathcal{D}_N) \,\mathrm{d}y \\ &= \ell(\mathbf{x}) \int_{-\infty}^{\tau} p(y \mid \mathcal{D}_N) \,\mathrm{d}y + g(\mathbf{x}) \int_{\tau}^{\infty} p(y \mid \mathcal{D}_N) \,\mathrm{d}y \\ &= \gamma \ell(\mathbf{x}) + (1 - \gamma) g(\mathbf{x}),\end{aligned}$$

since, by definition, $\gamma = \Phi(\tau) \coloneqq p(y \leq \tau \mid \mathcal{D}_N)$. Finally, we evaluate the numerator,

$$\begin{aligned}\int_{-\infty}^{\tau} (\tau - y) p(\mathbf{x} \mid y, \mathcal{D}_N) p(y \mid \mathcal{D}_N) \,\mathrm{d}y &= \ell(\mathbf{x}) \int_{-\infty}^{\tau} (\tau - y) p(y \mid \mathcal{D}_N) \,\mathrm{d}y \\ &= \ell(\mathbf{x}) \tau \int_{-\infty}^{\tau} p(y \mid \mathcal{D}_N) \,\mathrm{d}y - \ell(\mathbf{x}) \int_{-\infty}^{\tau} y p(y \mid \mathcal{D}_N) \,\mathrm{d}y \\ &= \gamma \tau \ell(\mathbf{x}) - \ell(\mathbf{x}) \int_{-\infty}^{\tau} y p(y \mid \mathcal{D}_N) \,\mathrm{d}y \\ &= K \cdot \ell(\mathbf{x}),\end{aligned}$$

where

$$K = \gamma \tau - \int_{-\infty}^{\tau} y p(y \mid \mathcal{D}_N) \,\mathrm{d}y.$$

Hence, this shows that the EI function is equivalent to the γ-relative density ratio (Yamada et al., 2011) up to a constant factor K,

$$\begin{aligned}\alpha(\mathbf{x}; \mathcal{D}_N, \tau) &\propto \frac{\ell(\mathbf{x})}{\gamma \ell(\mathbf{x}) + (1 - \gamma) g(\mathbf{x})} \\ &= \left(\gamma + \frac{g(\mathbf{x})}{\ell(\mathbf{x})} (1 - \gamma) \right)^{-1}.\end{aligned}$$

A.1. Probability of Improvement

In a nutshell, the derivation of eq. 5 (main paper) reproduced above hinges upon the invocation of Bayes' rule to express the predictive as $p(y \,|\, \mathbf{x}, \mathcal{D}_N) = {}^{p(\mathbf{x} \,|\, y, \mathcal{D}_N) p(y \,|\, \mathcal{D}_N)}/_{p(\mathbf{x} \,|\, \mathcal{D}_N)}$ and then estimating the terms $p(\mathbf{x} \,|\, y, \mathcal{D}_N)$ and $p(y \,|\, \mathcal{D}_N)$ or, more precisely, estimating the conditionals $\ell(\mathbf{x}) \coloneqq p\left(\mathbf{x} \,|\, y \leq \tau; \mathcal{D}_N\right)$ and $g(\mathbf{x}) \coloneqq p\left(\mathbf{x} \,|\, y > \tau; \mathcal{D}_N\right)$ while defining $\gamma \coloneqq p(y \leq \tau \,|\, \mathcal{D}_N)$.

Under this formulation, it is instructive to derive the probability of improvement (PI) function (Kushner, 1964), another improvement-based acquisition function that seeks to quantify the amount by which y decreases over some threshold τ. The PI function is defined as

$$\alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_N, \tau) \coloneqq \mathbb{E}_{p(y \,|\, \mathbf{x}, \mathcal{D}_N)}[U_{\text{PI}}(\mathbf{x}, y, \tau)],$$

with utility function $U_{\text{PI}}(\mathbf{x}, y, \tau) \coloneqq \mathbb{I}[y \leq \tau]$. In particular, we have

$$\begin{aligned}
\alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_N, \tau) &\coloneqq \int_{-\infty}^{\infty} \mathbb{I}[y \leq \tau] p(y \,|\, \mathbf{x}, \mathcal{D}_N) \,\mathrm{d}y \\
&= \int_{-\infty}^{\tau} p(y \,|\, \mathbf{x}, \mathcal{D}_N) \,\mathrm{d}y \\
&= \frac{1}{p(\mathbf{x} \,|\, \mathcal{D}_N)} \int_{-\infty}^{\tau} p(\mathbf{x} \,|\, y, \mathcal{D}_N) p(y \,|\, \mathcal{D}_N) \,\mathrm{d}y.
\end{aligned}$$

As before, the denominator is $p(\mathbf{x} \,|\, \mathcal{D}_N) = \gamma \ell(\mathbf{x}) + (1 - \gamma) g(\mathbf{x})$, while the numerator evaluates to

$$\int_{-\infty}^{\tau} p(\mathbf{x} \,|\, y, \mathcal{D}_N) p(y \,|\, \mathcal{D}_N) \,\mathrm{d}y = \ell(\mathbf{x}) \int_{-\infty}^{\tau} p(y \,|\, \mathcal{D}_N) \,\mathrm{d}y = \gamma \ell(\mathbf{x}).$$

Hence

$$\begin{aligned}
\alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_N, \tau) &= \frac{\gamma \ell(\mathbf{x})}{\gamma \ell(\mathbf{x}) + (1 - \gamma) g(\mathbf{x})} \\
&= \gamma \cdot r_\gamma(\mathbf{x}).
\end{aligned}$$

Therefore, under the formulation of Bergstra et al. (2011), the PI and EI functions turn out to be proportional. In contrast, recall that if the predictive has the Gaussian form of eq. 2 (main paper), we simply get

$$\alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_N, \tau) = \Psi\left(\frac{\tau - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right),$$

where Ψ denotes the cdf of the normal distribution. Clearly, as a function of $\mathbf{x}$, this is not equivalent to the EI function derived in eq. 3 (main paper) under the same assumptions of having a Gaussian predictive.

B. Relative density-ratio: unabridged notation

In § 2, for notational simplicity, we had excluded the dependencies of ℓ, g and r_γ on τ. Let us now define these densities more explicitly as

$$\ell(\mathbf{x}; \tau) \coloneqq p\left(\mathbf{x} \,|\, y \leq \tau, \mathcal{D}_N\right), \qquad \text{and} \qquad g(\mathbf{x}; \tau) \coloneqq p\left(\mathbf{x} \,|\, y > \tau, \mathcal{D}_N\right),$$

and accordingly, the γ-relative density-ratio from eq. 4 (main paper) as

$$r(\mathbf{x}; \gamma, \tau) = \frac{\ell(\mathbf{x}; \tau)}{\gamma \ell(\mathbf{x}; \tau) + (1 - \gamma) g(\mathbf{x}; \tau)}.$$

Recall from eq. 5 (main paper) that

$$\alpha\left(\mathbf{x}; \mathcal{D}_N, \Phi^{-1}(\gamma)\right) \propto r\left(\mathbf{x}; \gamma, \Phi^{-1}(\gamma)\right). \tag{1}$$

In step 1 (main paper), Bergstra et al. (2011) resort to optimizing $r(\mathbf{x}; 0, \Phi^{-1}(\gamma))$, which is justified by the fact that

$$r(\mathbf{x}; \gamma, \Phi^{-1}(\gamma)) = h_\gamma\left[r\left(\mathbf{x}; 0, \Phi^{-1}(\gamma)\right)\right],$$

for strictly nondecreasing h_γ. Note we have used a blue and orange color coding to emphasize the differences in the setting of γ (best viewed on a computer screen). Recall that $\Phi^{-1}(0) = \min_n y_n$ corresponds to the conventional setting of threshold τ. However, make no mistake, for any $\gamma > 0$,

$$\alpha\left(\mathbf{x}; \mathcal{D}_N, \Phi^{-1}(0)\right) \not\propto r\left(\mathbf{x}; 0, \Phi^{-1}(\gamma)\right).$$

Therefore, given the numerical instabilities associated with this approach as discussed in § 2.4, there is no advantage to be gained from taking this direction. Moreover, eq. 1 only holds for $\gamma > 0$. To see this, suppose $\gamma = 0$, which gives

$$\alpha\left(\mathbf{x}; \mathcal{D}_N, \Phi^{-1}(0)\right) \propto r\left(\mathbf{x}; 0, \Phi^{-1}(0)\right).$$

However, since by definition $\ell\left(\mathbf{x}; \Phi^{-1}(0)\right)$ has no mass, the RHS is undefined.

C. Class-posterior probability

We provide an unabridged derivation of the identity of eq. 8 (main paper). First, the γ-relative density ratio is given by

$$\begin{aligned} r_\gamma(\mathbf{x}) &\coloneqq \frac{\ell(\mathbf{x})}{\gamma\ell(\mathbf{x}) + (1-\gamma)g(\mathbf{x})} \\ &= \frac{p(\mathbf{x} \,|\, z=1)}{\gamma \cdot p(\mathbf{x} \,|\, z=1) + (1-\gamma) \cdot p(\mathbf{x} \,|\, z=0)} \\ &= \left(\frac{p(z=1 \,|\, \mathbf{x})\cancel{p(\mathbf{x})}}{p(z=1)}\right)\left(\gamma \cdot \frac{p(z=1 \,|\, \mathbf{x})\cancel{p(\mathbf{x})}}{p(z=1)} + (1-\gamma) \cdot \frac{p(z=0 \,|\, \mathbf{x})\cancel{p(\mathbf{x})}}{p(z=0)}\right)^{-1}. \end{aligned}$$

By construction, we have $p(z=1) \coloneqq p(y \leq \tau) = \gamma$ and $\pi(\mathbf{x}) \coloneqq p(z=1 \,|\, \mathbf{x})$. Therefore,

$$\begin{aligned} r_\gamma(\mathbf{x}) &= \gamma^{-1}\pi(\mathbf{x})\left(\cancel{\gamma} \cdot \frac{\pi(\mathbf{x})}{\cancel{\gamma}} + \cancel{(1-\gamma)} \cdot \frac{1-\pi(\mathbf{x})}{\cancel{1-\gamma}}\right)^{-1} \\ &= \gamma^{-1}\pi(\mathbf{x}). \end{aligned}$$

Alternatively, we can also arrive at the same result by writing the ordinary density ratio $r_0(\mathbf{x})$ in terms of $\pi(\mathbf{x})$ and γ, which is well-known to be

$$r_0(\mathbf{x}) = \left(\frac{\gamma}{1-\gamma}\right)^{-1} \frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}.$$

Plugging this into function h_γ, we get

$$\begin{aligned} r_\gamma(\mathbf{x}) = h_\gamma(r_0(\mathbf{x})) &= h_\gamma\left(\left(\frac{\gamma}{1-\gamma}\right)^{-1} \frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right) \\ &= \left(\gamma + (1-\gamma)\left(\frac{\gamma}{1-\gamma}\right)\left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right)^{-1}\right)^{-1} \\ &= \gamma^{-1}\left(1 + \left(\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}\right)^{-1}\right)^{-1} \\ &= \gamma^{-1}\pi(\mathbf{x}). \end{aligned}$$

D. Log loss

The log loss, also known as the binary cross-entropy (BCE) loss, is given by

$$\mathcal{L}^*(\boldsymbol{\theta}) \coloneqq -\beta \cdot \mathbb{E}_{\ell(\mathbf{x})}[\log \pi_{\boldsymbol{\theta}}(\mathbf{x})] - (1-\beta) \cdot \mathbb{E}_{g(\mathbf{x})}[\log(1 - \pi_{\boldsymbol{\theta}}(\mathbf{x}))], \tag{2}$$

where β denotes the class balance rate. In particular, let N_ℓ and N_g be the sizes of the support of $\ell(\mathbf{x})$ and $g(\mathbf{x})$, respectively. Then, we have

$$\beta = \frac{N_\ell}{N}, \qquad \text{and} \qquad 1-\beta = \frac{N_g}{N},$$

where $N = N_\ell + N_g$. In practice, we approximate the log loss $\mathcal{L}^*(\boldsymbol{\theta})$ by the empirical risk of eq. 9 (main paper), given by

$$\mathcal{L}(\boldsymbol{\theta}) \coloneqq -\frac{1}{N}\left(\sum_{n=1}^{N} z_n \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_n) + (1-z_n)\log\left(1-\pi_{\boldsymbol{\theta}}(\mathbf{x}_n)\right)\right).$$

In this section, we show that the approximation of eq. 10 (main paper), that is,

$$\pi_{\boldsymbol{\theta}}(\mathbf{x}) \simeq \gamma \cdot r_\gamma(\mathbf{x}),$$

attains equality at $\boldsymbol{\theta}_\star = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^*(\boldsymbol{\theta})$.

D.1. Optimum

Taking the functional derivative of $\mathcal{L}^*$ in eq. 2, we get

$$\begin{aligned}\frac{\partial \mathcal{L}^*}{\partial \pi_{\boldsymbol{\theta}}} &= -\mathbb{E}_{\ell(\mathbf{x})}\left[\frac{\beta}{\pi_{\boldsymbol{\theta}}(\mathbf{x})}\right] + \mathbb{E}_{g(\mathbf{x})}\left[\frac{1-\beta}{1-\pi_{\boldsymbol{\theta}}(\mathbf{x})}\right] \\ &= \int\left(-\beta\frac{\ell(\mathbf{x})}{\pi_{\boldsymbol{\theta}}(\mathbf{x})} + (1-\beta)\frac{g(\mathbf{x})}{1-\pi_{\boldsymbol{\theta}}(\mathbf{x})}\right)\mathrm{d}\mathbf{x}\end{aligned}$$

This integral evaluates to zero iff the integrand itself evaluates to zero. Hence, we solve the following for $\pi_{\boldsymbol{\theta}_\star}(\mathbf{x})$,

$$\beta\frac{\ell(\mathbf{x})}{\pi_{\boldsymbol{\theta}_\star}(\mathbf{x})} = (1-\beta)\frac{g(\mathbf{x})}{1-\pi_{\boldsymbol{\theta}_\star}(\mathbf{x})}.$$

We re-arrange this expression to give

$$\frac{1-\pi_{\boldsymbol{\theta}_\star}(\mathbf{x})}{\pi_{\boldsymbol{\theta}_\star}(\mathbf{x})} = \left(\frac{1-\beta}{\beta}\right)\frac{g(\mathbf{x})}{\ell(\mathbf{x})} \qquad \Leftrightarrow \qquad \frac{1}{\pi_{\boldsymbol{\theta}_\star}(\mathbf{x})} - 1 = \frac{\beta\ell(\mathbf{x}) + (1-\beta)g(\mathbf{x})}{\beta\ell(\mathbf{x})} - 1.$$

Finally, we add one to both sides and invert the result to give

$$\begin{aligned}\pi_{\boldsymbol{\theta}_\star}(\mathbf{x}) &= \frac{\beta\ell(\mathbf{x})}{\beta\ell(\mathbf{x}) + (1-\beta)g(\mathbf{x})} \\ &= \beta \cdot r_\beta(\mathbf{x}).\end{aligned}$$

Since, by definition $\beta = \gamma$, this leads to $\pi_{\boldsymbol{\theta}_\star}(\mathbf{x}) = \gamma \cdot r_\gamma(\mathbf{x})$ as required.

D.2. Empirical risk minimization

For completeness, we show that the log loss $\mathcal{L}^*(\boldsymbol{\theta})$ of eq. 2 can be approximated by $\mathcal{L}(\boldsymbol{\theta})$ of eq. 9 (main paper). First, let ρ be the permutation of the set $\{1,\ldots,N\}$, i.e. the bijection from $\{1,\ldots,N\}$ to itself, such that $y_{\rho(n)} \leq \tau$ if $0 < \rho(n) \leq N_\ell$, and $y_{\rho(n)} > \tau$ if $N_\ell < \rho(n) \leq N_g$. That is to say,

$$\mathbf{x}_{\rho(n)} \sim \begin{cases}\ell(\mathbf{x}) & \text{if } 0 < \rho(n) \leq N_\ell, \\ g(\mathbf{x}) & \text{if } N_\ell < \rho(n) \leq N_g.\end{cases} \quad \text{and} \quad z_{\rho(n)} \coloneqq \begin{cases}1 & \text{if } 0 < \rho(n) \leq N_\ell, \\ 0 & \text{if } N_\ell < \rho(n) \leq N_g.\end{cases}$$

Then, we have

$$\begin{aligned}\mathcal{L}^*(\boldsymbol{\theta}) &\coloneqq -\frac{1}{N}\left(N_\ell \cdot \mathbb{E}_{\ell(\mathbf{x})}[\log \pi_{\boldsymbol{\theta}}(\mathbf{x})] + N_g \cdot \mathbb{E}_{g(\mathbf{x})}[\log\left(1-\pi_{\boldsymbol{\theta}}(\mathbf{x})\right)]\right) \\ &\simeq -\frac{1}{N}\left(\cancel{N_\ell} \cdot \frac{1}{\cancel{N_\ell}}\sum_{n=1}^{N_\ell}\log \pi_{\boldsymbol{\theta}}(\mathbf{x}_{\rho(n)}) + \cancel{N_g} \cdot \frac{1}{\cancel{N_g}}\sum_{n=N_\ell+1}^{N_g}\log\left(1-\pi_{\boldsymbol{\theta}}(\mathbf{x}_{\rho(n)})\right)\right) \\ &= -\frac{1}{N}\left(\sum_{n=1}^{N} z_{\rho(n)}\log \pi_{\boldsymbol{\theta}}(\mathbf{x}_{\rho(n)}) + (1-z_{\rho(n)})\log\left(1-\pi_{\boldsymbol{\theta}}(\mathbf{x}_{\rho(n)})\right)\right) \\ &= -\frac{1}{N}\left(\sum_{n=1}^{N} z_n \log \pi_{\boldsymbol{\theta}}(\mathbf{x}_n) + (1-z_n)\log\left(1-\pi_{\boldsymbol{\theta}}(\mathbf{x}_n)\right)\right) \coloneqq \mathcal{L}(\boldsymbol{\theta}),\end{aligned}$$

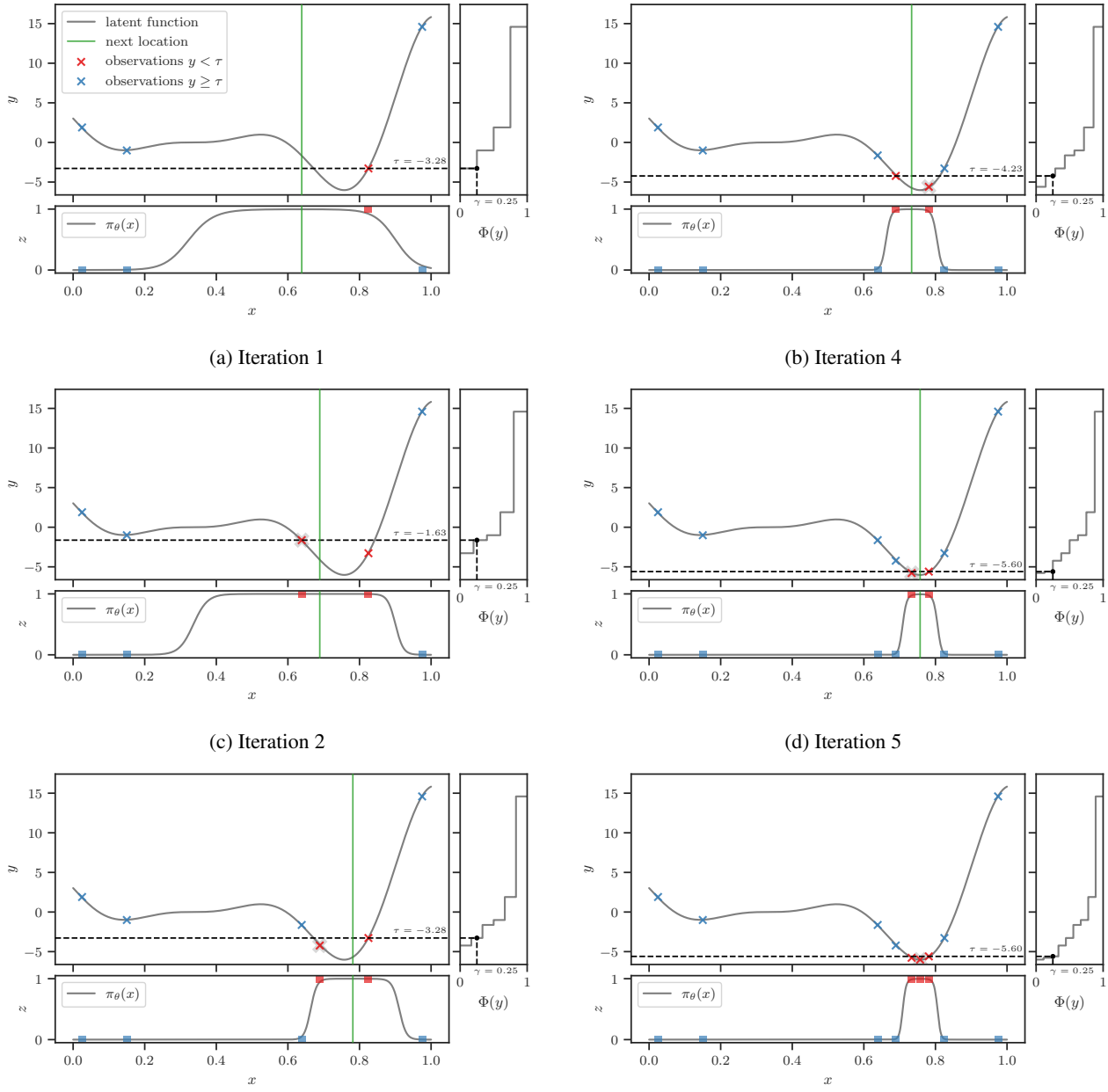
as required.

E. Step-through visualization

For illustration purposes, we go through Algorithm 1 (main paper) step-by-step on a synthetic problem for a half-dozen iterations. Specifically, we minimize the FORRESTER function

$$f(x) \coloneqq (6x-2)^2 \sin(12x-4),$$

in the domain $x \in [0,1]$ with observation noise $\varepsilon \sim \mathcal{N}(0, 0.05^2)$. See Figure 1.



(a) Iteration 1

(b) Iteration 4

(c) Iteration 2

(d) Iteration 5

(e) Iteration 3

(f) Iteration 6

Figure 1. We go through Algorithm 1 (main paper) step-by-step on a synthetic problem for a half-dozen iterations. Specifically, we minimize the FORRESTER function. The algorithm is started with 4 random initial designs. Each subfigure depicts the state after Lines 6 and 8—namely, after *updating* and *maximizing* the classifier. In every subfigure, the main pane depicts the noise-free function, represented by the *solid gray* curve, and the set of observations, represented by *crosses* '×'. The location that was evaluated in the previous iteration is highlighted with a *gray outline*. The right pane shows the ECDF of the observed y values. The *vertical dashed black line* in this pane is located at $\gamma = \frac{1}{4}$. The *horizontal dashed black line* is located at τ, the value of y such that $\Phi(y) = \frac{1}{4}$, i.e., $\tau = \Phi^{-1}\left(\frac{1}{4}\right)$. The instances below this horizontal line are assigned binary label $z = 1$, while those above are assigned $z = 0$. This is visualized in the bottom pane, alongside the probabilistic classifier $\pi_{\boldsymbol{\theta}}(\mathbf{x})$, represented by the *solid gray* curve. Finally, the maximizer of the classifier is represented by the *vertical solid green* line—this denotes the location to be evaluated in the next iteration.

F. Properties of the classification problem

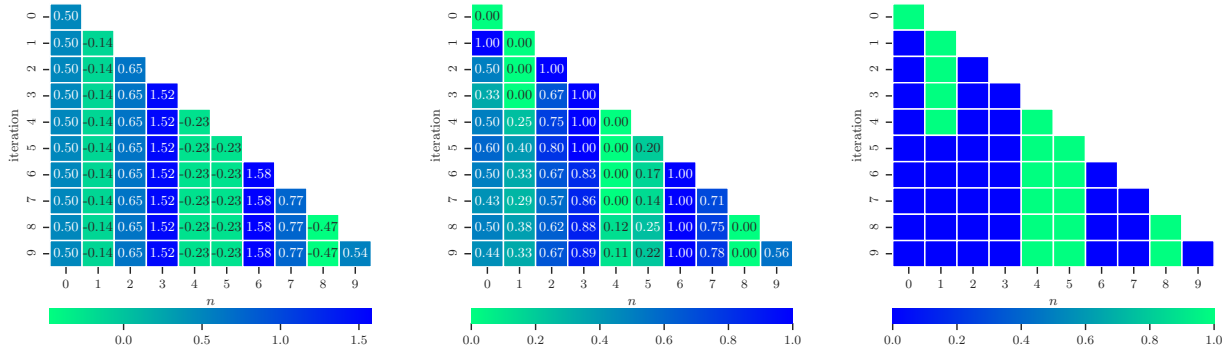
We outline some notable properties of the BORE classification problem as alluded to in § 3 (main paper).

Class imbalance. By construction, this problem has class balance rate γ.

Label changes across iterations. Assuming the proportion γ is fixed across iterations, then, in each iteration, we are guaranteed the following changes:

1. a new input and its corresponding output $(\mathbf{x}_N, y_N)$ will be added to the dataset, thus
2. creating a shift in the rankings and, by extension, quantiles of the observed y values, in turn
3. leading to the binary label of *at most* one instance to flip.

Therefore, between consecutive iterations, changes to the classification dataset are fairly incremental. This property can be exploited to make classifier training more efficient, especially in families of classifiers for which re-training entirely from scratch in each iteration may be superfluous and wasteful. See Figure 2 for an illustrated example.



(a) Continuous targets y_n (b) Empirical distribution $\Phi(y_n)$ (c) Binary labels z_n

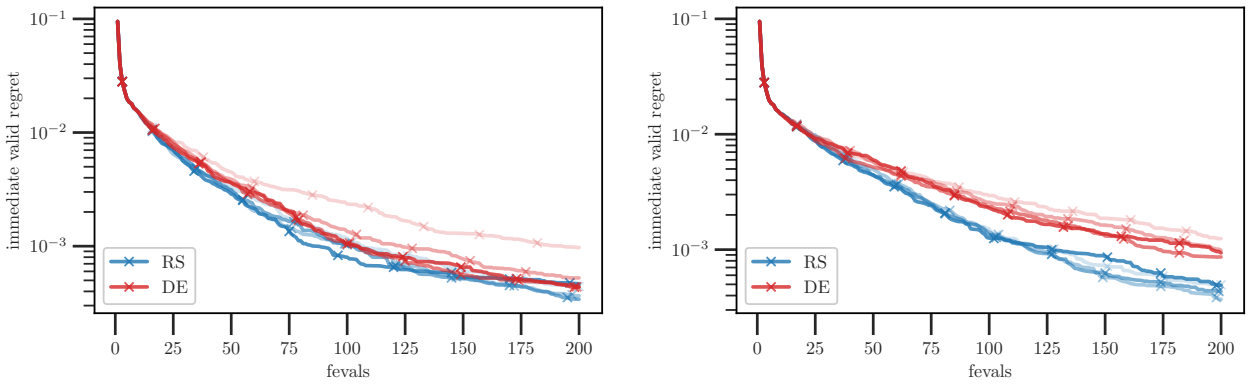
Figure 2. Optimizing a "noise-only" synthetic function $f(x) = 0$ with observation noise $\varepsilon \sim \mathcal{N}(0, 1)$. The proportion is set to $\gamma = 1/4$. As we iterate through the BO loop from top to bottom, the array of targets grows from left to right. Suffice it to say, in each iteration the size of the array increases by one, resulting in a re-shuffling of the rankings and, by extension, quantiles. This in turn leads to the label for *at most* one instance to flip. Hence, between consecutive iterations, changes to the classification dataset are fairly incremental. This property can be exploited to make classifier training more efficient in each iteration.

Some viable strategies for reducing per-iteration classier learning overhead may include speeding up convergence by (i) *importance sampling* (e.g. re-weighting new samples and those for which the label have flipped), (ii) *early-stopping* (stop training early if either the loss or accuracy have not changed for some number of epochs) and (iii) *annealing* (decaying the number of epochs or batch-wise training steps as optimization progresses).

G. Ablation studies

G.1. Maximizing the acquisition function

We examine different strategies for maximizing the acquisition function (i.e. the classifier) in the tree-based variants of BORE, namely, BORE-RF and BORE-XGB. Decision trees are difficult to maximize since their response surfaces are discontinuous and nondifferentiable. Hence, we consider the following methods: random search (RS) and differential evolution (DE). For each method, we further consider different evaluation budgets, i.e. limits on the number of evaluations of the acquisition function. Specifically, we consider the limits $50, 100, 200$, and 500.



(a) BORE-RF on CIFAR-10

(b) BORE-XGB on CIFAR-10

Figure 3. A comparison of various acquisition optimization strategies on the NASBench201 problem.

In Figures 3a and 3b, we show the results of BORE-RF and BORE-XGB, respectively, on the CIFAR-10 dataset of the NASBench201 benchmark, as described in Appendix K.2. Each curve represents the mean across 100 repeated runs. The opacity is proportional to the function evaluation limit, with the most transparent having the lowest limit and the most opaque having the highest limit. We find that RS appears to outperform DE by a narrow margin. Additionally, for DE, a higher evaluation limit appears to be somewhat beneficial, while the opposite holds for RS.

G.2. Effects of calibration

XGBOOST. We apply the calibration approaches (Niculescu-Mizil & Caruana, 2005) we considered in § 5 to XGBOOST for the BORE-XGB variant, namely, Platt scaling (Platt et al., 1999) and isotonic regression (Zadrozny & Elkan, 2001; 2002). As before, the results shown in Figure 4 seem to suggest that these calibration methods have deleterious effects—at least when considering optimization problems which require only a small number of function evaluations to reach the global minimum, since this yields a small dataset with which to calibrate the probabilistic classifier, making it susceptible to overfitting.

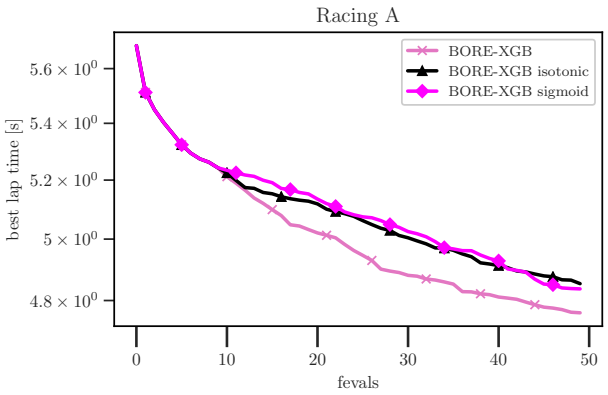


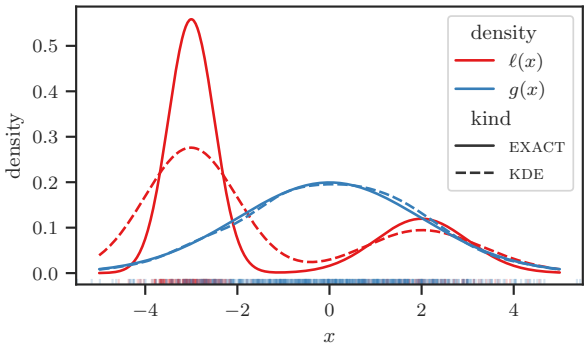
Figure 4. Effects of calibrating XGBOOST in the BORE-XGB variant. Results of racing line optimization on the UC BERKELEY track.

H. Toy example: relative density-ratio estimation by probabilistic classification

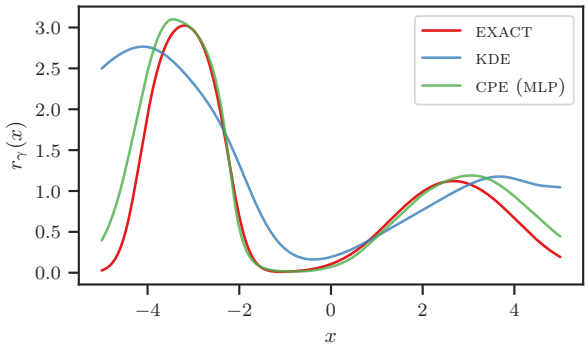
Consider the following toy example where the densities $\ell(x)$ and $g(x)$ are *known* and given exactly by the following (mixture of) Gaussians,

$$\ell(x) \coloneqq 0.3\mathcal{N}(2, 1^2) + 0.7\mathcal{N}(-3, 0.5^2), \qquad \text{and} \qquad g(x) \coloneqq \mathcal{N}(0, 2^2),$$

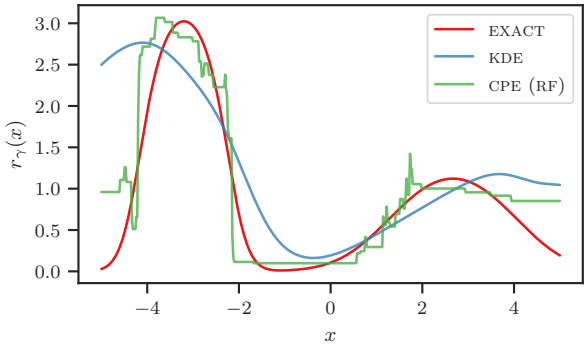
as illustrated by the *solid red* and *blue* lines in Figure 5a, respectively.



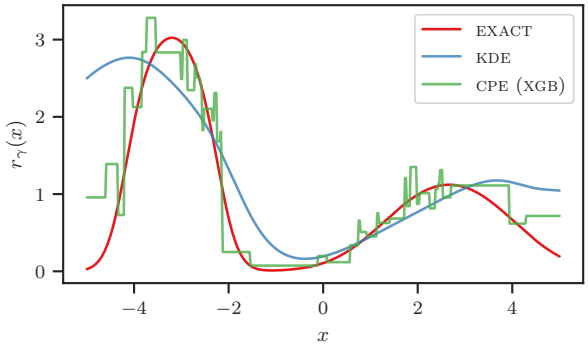
(a) Densities $\ell(x)$ and $g(x)$.



(b) Relative density-ratio, estimated with an MLP classifier.



(c) Relative density-ratio, estimated with a RF classifier.



(d) Relative density-ratio, estimated with an XGBOOST classifier.

Figure 5. Synthetic toy example with (mixtures of) Gaussians.

We draw a total of $N = 1000$ samples from these distributions, with a fraction $\gamma = 1/4$ drawn from $\ell(x)$ and the remainder from $g(x)$. These are represented by the vertical markers along the bottom of the x-axis (a so-called "rug plot"). Then, two kernel density estimations (KDEs), shown with *dashed* lines, are fit on these respective sample sets, with kernel bandwidths selected according to the "normal reference" rule-of-thumb. We see that, for both densities, the modes are recovered well, while for $\ell(x)$, the variances are overestimated in both of its mixture components. As we shall see, this has deleterious effects on the resulting density ratio estimate.

In Figure 5b, we represent the *true* relative density-ratio with the *red* line. The estimate resulting from taking the ratio of the KDEs is shown in *blue*, while that of the class-probability estimation (CPE) method described in § 3 (main paper) is shown in *green*. In this subfigure, the probabilistic classifier consists of a simple multi-layer perceptron (MLP) with 3 hidden layers, each with and 32 units and `elu` activations. In Figures 5c and 5d, we show the same, but with random forest (RF) and XGBOOST classifiers.

The CPE methods appear, at least visually, to recover the exact density ratios well, whereas the KDE method does so quite poorly. Perhaps the more important quality to focus on, for the purposes of BO, is the *mode* of the density-ratio functions. In the case of the KDE method, we can see that this deviates significantly from that of the true density-ratio. In this instance, even though KDE fit $g(x)$ well and recovered the modes of $\ell(x)$ accurately, a slight overestimation of the variance in the latter led to a significant shift in the maximum of the resulting density-ratio functions.

I. Implementation of Baselines

The software implementations of the baseline methods considered in our comparisons are described in Table 1.

Table 1. Implementations of baseline methods.

Method	Software Library	URL (`github.com/*`)	Notes
TPE	HyperOpt	`hyperopt/hyperopt`	
SMAC	SMAC3	`automl/SMAC3`	
GP-BO	AutoGluon	`awslabs/autogluon`	in `autogluon.searcher.GPFIFOSearcher`
DE	-	–	custom implementation
RE	NASBench-101	`automl/nas_benchmarks`	in `experiment_scripts/run_regularized_evolution.py`

J. Experimental Set-up and Implementation Details

Hardware. In our experiments, we employ `m4.xlarge` AWS EC2 instances, which have the following specifications:

- **CPU:** Intel(R) Xeon(R) E5-2676 v3 (4 Cores) @ 2.4 GHz
- **Memory:** 16GiB (DDR3)

Software. Our method is implemented as a *configuration generator* plug-in in the HpBandSter library of Falkner et al. (2018). The code will be released as open-source software upon publication.

The implementations of the classifiers on which the proposed variants of BORE are based are described in Table 2.

Table 2. Implementations of classifiers.

Model	Software Library	URL
Multi-layer perceptron (MLP)	Keras	`https://keras.io`
Random forest (RF)	scikit-learn	`https://scikit-learn.org`
Gradient-boosted trees (XGBOOST)	XGBoost	`https://xgboost.readthedocs.io`

We set out with the aim of devising a practical method that is not only agnostic to the of choice classifier, but also robust to underlying implementation details—down to the choice of algorithmic settings. Ideally, any instantiation of BORE should work well out-of-the-box without the need to tweak the sensible default settings that are typically provided by software libraries. Therefore, unless otherwise stated, we emphasize that made no effort was made to adjust any settings and all reported results were obtained using the defaults. For reproducibility, we explicitly enumerate them in turn for each of the proposed variants.

J.1. BORE-RF

We limit our description to the most salient hyperparameters. We do not deviate from the default settings which, at the time of this writing, are:

- *number of trees* – 100
- *minimum number of samples required to split an internal node* (`min_samples_split`) – 2
- *maximum depth* – unspecified (nodes are expanded until all leaves contain less than `min_samples_split` samples)

J.2. BORE-XGB

- *number of trees (boosting rounds)* – 100
- *learning rate* (η) – 0.3
- *minimum sum of instance weight (Hessian) needed in a child* (`min_child_weight`) – 1
- *maximum depth* – 6

J.3. BORE-MLP

In the BORE-MLP variant, the classifier is a MLP with 2 hidden layers, each with 32 units. We consistently found `elu` activations (Clevert et al., 2015) to be particularly effective for lower-dimensional problems, with `relu` remaining otherwise the best choice. We optimize the weights with ADAM (Kingma & Ba, 2014) using batch size of $B = 64$. For candidate suggestion, we optimize the input of the classifier wrt to its output using multi-started L-BFGS with three random restarts.

Epochs per iteration. To ensure the training time on BO iteration N is nonincreasing as a function of N, instead of directly specifying the number of epochs (i.e. full passes over the data), we specify the number of (batch-wise gradient) steps S to train for in each iteration. Since the number of steps per epoch is $M = \lceil N/B \rceil$, the effective number of epochs on the N-th BO iteration is then $E = \lfloor S/M \rfloor$. For example, if $S = 800$ and $B = 64$, the number of epochs for iteration $N = 512$ would be $E = 100$. As another example, for all $0 < N \leq B$ (i.e. we have yet to observe enough data to fill a batch), we have $E = S = 800$. See Figure 6 for a plot of the effective number of epochs against iterations for different settings of batch size B and number of steps per epoch S. Across all our experiments, we fix $S = 100$.

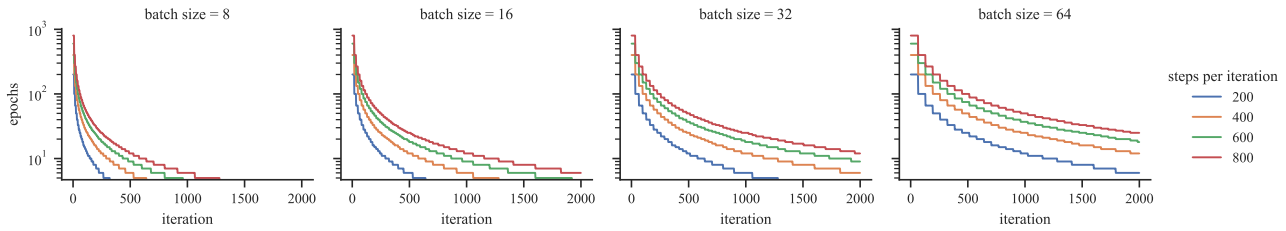


Figure 6. Effective number of epochs on the nth iteration for different settings of batch size B and number of steps per epoch S.

K. Details of Benchmarks

K.1. HPOBench

The hyperparameters for the HPOBench problem and their ranges are summarized in Table 3. All hyperparameters are

Table 3. Configuration space for HPOBench.

HYPERPARAMETER		RANGE
INITIAL LEARNING RATE (LR)		$\{5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-1}\}$
LR SCHEDULE		{`cosine`, `fixed`}
BATCH SIZE		$\{2^3, 2^4, 2^5, 2^6\}$
LAYER 1	WIDTH	$\{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$
	ACTIVATION	{`relu`, `tanh`}
	DROPOUT RATE	$\{0.0, 0.3, 0.6\}$
LAYER 2	WIDTH	$\{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$
	ACTIVATION	{`relu`, `tanh`}
	DROPOUT RATE	$\{0.0, 0.3, 0.6\}$

discrete—either ordered or unordered. All told, there are $6 \times 2 \times 4 \times 6 \times 2 \times 3 \times 6 \times 2 \times 3 = 66,208$ possible combinations. Further details on this problem can be found in (Klein & Hutter, 2019).

K.2. NASBench201

The hyperparameters for the HPOBench problem and their ranges are summarized in Table 4. The operation associated

Table 4. Configuration space for NASBench-201.

HYPERPARAMETER	RANGE
ARC 0	{none, skip-connect, conv-1 × 1, conv-3 × 3, avg-pool-3 × 3}
ARC 1	{none, skip-connect, conv-1 × 1, conv-3 × 3, avg-pool-3 × 3}
ARC 2	{none, skip-connect, conv-1 × 1, conv-3 × 3, avg-pool-3 × 3}
ARC 3	{none, skip-connect, conv-1 × 1, conv-3 × 3, avg-pool-3 × 3}
ARC 4	{none, skip-connect, conv-1 × 1, conv-3 × 3, avg-pool-3 × 3}
ARC 5	{none, skip-connect, conv-1 × 1, conv-3 × 3, avg-pool-3 × 3}

with each of the $\binom{4}{2} = 6$ arcs can belong to one of five categories. Hence, there are $5^6 = 15,625$ possible combinations of hyperparameter configurations. Further details on this problem can be found in (Dong & Yang, 2020).

K.3. Robot pushing control

This problem is concerned with tuning the controllers of two robot hands, with the goal of each pushing an object to some prescribed goal location $\mathbf{p}_g^{(1)}$ and $\mathbf{p}_g^{(2)}$, respectively. Let $\mathbf{p}_s^{(1)}$ and $\mathbf{p}_s^{(2)}$ denote the specified starting positions, and $\mathbf{p}_f^{(1)}$ and $\mathbf{p}_f^{(2)}$ the final positions (the latter of which are functions of the control parameters $\mathbf{x}$). The reward is defined as

$$R(\mathbf{x}) := \underbrace{\|\mathbf{p}_g^{(1)} - \mathbf{p}_s^{(1)}\| + \|\mathbf{p}_g^{(2)} - \mathbf{p}_s^{(2)}\|}_{\text{initial distances}} - \underbrace{(\|\mathbf{p}_g^{(1)} - \mathbf{p}_f^{(1)}\| + \|\mathbf{p}_g^{(2)} - \mathbf{p}_f^{(2)}\|)}_{\text{final distances}},$$

which effectively quantifies the amount of progress made toward pushing the objects to the desired goal. For each robot, the control parameters include the location and orientation of its hands, the pushing speed, moving direction and push duration. These parameters and their ranges are summarized in Table 5.

Table 5. Configuration space for the robot pushing control problem.

HYPERPARAMETER		RANGE
ROBOT 1	POSITION x	$[-5, 5]$
	POSITION y	$[-5, 5]$
	ANGLE θ	$[0, 2\pi]$
	VELOCITY v_x	$[-10, 10]$
	VELOCITY v_y	$[-10, 10]$
	PUSH DURATION	$[2, 30]$
	TORQUE	$[-5, 5]$
ROBOT 2	POSITION x	$[-5, 5]$
	POSITION y	$[-5, 5]$
	ANGLE θ	$[0, 2\pi]$
	VELOCITY v_x	$[-10, 10]$
	VELOCITY v_y	$[-10, 10]$
	PUSH DURATION	$[2, 30]$
	TORQUE	$[-5, 5]$

Further details on this problem can be found in (Wang et al., 2018). This simulation is implemented with the Box2D library, and the associated code repository can be found at `https://github.com/zi-w/Ensemble-Bayesian-Optimization`.

K.4. Racing line optimization

This problem is concerned with finding the optimal racing line. Namely, given a racetrack and a vehicle with known dynamics, the task is to determine a trajectory around the track for which the minimum time required to traverse it is minimal.

We adopt the set-up of Jain & Morari (2020), who consider 1:10 and 1:43 scale miniature remote-controlled cars traversing tracks at UC Berkeley (Liniger et al., 2015) and ETH Zürich (Rosolia & Borrelli, 2019), respectively.

The trajectory is represented by a cubic spline parameterized by the 2D coordinates of D waypoints, each placed at locations along the length of the track, where the ith waypoint deviates from the centerline of the track by $x_i \in \left[-\frac{W}{2}, \frac{W}{2}\right]$, for some track width W. Hence, the parameters are the distances by which each waypoint deviates from the centerline, $\mathbf{x} = [x_1 \cdots x_D]^\top$.

Our blackbox function of interest, namely, the minimum time to traverse a given trajectory, is determined by the solution to a convex optimization problem involving partial differential equations (PDEs) (Lipp & Boyd, 2014). Further details on this problem can be found in (Jain & Morari, 2020), and the associated code repository can be found at `https://github.com/jainachin/bayesrace`.

L. Parameters, hyperparameters, and meta-hyperparameters

We explicitly identify the parameters $\boldsymbol{\omega}$, hyperparameters $\boldsymbol{\theta}$, and meta-hyperparameters $\boldsymbol{\lambda}$ in our approach, making clear their distinction, examining their roles in comparison with other methods and discuss their treatment.

Table 6. A taxonomy of parameters, hyperparameters, and meta-hyperparameters.

	BO with Gaussian processes (GPs)	BO with Bayesian neural networks (BNNs)	BORE with neural networks (NNs)
Meta-hyperparameters $\boldsymbol{\lambda}$	kernel family, kernel isotropy (ARD), etc.	layer depth, widths, activations, etc.	prior precision α, likelihood precision β, layer depth, widths, activations, etc.
Hyperparameters $\boldsymbol{\theta}$	kernel lengthscale and amplitude, ℓ and σ, likelihood precision β	prior precision α, likelihood precision β	weights $\mathbf{W}$, biases $\mathbf{b}$
Parameters $\boldsymbol{\omega}$	None $\varnothing$ (nonparametric)	weights $\mathbf{W}$, biases $\mathbf{b}$	None $\varnothing$ (by design)

L.1. Parameters

Since we seek to *directly* approximate the acquisition function, our method is by design free of parameters $\boldsymbol{\omega}$. By contrast, in classical BO, the acquisition function is derived from the analytical properties of the posterior predictive $p(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathcal{D}_N)$. To compute this, the uncertainty about parameters $\boldsymbol{\omega}$ must be marginalized out

$$p(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathcal{D}_N) = \int p(y \mid \mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\theta}) p(\boldsymbol{\omega} \mid \boldsymbol{\theta}, \mathcal{D}_N) \, \mathrm{d}\boldsymbol{\omega}, \quad \text{where} \quad p(\boldsymbol{\omega} \mid \boldsymbol{\theta}, \mathcal{D}_N) = \frac{p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\omega}, \boldsymbol{\theta}) p(\boldsymbol{\omega} \mid \boldsymbol{\theta})}{p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})}. \tag{3}$$

While GPs are free of parameters, the latent function values $\mathbf{f}$ must be marginalized out

$$p(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathcal{D}_N) = \int p(y \mid \mathbf{x}, \mathbf{f}, \boldsymbol{\theta}) p(\mathbf{f} \mid \boldsymbol{\theta}, \mathcal{D}_N) \, \mathrm{d}\mathbf{f}.$$

In the case of GP regression, this is easily computed by applying straightforward rules of Gaussian conditioning. Unfortunately, few other models enjoy this convenience.

Case study: BNNs. As a concrete example, consider BNNs. The parameters $\boldsymbol{\omega}$ consist of the weights $\mathbf{W}$ and biases $\mathbf{b}$ in the NN, while the hyperparameters $\boldsymbol{\theta}$ consist of the prior and likelihood precisions, α and β, respectively. In general, $p(\boldsymbol{\omega} \mid \mathcal{D}_N, \boldsymbol{\theta})$ is not analytically tractable.

- To work around this, DNGO (Snoek et al., 2015) and ABLR (Perrone et al., 2018) both constrain the parameters $\boldsymbol{\omega}$ to include the weights and biases of only the *final* layer, $\mathbf{W}_L$ and $\mathbf{b}_L$, and relegate those of all preceding layers, $\mathbf{W}_{1:L-1}$ and $\mathbf{b}_{1:L-1}$, to the hyperparameters $\boldsymbol{\theta}$. This yields an exact (Gaussian) expression for $p(\boldsymbol{\omega} \mid \mathcal{D}_N, \boldsymbol{\theta})$ and $p(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathcal{D}_N)$. To treat the hyperparameters, Perrone et al. (2018) estimate $\mathbf{W}_{1:L-1}$, $\mathbf{b}_{1:L-1}$, α and β using type-II maximum likelihood estimation (MLE), while Snoek et al. (2015) use a combination of type-II MLE and slice sampling (Neal, 2003).

- In contrast, BOHAMIANN (Springenberg et al., 2016) makes no such simplifying distinctions regarding the layer weights and biases. Consequently, they must resort to sampling-based approximations of $p(\boldsymbol{\omega} \mid \boldsymbol{\theta}, \mathcal{D}_N)$, in their case by adopting stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014).

In both approaches, compromises needed to be made in order to negotiate the computation of $p(\boldsymbol{\omega} \mid \boldsymbol{\theta}, \mathcal{D}_N)$. This is not to mention the problem of computing the posterior over hyperparameters $p(\boldsymbol{\theta} \mid \mathcal{D}_N)$, which we discuss next. In contrast, BORE avoids the problems associated with computing the posterior predictive $p(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathcal{D}_N)$ and, by extension, posterior $p(\boldsymbol{\omega} \mid \boldsymbol{\theta}, \mathcal{D}_N)$ of eq. 3. Therefore, such compromises are simply unnecessary.

L.2. Hyperparameters

For the sake of notational simplicity, we have thus far not been explicit about how the acquisition function depends on the hyperparameters $\boldsymbol{\theta}$ and how they are handled. We first discuss generically how hyperparameters $\boldsymbol{\theta}$ are treated in BO. Refer to (Shahriari et al., 2015) for a full discussion. In particular, we rewrite the EI function, expressed in eq. 1 (main paper), to explicitly include $\boldsymbol{\theta}$

$$\alpha_\gamma(\mathbf{x}; \boldsymbol{\theta}, \mathcal{D}_N) \coloneqq \mathbb{E}_{p(y \mid \mathbf{x}, \boldsymbol{\theta}, \mathcal{D}_N)}[U(\mathbf{x}, y, \tau)].$$

Marginal acquisition function. Ultimately, one wishes to maximize the *marginal* acquisition function $A_\gamma(\mathbf{x}; \mathcal{D}_N)$, which marginalizes out the uncertainty about the hyperparameters,

$$A_\gamma(\mathbf{x}; \mathcal{D}_N) = \int \alpha_\gamma(\mathbf{x}; \mathcal{D}_N, \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}_N) \, \mathrm{d}\boldsymbol{\theta} \quad \text{where} \quad p(\boldsymbol{\theta} \mid \mathcal{D}_N) = \frac{p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{y} \mid \mathbf{X})}.$$

This consists of an expectation over the posterior $p(\boldsymbol{\theta} \mid \mathcal{D}_N)$ which is, generally speaking, analytically intractable. In practice, the most straightforward way to compute $A_\gamma(\mathbf{x}; \mathcal{D}_N)$ is to approximate the posterior using a delta measure centered at some point estimate $\hat{\boldsymbol{\theta}}$, either the type-II MLE $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ or the maximum *a posteriori* (MAP) estimate $\hat{\boldsymbol{\theta}}_{\text{MAP}}$. This leads to

$$A_\gamma(\mathbf{x}; \mathcal{D}_N) \simeq \alpha_\gamma(\mathbf{x}; \mathcal{D}_N, \hat{\boldsymbol{\theta}}).$$

Suffice it to say, sound uncertainty quantification is paramount to guiding exploration. Since point estimates fail to capture uncertainty about hyperparameters $\boldsymbol{\theta}$, it is often beneficial to turn instead to Monte Carlo (MC) estimation (Snoek et al., 2012)

$$A_\gamma(\mathbf{x}; \mathcal{D}_N) \simeq \frac{1}{S} \sum_{s=1}^{S} \alpha_\gamma(\mathbf{x}; \mathcal{D}_N, \boldsymbol{\theta}^{(s)}), \qquad \boldsymbol{\theta}^{(s)} \sim p(\boldsymbol{\theta} \mid \mathcal{D}_N).$$

Marginal class-posterior probabilities. Recall that the likelihood of our model is

$$p(z \mid \mathbf{x}, \boldsymbol{\theta}) \coloneqq \text{Bernoulli}(z \mid \pi_{\boldsymbol{\theta}}(\mathbf{x})),$$

or more succinctly $\pi_{\boldsymbol{\theta}}(\mathbf{x}) = p(z = 1 \mid \mathbf{x}, \boldsymbol{\theta})$. We specify a prior $p(\boldsymbol{\theta})$ on hyperparameters $\boldsymbol{\theta}$ and marginalize out its uncertainty to produce our analog to the marginal acquisition function

$$\Pi(\mathbf{x}; \mathcal{D}_N) = \int \pi_{\boldsymbol{\theta}}(\mathbf{x}) p(\boldsymbol{\theta} \mid \mathcal{D}_N) \, \mathrm{d}\boldsymbol{\theta}, \quad \text{where} \quad p(\boldsymbol{\theta} \mid \mathcal{D}_N) = \frac{p(\mathbf{z} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbf{z} \mid \mathbf{X})}.$$

As in the generic case, we are ultimately interested in maximizing the marginal class-posterior probabilities $\Pi(\mathbf{x}; \mathcal{D}_N)$. However, much like $A_\gamma(\mathbf{x}; \mathcal{D}_N)$, the marginal $\Pi(\mathbf{x}; \mathcal{D}_N)$ is analytically intractable in turn due to the intractability of $p(\boldsymbol{\theta} \mid \mathcal{D}_N)$. In this work, we focus on minimizing the log loss of eq. 9 (main paper), which is proportional to the negative log-likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \log p(z_n \mid \mathbf{x}_n, \boldsymbol{\theta}) \propto -\log p(\mathbf{z} \mid \mathbf{X}, \boldsymbol{\theta}).$$

Therefore, we're effectively performing the equivalent of type-II MLE,

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \log p(\mathbf{z} \mid \mathbf{X}, \boldsymbol{\theta}).$$

In the interest of improving exploration and, of particular interest in our case, calibration of class-membership probabilities, it may be beneficial to consider MC and other approximate inference methods (Blundell et al., 2015; Gal & Ghahramani, 2016; Lakshminarayanan et al., 2016). This remains fertile ground for future work.

L.3. Meta-hyperparameters

In the case of BORE-MLP, the meta-hyperparameters might consist of, e.g. layer depth, widths, activations, etc—the tuning of which is often the reason one appeals to BO in the first place. For improvements in calibration, and therefore sample diversity, it may be beneficial to marginalize out the uncertainty about these, or considering some approximation thereof, such as hyper-deep ensembles (Wenzel et al., 2020).

References

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622. PMLR, 2015.

Chen, T., Fox, E., and Guestrin, C. Stochastic Gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, pp. 1683–1691, 2014.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Dong, X. and Yang, Y. Nas-bench-102: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

Falkner, S., Klein, A., and Hutter, F. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pp. 1050–1059. PMLR, 2016.

Jain, A. and Morari, M. Computing the racing line using Bayesian optimization. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 6192–6197. IEEE, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klein, A. and Hutter, F. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*, 2019.

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

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Liniger, A., Domahidi, A., and Morari, M. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

Lipp, T. and Boyd, S. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6): 1297–1311, 2014.

Neal, R. M. Slice sampling. *Annals of Statistics*, pp. 705–741, 2003.

Niculescu-Mizil, A. and Caruana, R. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 625–632, 2005.

Perrone, V., Jenatton, R., Seeger, M. W., and Archambeau, C. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, pp. 6845–6855, 2018.

Platt, J. et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74, 1999.

Rosolia, U. and Borrelli, F. Learning how to autonomously race a car: a predictive control approach. *IEEE Transactions on Control Systems Technology*, 28(6):2713–2719, 2019.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25:2951–2959, 2012.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pp. 2171–2180, 2015.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems*, pp. 4134–4142, 2016.

Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pp. 745–754. PMLR, 2018.

Wenzel, F., Snoek, J., Tran, D., and Jenatton, R. Hyperparameter ensembles for robustness and uncertainty quantification. *arXiv preprint arXiv:2006.13570*, 2020.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. Relative density-ratio estimation for robust distribution comparison. In *Advances in Neural Information Processing Systems*, pp. 594–602, 2011.

Zadrozny, B. and Elkan, C. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *ICML*, volume 1, pp. 609–616. Citeseer, 2001.

Zadrozny, B. and Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 694–699, 2002.