Stochastic Gradient Ascent for Rollout Bayesian Optimization

Darian Nwankwo
Department of Computer Science
Cornell University
Ithaca, NY 14850
don4@cornell.edu

David Bindel
Department of Computer Science
Cornell University
Ithaca, NY 14850
bindel@cornell.edu

Abstract

Several strategies for myopic Bayesian optimization have been proposed where the immediate reward is maximized. Myopia isn't inherently bad; rather than under-weighting future consequences, we ignore them altogether. Non-myopic Bayesian optimization aims to resolve these issues by using "lookahead" algorithms that maximize a reward over a finite horizon. In this work, we provide a novel formulation for constructing non-myopic heuristics using well-tested myopic heuristics as building blocks. Our formulation creates a family of non-myopic acquisition functions that is highly parametric; the choice of "base acquisition function" and horizon creates this familial space.

1 Introduction

When making decisions that involve uncertainty and expensive operations—from finding optimal hyper-parameters in deep neural networks to mining oil—we want to minimize the time it takes to find an optimal decision. Given these circumstances, Bayesian optimization (BO) is a set of methods that performs well. Some of the most promising applications of Bayesian optimization use myopic strategies—performing the best with what is immediately present while ignoring its impact on future decision making.

In some settings, however, we want to behave non-myopically [27]. It has been shown that behaving non-myopically may produce solutions quicker than when behaving myopically. A quicker solution means less expensive operations—this is our primary goal in the first place. Non-myopic formulations tend to explore the search space better than greedy myopic formulations, so the benefit extends to applications in active learning as well. Philosophically, behaving optimally in the short-term proves to be naive in some settings.

Behaving non-myopically often suffers from the curse of dimensionality. Extending the greedy method of behaving optimally in the short-term to the long-term quickly produces a program that is computationally intractable. We need a formulation that considers the impact on future decision making while being computational tractable. Non-myopic formulations tend to be computationally tractable in a few circumstances:

1. When the number of dimensions is extremely low, say ≤ 2; or
2. The data-generating process takes more time to get data from than our algorithm takes to converge.

Non-myopic formulations suffer from the curse of dimensionality; as we consider the impact of more subsequent decisions, our problem space increases exponentially. Oftentimes, solutions are

Preprint. Under review.

approximate dynamic programs and the heuristics used to solve them are approximations to the underlying value function.

This paper aims to make non-myopic BO more practical; In particular, our main contributions are:

- We compute rollout acquisition functions via quasi-Monte Carlo integration and use variance reduction techniques to decrease the estimation error.
- We introduce a trajectory-based formulation for deriving non-myopic acquisition functions using myopic functions—or any heuristic—as base heuristics.
- We provide a way to differentiate rollout acquisition functions given a differentiable base policy.

2 Background and Related Work

The development of non-myopic Bayesian optimization has received a lot of attention over the past few years (include references). A lot of this research is about rollout, where future realizations of BO are simulated over a finite horizon h using a GP model and averaged to determine the acquisition function. Rollout acquisition functions represent state-of-the-art in BO and are integrals over h dimensions, where the integrand itself is evaluated through inner optimizations, resulting in an expensive integral. The rollout acquisition function is then maximized to determine the next BO evaluation, further increasing the cost. This large computational overhead has been observed by Osborn et al.[20], who are only able to compute rollout acquisition for horizon 2, dimension 1. Lam et al. [15], who use Gauss-Hermite quadrature in horizons up to five saw runtimes on the order of hours for small, synthetic functions [2].

Recent work focuses on making rollout more practical. Wu and Frazier [2] consider horizon 2, using a combination of Gauss-Hermite quadrature [17] and Monte Carlo (MC) integration to quickly calculate the acquisition function and its gradient. Non-myopic active learning also uses rollout [3; 8; 9; 13] and recent work develops a fast implementation by truncating the horizon and selecting a batch of points to collect future rewards[8; 9].

2.1 Gaussian process regression and Bayesian optimization

Consider the problem of seeking a global minimum of a continuous objective $f(\mathbf{x})$ over a compact set $\Omega \subseteq \mathbb{R}$. If $f(\mathbf{x})$ is expensive to evaluate, then finding a minimum should be sample-efficient. BO typically uses a Gaussian process (GP) to model $f(\mathbf{x})$ from the data $\mathcal{D}_n = \{(\mathbf{x}^i, y_i) : 0 \leq i \leq n, i \in \mathbb{N}\}$. The next evaluation location $\mathbf{x}^{n+1}$ is determined by maximizing an acquisition function $\alpha(\mathbf{x} \mid \mathcal{D}_n)$:

$$\mathbf{x}^{n+1} = \arg\max_{\Omega} \alpha(\mathbf{x} \mid \mathcal{D}_n).$$

We place a GP prior on $f(\mathbf{x})$, denoted by $f \sim \mathcal{GP}(\mu, k)$, where $\mu : \Omega \to \mathbb{R}$ and $k : \Omega \times \Omega \to \mathbb{R}$ are the mean and covariance function, respectively. Here, k is a kernel that correlates points in our sample space and it typically contains hyper-parameters—like a lengthscale factor—that are learned to improve the quality of the approximation [23]. Given $\mathcal{D}_n$, we define the following for convenience of representation:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \ \mathbf{k}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}^1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}^n) \end{bmatrix}, \ K = \begin{bmatrix} \mathbf{k}(\mathbf{x}^1)^T \\ \vdots \\ \mathbf{k}(\mathbf{x}^n)^T \end{bmatrix}.$$

We assume each observation y_i is tainted with Gaussian white noise: $y_i = f(\mathbf{x}^i) + \epsilon_i$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Given a GP prior and data $\mathcal{D}_n$, the resulting posterior distribution for function values at a location $\mathbf{x}$ is the Normal distribution $\mathcal{N}(\mu^{(n)}(\mathbf{x} \mid \mathcal{D}_n),\ K^{(n)}(\mathbf{x}, \mathbf{x} \mid \mathcal{D}_n))$:

$$\mu^{(n)}(\mathbf{x}| \ \mathcal{D}_n) = \mu(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T (K + \sigma^2 I_n)^{-1} (\mathbf{y} - \mu(\mathbf{x})) \tag{1}$$

$$K^{(n)}(\mathbf{x}, \mathbf{x}|\ \mathcal{D}_n) = K(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T (K + \sigma^2 I_n)^{-1} \mathbf{k}(\mathbf{x}) \tag{2}$$

where $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix.

2.2 Non-myopic Bayesian optimization

Non-myopic BO frames the exploration-exploitation problem as a balance of immediate and future rewards. Lam et al.[15] formulate non-myopic BO as a finite horizon dynamic program; the equivalent Markov decision process follows.

The notation used is standard in Puterman[21]: an MDP is a collection $(T, \mathbb{S}, \mathbb{A}, P, R)$. Where $T = \{0, 1, \ldots, h-1\}$, and $h < \infty$ is the set of decision epochs, assumed finite for our problem. The state space, $\mathbb{S}$, encapsulates all the information needed to model the system from time $t \in T$. Where $\mathbb{A}$ is the action space; given a state $s \in \mathbb{S}$ and an action $a \in \mathbb{A}$, $P(s'|s, a)$ is the transition probability of the next state being s'. $R(s, a, s')$ is the reward received for choosing action a in state s, and ending in state s'.

A decision rule, $\pi_t : \mathbb{S} \to \mathbb{A}$, maps states to actions at time t. A policy π is a series of decision rules $\pi = (\pi_0, \pi_1, \ldots, \pi_{h-1})$, one at each decision epoch. Given a policy π, a starting state s_0, and horizon h, we can define the expected total reward $V_h^\pi(s_0)$ as:

$$V_h^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{h-1} R(s_t, \pi_t(s_t), s_{t+1})\right].$$

Since our sequence of decisions is formulated as an MDP, our objective is to find the optimal policy π^* that maximizes the expected total reward, i.e., $sup_{\pi \in \Pi} V_h^\pi(s_0)$, where Π is the space of all admissible policies.

If we can sample from the transition probability P, we can estimate the expected total reward of any base policy—the decisions made using the base acquisition function—$\hat{\pi}$ with MC integration (site Sutton RL Book):

$$V_h^{\hat{\pi}}(s_0) \approx \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{t=0}^{h-1} R(s_t^i, \hat{\pi}_t(s_t^i), s_{t+1}^i)\right].$$

Given a GP prior over data $\mathcal{D}_t$ with mean $\mu^{(t)}$ and covariance matrix $K^{(t)}$, we model h steps of BO as an MDP. This MDP's state space is all possible data sets reachable from starting-state $\mathcal{D}_t$ with h steps of BO. Its action space is Ω; actions correspond to sampling a point in Ω. Its transition probability and reward function are defined as follows. Given an action x^{t+1}, the transition probability from $\mathcal{D}_t$ to $\mathcal{D}_{t+1}$, where $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{x}^{t+1}, y_{t+1})\}$ is:

$$P(\mathcal{D}_t, \mathbf{x}^{t+1}, \mathcal{D}_{t+1}) \sim \mathcal{N}(\mu^{(t)}(\mathbf{x}^{t+1}; \mathcal{D}_t), K^{(t)}(\mathbf{x}^{t+1}, \mathbf{x}^{t+1}; \mathcal{D}_t)).$$

Thus, the transition probability from $\mathcal{D}_t$ to $\mathcal{D}_{t+1}$ is the probability of sampling y_{t+1} from the posterior $\mathcal{GP}(\mu^{(t)}, K^{(t)})$ at $\mathbf{x}^{t+1}$. We define a reward according to expected improvement (EI) (reference D. R. Jones 1998). Let f_t^* be the minimum observed value in the observed set $\mathcal{D}_t$, i.e., $f_t^* = \min\{y_0, \ldots, y_t\}$. Then our reward is expressed as follows:

$$R(\mathcal{D}_t, \mathbf{x}^{t+1}, \mathcal{D}_{t+1}) = (f_t^* - f_{t+1})^+ \equiv \max(f_t^* - f_{t+1}, 0).$$

EI can be defined as the optimal policy for horizon one, obtained by maximizing the immediate reward:

$$\pi_{EI} = \arg\max_{\pi} V_1^\pi(\mathcal{D}_t) = \arg\max_{\mathbf{x}^{t+1} \in \Omega} \mathbb{E}\left[(f_t^* - f_{t+1})^+\right] \equiv \arg\max_{\mathbf{x}^{t+1} \in \Omega} EI(\mathbf{x}^{t+1}|\mathcal{D}_t),$$

where the starting state is $\mathcal{D}_t$—our initial samples. We define the non-myopic policy, however, as the optimal solution to an h-horizon MDP. The expected total reward of this MDP can be expressed as:

$$V_h^\pi(\mathcal{D}_n) = \mathbb{E}\left[\sum_{t=n}^{n+h-1} R(\mathcal{D}_t, \pi_t(\mathcal{D}_t), \mathcal{D}_{t+1})\right] = \mathbb{E}\left[\sum_{t=n}^{n+h-1} (f_t^* - f_{t+1})^+\right].$$

When $h > 2$, the optimal policy is difficult to compute.

2.3 Rollout acquisition functions

Rollout policies are sub-optimal approximations to our MDP program, yet yield promising results; they are tractable alternatives to optimal policies. For a given state $\mathcal{D}_n$, we denote our base policy $\tilde{\pi} = (\tilde{\pi}_0, \tilde{\pi}_1, \ldots, \tilde{\pi}_h)$. We let $\mathcal{D}_n$ denote the initial state of our MDP and $\mathcal{D}_{n,k}$ for $0 \leq k \leq h$ to denote the random variable that is the state at each decision epoch. Each individual decision rule $\tilde{\pi}_k$ consists of maximizing the base acquisition function $\bar{\alpha}$ given the current state $s_k = \mathcal{D}_{n,k}$, i.e.

$$\tilde{\pi}_k = \arg\max_{\mathbf{x}\in\Omega} \bar{\alpha}\left(\mathbf{x}|\mathcal{D}_{n,k}\right).$$

Using this policy, we define the non-myopic acquisition function $\alpha_h(\mathbf{x})$ as the rollout of $\tilde{\pi}$ to horizon h, i.e. the expected reward of $\tilde{\pi}$ starting with action $\tilde{\pi}_0 = \mathbf{x}$:

$$\alpha_h\left(\mathbf{x}^{n+1}\right) \coloneqq \mathbb{E}\left[V_h^{\tilde{\pi}}\left(\mathcal{D}_n \cup \left\{\left(\mathbf{x}^{n+1}, y_{n+1}\right)\right\}\right)\right],$$

where y_{n+1} is the noisy observed value of f at $\mathbf{x}^{n+1}$. Thus, as is the case with any acquisition function, the next BO evaluation is:

$$\mathbf{x}^{n+1} = \arg\max_{\mathbf{x}\in\Omega} \alpha_h(\mathbf{x}).$$

Rollout is tractable and conceptually straightforward, however, it is still computationally demanding. To rollout $\tilde{\pi}$ once, we must do h steps of BO with $\bar{\alpha}$. Many of the aforementioned rollouts must then be averaged to reasonably estimate α_h, which is an h-dimensional integral. Estimation can be done either through explicit quadrature or MC integration, and is the primary computational bottleneck of rollout. Our paper... (list contributions)

3 Models and methods

3.1 Model

We build our intuition behind our approach from a top-down perspective. We've seen that non-myopic Bayesian optimization is promising, though tends to be computationally intractable. To circumvent this problem, we formulate a sub-optimal approximation to solve the intractable dynamic program; namely, a rollout acquisition function. Though relatively more tractable, rollout acquisition functions can be computationally burdensome. We're interested in solving $x^* = \arg\max_{x\in\mathcal{X}} \alpha(x)$ where

$$\alpha(x) = \mathbb{E}_{\hat{f}\sim\mathcal{G}}\left[\alpha\big(x|\tau(h, x, \bar{\alpha}, \hat{f})\big)\right] \approx \frac{1}{N}\sum_{i=1}^{N}(f^* - \big(t^i(x)\big)^-)^+ \tag{3}$$

and $t^i(x) \sim \tau(h, x, \bar{\alpha}, \hat{f})$ are sample trajectories, soon to be defined. Unfortunately, derivative-free optimization in high-dimensional spaces is expensive, so we'd like estimates of $\nabla\alpha(x)$. In particular, differentiating with respect to x yields the following:

$$\nabla_x\left[\alpha(x)\right] \approx \nabla_x\left[\frac{1}{N}\sum_{i=1}^{N}(f^* - \big(t^i(x)\big)^-)^+\right] = \frac{1}{N}\sum_{i=1}^{N}\left(-\nabla_x\left[\big(t^i(x)\big)^-\right]\right)^+ \tag{4}$$

which requires some notion of differentiating sample trajectories. Thus, our problem can be expressed as two interrelated optimizations:

1. An inner optimization for determining sample trajectories $t^i(x) \sim \tau(h, x, \bar{\alpha}, \hat{f})$
2. An outer optimization for determining our next query location x^*

In what follows, we define how to compute/differentiate sample trajectories and how to differentiate the rollout acquisition function.

We introduce the following notation to distinguish between two GP sequences that must be maintained. Suppose we have function values and gradients denoted as follows: $(\hat{f}_0, \nabla\hat{f}_0), \dots (\hat{f}_h, \nabla\hat{f}_h)$. We define two GP sequences as:

$$\begin{aligned}\mathcal{F}_h \;&\sim\; \mathcal{F}_0 | \mathcal{F}_h(x^0) = \hat{f}_0, \dots, \mathcal{F}_h(x^h) = \hat{f}_h \\ \mathcal{G}_h \;&\sim\; \mathcal{F}_0 | \mathcal{F}_h(x^0) = \hat{f}_0, \dots, \mathcal{F}_h(x^h) = \hat{f}_h, \\ &\qquad \nabla\mathcal{F}_h(x^0) = \nabla\hat{f}_0, \dots, \nabla\mathcal{F}_h(x^h) = \nabla\hat{f}_h.\end{aligned}$$

Our model, fundamentally, relies on the distinction amongst known sample locations $\{\mathbf{x}^{-j} \in \mathbb{R}^d \mid 1 \leq j \leq m\}$, deterministic start location $\{\mathbf{x}^0 \in \mathbb{R}^d\}$, and the stochastic fantasized sample locations $\{\mathbf{x}^j \in \mathbb{R}^d \mid 1 \leq j \leq h\}$. Alternatively, this can be visualized as follows:

$$X^{m+h+1} := \left[\mathbf{x}^{-m} \;\dots\; \mathbf{x}^{-1} |\; \mathbf{x}^0 |\; \mathbf{x}^1 \;\dots\; \mathbf{x}^h\right] \in \mathbb{R}^{d \times (m+h+1)}.$$

Moreover, we collect the m known samples into $y = \left[y_{-m} \;\dots\; y_{-1}\right]^T$. The distinction between negative, null, and positive superscripts serves a useful purpose. Negative superscripts can be thought of as the past; potentially noisy observations that are fixed and immutable. The null superscripts denotes our freedom of choice; we are not bound to start at a specific location. Positive superscripts can be thought of as our ability to see into the future; things we anticipate on observing given our current beliefs.

Our problem is how to choose $\mathbf{x}^0$ to maximize our expected reward over some finite horizon h. We focus on developing an h-step expected improvement (EI) rollout policy that involves choosing $\mathbf{x}^0$ by using Stochastic Gradient Ascent (SGA) to optimize our rollout acquisition function.

An h-step EI *rollout* policy involves choosing $\mathbf{x}^0$ based on the anticipated behavior of the EI ($\bar{\alpha}$) algorithm starting from $\mathbf{x}^0$ and proceeding for h steps. That is, we consider the iteration

$$\mathbf{x}^r = \arg\max_x \bar{\alpha}(\mathbf{x} \mid \mathcal{F}_{r-1}),\; 1 \leq r \leq h \tag{5}$$

where the trajectory relations are

$$\nabla_x \bar{\alpha}(\mathbf{x}^r | \mathcal{F}_{r-1}) = \mathbf{0} \;\wedge\; \left(\hat{f}_r, \nabla\hat{f}_r\right) \sim \mathcal{G}_{r-1}(\mathbf{x}^r).$$

Trajectories are fundamentally random variables; their behavior is determined by the rollout horizon h, start location $\mathbf{x}$, base policy $\bar{\alpha}$, and surrogate model $\mathcal{G}$—denoted as $\tau(h, \mathbf{x}, \bar{\alpha}, \hat{f})$. We denote sample draws $t^i \sim \tau(h, \mathbf{x}, \bar{\alpha}, \mathcal{G})$ as follows:

$$t^i = \left(\left(x^j, \hat{f}_j^i(x^j), \nabla\hat{f}_j^i(x^j)\right)\right)_{j=0}^{h} \tag{6}$$

Computing this sample path, however, requires we solve the iteration defined above. We also use the notation $t^i_{j,k}$ to denote the k-th element of the j-th 3-tuple associated with the i-th sample. Now that we have some provisional notation, we define how to evaluate $t^i(x)$ as follows:

$$(t^i(x))^- \equiv \min(t^i(x)) \equiv \min_{0 \leq j \leq h} t^i_{j,2} \tag{7}$$

If we let $b = \arg\min_j t^i_{j,2}$, we can rewrite the minimum of the sample trajectory as follows:

$$(t^i(x))^- \equiv \hat{f}_b^i(x^b(x)). \tag{8}$$

Now, we're able to differentiate trajectories given by the following

$$\nabla_x \left[(t^i(x))^-\right] = \left[\tfrac{\partial}{\partial x_1}\left[(t^i(x))^-\right], \;\dots,\; \tfrac{\partial}{\partial x_d}\left[(t^i(x))^-\right]\right]^T \tag{9}$$

which requires us to compute $\frac{\partial}{\partial x_k}\left[(t^i(x))^-\right]$ where $1 \leq k \leq d$. Hence, we have

$$\frac{\partial}{\partial x_k}\left[(t^i(x))^-\right] = \frac{\partial}{\partial x_k}\left[\hat{f}_b(x^b(x))\right]. \tag{10}$$

Since the best function value $\hat{f}_b$ found is a function of the best location x^b, which is also a function of x, we have

$$\frac{\partial}{\partial x_k}\left[\hat{f}_b(x^b(x))\right] = \frac{\partial \hat{f}_b}{\partial x^b}\frac{\partial x^b}{\partial x_k}. \tag{11}$$

This captures how the best value found thus far changes as we vary the k-th dimension of x which is equivalently expressed as

$$\nabla_x\left[\hat{f}_b(x^b(x))\right] = \left(\hat{f}_b'(x^b(x)) \cdot \frac{\partial x^b}{\partial x}\right)^T. \tag{12}$$

From the trajectory relations defined above, we're able to derive the following relationship via the Implicit Function Theorem about x^r:

$$\frac{\partial \bar{\alpha}}{\partial x^r}(x^r|\mathcal{F}_{r-1}) = \mathbf{0} \rightarrow \frac{\partial^2 \bar{\alpha}}{(\partial x^r)^2}\frac{\partial x^r}{\partial x} + \frac{\partial^2 \bar{\alpha}}{\partial x^r \partial \mathcal{F}_{r-1}}\frac{\partial \mathcal{F}_{r-1}}{\partial x} = \mathbf{0}.$$

We're able to compute the Jacobian matrix $J(x^b) = \frac{\partial x^b}{\partial x}$ as

$$\frac{\partial x^k}{\partial x} = -\left(\frac{\partial^2 \bar{\alpha}}{(\partial x^k)^2}\right)^{-1}\left(\frac{\partial^2 \bar{\alpha}}{\partial x^k \partial \mathcal{F}_{k-1}}\frac{\partial \mathcal{F}_{k-1}}{\partial x}\right) \tag{13}$$

defining all of the computations necessary to solve the inner and outer optimization problem.

3.2 Methods

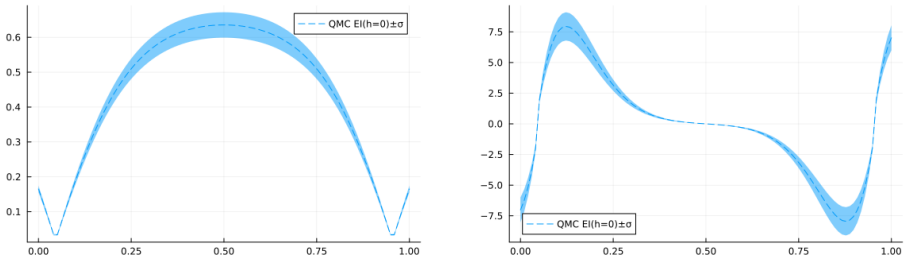

Figure 1: A demonstration of computing the rollout acquisition function and it's gradients for the trivial horizon $h = 0$.

Our objective $x^* = \arg\max_{x \in \mathcal{X}} \alpha(x)$ is fundamentally a stochastic program. We make use of a combination of quasi-monte carlo integration, common random numbers, and control variates.

Compared to traditional Monte Carlo integration, which uses random samples, quasi-Monte Carlo integration uses low-discrepancy sequences to generate sample points that are more evenly distributed across the integration domain. This results in a more efficient and accurate estimation of the integral.

One of the challenges of using quasi-Monte Carlo integration in Bayesian optimization is that the estimator can still have a high variance due to the presence of noise in the objective function. To address this, we make use of common random numbers, which involves using the same set of random numbers across different iterations of the algorithm. This helps to reduce the variance of the estimator and improve the overall performance of the algorithm.

In addition to common random numbers, we also make use of control variates to further reduce the variance of the estimator. Control variates involve introducing a known function as a control variable, which is correlated with the objective function. By using the control variable in conjunction with

the objective function, we can reduce the variance of the estimator and improve the accuracy of the estimation.

Overall, by combining quasi-Monte Carlo integration, common random numbers, and control variates, we can improve the efficiency and accuracy of our stochastic estimates of the acquisition function and it's gradient.

4 Experiments and discussion

In this section, we present a series of experiments designed to validate the effectiveness of our proposed approach for non-myopic Bayesian optimization. We compare our method to several myopic baselines across various synthetic problems. Throughout our experiments we use a GP with the Matérn 5/2 ARD kernel and learn its hyperparameters via maximum likelihood estimation. When rolling out acquisition functions, we maximize using Stochastic Gradient Ascent with expected improvement used as the base policy.

Table 1: Mean and median gap G over 40 initial guesses.

Function name		PI	EI	UCB	α_0	α_1
Branin-Hoo	Mean	0.847	0.818	0.848	0.861	0.846
	Median	0.922	0.909	0.91	0.983	0.909

5 Conclusion

We investigated developing non-myopic strategies for Bayesian optimization that balances exploration against exploitation in a principled manner by incorporating derivative information to reduce the computational burden of the outer optimization. To further enhance the efficiency of our approach, we focused on specifying a differentiable policy for the inner optimization, as the availability of derivative information streamlines the optimization process, making non-myopic Bayesian optimization more computationally tractable and practical for real-world applications.

References

[1] P. I. Frazier. A Tutorial on Bayesian Optimization. in(Section 5):1–22, 2018.

[2] P. I. Frazier. Practical Two-Step Look-Ahead Bayesian Optimization. (NeurIPS), 2019.

[3] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. Mann. Bayesian optimal active search and surveying. *arXiv preprint arXiv:1206.6406*, 2012.

[4] D. Ginsbourger and R. Le Riche. Towards Gaussian Process-based Optimization with Finite Time Horizon. (Umr 5146):89–96, 2010.

[5] J. González, M. Osborne, and N. D. Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, 41:790–799, 2016.

[6] J. C. Goodson, B. W. Thomas, and J. W. Ohlmann. A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, 258(1):216–229, 2017.

[7] R. R. Griffiths and J. M. Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chemical Science*, 11(2):577–586, 2020.

[8] S. Jiang, G. Malkomes, G. Converse, A. Shofner, B. Moseley, and R. Garnett. Efficient nonmyopic active search. In *International Conference on Machine Learning*, pages 1714–1723. PMLR, 2017.

[9] S. Jiang, G. Malkomes, M. Abbott, B. Moseley, and R. Garnett. Efficient nonmyopic batch active search. *Advances in Neural Information Processing Systems*, 31, 2018.

[10] S. Jiang, D. R. Jiang, M. Balandat, B. Karrer, J. R. Gardner, and R. Garnett. Efficient nonmyopic bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS), 2020.

[11] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions," , vol. 13, no. 4, pp. 455-492, 1998. *Journal of Global Optimization*, 13:455–492, 1998.

[12] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 1996.

[13] A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th international conference on Machine learning*, pages 449–456, 2007.

[14] R. R. Lam and K. E. Willcox. Lookahead Bayesian optimization with inequality constraints. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):1891–1901, 2017.

[15] R. R. Lam, K. E. Willcox, and D. H. Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, (Nips):883–891, 2016.

[16] E. H. Lee, D. Eriksson, B. Cheng, M. McCourt, and D. Bindel. Efficient Rollout Strategies for Bayesian Optimization. 2020.

[17] Q. Liu and D. A. Pierce. A note on gauss-hermite quadrature. *Biometrika*, 81(3):624–629, 1994. ISSN 00063444. URL `http://www.jstor.org/stable/2337136`.

[18] A. Mahajan and D. Teneketzis. *Multi-Armed Bandit Problems*. Number February 2014. 2008.

[19] J. Mockus. *Bayesian Approach to Global Optimization*. Springer, 1989.

[20] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimization. *3rd International Conference on Learning and Intelligent Optimization LION3*, (x):1–15, 2009.

[21] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[22] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[23] C. E. Rasmussen and W. C. K. I. *Gaussian processes for machine learning*. MIT Press, 2006.

[24] S. Sethi and G. Sorger. A theory of rolling horizon decision making. *Annals of Operations Research*, 29 (1):387–415, 1991.

[25] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 4:2951–2959, 2012.

[26] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[27] X. Yue and R. A. Kontar. Why Non-myopic Bayesian Optimization is Promising and How Far Should We Look-Ahead? A Study via Rollout. 108, 2019.

Appendix

A Kernels

The kernel functions used in this paper is the Matérn 5/2 kernel:

$$K_{5/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{5}}{\ell} + \frac{5}{3\ell^2}\right) \exp(-\frac{\sqrt{5}r}{\ell})$$

B Expected Improvement

$$\begin{aligned}
\alpha(x) &= \sigma(x) g(z(x)) \\
g(z) &= z\Phi(z) + \phi(z) \\
z(x) &= \sigma(x)^{-1} \left[\mu(x) - f^+ - \xi\right]
\end{aligned}$$

C Differentiating α

Differentiating with respect to spatial coordinates:

$$\begin{aligned}\alpha_{,i} &= \sigma_{,i} g(z) + \sigma g'(z) z_{,i} \\ \alpha_{,ij} &= \sigma_{,ij} g(z) + \sigma_{,i} g'(z) z_{,j} + \sigma_{,j} g'(z) z_{,i} + \sigma g'(z) z_{,ij} + \sigma g''(z) z_{,i} z_{,j} \\ &= \sigma_{,ij} g(z) + [\sigma_{,i} z_{,j} + \sigma_{,j} z_{,i} + \sigma z_{,ij}] g'(z) + \sigma g''(z) z_{,i} z_{,j}\end{aligned}$$

Mixed derivative with respect to spatial coordinates and data and hypers:

$$\dot{\alpha}_{,i} = \dot{\sigma}_{,i} g(z) + \sigma_{,i} g'(z) \dot{z} + \dot{\sigma} g'(z) z_{,i} + \sigma g''(z) \dot{z} z_{,i} + \sigma g'(z) \dot{z}_{,i}$$

Finally, we differentiate $g(z) = z\Phi(z) + \phi(z)$, noting that $\phi'(z) = -z\phi(z)$ and $\Phi'(z) = \phi(z)$. This gives

$$\begin{aligned}g(z) &= z\Phi(z) + \phi(z) \\ g'(z) &= \Phi(z) + z\phi(z) + \phi'(z) = \Phi(z) \\ g''(z) &= \phi(z).\end{aligned}$$

D Differentiating z

Now consider $z = \sigma^{-1}[\mu - f^+ - \xi]$. As before, we begin with spatial derivatives:

$$\begin{aligned}z_{,i} &= -\sigma^{-2}\sigma_{,i}[\mu - f^+ - \xi] + \sigma^{-1}\mu_{,i} \\ &= \sigma^{-1}\left[\mu_{,i} - \sigma_{,i} z\right] \\ z_{,ij} &= -\sigma^{-2}\sigma_{,j}\left[\mu_{,i} - \sigma_{,i} z\right] + \sigma^{-1}\left[\mu_{,ij} - \sigma_{,ij} z - \sigma_{,i} z_{,j}\right] \\ &= \sigma^{-1}\left[\mu_{,ij} - \sigma_{,ij} z - \sigma_{,i} z_{,j} - \sigma_{,j} z_{,i}\right]\end{aligned}$$

Now we differentiate with respect to data and hypers:

$$\begin{aligned}\dot{z} &= -\sigma^{-2}\dot{\sigma}[\mu - f^+ - \xi] + \sigma^{-1}[\dot{\mu} - \dot{f}^+ - \dot{\xi}] \\ &= \sigma^{-1}[\dot{\mu} - \dot{f}^+ - \dot{\xi} - \dot{\sigma} z] \\ \dot{z}_{,i} &= \sigma^{-1}[\dot{\mu}_{,i} - \dot{\sigma}_{,i} z - \dot{\sigma} z_{,i} - \sigma_{,i} \dot{z}]\end{aligned}$$

E Differentiating σ

The predictive variance is

$$\sigma^2 = k_{xx} - k_{xX} K_{XX}^{-1} k_{Xx}.$$

Differentiating the predictive variance twice in space — assuming k_{xx} is independent of x by stationarity — gives us

$$\begin{aligned}2\sigma\sigma_{,i} &= -2k_{xX,i} K_{XX}^{-1} k_{Xx} = -2k_{xX,i} d \\ 2\sigma_{,i}\sigma_{,j} + 2\sigma\sigma_{,ij} &= -2k_{xX,ij} K_{XX}^{-1} k_{Xx} - 2k_{xX,i} K_{XX}^{-1} k_{Xx,j} \\ &= -2k_{xX,ij} d - 2k_{xX,i} w^{(j)}\end{aligned}$$

Rearranging to get spatial derivatives of σ on their own gives us

$$\begin{aligned}\sigma_{,i} &= -\sigma^{-1} k_{xX,i} d \\ \sigma_{,ij} &= -\sigma^{-1}\left[k_{xX,ij} d + k_{xX,i} w^{(j)} + \sigma_{,i}\sigma_{,j}\right].\end{aligned}$$

Differentiating with respect to data (and locations) and kernel hypers requires more work. First, note that

$$\begin{aligned}2\sigma\dot{\sigma} &= \dot{k}_{xx} - 2\dot{k}_{xX} K_{XX}^{-1} k_{Xx} + k_{xX} K_{XX}^{-1} \dot{K}_{XX} K_{XX}^{-1} k_{Xx} \\ &= \dot{k}_{xx} - 2\dot{k}_{xX} d + d^T \dot{K}_{XX} d\end{aligned}$$

Now, differentiating σ^{-1} with respect to data and hypers gives

$$\begin{aligned}\dot{\sigma}_{,i} &= \sigma^{-2}\dot{\sigma} k_{xX,i} K_{XX}^{-1} k_{Xx} - \sigma^{-1}\left[\dot{k}_{xX,i} K_{XX}^{-1} k_{Xx} + k_{xX,i} K_{XX}^{-1} \dot{k}_{Xx} - k_{xX} K_{XX}^{-1} \dot{K}_{XX} K_{XX}^{-1} k_{Xx}\right] \\ &= -\sigma^{-1}\left[\dot{\sigma}\sigma_{,i} + \dot{k}_{xX,i} d + (w^{(i)})^T \dot{k}_{Xx} - d^T \dot{K}_{XX} d\right]\end{aligned}$$

F Differentiating μ

Let K_{XX} denote the kernel matrix, and k_{Xx} the column vector of kernel evaluations at x. The posterior mean function for the GP (assuming a zero-mean prior) is

$$\mu = k_{xX}c$$

where $K_{XX}c = y$. Note that c does not depend on x, but it does depend on the data and hyperparameters.

Differentiating in space is straightforward, as we only invoke the kernel derivatives:

$$\begin{aligned}\mu_{,i} &= k_{xX,i}c \\ \mu_{,ij} &= k_{xX,ij}c\end{aligned}$$

Differentiating in the data and hyperparameters requires that we also differentiate through a matrix solve:

$$\dot{\mu} = \dot{k}_{xX}K_{XX}^{-1}y + k_{xX}K_{XX}^{-1}\dot{y} - k_{xX}K_{XX}^{-1}\dot{K}_{XX}K_{XX}^{-1}y.$$

Defining $d = K_{XX}^{-1}k_{Xx}$, we have

$$\dot{\mu} = \dot{k}_{xX}c + d^T(\dot{y} - \dot{K}_{XX}c).$$

Now differentiating in space and defining $K_{XX}^{-1}k_{Xx,i}$ as $w^{(i)}$, we have

$$\dot{\mu}_{,i} = \dot{k}_{xX,i}c + (w^{(i)})^T(\dot{y} - \dot{K}_{XX}c).$$

G Differentiating kernels

We assume the kernel has the form $k(x, y) = \psi(\rho)$ where $\rho = \|r\|$ and $r = x - y$. Recall that

$$\begin{aligned}\rho &= \sqrt{r_k r_k} \\ \rho_{,i} &= \rho^{-1}r_k r_{k,i} = \rho^{-1}r_i \\ \rho_{,ij} &= \rho^{-1}\delta_{ij} - \rho^{-2}r_i\rho_{,j} \\ &= \rho^{-1}\left[\delta_{ij} - \rho^{-2}r_i r_j\right]\end{aligned}$$

Applying this together with the chain rule yields

$$\begin{aligned}k &= \psi(\rho) \\ k_{,i} &= \psi'(\rho)\rho_{,i} = \psi'(\rho)\rho^{-1}r_i \\ k_{,ij} &= \psi''(\rho)\rho_{,i}\rho_{,j} + \psi'(\rho)\rho_{,ij} \\ &= \left[\psi''(\rho) - \rho^{-1}\psi'(\rho)\right]\rho^{-2}r_i r_j + \rho^{-1}\psi'(\rho)\delta_{ij}.\end{aligned}$$