Bayesian Optimization with Gradients

Jian Wu, Matthias Poloczek, Andrew Gordon Wilson, and Peter I. Frazier
School of Operations Research and Information Engineering
Cornell University
\{jw926,poloczek,andrew,pf98\}@cornell.edu

Abstract

In recent years, Bayesian optimization has proven to be exceptionally successful for global optimization of expensive-to-evaluate multimodal objective functions. However, unlike most optimization methods, Bayesian optimization typically does not make use of derivative information. In this paper we show how Bayesian optimization can exploit derivative information to greatly decrease the number of objective function evaluations required for a good performance. We find derivative information particularly complements the knowledge gradient acquisition function, relative to the popular expected improvement acquisition function. Moreover, we develop a batch Bayesian optimization procedure which exploits noisy and incomplete derivative information, with state-of-the-art performance compared to a wide range of optimization procedures with and without gradients, on several benchmarks as well as kernel learning and k-nearest neighbor applications.

1 Introduction

Bayesian optimization is able to find global optima with a remarkably small number of function evaluations [Brochu et al., 2010, Kleijnen, 2014, Jones et al., 1998]. Bayesian optimization has thus been particularly adopted for automatic hyperparameter tuning of machine learning algorithms [Snoek et al., 2012, Swersky et al., 2013, Gelbart et al., 2014, Gardner et al., 2014], where learning objectives can be extremely expensive to evaluate, noisy, and multimodal.

Bayesian optimization models observations of an objective function as being sampled from a probabilistic distribution over functions, typically given by a Gaussian process (GP).

The objective function, for example, could be predictive performance, and its input could be the hyperparameters of a deep neural network, such as the dropout rate. The goal is then to find the optimal setting of hyperparameters with as few evaluations of the objective as possible, since each evaluation requires retraining all the weights of a neural network, which is a computationally expensive procedure. To decide where to query the objective function next, one uses an acquisition function such as expected improvement [Jones et al., 1998, Huang et al., 2006, Picheny et al., 2013], upper confidence bound [Srinivas et al., 2010], or knowledge gradient [Frazier et al., 2009, Scott et al., 2011], which determines the trade-off between exploration (moving to where the GP has high predictive uncertainty) and exploitation (moving to where the GP has a high predictive mean).

Unlike alternative optimization approaches, Bayesian optimization procedures do not generally leverage derivative information. Derivative information may seemingly be unavailable, or lead to more expensive Gaussian process inference procedures.

On the other hand: (1) The value of gradient information is particularly compelling in the context of Bayesian optimization: one can update an estimate of the optimum using all of the derivative information at previous evaluations of the objective function rather than just local gradient information at the current iteration. (2) In many Bayesian optimization applications it is not GP inference which is the computational bottleneck, but rather evaluating an expensive objective function. Reducing the number of times one needs to query the objective can more than compensate for some additional computational overhead during GP inference. (3) Derivative information is available in many applications, often at little additional cost. For example, in PDE-constrained problems, gradients can be obtained cheaply via adjoint methods [Plessix, 2006, Duffy, 2009]. Moreover, recent work [Maclaurin et al., 2015, Luketina et al., 2015, Fu et al., 2016] makes gradient information available for hyperparameter tuning problems. (4) Even when derivative information is not readily available, we can usually compute approximative derivatives, for instance through finite differences. (5) Moreover, Bayesian optimization procedures can be designed to naturally accommodate noisy or incomplete derivative information.

This paper investigates the use of derivative infor-
We show how Bayesian optimization can efficiently exploit derivative information to greatly decrease the number of objective function evaluations required for good performance.

We show that gradient information naturally complements the knowledge gradient acquisition function, outperforming the popular expected improvement acquisition function, and we provide insights into this result.

We show that random feature expansions [Lázaro-Gredilla et al., 2010] can effectively manage the additional computational expenses of gradient methods.

We show how one can naturally handle noisy and incomplete gradient information.

We propose a batch Bayesian optimization approach which leverages derivative information, and provides substantially improved performance over many alternatives. We compare with state-of-the-art batch Bayesian optimization algorithms with and without derivative information, and the gradient-based optimizer BFGS when full gradients are available.

Our approach is focused on using derivative information for fast convergence, and is more closely related to Lizotte [2008], with several key differences: (i) we consider derivative information with Bayesian optimization in much greater generality; (ii) we explore the effects of gradients with different acquisition functions, showing empirical and theoretical advantages to combining derivatives with the knowledge gradient acquisition function; (iii) we consider how to leverage derivative information efficiently; (iv) we accommodate partial and noisy derivative information; (v) we develop a powerful batch parallel Bayesian optimization method that leverages derivative information and outperforms state-of-the-art alternatives on several benchmarks and consequential applications such as kernel learning.

Recently, several batch Bayesian optimization algorithms have been proposed that in each iteration choose a set of points rather than a single point at which the function is evaluated. Snoek et al. [2012] construct the batch by iteratively adding a point of maximum value under a single point EI criterion, averaged over the posterior distribution of previously selected points. Wang et al. [2015] develop a Monte Carlo method to determine a set of points that maximizes a batch EI criterion. Marmin et al. [2016] proposes a fast way to evaluate the closed-form formula of the batch EI criterion.

Batch acquisition algorithms can also be developed from upper confidence bounds [Contal et al., 2013, Desautels et al., 2014] or entropy search [Shah and Ghahramani, 2015].

Our approach to handling batch observations is most closely related to the batch Knowledge Gradient (KG) of Wu and Frazier [2016], who extended the knowledge gradient policy of Frazier et al. [2009] to the batch setting. We provide a detailed description of this approach in section 3.2.

2 Related Work

Despite its fundamental importance, surprisingly little work has been done to extend or understand Bayesian optimization with derivatives.

Lizotte [2008, Sect. 4.2.1 and Sect. 5.2.4] incorporates derivatives into Bayesian optimization, modelling the derivatives of a Gaussian process as in Rasmussen and Williams [2006, Sect. 9.4]. Lizotte [2008] shows that Bayesian optimization with the expected improvement acquisition and complete gradient information at each sample can outperform BFGS.

Alternatively, Osborne et al. [2009] propose fully Bayesian optimization procedures that use observations of derivatives to improve the conditioning of the Gaussian process covariance matrix. If a sample is taken near a previously observed point, only the derivative information at the new location is used to update the covariance matrix.

Our approach is focused on using derivative information for fast convergence, and is more closely related to Lizotte [2008], with several key differences: (i) we consider derivative information with Bayesian optimization in much greater generality; (ii) we explore the effects of gradients with different acquisition functions, showing empirical and theoretical advantages to combining derivatives with the knowledge gradient acquisition function; (iii) we consider how to leverage derivative information efficiently; (iv) we accommodate partial and noisy derivative information; (v) we develop a powerful batch parallel Bayesian optimization method that leverages derivative information and outperforms state-of-the-art alternatives on several benchmarks and consequential applications such as kernel learning.

3 Batch Knowledge Gradient with Derivatives

In section 3.1 we discuss a general approach to incorporating derivative information into Gaussian processes for Bayesian optimization. We propose an acquisition function based on the knowledge gradient that selects a batch of points to sample in each iteration, utilizing derivative information. This approach is introduced in section 3.2. We then detail how to implement the algorithm efficiently in section 3.4.
3.1 Derivative Information

Given an expensive-to-evaluate function \( f \), our goal is to find an \( \arg\min_{x \in A} f(x) \), where \( A \subset \mathbb{R}^d \) is the domain we optimize over. We place a Gaussian process prior over the function \( f : A \to \mathbb{R} \), which is specified by its mean function \( \mu(\cdot) : A \to \mathbb{R} \) and the kernel function \( K(\cdot, \cdot) : A \times A \to \mathbb{R}_{\geq 0} \), where \( \mathbb{R}_{\geq 0} \) denotes the nonnegative reals. We initially suppose that for each sample we observe the function value and all \( d \) partial derivatives, and then later show how to relax this assumption.

For \( x \in A \) we denote the function value by \( f(x) \) and the gradient by \( \nabla f(x) \). It is convenient to jointly model the function and its gradient via a multi-output Gaussian process with mean function \( \tilde{\mu} \) and kernel function \( \tilde{K} \) defined as follows:

\[
\begin{align*}
\tilde{\mu}(x) &= (\mu(x), \nabla \mu(x))^T, \\
\tilde{K}(x, x') &= \begin{pmatrix} K(x, x') & J(x, x') \\ J(x', x')^T & H(x, x') \end{pmatrix},
\end{align*}
\]

(3.1)

where \( J(x, x') = \left( \frac{\partial K(x, x')}{\partial x_1}, \ldots, \frac{\partial K(x, x')}{\partial x_d} \right) \) and \( H(x, x') \) is the \( d \times d \) Hessian of \( K(x, x') \). Since the gradient is a linear operator, the gradient of a GP is also a GP (see also section 9.4 in Rasmussen and Williams [2006]).

We are particularly interested in the ability of acquisition algorithms to leverage noisy observations of partial derivatives. Accordingly, we suppose that the observations of the function value and the gradient are subject to noise. That is, when evaluating \( f(x) \) at point \( x \), we observe the \((d+1)\)-dimensional vector

\[
y(x) \mid (f(x), \nabla f(x)) \sim \mathcal{N}\left( \begin{pmatrix} f(x) \\ \nabla f(x) \end{pmatrix}, \text{diag}(\sigma^2(x)) \right),
\]

where \( \sigma^2 : A \to \mathbb{R}^{d+1} \) gives the variance of the observational noise at each point for the function value and its \( d \) partial derivatives. Then \( \text{diag}(\sigma^2(x)) \) is the diagonal matrix that gives the variance for each observation, i.e. either of the function \( f \) or of a partial derivative. If \( \sigma^2 \) is not known, we will estimate it from data.

The posterior distribution is again a GP with mean function \( \tilde{\mu}^{(n)}(\cdot) \) and kernel function \( \tilde{K}^{(n)}(\cdot, \cdot) \). Their formulae are given in the appendix for completeness.

To relax the assumption of complete derivatives, we note that if some entries of \( (f(x), \nabla f(x)) \) are not provided, then the remaining values associated with \( x \) still obey the multivariate normal distribution imposed by the GP. Thus, we may simply omit the entries of the mean vector corresponding to outputs of the GP that are not available. Accordingly, we omit the rows and columns of the covariance matrix that correspond to values that were not provided.

3.2 The Acquisition Algorithm \( dKG \)

We extend the batch version of the knowledge gradient of Wu and Frazier [2016] to exploit available derivative information. We refer to this algorithm as derivative-enabled knowledge gradient (\( dKG \)).

The algorithm proceeds iteratively: in each iteration \( dKG \) selects a batch of \( q \) points in \( A \) that has a maximum value of information (VOI). Suppose that we have observed \( n \) points and let \( \tilde{\mu}^{(n)}(x) \) for each \( x \in A \) be the \((d+1)\)-dimensional vector that gives the posterior mean for \( f(x) \) and its \( d \) partial derivatives at \( x \). Note that we showed in Sect. 3.1 how to remove the assumption that all \( d + 1 \) values are provided.

We examine how observing derivative information affects the posterior distribution and the value of information analyses of the knowledge gradient and the expected improvement criteria.

The two topmost plots of Fig. 1 depict the posterior surfaces of a function sampled from a one dimensional...
Gaussian process (without taking into account partial derivatives, on the left-hand side) and after incorporating observations of the full respective gradients at the sample locations (on the right-hand side). We see that the uncertainty is considerably reduced if derivative information is taken into account.

The two plots in the second row illustrate how the acquisition criteria of the knowledge gradient and expected improvement are affected by including derivative information. Here we suppose a batch size of one. Note that EI, KG, and even dEI pick essentially the same location for the next sample, where dKG prefers a different sample.

The plots in the third and fourth row show the posterior surface after observing the next sample chosen by the respective acquisition criterion. We see that the posterior uncertainty is smaller away from the global optimum for the algorithms that utilize the gradient observations than for those that do not. Interestingly, we notice that the knowledge gradient seems to benefit considerably more from derivative information than expected improvement (fourth row): dKG has sampled a point whose observation gives an accurate knowledge of the location of the optimum, while dEI still is forced to make a greedy sampling decision. We will investigate this observation in more detail in our experimental evaluation.

### 3.4 An Efficient Formulation of dKG

Recall that the computation of a batch of maximum value of information is difficult since each evaluation of the objective function $dKG(z^{(1:q)}, A)$ requires an optimal solution to the inner optimization problem in Eq. (3.2) that is stated over the continuous space $A$. To make this problem tractable in practice, we propose a novel discretization that improves over previous approaches. Then we can compute the $dKG$ factor and its gradient over the discrete set, which allows us to find a batch of maximum value of information efficiently via a gradient-based optimizer.

#### An Improved Discretization of $A$.

We discretize the set $A$ in the inner optimization problem stated in Eq. (3.2). How to perform this step is an interesting topic of research itself [Scott et al., 2011]. The discrete set $A_n$ is not chosen statically, but evolves over time. For example, one can draw $M$ samples from the posterior over the global maximizer (please refer to Sect. B and also to Hernández-Lobato et al. [2014], Shah and Ghahramani [2015] for a description of this technique). This sample set, denoted by $A_n^M$, is then extended by the locations of previously sampled points $x^{(1:n)}$ and the set of points $z^{(1:q)}$ whose value of information we wish to compute. Then the inner optimization problem can be restated as

$$dKG(z^{(1:q)}, A_n) = \min_{x \in A_n} \mathbb{E}_n \left[ \min_{y \in z^{(1:q)}} f^T(x) y \right],$$

where $A_n = \bigcup A_n^{(n)} \cup A^{(1:n)} \cup z^{(1:q)}$. For the experimental evaluation we recompute $A_n^{M}$ in every iteration after updating the posterior of the Gaussian process.

#### The Computation of $dKG$ and its Gradient.

Next we show how the $dKG$ factor can be computed efficiently, using the above discretization in the inner optimization problem. Recall that $\hat{K}^{(n)}$ and $\hat{\mu}^{(n)}$ are the kernel and mean function respectively of the posterior after evaluating $n$ points. It is well-known (e.g., see Frazier et al. [2009], Wu and Frazier [2016]) that, conditioned on $z^{(1:q)}$ and the knowledge after $n$ evaluations, $y(z^{(1:q)}) - \hat{\mu}^{(n)}(z^{(1:q)})$ is normally distributed with zero mean and covariance matrix $\hat{K}^{(n)}(z^{(1:q)}, z^{(1:q)}) + \text{diag}(\sigma^2(z^{(1)}), \ldots, \sigma^2(z^{(q)}))$. Recall that $y(z^{(1:q)})$ contains the function value and the $d$ partial derivatives for each of the $q$ points in the batch.

Following Wu and Frazier [2016], we express $\hat{\mu}^{(n+q)}(x)$ as

$$\hat{\mu}^{(n+q)}(x) = \hat{\mu}^{(n)}(x) + \hat{K}^{(n)}(x, z^{(1:q)})$$

$$(\hat{K}^{(n)}(z^{(1:q)}, z^{(1:q)})$$

$$+ \text{diag}(\sigma^2(z^{(1)}), \ldots, \sigma^2(z^{(q)})))^{-1}$$

$$\left(y(z^{(1:q)}) - \hat{\mu}^{(n)}(z^{(1:q)})\right).$$

Thus, we can rewrite $\hat{\mu}^{(n+q)}(x)$ as

$$\hat{\mu}^{(n+q)}(x) = \hat{\mu}^{(n)}(x) + \hat{\sigma}(n, x, z^{(1:q)}) Z_{q(d+1)},$$

where $Z_{q(d+1)}$ is a $q \cdot (d+1)$-dimensional standard normal vector and

$$\hat{\sigma}(n, x, z^{(1:q)}) = \hat{K}^{(n)}(x, z^{(1:q)}) \left(\hat{D}^{(n)}(z^{(1:q)}) \hat{D}^{(n)}(z^{(1:q)})^T\right)^{-1}.$$
Figure 1: The topmost plots show the posterior surfaces of a function sampled from a one dimensional Gaussian process with and without incorporating observations of the gradients. Note that the posterior variance is considerably smaller if the gradients are incorporated. The plots in the second row show the utility of sampling each point under the value of information criteria of KG and EI in both settings. If no derivatives are observed, both KG and EI will query a point with high potential gain (i.e. a small expected function value). On the other hand, when gradients are observed, $dKG$ makes a considerably better sampling decision, whereas $dEI$ samples essentially the same location as $EI$. The plots in the third and fourth row depict the posterior surface after the respective sample. Interestingly, KG benefits more from observing the gradients than EI (fourth row): $dKG$ samples a point whose observation yields an accurate knowledge of the location of the optimum, while $dEI$ still has considerable uncertainty around the optimum.
as
\[
    dKG(z^{(1:q)}, \mathbb{A}^n) = \min_{x \in \mathbb{A}^n} e^T \mu(n)(x) - \mathbb{E}_n \left[ \min_{x \in \mathbb{A}^n} e^T \mu(n+q)(x) \right]
\]
\[
    = \mathbb{E}_{Z \in \mathbb{Q}^{(1+q)}} \left[ \min_{x \in \mathbb{A}^n} e^T \mu(n)(x) - \frac{\partial}{\partial z_{ij}} \left( \mu(n)(x) + \sigma_n(x, z^{(1:q)})Z_{q(d+1)} \right) \right].
\]

Now let
\[
x^{1,*} = \arg\min_{x \in \mathbb{A}^n} e^T \mu(n)(x) \quad \text{and} \quad x^{2,*} = \arg\min_{x \in \mathbb{A}^n} e^T \left( \tilde{\mu}(n)(x) + \sigma_n(x, z^{(1:q)})Z_{q(d+1)} \right),
\]
then the partial derivative of \( dKG(z^{(1:q)}, \mathbb{A}^n) \) with respect to \( z_{ij} \) is
\[
    \frac{\partial dKG(z^{(1:q)}, \mathbb{A}^n)}{\partial z_{ij}} = \mathbb{E}_{Z \in \mathbb{Q}^{(1+q)}} \left[ \frac{\partial}{\partial z_{ij}} e^T \mu(n)(x^{1,*}) - \frac{\partial}{\partial z_{ij}} e^T \left( \tilde{\mu}(n)(x^{1,*}) + \sigma_n(x^{1,*}, z^{(1:q)})Z_{q(d+1)} \right) \right].
\]

where \( z_{ij} \) is the \( j \)-th dimension of \( i \)-th point in \( z^{(1:q)} \). Therefore, we can utilize a multi-start gradient-descent to select the next batch.

4 Experiments

We evaluate the performance of the proposed algorithm \( dKG \) on six standard synthetic benchmarks in Sect. 4.1. Moreover, we examine its ability to tune the hyperparameters for the kernel-weighted k-Nearest Neighbor algorithm (KNN) (see Sect. 4.2) and for a spectral mixture kernel (cp. Sect. 4.3). Note that in the former application not all hyperparameters are differentiable. We compare its performance to state-of-the-art methods in Bayesian optimization:

- the batch expected improvement method (\( EI \)) of Wang et al. [2014] that does not utilize derivative information,
- our extension of the above batch expected improvement method that incorporates derivative information (\( dEI \)), and
- the batch knowledge gradient algorithm without derivative information (\( KG \)) of Wu and Frazier [2016].

Note that all above algorithms can be run even if not all partial derivatives are given. In benchmarks that provide the full gradient, we additionally compare to the gradient-descent method L-BFGS-B provided in \texttt{scipy}. We suppose that the objective function \( f \) is drawn from a Gaussian process \( GP(\mu, \Sigma) \), where \( \mu \) is a constant mean function and \( \Sigma \) is the squared exponential kernel. The hyperparameters of the kernel are obtained via maximum marginal likelihood estimation and updated after every iteration. The parameter \( M \) that determines the number of samples drawn from the posterior over the global maximizer is set to 200 (cp. Sect. 3.4).

The plots for synthetic benchmark functions in Sect. 4.1 we report the immediate regret of the solution that each algorithm would pick as a function of the number of function evaluations. Recall that the immediate regret is defined as the loss with respect to a global optimum. For the other experiments the plots depict the objective value of the solution instead of the immediate regret. The error bars give the mean value plus and minus one standard error. The number of replications varies and is stated in the description of the respective benchmark below. Our method was implemented in C++ with a Python interface, inheriting the open-source implementation of batch EI from the \texttt{Metrics Optimization Engine} [Wang et al., 2014].

4.1 Results on synthetic functions

We evaluate the algorithms on six test functions chosen from Bingham [2015]. In order to demonstrate the ability to benefit from \textit{noisy derivative information}, we suppose an additive normally distributed noise with zero mean and variance \( \sigma^2 = 0.25 \) for both the function value and its partial derivatives. Note that \( \sigma \) is not known to the algorithms but has to be estimated from observations. Moreover, we investigate how the performance of the algorithms is affected if partial derivatives are not given for all parameters. We also experiment with two different batch sizes: batch size \( q = 4 \) is used for the Branin, Rosenbrock, and Ackley functions. Otherwise we use batch size \( q = 8 \).

The experimental results are summarized in Fig. 2.

\textbf{Functions with Full Gradient Information.} For 2d Branin on domain \([-15, 15]^2 \), 5d Ackley on \([-2, 2]^5 \), 6d Hartmann function on \([0, 1]^6 \), we assume that the full gradient is available.

Looking at the results for the Branin function (cp. Fig. 2), \( dKG \) outperforms its competitors after 40 function evaluations and obtains the best solution overall (within the limit of function evaluations). BFGS makes faster progress than the Bayesian optimization methods during the first 20 evaluations; however, it stalls subsequently and fails to obtain a competitive solution. On the Ackley function \( dEI \) makes fast progress during the first 50 evaluations but fails to make any
progress subsequently. dKG requires about 100 evaluations to improve on the performance of dEI; dKG exhibits the best overall performance again. For the Hartmann function dKG clearly dominates its competitors over all function evaluations.

Functions with Incomplete Derivative Information. For the 3d Rosenbrock function on \([-2, 2]^3\) we only provide a noisy observation of the third partial derivative. Both EI and dEI get stuck early. dKG on the other hand finds almost optimal solution after about 50 function evaluations; KG catches up after about 75 evaluations and has a comparable performance afterwards. The 4d Levy benchmark on \([-10, 10]^4\), where the fourth partial derivative is observable with noise, shows a different ordering of the algorithms: here EI has the best performance, beating even its formulation that utilizes derivative information. A possible explanation could be that the smoothness and regularized shape of the function surface benefits this acquisition criterion. For the 8d Cosine mixture function on \([-1, 1]^8\) we provide two noisy partial derivatives. Both KG-based acquisition functions show surprising behavior: their function values actually become worse initially. After about 50 function evaluation both algorithms recover and achieve the best performances, with dKG beating KG slightly.

Summing up, we see that dKG successfully exploits noisy derivative information and has the best overall performance.

4.2 Kernel-weighted k-Nearest Neighbor

Suppose a cab company wishes to predict the duration of trips for its vehicles and customers. Clearly, the duration not only depends on the endpoints of the trip, but also on the day and time. In this benchmark we tune a kernel-weighted k-nearest neighbor (KNN) metric to optimize predictions of these durations, based on historical data. A trip is described by the pick-up time \(t\), the pick-up location \((p_1, p_2)\), and the drop-off point \((d_1, d_2)\). Then the estimate of the duration is obtained as a weighted average over all trips \(D_{m,t}\) in our database that happened in the time interval \(t \pm m\) minutes, where \(m\) is a tunable hyperparameter:

\[
\text{Prediction}(t, p_1, p_2, d_1, d_2) = \frac{\sum_{i \in D_{m,t}} \text{duration}_i \times \text{weight}(i)}{\sum_{i \in D_{m,t}} \text{weight}(i)}.
\]

The weight of trip \(i \in D_{m,t}\) in this prediction is given by

\[
\text{weight}(i) = \exp \left\{ -\frac{(t - t_i)^2}{l_1^2} - \frac{(p_1 - p_{1i})^2}{l_2^2} - \frac{(p_2 - p_{2i})^2}{l_3^2} - \frac{(d_1 - d_{1i})^2}{l_4^2} - \frac{(d_2 - d_{2i})^2}{l_5^2} \right\},
\]

where \((t_i, p_{1i}, p_{2i}, d_{1i}, d_{2i})\) are the respective parameter values for trip \(i\), and \((l_1, l_2, l_3, l_4, l_5)\) are tunable hyperparameters. Thus, we have 6 hyperparameters to tune: \((m, l_1, l_2, l_3, l_4, l_5)\). We choose \(m\) in \([30, 100]\), \(l_1^2\) in \([10^2, 10^4]\), and \(l_2^2, l_3^2, l_4^2, l_5^2\) each in \([10^{-5}, 10^{-2}]\).

We use the yellow cab NYC public data set from June 2016, sampling 4000 records from June 1st to June 25th as training data and 1000 trip records from June 26th to 30th as validation data. Our test criterion is the root mean squared error (RMSE), for which we compute the partial derivatives on the validation dataset with respect to the hyperparameters \((l_1, l_2, l_3, l_4, l_5)\), while the hyperparameter \(m\) is not differentiable.

The experimental results show that our algorithm as well as the batch knowledge gradient perform considerably better than both algorithms based on the expected improvement criterion. For either pair exploiting derivative information provides an advantage. Fig. 3 summarizes the results for batch size \(q = 8\).

![Figure 3: The best function value for the KNN benchmark, averaged over 100 replications.](image)

4.3 Kernel Learning

In this benchmark we examine the performance of the optimization algorithms for a complex kernel learning task. Although we have access to an analytic closed form (marginal likelihood) objective, this objective is (i) expensive to evaluate, (ii) highly multimodal, and (iii) derivative information is available. Thus learning flexible kernel functions is a perfect candidate for our approach.
Figure 2: The average performance of 100 replications (the log10 of the immediate regret vs. the number of function evaluations). For the Branin, Ackley, and Hartmann functions, we assume that a noisy observation of the full gradient is available. On the other functions only one or two partial derivatives can be observed (with noise). dKG performs significantly better than its competitors for all benchmarks except the Levy function.
Spectral mixture kernels [Wilson and Adams, 2013] can be used for flexible kernel learning to enable long-term extrapolation. These kernels are obtained by modeling a spectral density by a mixture of Gaussians. While any stationary kernel can be described by a spectral mixture kernel with a particular setting of its hyperparameters, initializing and learning these parameters can be difficult, due to a highly multimodal marginal likelihood objective.

In this experiment, the task is to train a 3-component spectral mixture kernel on an airline data set used by Wilson and Adams [2013]. We have to determine the mixture weights, means, and variances, for each of the three Gaussians. We run the algorithms with batch size $q = 8$ on this highly multi-modal function. Their performance is summarized in Fig. 4. On this application, BFGS tends to either perform reasonably well, or become trapped in a very bad local optima, depending highly on initialization and human intervention. We have chosen a run where BFGS performs reasonably. $dKG$, on the other hand, can more consistently find a good solution. Here $dKG$ finds the best solution within the step limit. Overall, we observe that gradient information is highly valuable in performing this kernel learning task.

5 Discussion

Bayesian optimization is primarily applied to low dimensional problems where we wish to find a good solution with a very small number of objective function evaluations. We have shown that in this context derivative information can be extremely useful: we can greatly decrease the number of objective function evaluations, especially when using the knowledge gradient acquisition function. Moreover, our approach provides considerably better performance even when derivative information is noisy and only available for some variables. Our batch Bayesian optimization method outperforms state-of-the-art techniques with and without derivatives on a number of common benchmarks in addition to significant kernel learning and k-nearest neighbor applications.

Bayesian optimization is increasingly being used to automate parameter tuning in machine learning, where objective functions can be extremely expensive to evaluate. For example, the parameters could even represent the architecture of a deep neural network. In the future we expect derivative information with Bayesian optimization to enable such promising applications, moving us towards fully automatic and principled approaches to statistical machine learning.

Acknowledgments

Wilson was partially supported by NSF IIS-1563887. Frazier, Poloczek, and Wu were partially supported by NSF CAREER CMMI-1254298, NSF CMMI-1536895, NSF IIS-1247696, AFOSR FA9550-12-1-0200, AFOSR FA9550-15-1-0038, and AFOSR FA9550-16-1-0046.
A The Posterior Distribution of the Multivariate GP

Suppose that we have sampled \( f \) at \( n \) points \( X := \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\} \) so far and observed \( y^{(1:n)} \), where each observation consists of the function value and the gradient at \( x^{(i)} \). Then the posterior distribution is a multivariate Gaussian process with mean function \( \tilde{\mu}^{(n)}(\cdot) \) and kernel function \( \tilde{K}^{(n)}(\cdot, \cdot) \), where

\[
\tilde{\mu}^{(n)}(x) = \tilde{\mu}(x) + \tilde{K}(x, X) \left( \tilde{K}(X, X) + \text{diag}\{\sigma^2(x^{(1)}), \ldots, \sigma^2(x^{(n)})\} \right)^{-1} (y^{(1:n)} - \tilde{\mu}(X)),
\]

\[
\tilde{K}^{(n)}(x_1, x_2) = \tilde{K}(x_1, x_2) - \tilde{K}(x_1, X) \left( \tilde{K}(X, X) + \text{diag}\{\sigma^2(x^{(1)}), \ldots, \sigma^2(x^{(n)})\} \right)^{-1} \tilde{K}(X, x_2).
\]

(A.1)

The rows and columns in Eq. (A.1) corresponding to partial derivatives (or function values) that were not observed are to be omitted.

B Spectral Density Approximation of the Gaussian Process

In this paper, we use random features to approximate a Gaussian process to

- obtain a better discretization of set \( A \) used in the inner optimization problem of \( dKG \) (see Sect. 3.4), and

- improve the scalability of kernel learning, thereby following ideas of [Hernández-Lobato et al., 2014, Lázaro-Gredilla et al., 2010].

Denote by \( s(w) \) the Fourier dual of a stationary kernel function and \( p(w) := s(w)/\alpha \) the associated normalized density, where \( \alpha = \int s(w)dw \). We approximate the Gaussian process with a finite set of \( m \) random features, specifically,

\[
K(x_1, x_2) = \frac{2\alpha}{m} \mathbb{E}_{p(W, b)} \left[ \cos(Wx_1 + b) \cos(Wx_2 + b) \right],
\]

where \( W \) is a \( m \times d \) random matrix with \( W_{ij} \sim p(w) \) and \( b \) is a \( m \times 1 \) random vector with \( b_i \sim \mathcal{U}(0, 2\pi) \) [Hernández-Lobato et al., 2014, Sect. A].

Let \( \Phi(x) = \sqrt{2\alpha/m} \cos(Wx + b) \). We approximate the Gaussian process prior for \( f \) via a Bayesian linear model \( f(x) = \Phi(x)^T\theta \), where \( \theta \sim \mathcal{N}(0, I) \). Conditioned on the collected data, the posterior of the \( \theta \) is multivariate normal with mean and covariance

\[
m = (\Phi^T \Phi + \Sigma)^{-1} \Phi^T y
\]

\[
V = (\Phi^T \Phi + \Sigma)^{-1},
\]

where \( \Sigma = \text{diag}(\sigma^2(x)) \) denotes the variance for observations of function values and partial derivatives (see Sect. 3.1).

To sample from the posterior of the global maxima, we first sample \( m \) random features \( \Phi^{(i)}(x) \) and their corresponding weights \( \theta^{(i)} \), and then construct \( f^{(i)}(x) = \Phi^{(i)}(x)^T \theta^{(i)} \). This is a sample from the approximate posterior of \( f \) conditioned on the data, on which we locate global optima using a gradient-based optimizer (see also Sect. 2.1 in Hernández-Lobato et al. [2014] and Sect. 3.2 in Shah and Ghahramani [2015]).

Recall that we re-estimate the hyperparameters of the kernel regularly as more function observations are obtained. To speed up the kernel learning when the number of samples \( n \) exceeds \( m = 1000 \), we apply the above approximation. Then we seek hyperparameters that maximize \( -\log \text{det}(\Phi\Phi^T + \Sigma) - y^T[\Phi\Phi^T + \Sigma]^{-1}y \). With this approximation, the computation time is \( O(m^2n) \) instead of \( O(n^3) \).
References


W. Scott, P. Frazier, and W. Powell. The correlated knowledge gradient for simulation optimization of


