Discretization-free Knowledge Gradient Methods for Bayesian Optimization

Jian Wu and Peter I. Frazier

School of Operations Research and Information Engineering
Cornell University
{jw926,pf98}@cornell.edu

Abstract

This paper studies Bayesian ranking and selection (R&S) problems with correlated prior beliefs and continuous domains, i.e. Bayesian optimization (BO). Knowledge gradient methods [Frazier et al., 2008, 2009] have been widely studied for discrete R&S problems, which sample the one-step Bayes-optimal point. When used over continuous domains, previous work on the knowledge gradient [Scott et al., 2011, Wu and Frazier, 2016, Wu et al., 2017] often rely on a discretized finite approximation. However, the discretization introduces error, and scales poorly as the dimension of domain grows. In this paper, we develop a fast discretization-free knowledge gradient method for Bayesian optimization. Our method is not restricted to the fully sequential setting, but useful in all settings where knowledge gradient can be used over continuous domains. We show how our method can be generalized to handle (i) batch of points suggestion (parallel knowledge gradient); (ii) the setting where derivative information is available in the optimization process (derivative-enabled knowledge gradient). In numerical experiments, we demonstrate that the discretization-free knowledge gradient method finds global optima significantly faster than previous Bayesian optimization algorithms on both synthetic test functions and real-world applications, especially when function evaluations are noisy; and derivative-enabled knowledge gradient can further improve the performances, even outperforming the gradient-based optimizer such as BFGS when derivative information is available.

1 Introduction

In conventional Bayesian optimization (BO) [Jones et al., 1998], we wish to optimize a derivative-free expensive-to-evaluate function $f$ with feasible domain $A \subseteq \mathbb{R}^d$,

$$\min_{x \in A} f(x),$$

with as few function evaluations as possible. In this paper, we assume that membership in the domain $A$ is easy to evaluate and we can evaluate $f$ only at points in $A$. We assume that evaluations of $f$ are either noise-free, or have additive independent normally distributed noise.

BO typically puts a Gaussian process prior distribution on the function $f$, updating this prior distribution with each new observation of $f$, and choosing the next point or points to evaluate by maximizing an acquisition function that quantifies the benefit of evaluating the objective as a function of where it is evaluated. In comparison with other global optimization algorithms, BO often finds “near optimal” function values with fewer evaluations [Snoek et al., 2012]. As a consequence, BO is useful when function evaluation is time-consuming, such as optimizing the configurations of a complex simulator. Recently, BO also becomes popular in machine learning as it is highly effective in tuning hyperparameters of machine learning algorithms [Snoek et al., 2012, Swersky et al., 2013, Gelbart et al., 2014, Gardner et al., 2014].

When the domain $A$ is discretized, the problem is called Bayesian ranking and selection problem. Knowledge gradient (KG) methods have been widely studied for this problem [Frazier et al., 2008, 2009], which provide one-step Bayes-optimal point to sample. All the previous efforts generalizing KG to continuous domains rely on a discretized finite approximation due to the computational challenges in calculating KG. However, the discretization introduces error, and scales poorly as the dimension of domain grows. In this paper, we develop...
a fast discretization-free knowledge gradient method for Bayesian optimization. In addition, we show that our method is not only useful in the fully sequential setting, but also can be adopted in two new settings: (i) batch of points suggestion (parallel knowledge gradient); (ii) the setting where derivative information are available in the optimization process (derivative-enabled knowledge gradient). We will highlight these two new settings one by one.

First, most previous work in BO assumes that we evaluate the objective function sequentially [Jones et al., 1998], though a few recent papers have considered parallel evaluations [Wang et al., 2016, Marmin et al., 2016, Contal et al., 2013, Desautels et al., 2014, Kathuria et al., 2016, Shah and Ghahramani, 2015, Gonzalez et al., 2016]. While in practice, we can often evaluate several different choices in parallel, such as multiple machines can simultaneously train the machine learning algorithm with different sets of hyperparameters. In this paper, we assume that we can access \( q \geq 1 \) evaluations simultaneously at each iteration. Then we generalize knowledge gradient to guide where to evaluate next based on the decision-theoretical analysis.

Second, Bayesian optimization procedures do not generally leverage derivative information, beyond a few exceptions described in Sect. 2. By contrast, other types of continuous optimization methods [Snyman, 2005] use gradient information extensively. The broader use of gradients for optimization suggests that gradients should also be quite useful in Bayesian optimization: (1) Gradients inform us about the objective’s relative value as a function of location, which is well-aligned with optimization. (2) In \( d \)-dimensional problems, gradients provide \( d \) distinct pieces of information about the objective’s relative value in each direction, constituting \( d + 1 \) values per query together with the objective value itself. This advantage is particularly significant for high-dimensional problems. (3) Derivative information is available in many applications at little additional cost: in the optimization of engineering systems modeled by partial differential equations [Forrester et al., 2008], adjoint methods provide gradients cheaply [Plessix, 2006, Jameson, 1999]. In this paper, we show how a modification of our method, called derivative-enabled knowledge-gradient (\( d\)-KG), can effectively leverage derivative observations, and can optionally reduce the computational overhead of inference by selecting the single most valuable directional derivatives to retain.

Formally, we list our contributions as follows.

- We propose a novel batch BO method which measures the information gain of evaluating \( q \) points via a new acquisition function, the parallel knowledge gradient (\( q\)-KG). This method is derived using a decision-theoretic analysis that chooses the set of points to evaluate next that is optimal in the average-case with respect to the posterior when there is only one batch of points remaining.

- Naively maximizing \( q\)-KG would be extremely computationally intensive, especially when \( q \) is large, and so, in this paper, we develop a fast discretization-free method based on infinitesimal perturbation analysis (IPA) [L’Ecuyer, 1990] to evaluate \( q\)-KG’s gradient efficiently, allowing its efficient optimization.

- In addition, we show that a modification of our method (\( d\)-KG) can effectively leverage gradients when they are available to further improve the optimization performance. This algorithm can optionally reduce the computational overhead of inference by selecting the single most valuable directional derivative to retain.

- In numerical experiments on both synthetic functions and tuning practical machine learning algorithms, \( q\)-KG consistently finds better function values than other parallel BO algorithms, such as parallel EI [Snoek et al., 2012, Chevalier and Ginsbourger, 2013, Wang et al., 2016], batch UCB [Desautels et al., 2014] and parallel UCB with exploration [Contal et al., 2013]. \( q\)-KG provides especially large value when function evaluations are noisy. \( d\)-KG outperforms state-of-the-art batch Bayesian optimization algorithms with and without derivative information, and the gradient-based optimizer BFGS with full gradients. The code in this paper is available at https://github.com/wujian16/qKG.

The rest of the paper is organized as follows. Sect. 2 reviews related work. Sect. 3 gives background on Gaussian processes and defines notation used later. Sect. 4 proposes our new acquisition function \( q\)-KG for batch BO. Sect. 5 provides our computationally efficient discretization-free approach to maximizing \( q\)-KG. Sect. 6 extends our method to the settings where gradient observations of \( f \) are available. Sect. 7 proves that our gradient estimator of \( q\)-KG is unbiased. Sect. 8 presents the empirical performance of \( q\)-KG, \( d\)-KG and several benchmarks on synthetic functions and real-world problems. Finally, Sect. 9 concludes the paper.
2 Related work

Within the past several years, BO has shown success in global optimization of expensive-to-evaluate multimodal objective functions [Snoek et al., 2012, Swersky et al., 2013, Gelbart et al., 2014, Gardner et al., 2014, Hernández-Lobato et al., 2014, Snoek et al., 2014]. BO algorithms consist of two components: a statistical model describing the function and an acquisition function guiding evaluations. In practice, Gaussian Process (GP) [Rasmussen and Williams, 2006] is the mostly widely used statistical model due to its flexibility and tractability. Much of the literature in BO focuses on designing good acquisition functions that reach optima with as few evaluations as possible. Maximizing this acquisition function usually provides a single point to evaluate next, with common acquisition functions for sequential Bayesian optimization including probability of improvement (PI) [Törn and Zilinskas, 1989], expected improvement (EI) [Jones et al., 1998], upper confidence bound (UCB) [Srinivas et al., 2010], entropy search (ES) [Hernández-Lobato et al., 2014], and knowledge gradient (KG) [Scott et al., 2011].

Recently, a few papers have extended BO to the parallel setting, aiming to choose a batch of points to evaluate next in each iteration, rather than just a single point. Ginsbourger et al. [2010], Snoek et al. [2012] suggests parallelizing EI by iteratively constructing a batch, in each iteration adding the point with maximal single-evaluation EI averaged over the posterior distribution of previously selected points. Ginsbourger et al. [2010] also proposes an algorithm called “constant liar”, which iteratively constructs a batch of points to sample by maximizing single-evaluation while pretending that points previously added to the batch have already returned values.

There are also work extending UCB to the parallel setting. Desautels et al. [2014] proposes the GP-BUCB policy, which selects points sequentially by a UCB criterion until filling the batch. Each time one point is selected, the algorithm updates the kernel function while keeping the mean function fixed. Contal et al. [2013], Kathuria et al. [2016] propose algorithms combining UCB with pure exploration or determinant point process, called GP-UCB-PE (UCB-DPP-PE). In these algorithms, the first point is selected according to a UCB criterion; then the remaining points are selected to encourage the diversity of the batch. These algorithms extending UCB do not require Monte Carlo sampling, making them fast and scalable. However, UCB criteria are usually designed to minimize cumulative regret rather than immediate regret, causing these methods to underperform in BO, where we wish to minimize simple regret.

The parallel methods above construct the batch of points in an iterative greedy fashion, optimizing some single-evaluation acquisition function while holding the other points in the batch fixed. The acquisition function we propose considers the batch of points collectively, and we choose the batch to jointly optimize this acquisition function. Other recent papers that value points collectively include Chevalier and Ginsbourger [2013] which optimizes the parallel EI by a closed-form formula, Marmin et al. [2015], Wang et al. [2016], in which gradient-based methods are proposed to jointly optimize a parallel EI criterion, and Shah and Ghahramani [2015], which proposes a parallel version of the ES algorithm and uses Monte Carlo Sampling to optimize the parallel ES acquisition function.

We compare against methods from a number of these previous papers in our numerical experiments, and demonstrate that we provide an improvement, especially in problems with noisy evaluations.

For Bayesian optimization with gradients, Osborne et al. [2009] proposes fully Bayesian optimization procedures that use derivative observations to improve the conditioning of the GP covariance matrix. Samples taken near previously observed points use only the derivative information to update the covariance matrix. Unlike our current work, derivative information is not fully utilized for optimization in this previous work in the sense that derivatives’ availability do not affect the acquisition function. We directly compare with Osborne et al. [2009] within the KNN benchmark in Sect. 8.3. Lizotte [2008, Sect. 4.2.1 and Sect. 5.2.4] incorporates derivatives into Bayesian optimization, modeling the derivatives of a GP as in Rasmussen and Williams [2006, Sect. 9.4]. Lizotte [2008] shows that Bayesian optimization with the expected improvement (EI) acquisition function and complete gradient information at each sample can outperform BFGS.

Very recently, Koistinen et al. [2016] use GPs with derivative observations for minimum energy path calculations of atomic rearrangements and Ahmed et al. [2016] studies expected improvement with gradient observations. In Ahmed et al. [2016], a randomly selected directional derivative is retained in each iteration for computational reasons, which is similar to our approach of retaining a single directional derivative, though differs in its random selection in contrast with our value-of-information-based selection. Our approach is complementary to these works.

Our method is also closely related to the knowledge gradient (KG) method [Scott et al., 2011, Frazier et al., 2009] for the non-batch (sequential) and derivative-free setting, which chooses the Bayes-optimal point to evaluate if only one iteration is left [Scott et al., 2011], and the final solution that we choose is not restricted to be one
of the points we evaluate. (Expected improvement is Bayes-optimal if the solution is restricted to be one of the points we evaluate.) We go beyond this previous work in four aspects. First, we generalize to the parallel setting. Second, previous KG methods with continuous alternatives all rely on the discretization of the search space. We provide the first fast discretization-free method for optimizing the knowledge gradient function. Third, while the sequential setting allows evaluating the KG acquisition function exactly if one discretizes the space, evaluation requires Monte Carlo in the parallel setting, and so we develop more sophisticated computational techniques to optimize our acquisition function. Fourth, we generalize to the derivative-enabled setting.

3 The Gaussian processes model

In this section, we state our prior on $f$, briefly discuss well known results about Gaussian processes (GP), and introduce notation used later. We put a Gaussian process prior over the function $f : \mathcal{A} \to \mathbb{R}$, which is specified by its mean function $\mu(x) : \mathcal{A} \to \mathbb{R}$ and kernel function $K(x_1, x_2) : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$. We assume either exact or independent normally distributed measurement errors, i.e. the evaluation $y(x_i)$ at point $x_i$ satisfies

$$y(x_i) \mid f(x_i) \sim \mathcal{N}(f(x_i), \sigma^2(x_i)),$$

where $\sigma^2 : \mathcal{A} \to \mathbb{R}^+$ is a known function describing the variance of the measurement errors. If $\sigma^2$ is not known, we can also estimate it as we do in Sect. 8.

Supposing we have measured $f$ at $n$ points $x^{(1:n)} := \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ and obtained corresponding measurements $y^{(1:n)}$, we can then combine these observed function values with our prior to obtain a posterior distribution on $f$. This posterior distribution is still a Gaussian process with the mean function $\mu^{(n)}$ and the kernel function $K^{(n)}$ as follows

$$\begin{align*}
\mu^{(n)}(x) &= \mu(x) \\
&+ K(x, x^{(1:n)}) \left( K(x^{(1:n)}, x^{(1:n)}) + \text{diag}(\sigma^2(x^{(1)}), \ldots, \sigma^2(x^{(n)})) \right)^{-1} (y^{(1:n)} - \mu(x^{(1:n)})), \\
K^{(n)}(x_1, x_2) &= K(x_1, x_2) \\
&- K(x_1, x^{(1:n)}) K(x^{(1:n)}, x^{(1:n)}) \left( K(x^{(1:n)}, x^{(1:n)}) + \text{diag}(\sigma^2(x^{(1)}), \ldots, \sigma^2(x^{(n)})) \right)^{-1} K(x^{(1:n)}, x_2).
\end{align*}
$$

4 Parallel knowledge gradient ($q$-KG)

In this section, we propose a novel parallel Bayesian optimization algorithm by generalizing the concept of the knowledge gradient from Frazier et al. [2009] to the parallel setting. The knowledge gradient policy in Frazier et al. [2009] for discrete $\mathcal{A}$ chooses the next sampling decision by maximizing the expected incremental value of a measurement, without assuming (as expected improvement does) that the point returned as the optimum must be a previously sampled point.

We now show how to compute this expected incremental value of an additional iteration in the parallel setting. Suppose that we have observed $n$ function values. If we were to stop measuring now, $\min_{x \in \mathcal{A}} \mu^{(n)}(x)$ would be the minimum of the predictor of the GP. If instead we took one more batch of samples, $\min_{x \in \mathcal{A}} \mu^{(n+q)}(x)$ would be the minimum of the predictor of the GP. The difference between these quantities, $\min_{x \in \mathcal{A}} \mu^{(n)}(x) - \min_{x \in \mathcal{A}} \mu^{(n+q)}(x)$, is the increment in expected solution quality (given the posterior after $n + q$ samples) that results from the additional batch of samples.

This increment in solution quality is random given the posterior after $n$ samples, because $\min_{x \in \mathcal{A}} \mu^{(n+q)}(x)$ is itself a random vector due to its dependence on the outcome of the samples. We can compute the probability distribution of this difference (with more details given below), and the $q$-KG algorithm values the sampling decision $z^{(1:q)} := \{z_1, z_2, \ldots, z_q\}$ according to its expected value, which we call the parallel knowledge gradient factor, and indicate it using the notation $q$-KG. Formally, we define the $q$-KG factor for a set of candidate points to sample $z^{(1:q)}$ as

$$q\text{-KG}(z^{(1:q)}, \mathcal{A}) = \min_{x \in \mathcal{A}} \mu^{(n)}(x) - \mathbb{E}_n \left[ \min_{x \in \mathcal{A}} \mu^{(n+q)}(x) \mid z^{(1:q)} \right],$$

where $\mathbb{E}_n [\cdot] := \mathbb{E}[\cdot \mid y^{(1:n)}]$, is the expectation taken with respect to the posterior distribution after $n$ evaluations. Then we choose to evaluate the next batch of $q$ points that maximizes the parallel knowledge.
By construction, the parallel knowledge gradient policy is Bayes-optimal for minimizing the minimum of the predictor of the GP if only one decision is remaining. The \( q\text{-KG} \) algorithm will reduce to the parallel EI algorithm if function evaluations are noise-free and the final recommendation is restricted to the previous sampling decisions. Because under the two conditions above, the increment in expected solution quality will become

\[
\min_{x \in \mathbb{A}^{(1:n)}} \mu^{(n)}(x) - \min_{x \in \mathbb{A}^{(1:n)} \cup \mathbb{A}^{(1:q)}} \mu^{(n+q)}(x) = \min y^{(1:n)} - \min \left\{ y^{(1:n)} \mid x \in \mathbb{A}^{(1:q)} \right\}
\]

which is exactly the parallel EI acquisition function. However, computing \( q\text{-KG} \) and its gradient is very expensive. We will address the computational issues in Sect. 5. The full description of the \( q\text{-KG} \) algorithm is summarized as follows.

**Algorithm 1** The \( q\text{-KG} \) algorithm

**Require:** the number of initial stage samples \( I \), and the number of main stage sampling iterations \( N \).

1: Initial Stage: draw \( I \) initial samples from a latin hypercube design in \( \mathbb{A} \), \( x^{(i)} \) for \( i = 1, \ldots, I \).

2: Main Stage:

3: for \( s = 1 \) to \( N \) do

4: \quad Solve (4.2), i.e. get \((z^*_1, z^*_2, \ldots, z^*_q) = \arg\max_{x \in \mathbb{A}^{(1:q)}} q\text{-KG}(x, \mathbb{A})\)

5: \quad Sample these points \((z^*_1, z^*_2, \ldots, z^*_q)\), re-train the hyperparameters of the GP, and update the posterior distribution of \( f \).

6: end for

7: return \( x^* = \arg\min_{x \in \mathbb{A}} \mu(I+Nq)(x) \).

## 5 Computation of \( q\text{-KG} \)

Calculating and maximizing \( q\text{-KG} \) is difficult when \( \mathbb{A} \) is continuous because the term \( \min_{x \in \mathbb{A}} \mu^{(n+q)}(x) \) in Eq. (4.1) requires optimizing over an continuous domain, and then we must integrate this optimal value through its dependence on \( y(z^{(1:q)}) \). Previous work on the knowledge gradient in continuous domains [Scott et al., 2011, Poloczek et al., 2016, Wu and Frazier, 2016, Wu et al., 2017] overcomes this by taking minima within expectations over the full domain \( \mathbb{A} \) but over a discretized finite approximation. This supports analytic integration in Scott et al. [2011], Poloczek et al. [2016] and a sampling-based scheme in Wu and Frazier [2016], Wu et al. [2017].

The discretization introduces error, and scales poorly as the dimension of \( \mathbb{A} \) grows. In this section, we provide the first efficient discretization-free strategy to maximize \( q\text{-KG} \) by a gradient-based optimizer. In Sect. 5.1, we describe how to compute \( q\text{-KG} \) by solving the inner optimization problem in Eq. (4.1). In Sect. 5.2, we provides an efficient unbiased estimator of the gradient of \( q\text{-KG} \), which enables its fast optimization. Sect. 5.3 describes how we handle the hyperparameters of the GP model. Sect. 5.4 shows how to do asynchronous batch suggestion.

### 5.1 Estimating \( q\text{-KG} \)

Following Frazier et al. [2009], we express \( \mu^{(n+q)}(x) \) as

\[
\mu^{(n+q)}(x) = \mu^{(n)}(x) + \kappa^{(n)}(x, z^{(1:q)}) \left( \kappa^{(n)}(z^{(1:q)}, z^{(1:q)}) + \text{diag}(\sigma^2(z_1), \ldots, \sigma^2(z_q)) \right)^{-1} (y(z^{(1:q)}) - \mu^{(n)}(z^{(1:q)})).
\]

Because \( y(z^{(1:q)}) - \mu^{(n)}(z^{(1:q)}) \) is normally distributed with zero mean and covariance matrix \( \kappa^{(n)}(z^{(1:q)}, z^{(1:q)}) + \text{diag}(\sigma^2(z_1), \ldots, \sigma^2(z_q)) \) with respect to the posterior after \( n \) observations, we can rewrite \( \mu^{(n+q)}(x) \) as

\[
\mu^{(n+q)}(x) = \mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}) W_q,
\]

where \( W_q \) is a diagonal matrix whose diagonal entries are the square roots of the diagonal elements of the covariance matrix of the posterior distribution of \( y(z^{(1:q)}) - \mu^{(n)}(z^{(1:q)}) \).
where \( W_q \) is a standard \( q \)-dimensional normal random vector, and
\[
\tilde{\sigma}_n(x, z^{(1:q)}) = K^{(n)}(x, z^{(1:q)})(D^{(n)}(z^{(1:q)}))^T - 1,
\]
where \( D^{(n)}(z^{(1:q)}) \) is the Cholesky factor of the covariance matrix \( K^{(n)}(z^{(1:q)}, z^{(1:q)}) + \text{diag}(\sigma_1^2, \ldots, \sigma_q^2) \).

The gradient of \( \mu^{(n+q)}(x) \) with respect to \( x \) in (5.1) can also be computed by calculating the gradient of \( \mu^{(n)}(x) \) and \( K^{(n)}(x, z^{(1:q)}) \), which means that \( \min_{x \in \mathbb{A}} \mu^{(n+q)}(x) \) can be solved efficiently by an inner gradient-based optimizer. Now we can compute the \( q \)-KG factor using Monte Carlo sampling: we first sample \( W_q \), optimize \( \mu^{(n+q)}(x) \) in (5.1) by an inner gradient-based optimizer, then plug in (4.1), repeat many times and take average.

### 5.2 Estimating the gradient of \( q \)-KG

Here we propose a novel method for calculating an unbiased estimator of the gradient of \( q \)-KG which we then use within stochastic gradient ascent to maximize \( q \)-KG. This method avoids discretization, and thus is “exact”. It also improves speed significantly over a discretization-based scheme.

Exploiting (5.1), the \( q \)-KG factor can be expressed as
\[
q \text{-KG}(z^{(1:q)}, \mathbb{A}) = \min_{x \in \mathbb{A}} \mu^{(n)}(x) - \mathbb{E}_n \left[ \min_{x \in \mathbb{A}} \left( \mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}) \cdot W_q \right) \right].
\]

Under sufficient regularity conditions [L’Ecuyer, 1990], with details in Theorem 1 one can interchange the gradient and expectation operators,
\[
\nabla q \text{-KG}(z^{(1:q)}, \mathbb{A}) = -\mathbb{E}_n \left[ \nabla \min_{x \in \mathbb{A}} \left( \mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}) \cdot W_q \right) \right].
\]

Since the multiplication, the inverse (when the inverse exists) and the Cholesky operators [Smith, 1995] preserve continuous differentiability, we have the following proposition.

**Proposition 1.** When the mean function \( \mu(\cdot) \) and the kernel function \( K(\cdot) \) are continuous differentiable, \( (x, z^{(1:q)}) \mapsto (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}) \cdot W_q) \) is continuously differentiable.

When \( (x, z^{(1:q)}) \mapsto (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}) \cdot W_q) \) is continuously differentiable and \( \mathbb{A} \) is compact, the envelope theorem (Corollary 4 in Milgrom and Segal [2002]) implies
\[
\nabla q \text{-KG}(z^{(1:q)}, \mathbb{A}) = -\mathbb{E}_n \left[ \nabla \left( \mu^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), z^{(1:q)}) \cdot W_q \right) \right],
\]
where \( x^*(W_q) \in \arg \min_{x \in \mathbb{A}} (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}) \cdot W_q) \).

This implies that \( -\nabla \left( \mu^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), z^{(1:q)}) \cdot W_q \right) \) is an unbiased estimator of \( \nabla q \text{-KG}(z^{(1:q)}, \mathbb{A}) \).

Then,
\[
\nabla \left( \mu^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), z^{(1:q)}) \cdot W_q \right) = \nabla \tilde{\sigma}_n(x^*(W_q), z^{(1:q)}) \cdot W_q
\]
where
\[
\nabla \tilde{\sigma}_n(x^*(W_q), z^{(1:q)}) = \left( \nabla K^{(n)}(x^*(W_q), z^{(1:q)}) \right)(D^{(n)}(z^{(1:q)}))^T - 1 - K^{(n)}(x^*(W_q), z^{(1:q)})(D^{(n)}(z^{(1:q)}))^T - 1.
\]

We can use this unbiased gradient estimator within stochastic gradient ascent [Harold et al., 2003], optionally with multiple starts, to solve the optimization problem (4.2). This technique is called infinitesimal perturbation analysis (IPA) in gradient estimation [L’Ecuyer, 1990].

At the end of the section, we state the theorem which justifies the technique in the paper, whose proof is provided in Sect. 7.

**Theorem 1.** When the mean function \( \mu(\cdot) \) and the kernel function \( K(\cdot) \) are continuously differentiable and the domain \( \mathbb{A} \) is compact, the interchange of the gradient and expectation operators in (5.2) is valid.
5.3 Bayesian Treatment of Hyperparameters.

We adopt a fully Bayesian treatment of hyperparameters in GP similar to Snoek et al. [2012]. We draw $K$ samples of hyperparameters $\phi^{(i)}$ for $1 \leq i \leq K$ via slice sampling [Murray and Adams, 2010] and average our acquisition function across them to obtain

$$q\text{-}KG\text{Integrated}(z^{(1:n)}) = \frac{1}{K} \sum_{i=1}^{K} q\text{-}KG(z^{(1:n)}, \mathbb{A}; \phi^{(i)}), \tag{5.3}$$

where the additional argument $\phi^{(i)}$ in $q\text{-}KG$ indicates that the computation is performed conditioning on hyperparameters $\phi^{(i)}$. In our experiments, we found this method to be computationally efficient and robust, although a more principled treatment of unknown hyperparameters within the knowledge gradient framework would instead marginalize over them when computing $\mu^{(n+q)}(x)$ and $\mu^{(n)}$.

5.4 Asynchronous $q\text{-}KG$ Optimization

The (4.2) corresponds to the synchronous $q\text{-}KG$ optimization, in which we wait for all $q$ points from our previous batch to finish before searching for a new batch of $q$ points. However, in some applications, we may wish to generate a new batch of points to evaluate next while $p(<q)$ points are still being evaluated, before we have their values. This is common in training machine learning algorithms, where different machine learning models do not necessarily finish at the same time.

We can generalize (4.2) to the asynchronous $q\text{-}KG$ optimization. Given that $p$ points are still under evaluation, now we would like to recommend a batch of $q-p$ points to evaluate. As we did for the synchronous $q\text{-}KG$ optimization above, now we estimate the gradient of $q\text{-}KG$ of the combined $q$ points only with respect to the $q-p$ points that we need to recommend. Then we proceed the same way via gradient-based algorithms.

6 Knowledge Gradient with Derivatives

In this section, we will show how our method can be modified to effectively exploit derivative information. Sect. 6.1 reviews a general approach to incorporating derivative information into GPs for Bayesian optimization. Sect. 6.2 introduces a novel acquisition function $d\text{-}KG$, based on the knowledge gradient approach, which utilizes derivative information.

6.1 Derivative Information

Recall we place a GP prior over $f : \mathbb{A} \rightarrow \mathbb{R}$, which is specified by its mean function $\mu : \mathbb{A} \rightarrow \mathbb{R}$ and kernel function $K : \mathbb{A} \times \mathbb{A} \rightarrow \mathbb{R}$. We first suppose that for each sample of $x$ we observe the function value and all $d$ partial derivatives, possibly with independent normally distributed noise, and then later show discuss relaxation to observing only a single directional derivative.

Since the gradient is a linear operator, the gradient of a GP is also a GP (see also Sect. 9.4 in Rasmussen and Williams [2006]), and the function and its gradient follows a multi-output GP with mean function $\tilde{\mu}$ and kernel function $\tilde{K}$ defined below:

$$\tilde{\mu}(x) = (\mu(x), \nabla \mu(x))^T, \quad \tilde{K}(x, x') = \begin{pmatrix} K(x, x') & J(x, x') \\ J(x', x) & H(x, x') \end{pmatrix}, \tag{6.1}$$

where $J(x, x') = \left( \frac{\partial K(x, x')}{\partial x_i}, \ldots, \frac{\partial K(x, x')}{\partial x_d} \right)$ and $H(x, x')$ is the $d \times d$ Hessian of $K(x, x')$.

When evaluating at a point $x$, we observe the noise-obscured function value $y(x)$ and gradient $\nabla y(x)$. Jointly, these observations form a $(d+1)$-dimensional vector with conditional distribution

$$(y(x), \nabla y(x))^T \mid f(x), \nabla f(x) \sim \mathcal{N} \left( (f(x), \nabla f(x))^T, \text{diag}(\sigma^2(x)) \right), \tag{6.2}$$

where $\sigma^2 : \mathbb{A} \rightarrow \mathbb{R}_{\geq 0}^{d+1}$ gives the variance of the observational noise. If $\sigma^2$ is not known, we may estimate it from data. The posterior distribution is again a GP. We refer to the mean function of this posterior GP after $n$ samples as $\tilde{\mu}^{(n)}(\cdot)$ and its kernel function as $\tilde{K}^{(n)}(\cdot, \cdot)$. Their formulae are the same as in (3.1).
If some entries of \((y(x), \nabla y(x))\) are not observed, then the conditional distribution of the remaining vector is obtained by omitting the corresponding entries from \((f(x), \nabla f(x))\) and \(\sigma^2(x)\) in (6.2). Moreover, the vector remaining after these entries are removed from \((f(x), \nabla f(x))\) still obeys a multivariate normal distribution, with parameters obtained by omitting the corresponding entries of the mean vector and covariance matrix from (6.1). This allows performing inference similarly to the procedure described above for full gradient observations.

### 6.2 The \(d\)-KG Acquisition Function

We modify our parallel knowledge gradient method to exploit available derivative information. We call this the \textit{derivative-enabled knowledge gradient} (\(d\)-KG).

Suppose we have observed \(n\) points, and recall from Section 6.1 that \(\tilde{\mu}_1^{(n)}(x)\) is the \((d+1)\)-dimensional vector giving the posterior mean for \(f(x)\) and its \(d\) partial derivatives at \(x\). Section 6.1 discusses how to remove the assumption that all \(d+1\) values are provided. The expected value of \(f(x)\) under the posterior distribution is \(\tilde{\mu}_1^{(n)}(x)\). If after \(n\) samples we were to make an irrevocable (risk-neutral) decision now about the solution to our overarching optimization problem and receive a loss equal to the value of \(f\) at the chosen point, we would choose \(\text{argmin}_{x \in A} \tilde{\mu}_1^{(n)}(x)\) and suffer conditional expected loss \(\min_{x \in A} \tilde{\mu}_1^{(n+q)}(x)\). Similarly, if we made this decision after \(n + q\) samples our conditional expected loss would be \(\min_{x \in A} \tilde{\mu}_1^{(n+q)}(x)\). Therefore, we define the \(d\)-KG factor for a given set of \(q\) candidate points \(z^{(1:q)}\) as

\[
d-KG(z^{(1:q)}, A) = \min_{x \in A} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n \left[ \min_{x \in A} \tilde{\mu}_1^{(n+q)}(x) \mid z^{(1:q)} \right],
\]

where \(\mathbb{E}_n[\cdot]\) is the expectation taken with respect to the posterior distribution after \(n\) evaluations, and the distribution of \(\tilde{\mu}_1^{(n+q)}(\cdot)\) under this posterior marginalizes over the observations \((y(z^{(1)}), \nabla y(z^{(1)}) : i = 1, \ldots, q)\) upon which it depends.

The \(d\)-KG algorithm then seeks to evaluate the batch of points next that maximizes the \(d\)-KG factor,

\[
\max_{z^{(1:q)} \in A} d-KG(z^{(1:q)}, A).
\]

The \(d\)-KG acquisition function differs from \(q\)-KG because here the time-\(n+q\) posterior mean \(\tilde{\mu}_1^{(n+q)}(x)\) depends on \(\nabla y(z^{(1)})\). This in turn requires calculating the pre-posterior distribution of these gradient observations under the time-\(n\) posterior and marginalizing over them. Thus, the \(d\)-KG algorithm differs from \(q\)-KG not just in that gradient observations change the posterior, but also in that the prospect of future gradient observations changes the acquisition function.

Fig. 1 illustrates the behavior of \(d\)-KG and \(d\)-EI on a 1-d example. The latter generalizes expected improvement (EI) to batch-acquisition with derivative information [Lizotte, 2008]. \(d\)-KG clearly chooses a better point to evaluate than \(d\)-EI.

Including all \(d\) partial derivatives can be computationally prohibitive since GP inference scales as \(O(n^3(d + 1)^3)\). To overcome this challenge while retaining the value of derivative observations, we can include only one directional derivative from each iteration in our inference. \(d\)-KG can naturally decide which derivative to include, and can adjust our choice of where to best sample given that we observe more limited information. We define the \(d\)-KG acquisition function for observing only the function value and the derivative with direction \(\theta\) at \(z^{(1:q)}\) as

\[
d-KG(z^{(1:q)}, \theta, A) = \min_{x \in A} \tilde{\mu}_1^{(n)}(x) - \mathbb{E}_n \left[ \min_{x \in A} \tilde{\mu}_1^{(n+q)}(x) \mid z^{(1:q)}, \theta \right].
\]

where conditioning on \(\theta\) is here understood to mean that \(\tilde{\mu}_1^{(n+q)}(x)\) is the conditional mean of \(f(x)\) given \(y(z^{(1)})\) and \(\theta \cdot \nabla y(z^{(1)}) = (\theta \cdot \nabla y(z^{(i)}) : i = 1, \ldots, q)\). The optimization of \(d\)-KG can be solved similar to \(q\)-KG using the technique in Sect. 5. The full algorithm is as follows.
Figure 1: The topmost plots show (1) the posterior surfaces of a function sampled from a one dimensional GP with and without incorporating observations of the gradients. Note that the posterior variance is smaller if the gradients are incorporated; (2) 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, d-KG makes a considerably better sampling decision, whereas d-EI samples essentially the same location as q-EI. The plots in the bottom row depict the posterior surface after the respective sample. Interestingly, KG benefits more from observing the gradients than EI (the last two plots): d-KG samples a point whose observation yields an accurate knowledge of the location of the optimum, while d-EI still has considerable uncertainty around the optimum.

Algorithm 2 d-KG with Relevant Directional Derivative Detection

1: for $t = 1$ to $N$ do
2: \[(\mathbf{z}^{(1:q)*}, \theta^*) = \text{argmax}_{\mathbf{z}^{(1:q)}, \theta} d\text{-}KG(\mathbf{z}^{(1:q)}, \theta, \mathbb{A})\]
3: Augment data with \(y(\mathbf{z}^{(1:q)*})\) and \(\theta^T \nabla y(\mathbf{z}^{(1:q)*})\). Update our posterior on \((f(x), \nabla f(x))\).
4: end for
Return \(x^* = \text{argmin}_{x \in \mathbb{A}} \mu_1^{-N}(x)\)

7 Proof of Theorem 1

In this section, we will prove Theorem 1. Recall that in Sect. 5, we have expressed the \(q\)-KG factor as follows,

\[
q\text{-}KG(\mathbf{z}^{(1:q)}, \mathbb{A}) = \mathbb{E}_n \left( g(\mathbf{z}^{(1:q)}, W_q) \right) \tag{7.1}
\]

where the expectation is taken over \(W_q\) and

\[
g(\mathbf{z}^{(1:q)}, W_q) = \min_{x \in \mathbb{A}} \mu(n)(x) - \min_{x \in \mathbb{A}} \left( \mu(n)(x) + \tilde{\sigma}_n(x, \mathbf{z}^{(1:q)}) W_q \right),
\]

\[
\tilde{\sigma}_n(x, \mathbf{z}^{(1:q)}) = K(n)(x, \mathbf{z}^{(1:q)})(D(n)(\mathbf{z}^{(1:q)})^T)^{-1}.
\]

The main purpose of this section is to prove the following proposition, which is equivalent to Theorem 1 in Sect. 5.

Proposition 2. When \(\mu(\cdot)\) and \(K(\cdot)\) are continuous differentiable,

\[
\frac{\partial}{\partial z_{ij}} q\text{-}KG(\mathbf{z}^{(1:q)}, \mathbb{A}) \bigg|_{z^{(1:q)} = \theta^{(1:q)}} = \mathbb{E} \left( \frac{\partial}{\partial z_{ij}} g(\mathbf{z}^{(1:q)}, W_q) \right) \bigg|_{z^{(1:q)} = \theta^{(1:q)}}, \tag{7.2}
\]

where \(1 \leq i \leq q, 1 \leq j \leq d, z_{ij} \) is the \(j^{th}\) component of the \(i^{th}\) point in \(\mathbf{z}^{(1:q)}\) and \(\theta^{(1:q)} \in \text{the interior of } \mathbb{A}^q\).

Without loss of generality, we assume that \(\mathbb{A} = [0, 1]^d\). Before proceeding, we define one more notation \(f_{W_q}(z_{ij}) = g(\mathbf{z}^{(1:q)}, W_q)\) where \(\mathbf{z}^{(1:q)}\) equals to \(\theta^{(1:q)}\) component-wise except for \(z_{ij}\). To prove Proposition 2, we refer to Theorem 1 in L’Ecuyer [1990], which requires three conditions to make (7.2) valid: there exists an open neighborhood \(\Theta = (0, 1)\) of \(\theta_{ij}\) where \(\theta_{ij}\) is the \(j^{th}\) component of the \(i^{th}\) point in \(\theta^{(1:q)}\) such that (i) \(f_{W_q}(z_{ij})\) is continuous everywhere in \(\Theta\) for any fixed \(W_q\), (ii) \(f_{W_q}(z_{ij})\) is differentiable almost everywhere in \(\Theta\) for any
fixed $W_q$, (iii) the derivative of $f_{W_q}(z_{ij})$ (when it exists) is uniformly bounded by $\Gamma(W_q)$ for all $z_{ij} \in \Theta$, and the expectation of $\Gamma(W_q)$ is finite.

The proof of condition (i) and condition (ii). Under the condition that the mean function $\mu(\cdot)$ and the kernel function $K(\cdot)$ are continuous differentiable, by Proposition 1, $(x, z_{ij}) \mapsto (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}))W_q$ is continuously differentiable. When $(x, z_{ij}) \mapsto (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)}))W_q$ is continuously differentiable and $\mathcal{A}$ is compact, the envelope theorem (Corollary 4 in Milgrom and Segal [2002]) implies (1) $f_{W_q}(z_{ij})$ is absolute continuous, i.e. differentiable almost everywhere; (2) the derivative of $f_{W_q}(z_{ij})$ (when it exists) is

$$\frac{\partial}{\partial z_{ij}} f_{W_q}(z_{ij}) = -\frac{\partial}{\partial z_{ij}} \left( \mu^{(n)}(x^*(W_q)) + \tilde{\sigma}^{(n)}(x^*(W_q), z^{(1:q)}) \cdot W_q \right)$$

where $x^*(W_q) \in \arg\min_{x \in \mathcal{A}} (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)})) \cdot W_q$. One can see that $f_{W_q}(z_{ij})$ is continuous everywhere and differentiable almost everywhere for any fixed $W_q$.

The proof of condition (iii). Given that

$$\frac{\partial}{\partial z_{ij}} f_{W_q}(z_{ij}) = -\frac{\partial}{\partial z_{ij}} \tilde{\sigma}^{(n)}(x^*(W_q), z^{(1:q)}) \cdot W_q$$

where $x^*(W_q) \in \arg\min_{x \in \mathcal{A}} (\mu^{(n)}(x) + \tilde{\sigma}_n(x, z^{(1:q)})) \cdot W_q$. Recall that from Sect. 5.2,

$$\frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(z^{(1:q)}, x^*(W_q)) = \left( \frac{\partial}{\partial z_{ij}} K^{(n)}(x^*(W_q), z^{(1:q)}) \right) (D^{(n)}(z^{(1:q)})^T)^{-1} - K^{(n)}(x^*(W_q), z^{(1:q)}) (D^{(n)}(z^{(1:q)})^T)^{-1}$$

By the result that $(z_{ij}, x) \mapsto \frac{\partial}{\partial z_{ij}} \tilde{\sigma}_n(z^{(1:q)}, x)$ is continuous, one can see that $\frac{\partial}{\partial z_{ij}} \tilde{\sigma}^{(n)}(x^*(W_q), z^{(1:q)})$ is bounded by a vector $0 \leq \Lambda < \infty$ as $\mathcal{A}$ is compact. Then $\left| \frac{\partial}{\partial z_{ij}} f_{W_q}(z_{ij}) \right| \leq \sum_{i=1}^q \Lambda_i |w_i|$ where $W_q = (w_1, \cdots, w_q)^T$. And $E(\sum_{i=1}^q \Lambda_i |w_i|) = \sqrt{2/\pi} \sum_{i=1}^q \Lambda_i < \infty$.

In addition, we can prove that the second moment of the gradient estimator is finite, which implies that SGA converges to a stationary point with an appropriate sequence of step sizes. Since $|\frac{\partial}{\partial z_{ij}} g(z^{(1:q)}, W_q)| \leq \sum_{i=1}^q \Lambda_i |w_i|$, then $E(\frac{\partial}{\partial z_{ij}} g(z^{(1:q)}, W_q))^2 \leq \sum_{i=1}^q \Lambda_i^2 + 4/\pi \sum_{i,j} \Lambda_i \Lambda_j < \infty$.

8 Numerical experiments

We conduct experiments in three different settings. In the first setting, we analyze how $q$-$KG$ can save the wall-clock time as $q$ grows. In the latter two settings, we test the performances of $q$-$KG$ and $d$-$KG$ with several state-of-the-art methods on synthetic functions and real-world applications:

- The batch expected improvement method ($q$-EI) of Wang et al. [2016] that does not utilize derivative information and an extension of $q$-EI that incorporates derivative information denoted by $d$-EI. Note that $d$-EI is similar to Lizotte [2008] but handles incomplete gradients and supports batches.

- The batch GP-UCB-PE method of Contal et al. [2013] that does not utilize derivative information, and an extension of this method that does.

Moreover, we generalize the method of Osborne et al. [2009] to batches and evaluate it on the KNN benchmark. We stress that all of the 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-based method L-BFGS-B provided in scipy.

Following previous literature [Hernández-Lobato et al., 2014], we use a constant mean prior and the ARD square-exponential kernel. When the function evaluation is noisy, we assume that $\sigma^2(x)$ is constant across the domain $\mathcal{A}$, and we estimate it together with other hyperparameters in the GP using Bayesian treatment. We
sample $K = 10$ sets of hyperparameters by slice sampling. In general, the $q$-KG algorithm performs as well or better than state-of-art benchmark algorithms on both synthetic and real problems. It performs especially well in the noisy setting. $d$-KG provides a substantial gain when we incorporate the derivative information.

Before describing the details of the empirical results, we highlight the implementation details of our method. Our implementation inherits the open-source implementation of parallel EI from the Metrics Optimization Engine [Wang et al., 2014], which is fully implemented in C++ with a python interface. The code in this paper is available at https://github.com/wujian16/qKG.

### 8.1 Speed-up analysis

We compare $q$-KG at different levels of parallelism against the fully sequential KG algorithm. We test the algorithms with different batch sizes on two noisy synthetic functions Branin2 and Hartmann6, whose standard deviation of the noise is $\sigma = 0.5$. From the results, our parallel knowledge gradient method does provide a speed-up as $q$ goes up.

![Figure 2: The performances of $q$-KG with different batch sizes. We report the mean and the standard deviation of the log10 scale of the immediate regret vs. the number of iterations. Iteration 0 is the initial designs. For each iteration later, we evaluate $q$ points recommended by the $q$-KG algorithm.](image)

### 8.2 Synthetic Test Functions

We evaluate all methods on four test functions chosen from Bingham [2015]. To examine the gain from noisy derivative information, we sample additive normally distributed noise with mean zero and standard deviation $\sigma = 0.5$ for both the objective function and its partial derivatives. Note that $\sigma$ is not provided to the algorithms and thus 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: we use a batch size $q = 4$ for the Branin, and Rosenbrock functions; otherwise, we use a batch size $q = 8$. For 2d Branin on domain $[-15, 15]^2$ and 6d Hartmann function on $[0, 1]^6$, we assume that the full gradient is available. The 3d Rosenbrock benchmark is on $[-2, 2]^3$, where the third partial derivative is observable with noise. For the 8d Cosine mixture function on $[-1, 1]^8$ we provide noisy 1st and 2nd partial derivatives. The experimental results are summarized in Fig. 3.

Recall that the immediate regret is defined as the loss with respect to a global optimum. The plots for synthetic benchmark functions, shown in Fig. 3, report the log10 scale of immediate regret of the solution that each algorithm would pick as a function of the number of function evaluations. 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 deviation. The number of replications is stated in each benchmark’s description. Summing up, we see that $d$-KG performs the best among all the Bayesian optimization algorithms without derivatives, and $d$-KG successfully exploits noisy derivative information and has the best overall performance.

### 8.3 Real-World Test Functions

**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

$$\text{weight}(i) = \exp \left\{ -\frac{(t-t^i)^2}{l_1^2} - \frac{(p_1-p_1^i)^2}{l_2^2} - \frac{(p_2-p_2^i)^2}{l_3^2} - \frac{(d_1-d_1^i)^2}{l_4^2} - \frac{(d_2-d_2^i)^2}{l_5^2} \right\},$$

where $(t^i, p_1^i, p_2^i, d_1^i, d_2^i)$ 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 $[30, 200]$, $l_2^2$ in $[10^1, 10^8]$, and $l_1^2, l_3^2, l_4^2, l_5^2$ each in $[10^{-8}, 10^{-1}]$.

We use the yellow cab NYC public data set from June 2016, sampling 10000 records from June 1 – 25 as training data and 1000 trip records from June 26 – 30 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 in Fig. 4 demonstrate that d-KG outperforms the other algorithms eventually. For UCB and KG acquisition functions, exploiting derivative information provides an advantage.

**Kernel Learning.** 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. Although we have access to an analytic closed form of the (marginal likelihood) objective, this function is (i) expensive to evaluate and (ii) highly multimodal. Moreover, (iii) derivative information is available. Thus, learning flexible kernel functions is a perfect candidate for our approach.

The task is to train a 3-component spectral mixture kernel on an airline data set Wilson and Adams [2013]. We have to determine the mixture weights, means, and variances, for each of the three Gaussians. Fig. 4 summarizes the experimental performances for batch size $q = 8$. BFGS turns out to be sensitive to its initialization and human intervention and is often trapped in local optima. d-KG, on the other hand, more consistently finds a good solution, and obtains the best solution of all algorithms (within the step limit). Overall, we observe that gradient information is highly valuable in performing this kernel learning task. q-KG also performs extremely well even when no derivative information is used, significantly outperforming other BO methods without derivatives, and comparable to BFGS.

**Logistic Regression and Deep Learning.** We tune logistic regression and a feedforward neural network with 2 hidden layers on the MNIST dataset [LeCun et al., 1998], a standard classification task for handwritten digits. The training set contains 60000 images, the test set 10000. We tune 4 hyperparameters for logistic regression: the $\ell_2$ regularization parameter from 0 to 1, learning rate from 0 to 1, mini batch size from 20 to
2000 and training epochs from 5 to 50. The first derivatives of the first two parameters are can be obtained via the technique of Maclaurin et al. [2015]. For the neural network, we additionally tune the number of hidden units in [50, 500].

Fig. 4 reports the mean and standard deviation of the mean cross-entropy loss on the test set for 20 replications. $d$-KG outperforms the other approaches, which suggests that derivative information is helpful. Our algorithm proves its value in tuning a deep neural network, which harmonizes with research computing the gradient of hyperparameters [Maclaurin et al., 2015, Pedregosa, 2016].

Figure 4: Results for the weighted KNN benchmark, the spectral mixture kernel benchmark, logistic regression and deep neural network (from left to right), all with batch size 8 and averaged over 20 replications.

9 Conclusions

In this paper, we introduce a novel batch Bayesian optimization method $q$-KG, derived from a decision-theoretical perspective, and develop a fast discretization-free method to implement it efficiently. We show that $q$-KG outperforms or is competitive with the state-of-art benchmark algorithms on several synthetic functions and in tuning practical machine learning algorithms.

Moreover, we have shown that in the context of Bayesian optimization, derivative information can be extremely useful: we can greatly decrease the number of objective function evaluations, especially when building upon the knowledge gradient acquisition function, even when derivative information is noisy and only available for some variables.

10 Acknowledgments

The authors 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.

References

M. O. Ahmed, B. Shahriari, and M. Schmidt. Do we need “harmless” bayesian optimization and “first-order” bayesian optimization? In NIPS BayesOpt, 2016.

D. Bingham. Optimization test problems. http://www.sfu.ca/~ssurjano/optimization.html, 2015.

C. Chevalier and D. Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In Learning and Intelligent Optimization, pages 59–69. Springer, 2013.

N. T. . L. Commission. NYC Trip Record Data. http://www.nyc.gov/html/tlc/, June 2016. Last accessed on 2016-10-10.

E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In Machine Learning and Knowledge Discovery in Databases, pages 225–240. Springer, 2013.

T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. The Journal of Machine Learning Research, 15(1):3873–3923, 2014.
A. Forrester, A. Sobester, and A. Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009.

P. I. Frazier, W. B. Powell, and S. Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, pages 937–945, 2014.

M. Gelbart, J. Snoek, and R. Adams. Bayesian optimization with unknown constraints. In *ICML*, pages 250–259, Corvallis, Oregon, 2014. AUAI Press.

D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

J. Gonzalez, Z. Dai, P. Hennig, and N. Lawrence. Batch bayesian optimization via local penalization. In *AISTATS*, pages 648–657, 2016.

J. Harold, G. Kushner, and G. Yin. *Stochastic approximation and recursive algorithm and applications*. Springer, 2003.

J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pages 918–926, 2014.

A. Jameson. Re-engineering the design process through computation. *Journal of Aircraft*, 36(1):36–50, 1999.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

T. Kathuria, A. Deshpande, and P. Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

O.-P. Koistinen, E. Maras, A. Vehtari, and H. Jónsson. Minimum energy path calculations with gaussian process regression. *Nanosystems: Physics, Chemistry, Mathematics*, 7(6), 2016.

Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

P. L’Ecuyer. A unified view of the IPA, SF, and LR gradient estimation techniques. *Management Science*, 36(11):1364–1383, 1990.

D. J. Lizotte. *Practical bayesian optimization*. PhD thesis, University of Alberta, 2008.

D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. In *ICML*, pages 2113–2122, 2015.

S. Marmin, C. Chevalier, and D. Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. In *International Workshop on Machine Learning, Optimization and Big Data*, pages 37–48. Springer, 2015.

S. Marmin, C. Chevalier, and D. Ginsbourger. Efficient batch-sequential bayesian optimization with moments of truncated gaussian vectors. *arXiv preprint arXiv:1609.02700*, 2016.

P. Milgrom and I. Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of latent gaussian models. In *Advances in Neural Information Processing Systems*, pages 1732–1740, 2010.

M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In *3rd international conference on learning and intelligent optimization (LION3)*, pages 1–15. Citeseer, 2009.
F. Pedregosa. Hyperparameter optimization with approximate gradient. In Proceedings of the 33rd International Conference on Machine Learning, ICML, pages 737–746, 2016.

R.-É. Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. Geophysical Journal International, 167(2):495–503, 2006.

M. Poloczek, J. Wang, and P. I. Frazier. Multi-information source optimization. arXiv preprint arXiv:1603.00389, 2016.

C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. MIT Press, 2006. ISBN 0-262-18253-X.

W. Scott, P. Frazier, and W. Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. SIAM Journal on Optimization, 21(3):996–1026, 2011.

A. Shah and Z. Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In Advances in Neural Information Processing Systems, pages 2004–2012, 2013.

N. Srinivas, A. Krause, M. Seeger, and S. M. Kakade. Gaussian process optimization in the bandit setting: No regret and experimental design. In ICML, pages 1015–1022, 2010.

A. Törn and A. Zilinskas. Global optimization, volume 350 of lecture notes in computer science, 1989.

J. Wang, S. C. Clark, E. Liu, and P. I. Frazier. Metrics optimization engine. http://yelp.github.io/MOE/, 2014. Last accessed on 2016-01-21.

A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In ICML, pages 1067–1075, 2013.

J. Wu and P. Frazier. The parallel knowledge gradient method for batch bayesian optimization. In Advances in Neural Information Processing Systems, pages 3126–3134, 2016.