Bayesian Optimization with Unknown Constraints

Michael A. Gelbart
Harvard University
Cambridge, MA

Jasper Snoek
Harvard University
Cambridge, MA

Ryan P. Adams
Harvard University
Cambridge, MA

Abstract

Recent work on Bayesian optimization has shown its effectiveness in global optimization of difficult black-box objective functions. Many real-world optimization problems of interest also have constraints which are unknown *a priori*. In this paper, we study Bayesian optimization for constrained problems in the general case that noise may be present in the constraint functions, and the objective and constraints may be evaluated independently. We provide motivating practical examples, and present a general framework to solve such problems. We demonstrate the effectiveness of our approach on optimizing the performance of online latent Dirichlet allocation subject to topic sparsity constraints, tuning a neural network given test-time memory constraints, and optimizing Hamiltonian Monte Carlo to achieve maximal effectiveness in a fixed time, subject to passing standard convergence diagnostics.

1 INTRODUCTION

Bayesian optimization (Mockus et al., 1978) is a method for performing global optimization of unknown "black box" objectives that is particularly appropriate when objective function evaluations are expensive (in any sense, such as time or money). For example, consider a food company trying to design a low-calorie variant of a popular cookie. In this case, the design space is the space of possible recipes and might include several key parameters such as quantities of various ingredients and baking times. Each evaluation of a recipe entails computing (or perhaps actually measuring) the number of calories in the proposed cookie. Bayesian optimization can be used to propose new candidate recipes such that good results are found with few evaluations.

Now suppose the company also wants to ensure the *taste* of the cookie is not compromised when calories are reduced. Therefore, for each proposed low-calorie recipe, they perform a taste test with sample customers. Because different people, or the same people at different times, have differing opinions about the taste of cookies, the company decides to require that at least 95% of test subjects must like the new cookie. This is a constrained optimization problem:

$$\min_{\mathbf{x}} \; c(\mathbf{x}) \text{ s.t. } \rho(\mathbf{x}) \geq 1 - \epsilon \;,$$

where $\mathbf{x}$ is a real-valued vector representing a recipe, $c(\mathbf{x})$ is the number of calories in recipe $\mathbf{x}$, $\rho(\mathbf{x})$ is the fraction of test subjects that like recipe $\mathbf{x}$, and $1 - \epsilon$ is the minimum acceptable fraction, i.e., 95%.

This paper presents a general formulation of constrained Bayesian optimization that is suitable for a large class of problems such as this one. Other examples might include tuning speech recognition performance on a smart phone such that the user's speech is transcribed within some acceptable time limit, or minimizing the cost of materials for a new bridge, subject to the constraint that all safety margins are met.

Another use of constraints arises when the search space is known *a priori* but occupies a complicated volume that cannot be expressed as simple coordinate-wise bounds on the search variables. For example, in a chemical synthesis experiment, it may be known that certain combinations of reagents cause an explosion to occur. This constraint is not unknown in the sense of being a discovered property of the environment as in the examples above—we do not want to discover the constraint boundary by trial and error explosions of our laboratory. Rather, we would like to specify this constraint using a boolean noise-free oracle function that declares input vectors as valid or invalid. Our formulation of constrained Bayesian optimization naturally encapsulates such constraints.

1.1 BAYESIAN OPTIMIZATION

Bayesian optimization proceeds by iteratively developing a global statistical model of the unknown objective function. Starting with a prior over functions and a likelihood, at each

iteration a posterior distribution is computed by conditioning on the previous evaluations of the objective function, treating them as observations in a Bayesian nonlinear regression. An *acquisition function* is used to map beliefs about the objective function to a measure of how promising each location in input space is, if it were to be evaluated next. The goal is then to find the input that maximizes the acquisition function, and submit it for function evaluation.

Maximizing the acquisition function is ideally a relatively easy proxy optimization problem: evaluations of the acquisition function are often inexpensive, do not require the objective to be queried, and may have gradient information available. Under the assumption that evaluating the objective function is expensive, the time spent computing the best next evaluation via this inner optimization problem is well spent. Once a new result is obtained, the model is updated, the acquisition function is recomputed, and a new input is chosen for evaluation. This completes one iteration of the Bayesian optimization loop.

For an in-depth discussion of Bayesian optimization, see Brochu et al. (2010b) or Lizotte (2008). Recent work has extended Bayesian optimization to multiple tasks and objectives (Krause and Ong, 2011; Swersky et al., 2013; Zuluaga et al., 2013) and high dimensional problems (Wang et al., 2013; Djolonga et al., 2013). Strong theoretical results have also been developed (Srinivas et al., 2010; Bull, 2011; de Freitas et al., 2012). Bayesian optimization has been shown to be a powerful method for the meta-optimization of machine learning algorithms (Snoek et al., 2012; Bergstra et al., 2011) and algorithm configuration (Hutter et al., 2011).

1.2 EXPECTED IMPROVEMENT

An acquisition function for Bayesian optimization should address the exploitation vs. exploration tradeoff: the idea that we are interested both in regions where the model believes the objective function is low ("exploitation") and regions where uncertainty is high ("exploration"). One such choice is the Expected Improvement (EI) criterion (Mockus et al., 1978), an acquisition function shown to have strong theoretical guarantees (Bull, 2011) and empirical effectiveness (e.g., Snoek et al., 2012). The expected improvement, $\mathrm{EI}(\mathbf{x})$, is defined as the expected amount of improvement over some target t, if we were to evaluate the objective function at $\mathbf{x}$:

$$\mathrm{EI}(\mathbf{x}) = \mathbb{E}[(t-y)_+] = \int_{-\infty}^{\infty} (t-y)_+ p(y \,|\, \mathbf{x}) \, \mathrm{d}y \;, \quad (1)$$

where $p(y \,|\, \mathbf{x})$ is the predictive marginal density of the objective function at $\mathbf{x}$, and $(t-y)_+ \equiv \max(0, t-y)$ is the improvement (in the case of minimization) over the target t. EI encourages both exploitation and exploration because it is large for inputs with a low predictive mean (exploitation) and/or a high predictive variance (exploration). Often, t is set to be the minimum over previous observations (e.g., Snoek et al., 2012), or the minimum of the expected value of the objective (Brochu et al., 2010a). Following our formulation of the problem, we use the minimum expected value of the objective such that the probabilistic constraints are satisfied (see Section 1.5, Eq., 6).

When the predictive distribution under the model is Gaussian, EI has a closed-form expression (Jones, 2001):

$$\mathrm{EI}(\mathbf{x}) = \sigma(\mathbf{x}) \left(z(\mathbf{x}) \Phi\left(z(\mathbf{x})\right) + \phi\left(z(\mathbf{x})\right) \right) \quad (2)$$

where $z(\mathbf{x}) \equiv \frac{t-\mu(\mathbf{x})}{\sigma(\mathbf{x})}$, $\mu(\mathbf{x})$ is the predictive mean at $\mathbf{x}$, $\sigma^2(\mathbf{x})$ is the predictive variance at $\mathbf{x}$, $\Phi(\cdot)$ is the standard normal CDF, and $\phi(\cdot)$ is the standard normal PDF. This function is differentiable and fast to compute, and can therefore be maximized with a standard gradient-based optimizer. In Section 3 we present an acquisition function for constrained Bayesian optimization based on EI.

1.3 OUR CONTRIBUTIONS

The main contribution of this paper is a general formulation for constrained Bayesian optimization, along with an acquisition function that enables efficient optimization of such problems. Our formulation is suitable for addressing a large class of constrained problems, including those considered in previous work. The specific improvements are enumerated below.

First, our formulation allows the user to manage uncertainty when constraint observations are noisy. By reformulating the problem with probabilistic constraints, the user can directly address this uncertainty by specifying the required confidence that constraints are satisfied.

Second, we consider the class of problems for which the objective function and constraint function need not be evaluated jointly. In the cookie example, the number of calories might be predicted very cheaply with a simple calculation, while evaluating the taste is a large undertaking requiring human trials. Previous methods, which assume joint evaluations, might query a particular recipe only to discover that the objective (calorie) function for that recipe is highly unfavorable. The resources spent simultaneously evaluating the constraint (taste) function would then be very poorly spent. We present an acquisition function for such problems, which incorporates this user-specified cost information.

Third, our framework, which supports an arbitrary number of constraints, provides an expressive language for specifying arbitrarily complicated restrictions on the parameter search spaces. For example if the total memory usage of a neural network must be within some bound, this restriction could be encoded as a separate, noise-free constraint with very low cost. As described above, evaluating this low-cost constraint would take priority over the more expensive con-

straints and/or objective function.

1.4 PRIOR WORK

There has been some previous work on constrained Bayesian optimization. Gramacy and Lee (2010) propose an acquisition function called the integrated expected conditional improvement (IECI), defined as

$$\mathrm{IECI}(\mathbf{x}) = \int_{\mathcal{X}} [\mathrm{EI}(\mathbf{x}') - \mathrm{EI}(\mathbf{x}'|\mathbf{x})]\, h(\mathbf{x}')\mathrm{d}\mathbf{x}' \,. \quad (3)$$

In the above, $\mathrm{EI}(\mathbf{x}')$ is the expected improvement at $\mathbf{x}'$, $\mathrm{EI}(\mathbf{x}'|\mathbf{x})$ is the expected improvement at $\mathbf{x}'$ given that the objective has been observed at $\mathbf{x}$ (but without making any assumptions about the observed value), and $h(\mathbf{x}')$ is an arbitrary density over $\mathbf{x}'$. In words, the IECI at $\mathbf{x}$ is the expected reduction in EI at $\mathbf{x}'$, under the density $h(\mathbf{x}')$, caused by observing the objective at $\mathbf{x}$. Gramacy and Lee use IECI for constrained Bayesian optimization by setting $h(\mathbf{x}')$ to the probability of satisfying the constraint. This formulation encourages evaluations that inform the model in places that are likely to satisfy the constraint.

Zuluaga et al. (2013) propose the Pareto Active Learning (PAL) method for finding Pareto-optimal solutions when multiple objective functions are present and the input space is a discrete set. Their algorithm classifies each design candidate as either Pareto-optimal or not, and proceeds iteratively until all inputs are classified. The user may specify a confidence parameter determining the tradeoff between the number of function evaluations and prediction accuracy. Constrained optimization can be considered a special case of multi-objective optimization in which the user's utility function for the "constraint objectives" is an infinite step function: constant over the feasible region and negative infinity elsewhere. However, PAL solves different problems than those we intend to solve, because it is limited to discrete sets and aims to classify each point in the set versus finding a single optimal solution.

Snoek (2013) discusses constrained Bayesian optimization for cases in which constraint violations arise from a failure mode of the objective function, such as a simulation crashing or failing to terminate. The author introduces the weighted expected improvement acquisition function, namely expected improvement weighted by the predictive probability that the constraint is satisfied at that input.

1.5 FORMALIZING THE PROBLEM

In Bayesian optimization, the objective and constraint functions are in general unknown for two reasons. First, the functions have not been observed everywhere, and therefore we must interpolate or extrapolate their values to new inputs. Second, our observations may be noisy; even after multiple observations at the same input, the true function is not known. Accounting for this uncertainty is the role of the model, see Section 2.

However, before solving the problem, we must first define it. Returning to the cookie example, each taste test yields an estimate of $\rho(\mathbf{x})$, the fraction of test subjects that like recipe $\mathbf{x}$. But uncertainty is always present, even after many measurements. Therefore, it is impossible to be certain that the constraint $\rho(\mathbf{x}) \geq 1 - \epsilon$ is satisfied for any $\mathbf{x}$. Likewise, the objective function can only be evaluated point-wise and, if noise is present, it may never be determined with certainty.

This is a stochastic programming problem: namely, an optimization problem in which the objective and/or constraints contain uncertain quantities whose probability distributions are known or can be estimated (see e.g., Shapiro et al., 2009). A natural formulation of these problems is to minimize the objective function *in expectation*, while satisfying the constraints *with high probability*. The condition that the constraint be satisfied with high probability is called a *probabilistic constraint*. This concept is formalized below.

Let $f(\mathbf{x})$ represent the objective function. Let $\mathcal{C}(\mathbf{x})$ represent the the *constraint condition*, namely the boolean function indicating whether or not the constraint is satisfied for input $\mathbf{x}$. For example, in the cookie problem, $\mathcal{C}(\mathbf{x}) \iff \rho(\mathbf{x}) \geq 1 - \epsilon$. Then, our probabilistic constraint is

$$\Pr(\mathcal{C}(\mathbf{x})) \geq 1 - \delta \,, \quad (4)$$

for some user-specified minimum confidence $1 - \delta$.

If K constraints are present, for each constraint $k \in (1, \ldots, K)$ we define $\mathcal{C}_k(\mathbf{x})$ to be the constraint condition for constraint k. Each constraint may also have its own tolerance δ_k, so we have K probabilistic constraints of the form

$$\Pr(\mathcal{C}_k(\mathbf{x})) \geq 1 - \delta_k \,. \quad (5)$$

All K probabilistic constraints must ultimately be satisfied at a solution to the optimization problem.[1]

Given these definitions, a general class of constrained Bayesian optimization problems can be formulated as

$$\min_{\mathbf{x}} \ \mathbb{E}[f(\mathbf{x})] \ \text{s.t.} \ \forall k \ \Pr(\mathcal{C}_k(\mathbf{x})) \geq 1 - \delta_k \,. \quad (6)$$

The remainder of this paper proposes methods for solving problems in this class using Bayesian optimization. Two key ingredients are needed: a model of the objective and constraint functions (Section 2), and an acquisition function that determines which input $\mathbf{x}$ would be most beneficial to observe next (Section 3).

[1] Note: this formulation is based on individual constraint satisfaction for all constraints. Another reasonable formulation requires the (joint) probability that *all* constraints are satisfied to be above some single threshold.

2 MODELING THE CONSTRAINTS

2.1 GAUSSIAN PROCESSES

We use Gaussian processes (GPs) to model both the objective function $f(\mathbf{x})$ and the constraint functions. A GP is a generalization of the multivariate normal distribution to arbitrary index sets, including infinite length vectors or functions, and is specified by its positive definite covariance kernel function $K(\mathbf{x}, \mathbf{x}')$. GPs allow us to condition on observed data and tractably compute the posterior distribution of the model for any finite number of query points. A consequence of this property is that the marginal distribution at any single point is univariate Gaussian with a known mean and variance. See Rasmussen and Williams (2006) for an in-depth treatment of GPs for machine learning.

We assume the objective and all constraints are independent and model them with independent GPs. Note that since the objective and constraints are all modeled independently, they need not all be modeled with GPs or even with the same types of models as each other. Any combination of models suffices, so long as each one represents its uncertainty about the true function values.

2.2 THE LATENT CONSTRAINT FUNCTION, $g(\mathbf{x})$

In order to model constraint conditions $\mathcal{C}_k(\mathbf{x})$, we introduce real-valued latent *constraint functions* $g_k(\mathbf{x})$ such that for each constraint k, the constraint condition $\mathcal{C}_k(\mathbf{x})$ is satisfied if and only if $g_k(\mathbf{x}) \geq 0$.[2] Different observation models lead to different likelihoods on $g(\mathbf{x})$, as discussed below. By computing the posterior distribution of $g_k(\mathbf{x})$ for each constraint, we can then compute $\Pr(\mathcal{C}_k(\mathbf{x})) = \Pr(g_k(\mathbf{x}) \geq 0)$ by simply evaluating the Gaussian CDF using the predictive marginal mean and variance of the GP at $\mathbf{x}$.

Different constraints require different definitions of the constraint function $g(\mathbf{x})$. When the nature of the problem permits constraint observations to be modeled with a Gaussian likelihood, the posterior distribution of $g(\mathbf{x})$ can be computed in closed form. If not, approximations or sampling methods are needed (see Rasmussen and Williams, 2006, p. 41-75). We discuss two examples below, one of each type, respectively.

2.3 EXAMPLE I: BOUNDED RUNNING TIME

Consider optimizing some property of a computer program such that its running time $\tau(\mathbf{x})$ must not exceed some value $\tau_{\max}$. Because $\tau(\mathbf{x})$ is a measure of time, it is non-negative for all $\mathbf{x}$ and thus not well-modeled by a GP prior. We therefore choose to model time in logarithmic units. In particular, we define $g(\mathbf{x}) = \log \tau_{\max} - \log \tau$, so that the condition $g(\mathbf{x}) \geq 0$ corresponds to our constraint condition $\tau \leq \tau_{\max}$, and place a GP prior on $g(\mathbf{x})$. For every problem, this transformation implies a particular prior on the original variables; in this case, the implied prior on $\tau(\mathbf{x})$ is the log-normal distribution. In this problem we may also posit a Gaussian likelihood for observations of $g(\mathbf{x})$. This corresponds to the generative model that constraint observations are generated by some true latent function corrupted with i.i.d. Gaussian noise. As with the prior, this choice implies something about the original function $\tau(\mathbf{x})$, in this case a log-normal likelihood. The basis for these choices is their computational convenience. Given a Gaussian prior and likelihood, the posterior distribution is also Gaussian and can be computed in closed form using the standard GP predictive equations.

2.4 EXAMPLE II: MODELING COOKIE TASTINESS

Recall the cookie optimization, and let us assume that constraint observations arrive as a set of counts indicating the numbers of people who did and did not like the cookies. We call these *binomial constraint observations*. Because these observations are discrete, they are not modeled well by a GP prior. Instead, we model the (unknown) binomial probability $\rho(\mathbf{x})$ that a test subject likes cookie $\mathbf{x}$, which is linked to the observations through a binomial likelihood.[3] In Section 1.5, we selected the constraint condition $\rho(\mathbf{x}) \geq 1 - \epsilon$, where $1 - \epsilon$ is the user-specified threshold representing the minimum allowable probability that a test subject likes the new cookie. Because $\rho(\mathbf{x}) \in (0, 1)$ and $g(\mathbf{x}) \in \mathbb{R}$, we define $g(\mathbf{x}) = s^{-1}(\rho(\mathbf{x}))$, where $s(\cdot)$ is a monotonically increasing sigmoid function mapping $\mathbb{R} \to (0, 1)$ as in logistic or probit regression.[4] In our implementation, we use $s(z) = \Phi(z)$, the Gaussian CDF. The likelihood of $g(\mathbf{x})$ given the binomial observations is then the binomial likelihood composed with s^{-1}. Because this likelihood is non-Gaussian, the posterior distribution cannot be computed in closed form, and therefore approximation or sampling methods are needed.

2.5 INTEGRATING OUT THE GP HYPERPARAMETERS

Following Snoek et al. (2012), we use the Matérn 5/2 kernel for the Gaussian process prior, which corresponds to the

[2] Any inequality constraint $g(\mathbf{x}) \leq g_0$ or $g(\mathbf{x}) \geq g_1$ can be represented this way by transforming to a new variable $\hat{g}(\mathbf{x}) \equiv g_0 - g(\mathbf{x}) \geq 0$ or $\hat{g}(\mathbf{x}) \equiv g(\mathbf{x}) - g_1 \geq 0$, respectively, so we set the right-hand side to zero without loss of generality.

[3] We use the notation $\rho(\mathbf{x})$ both for the fraction of test subjects who like recipe $\mathbf{x}$ and for its generative interpretation as the probability that a subject likes recipe $\mathbf{x}$.

[4] When the number of binomial trials is one, this model is called Gaussian Process Classification.

assumption that the function being modeled is twice differentiable. This kernel has $D + 1$ hyperparameters in D dimensions: one characteristic length scale per dimension, and an overall amplitude. Again following Snoek et al. (2012), we perform a fully-Bayesian treatment by integrating out these kernel hyperparameters with Markov chain Monte Carlo (MCMC) via slice sampling (Neal, 2000).

When the posterior distribution cannot be computed in closed form due to a non-Gaussian likelihood, we use elliptical slice sampling (Murray et al., 2010) to sample $g(\mathbf{x})$. We also use the prior whitening procedure described in Murray and Adams (2010) to avoid poor mixing due to the strong coupling of the latent values and the kernel hyperparameters.

3 ACQUISITION FUNCTIONS

3.1 CONSTRAINT WEIGHTED EXPECTED IMPROVEMENT

Given the probabilistic constraints and the model for a particular problem, it remains to specify an acquisition function that leads to efficient optimization. Here, we present an acquisition function for constrained Bayesian optimization under the Expected Improvement (EI) criterion (Section 1.2). However, the general framework presented here does not depend on this specific choice and can be used in conjunction with any improvement criterion.

Because improvement is not possible when the constraint is violated, we can define an acquisition function for constrained Bayesian optimization by extending the expectation in Eq. 1 to include the additional constraint uncertainty. This results in a constraint-weighted expected improvement criterion, $a(\mathbf{x})$:

$$a(\mathbf{x}) = \mathrm{EI}(\mathbf{x}) \Pr(\mathcal{C}(\mathbf{x})) \tag{7}$$

$$= \mathrm{EI}(\mathbf{x}) \prod_{k=1}^{K} \Pr(\mathcal{C}_k(\mathbf{x})) \tag{8}$$

where the second line follows from the assumed independence of the constraints. The gradient of this acquisition function is readily computed and therefore this acquisition function can be maximized in the same manner as standard EI. In practice we maximize it following the method in Snoek et al. (2012).

Then, the full acquisition function $a(\mathbf{x})$, after integrating out the GP hyperparameters, is given by

$$a(\mathbf{x}) = \int \mathrm{EI}(\mathbf{x}|\theta) p(\theta|\mathbf{D}) p(\mathcal{C}(\mathbf{x})|\mathbf{x}, \mathbf{D}', \omega) p(\omega|\mathbf{D}') \mathrm{d}\theta \mathrm{d}\omega,$$

where θ is the set of GP hyperparameters for the objective function model, ω is the set of GP hyperparameters for the constraint model(s), $\mathbf{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$ are the previous objective function observations, and $\mathbf{D}'$ are the constraint function observations.

3.2 FINDING THE FEASIBLE REGION

The acquisition function given above is not defined when at least one probabilistic constraint is violated for all $\mathbf{x}$, because in this case the EI target does not exist and therefore EI cannot be computed. In this case we take the acquisition function to include only the second factor,

$$a(\mathbf{x}) = \prod_{k=1}^{K} \Pr(g_k(\mathbf{x}) \geq 0) \tag{9}$$

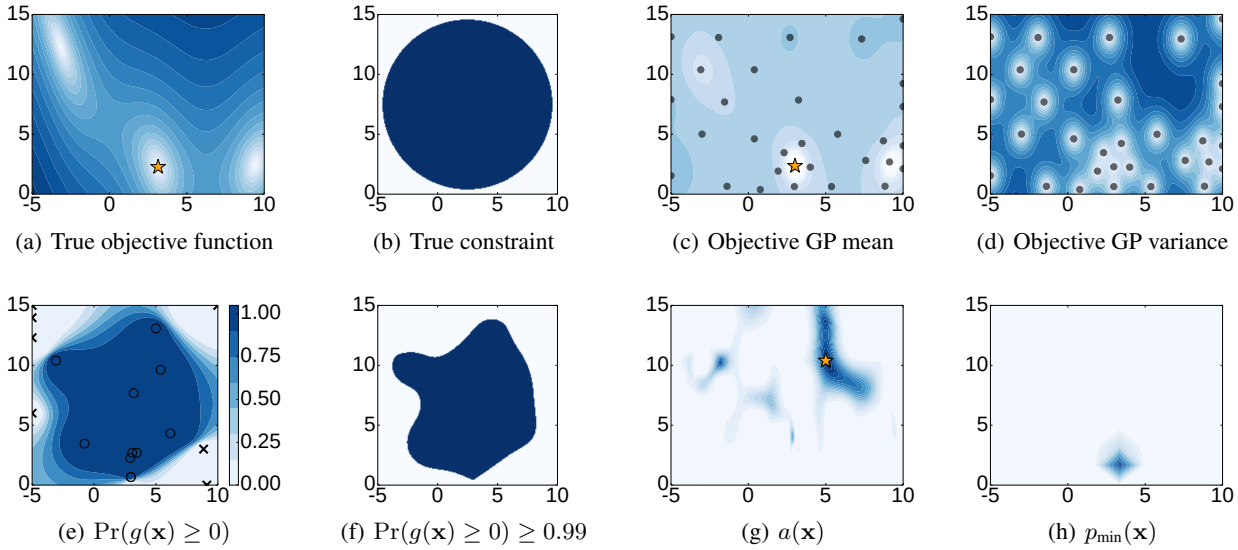
Intuitively, if the probabilistic constraint is violated everywhere, we ignore the objective function and try to satisfy the probabilistic constraint until it is satisfied somewhere. This acquisition function may also be used if no objective function exists, i.e., if the problem is just to search for any feasible input. This feasibility search is purely exploitative: it searches where the probability of satisfying the constraints is highest. This is possible because the true probability of constraint satisfaction is either zero or one. Therefore, as the algorithm continues to probe a particular region, it will either discover that the region is feasible, or the probability will drop and it will move on to a more promising region.

3.3 DECOUPLED OBSERVATIONS

In some problems, the objective and constraint functions may be evaluated independently. We call this property the *decoupling* of the objective and constraint functions. In decoupled problems, we must choose to evaluate either the objective function or one of the constraint functions at each iteration of Bayesian optimization. As discussed in Section 1.3, it is important to identify problems with this decoupled structure, because often some of the functions are much more expensive to evaluate than others. Bayesian optimization with decoupled constraints is a form of multi-task Bayesian optimization (Swersky et al., 2013), in which the different black-boxes or *tasks* are the objective and decoupled constraint(s), represented by the set $\{\text{objective}, 1, 2, \ldots, K\}$ for K constraints.

3.3.1 Chicken and Egg Pathology

One possible acquisition function for decoupled constraints is the expected improvement of individually evaluating each task. However, the myopic nature of the EI criterion causes a pathology in this formulation that prevents exploration of the design space. Consider a situation, with a single constraint, in which some feasible region has been identified and thus the current best input is defined, but a large unexplored region remains. Evaluating only the objective in this region could not cause improvement as our



(a) True objective function (b) True constraint (c) Objective GP mean (d) Objective GP variance

(e) $\Pr(g(\mathbf{x}) \geq 0)$ (f) $\Pr(g(\mathbf{x}) \geq 0) \geq 0.99$ (g) $a(\mathbf{x})$ (h) $p_{\min}(\mathbf{x})$

Figure 1: Constrained Bayesian optimization on the 2D Branin-Hoo function with a disk constraint, after 50 iterations (33 objective evaluations and 17 constraint evaluations): (a) Branin-Hoo function, (b) true constraint, (c) mean of objective function GP, (d) variance of objective function GP, (e) probability of constraint satisfaction, (f) probabilistic constraint, $\Pr(g(\mathbf{x}) \geq 0) \geq 0.99$, (g) acquisition function, $a(\mathbf{x})$, and (h) probability distribution over the location of the minimum, $p_{\min}(\mathbf{x})$. Lighter colors indicate lower values. Objective function observations are indicated with black circles in (c) and (d). Constraint observations are indicated with black $\times$'s (violations) and o's (satisfactions) in (e). Orange stars: (a) unique true minimum of the constrained problem, (c) best solution found by Bayesian optimization, (g) input chosen for the next evaluation, in this case an objective evaluation because $\Delta S_o(\mathbf{x}) > \Delta S_c(\mathbf{x})$ at the next observation location $\mathbf{x}$.

belief about $\Pr(g(\mathbf{x}) \geq 0)$ will follow the prior and not exceed the threshold $1 - \delta$. Likewise, evaluating only the constraint would not cause improvement because our belief about the objective will follow the prior and is unlikely to become the new best. This is a causality dilemma: we must learn that *both* the objective and the constraint are favorable for improvement to occur, but this is not possible when only a single task is observed. This difficulty suggests a non-myopic aquisition function which assesses the improvement after a sequence of objective and constraint observations. However, such a multi-step acquisition function is intractable in general (Ginsbourger and Riche, 2010).

Instead, to address this pathology, we propose to use the coupled acquisition function (Eq. 7) to select an input $\mathbf{x}$ for observation, followed by a second step to determine which task will be evaluated at $\mathbf{x}$. Following Swersky et al. (2013), we use the entropy search criterion (Hennig and Schuler, 2012) to select a task. However, our framework does not depend on this choice.

3.3.2 Entropy Search Criterion

Entropy search works by considering $p_{\min}(\mathbf{x})$, the probability distribution over the location of the minimum of the objective function. Here, we extend the definition of $p_{\min}$ to be the probability distribution over the location of the solution to the constrained problem. Entropy search seeks the action that, in expectation, most reduces the relative entropy between $p_{\min}(\mathbf{x})$ and an uninformative base distribution such as the uniform distribution. Intuitively speaking, we want to reduce our uncertainty about $p_{\min}$ as much as possible at each step, or, in other words, maximize our information gain at each step. Following Hennig and Schuler (2012), we choose $b(\mathbf{x})$ to be the uniform distribution on the input space. Given this choice, the relative entropy of $p_{\min}$ and b is the differential entropy of $p_{\min}$ up to a constant that does not affect the choice of task. Our decision criterion is then

$$T^* = \arg\min_T \mathbb{E}_y \left[S\left(p_{\min}^{(y_T)}\right) - S(p_{\min}) \right], \qquad (10)$$

where T is one of the tasks in $\{\text{objective}, 1, 2, \ldots, K\}$, T^* is the selected task, $S(\cdot)$ is the differential entropy functional, and $p_{\min}^{(y_T)}$ is $p_{\min}$ conditioned on observing the value y_T for task T. When integrating out the GP covariance hyperparameters, the full form is

$$T^* = \arg\min_T \int S\left(p_{\min}^{(y_T)}\right) p\left(y_T|\theta,\omega\right) \mathrm{d}y_T \, \mathrm{d}\theta \, \mathrm{d}\omega \qquad (11)$$

where y_T is a possible observed outcome of selecting task T and θ and ω are the objective and constraint GP hyperparameters respectively.[5]

[5]For brevity, we have omitted the base entropy term (which does not affect the decision T^*) and the explicit dependence of $p_{\min}$ on θ and ω.

3.3.3 Entropy Search in Practice

Solving Eq. 11 poses several practical difficulties, which we address here in turn. First, estimating $p_{\min}(\mathbf{x})$ requires a discretization of the space. In the spirit of Hennig and Schuler (2012), we form a discretization of N_d points by taking the top N_d points according to the weighted expected improvement criterion. Second, $p_{\min}$ cannot be computed in closed form and must be either estimated or approximated. Swersky et al. (2013) use Monte Carlo sampling to estimate $p_{\min}$ by drawing samples from the GP on the discretization set and finding the minimum. We use the analogous method for constrained optimization: we sample from the objective function GP and all K constraint GPs, and then find the minimum of the objective for which the constraint is satisfied for all K constraint samples.

3.3.4 Incorporating cost information

Following Swersky et al. (2013), we incorporate information about the relative cost of the tasks by simply scaling the acquisition functions by these costs (provided by the user). In doing so, we pick the task with the most information gain per unit cost. If λ_A is the cost of observing task A, then Eq. 10 becomes

$$A^* = \arg\min_A \frac{1}{\lambda_A} \mathbb{E}_y \left[S\left(p_{\min}^{(y_A)}\right) - S(p_{\min}) \right] . \quad (12)$$

4 EXPERIMENTS

4.1 BRANIN-HOO FUNCTION

We first illustrate constrained Bayesian optimization on the Branin-Hoo function, a 2D function with three global minima (Fig. 1(a)). We add a decoupled disk constraint $(\mathbf{x}_1 - 2.5)^2 + (\mathbf{x}_2 - 7.5)^2) \leq 50$, shown in Fig. 1(b). This constraint eliminates the upper-left and lower-right solutions, leaving a unique global minimum at $\mathbf{x} = (\pi, 2.275)$, indicated by the orange star in Fig. 1(a). After 33 objective function evaluations and 17 constraint evaluations, the best solution is $(3.01, 2.36)$, which satisfies the constraint and has value 0.48 (true best value = 0.40).

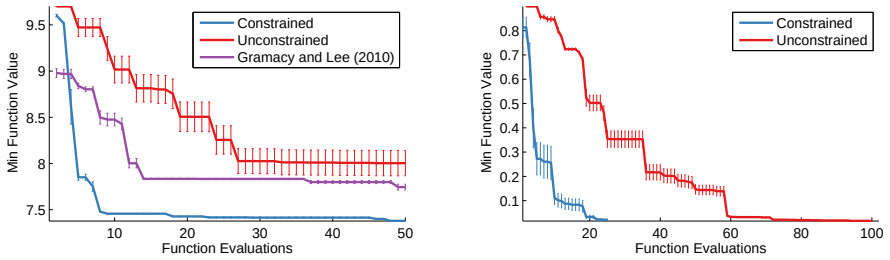
4.2 ONLINE LDA WITH SPARSE TOPICS

Online Latent Dirichlet Allocation (LDA, Hoffman et al., 2010) is an efficient variational formulation of a popular topic model for learning topics and corresponding word distributions given a corpus of documents. In order for topics to have meaningful semantic interpretations, it is desirable for the word distributions to exhibit sparsity. In this experiment we optimize the hyperparameters of online LDA subject to the constraint that the entropy of the per-topic word distribution averaged over topics is less than $\log_2 200$ bits, which is achieved, for example by allocating uniform density over 200 words. We used the online LDA implementation from Agarwal et al. (2011) and optimized five hyperparameters corresponding to the number of topics (from 2 to 100), two Dirichlet distribution prior base measures (from 0 to 2), and two learning rate parameters (rate from 0.1 to 1, decay from 10^{-5} to 1). As a baseline, we compare with unconstrained Bayesian optimization in which constraint violations are set to the worst possible value for this LDA problem. Fig. 2(a) shows that constrained Bayesian optimization significantly outperforms the baseline and the IECI method from Gramacy and Lee (2010) (see Section 1.4). Intuitively, the baseline is poor because the GP has difficulty modeling the sharp discontinuities caused by the large values.

4.3 MEMORY-LIMITED NEURAL NET

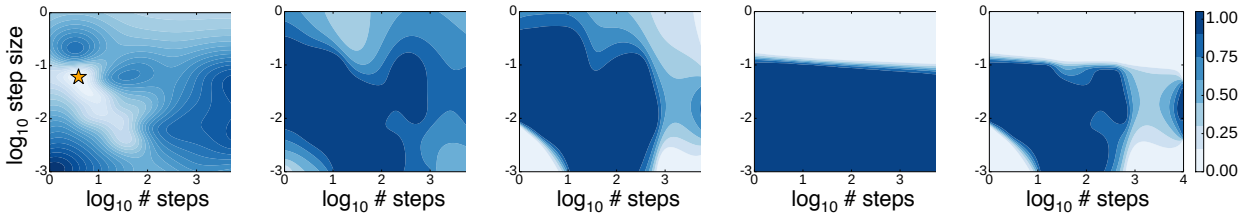
In the final experiment, we optimize the hyperparameters of a deep neural network on the MNIST handwritten digit classification task in a memory-constrained scenario. We optimize over 11 parameters: 1 learning rate, 2 momentum parameters (initial and final), the number of hidden units per layer (2 layers), the maximum norm on model weights (for 3 sets of weights), and the dropout regularization probabilities (for the inputs and 2 hidden layers). We optimize the classification error on a withheld validation set under the constraint that the total number of model parameters (weights in the network) must be less than one million. This constraint is decoupled from the objective and inexpensive to evaluate, because the number of weights can be calculated directly from the parameters, without training the network. We train the neural network using momentum-based stochastic gradient descent which is notoriously difficult to tune as training can diverge under various combinations of the momentum and learning rate. When training diverges, the objective function cannot be measured. Reporting the constraint violation as a large objective value performs poorly because it introduces sharp discontinuities that are hard to model (Fig. 2). This necessitates a second noisy, binary constraint which is violated when training diverges, for example when the both the learning rate and momentum are too large. The network is trained[6] for 25,000 weight updates and the objective is reported as classification error on the standard validation set. Our Bayesian optimization routine can thus choose between two decoupled tasks, evaluating the memory constraint or the validation error after a full training run. Evaluating the validation error can still cause a constraint violation when the training diverges, which is treated as a binary constraint in our model. Fig. 2(b) shows a comparison of our constrained Bayesian optimization against a baseline standard Bayesian optimization where constraint violations are treated as resulting in a random classifier (90% error). Only the objective evaluations are presented, since

[6]We use the Deepnet package: `https://github.com/nitishsrivastava/deepnet`.



(a) Online LDA (b) Neural Network

Figure 2: Empirical performance of constrained Bayesian optimization for (a) Online Latent Dirichlet Allocation and (b) turning a deep neural network. Blue curves: our method. Red curves: unconstrained Bayesian optimization with constraint violations as large values. Purple curve: Integrated Expected Conditional Improvement method from Gramacy and Lee (2010). Errors bars indicate standard error from 5 independent runs.



(a) Objective Function (b) Geweke (c) Gelman-Rubin (d) Stability (e) Overall

Figure 3: Tuning Hamiltonian Monte Carlo with constrained Bayesian optimization: (a) objective function model, (b-e) constraint satisfaction probability surfaces for (b) Geweke test, (c) Gelman-Rubin test, (d) stability of the numerical integration, (d) overall, which is the product of the preceding three probability surfaces. In (a), lighter colors correspond to more effective samples, circles indicate function evaluations, and the orange star indicates the best solution. Vertical axis label at left is for all subplots. Probability colormap at right is for (b-d).

constraint evaluations are extremely inexpensive compared to an entire training run. In the event that training diverges on an objective evaluation, we report 90% error. The optimized net has a learning rate of 0.1, dropout probabilities of 0.17 (inputs), 0.30 (first layer), and 0 (second layer), initial momentum 0.86, and final momentum 0.81. Interestingly, the optimization chooses a small first layer (size 312) and a large second layer (size 1772).

4.4 TUNING MCMC

Hamiltonian Monte Carlo (HMC) is a popular MCMC sampling technique that takes advantage of gradient information for rapid mixing. However, HMC contains several parameters that require careful tuning. The two basic parameters are the number of leapfrog steps τ, and the step size ϵ. HMC may also include a mass matrix which introduces $\mathcal{O}(D^2)$ additional parameters in D dimensions, although the matrix is often chosen to be diagonal (D parameters) or a multiple of the identity matrix (1 parameter) (Neal, 2011). In this experiment, we optimize the performance of HMC using Bayesian optimization; see Mahendran et al. (2012) for a similar approach. We optimize the following parameters: τ, ϵ, a mass parameter, and the fraction of the allotted computation time spent burning in the chain.

Our experiment measures the number of effective samples (ES) in a fixed computation time; this corresponds to finding chains that minimize estimator variance. We impose the constraints that the generated samples must pass the Geweke (Geweke, 1992) and Gelman-Rubin (Gelman and Rubin, 1992) convergence diagnostics. In particular, we require the worst (largest absolute value) Geweke test score across all variables and chains to be at most 2.0, and the worst (largest) Gelman-Rubin score between chains and across all variables to be at most 1.2. We use PyMC (Patil et al., 2010) for the convergence diagnostics and the LaplacesDemon R package to compute effective sample size. The chosen thresholds for the convergence diagnostics are based on the PyMC and LaplacesDemon documentation. The HMC integration may also diverge for large values of ϵ; we treat this as an additional constraint, and set $\delta = 0.05$ for all constraints. We optimize HMC sampling from the posterior of a logistic regression binary classification problem using the German credit data set from the UCI repository (Frank and Asuncion, 2010). The data set contains 1000 data points, and is normalized to have unit

Table 1: Tuning Hamiltonian Monte Carlo.

Experiment	burn-in	# steps, τ	step size, ϵ	mass	# samples	accept rate	**effective samples**
Baseline	10%	100	0.047	1	8.3×10^3	85%	1.1×10^3
BayesOpt	3.8%	2	0.048	1.55	3.3×10^5	70%	9.7×10^4

variance. We initialize each chain randomly with D independent draws from a Gaussian distribution with mean zero and standard deviation 10^{-3}. For each set of inputs, we compute two chains, each with 5 minutes of computation time on a single core of a compute node.

Fig. 3 shows constraint surfaces discovered by Bayesian optimization for a simpler experiment in which only τ and ϵ are varied; burn-in is fixed at 10% and the mass is fixed at 1. These diagrams yield interpretations of the feasible region; for example, Fig. 3(d) shows that the numerical integration diverges for values of ϵ above $\approx 10^{-1}$. Table 1 shows the results of our 4-parameter optimization after 50 iterations, compared with a baseline that is reflective of a typical HMC configuration: 10% burn in, 100 leapfrog steps, and the step size chosen to yield an 85% proposal accept rate. Each row in the table was produced by averaging 5 independent runs with the given parameters. The optimization chooses to perform very few ($\tau = 2$) leapfrog steps and spend relatively little time (3.8%) burning in the chain, and chooses an acceptance rate of 70%. In contrast, the baseline spends much more time generating each proposal ($\tau = 100$), which produces many fewer total samples and, correspondingly, significantly fewer effective samples.

5 CONCLUSION

In this paper we extended Bayesian optimization to constrained optimization problems. Because constraint observations may be noisy, we formulate the problem using probabilistic constraints, allowing the user to directly express the tradeoff between cost and risk by specifying the confidence parameter δ. We then propose an acquisition function to perform constrained Bayesian optimization, including the case where the objective and constraint(s) may be observed independently. We demonstrate the effectiveness of our system on the meta-optimization of machine learning algorithms and sampling techniques. Constrained optimization is a ubiquitous problem and we believe this work has applications in areas such as product design (e.g. designing a low-calorie cookie), machine learning meta-optimization (as in our experiments), real-time systems (such as a speech recognition system on a mobile device with speed, memory, and/or energy usage constraints), or any optimization problem in which the objective function and/or constraints are expensive to evaluate and possibly noisy.

Acknowledgements

The authors would like to thank Geoffrey Hinton, George Dahl, and Oren Rippel for helpful discussions, and Robert Nishihara for help with the experiments. This work was partially funded by DARPA Young Faculty Award N66001-12-1-4219. Jasper Snoek is a fellow in the Harvard Center for Research on Computation and Society.

References

Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system, 2011. arXiv: 1110.4198 [cs.LG].

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Bálázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS*. 2011.

Eric Brochu, Tyson Brochu, and Nando de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2010a.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, 2010b. arXiv:1012.2599 [cs.LG].

Adam D. Bull. Convergence rates of efficient global optimization algorithms. *JMLR*, (3-4):2879–2904, 2011.

Nando de Freitas, Alex Smola, and Masrour Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *ICML*, 2012.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High dimensional Gaussian Process bandits. In *NIPS*, 2013.

Andrew Frank and Arthur Asuncion. UCI machine learning repository, 2010.

Andrew Gelman and Donald R. Rubin. A single series from the Gibbs sampler provides a false sense of security. In *Bayesian Statistics*, pages 625–32. Oxford University Press, 1992.

John Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *Bayesian Statistics*, pages 169–193. University Press, 1992.

David Ginsbourger and Rodolphe Riche. Towards Gaussian process-based optimization with finite time horizon. In *Advances in Model-Oriented Design and Analysis*. Physica-Verlag HD, 2010.

Robert B. Gramacy and Herbert K. H. Lee. Optimization under unknown constraints, 2010. arXiv:1004.4027 [stat.ME].

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *JMLR*, 13, 2012.

Matthew Hoffman, David M. Blei, and Francis Bach. Online learning for latent Dirichlet allocation. In *NIPS*, 2010.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, 2011.

Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21, 2001.

Andreas Krause and Cheng Soon Ong. Contextual Gaussian Process bandit optimization. In *NIPS*, 2011.

Dan Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Canada, 2008.

Nimalan Mahendran, Ziyu Wang, Firas Hamze, and Nando de Freitas. Adaptive MCMC with Bayesian optimization. In *AISTATS*, 2012.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2, 1978.

Iain Murray and Ryan P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *NIPS*, 2010.

Iain Murray, Ryan P. Adams, and David J.C. MacKay. Elliptical slice sampling. *JMLR*, 9:541–548, 2010.

Radford Neal. Slice sampling. *Annals of Statistics*, 31: 705–767, 2000.

Radford Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.

Anand Patil, David Huard, and Christopher Fonnesbeck. PyMC: Bayesian stochastic modelling in Python. *Journal of Statistical Software*, 2010.

Carl Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

A. Shapiro, D. Dentcheva, and A. Ruszczynski. *Lectures on stochastic programming: modeling and theory*. MPS-SIAM Series on Optimization, Philadelphia, USA, 2009.

Jasper Snoek. *Bayesian Optimization and Semiparametric Models with Applications to Assistive Technology*. PhD thesis, University of Toronto, Toronto, Canada, 2013.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *ICML*, 2010.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task Bayesian optimization. In *NIPS*, 2013.

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, 2013.

Marcela Zuluaga, Andreas Krause, Guillaume Sergent, and Markus Püschel. Active learning for multi-objective optimization. In *ICML*, 2013.