Discovering and Exploiting Additive Structure for Bayesian Optimization

Jacob R. Gardner
Department of Computer Science
Cornell University
jrg365@cornell.edu

Chuan Guo
Department of Computer Science
Cornell University
cg563@cornell.edu

Kilian Q. Weinberger
Department of Computer Science
Cornell University
kqw4@cornell.edu

Roman Garnett
Computer Science and Engineering
Washington University in St. Louis
garnett@wustl.edu

Roger Grosse
Department of Computer Science
University of Toronto
rgrosse@cs.toronto.edu

Abstract

Bayesian optimization has proven invaluable for black-box optimization of expensive functions. Its main limitation is its exponential complexity with respect to the dimensionality of the search space using typical kernels. Luckily, many objective functions can be decomposed into additive sub-problems, which can be optimized independently. We investigate how to automatically discover such (typically unknown) additive structure while simultaneously exploiting it through Bayesian optimization. We propose an efficient algorithm based on Metropolis–Hastings sampling and demonstrate its efficacy empirically on synthetic and real-world data sets. Throughout all our experiments we reliably discover hidden additive structure whenever it exists and exploit it to yield significantly faster convergence.

1 INTRODUCTION

Gradient-free optimization of expensive black-box functions is a ubiquitous task in a many fields. Applications range from sensor placement [10], to robotic control [5], to hyperparameter tuning for complex machine learning algorithms such as deep convolutional neural networks [19]. These optimization tasks are challenging, because we typically have little knowledge about the objective function beyond the ability to evaluate it at a chosen location, and because the function is expensive to evaluate. For example, each lab experiment costs time and money, and evaluating a given set of hyperparameters for a deep neural network may take days or even weeks of training time.

Bayesian optimization (BayesOpt) [2, 17] is a widely used technique for optimizing expensive black-box functions. Broadly, BayesOpt operates by maintaining a probabilistic belief about the objective function. We place a prior distribution—most often a Gaussian process—over the objective function, which we condition on data as we observe it. We use this belief to drive the optimization process by sequentially evaluating the objective. At each iteration, the posterior distribution of the function is used to suggest queries to evaluate next, typically trading off exploration (i.e., querying areas with high posterior uncertainty) and exploitation (i.e., querying areas with low posterior mean).

A typical BayesOpt algorithm has two key factors that must be selected by the user: the acquisition function—how new queries are selected given the posterior belief—and the choice of prior. Whereas the former has been studied extensively, and many reasonable options exist, the choice of prior has received somewhat less attention. Almost all off-the-shelf BayesOpt solvers use general-purpose kernels, and do little in the way of discovering and exploiting structure that may underlie a particular objective function.

With poor or overly general choices of the kernel, BayesOpt may converge very slowly on complex functions, especially in moderate-to-high dimension. Hence, exploiting structure can have a substantial impact on optimization performance. For example, Snoek et al. [19] argue, for example, that for hyperparameter optimization a Matérn-5/2 kernel provides better results in general than the squared exponential kernel, because the latter models unrealistically smooth functions. Choosing (or learning) a kernel that exploits low-dimensional structure in the function can significantly improve performance when confronted with

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

10s or 100s of parameters [11, 24].

Here we seek to discover and exploit such *additive structure:* cases where the setting of some variables in the design space does not affect the optimal setting of others. For example, in hyperparameter tuning, some groups of parameters may have well-known interactions (e.g., learning rate and momentum), whereas others do not strongly interact (e.g., momentum and regularization weight). Intuitively, if different groups of parameters interact only additively, then they can be optimized independently. This intuition has been formalized many times in the context of kernels [1], generalized additive models [13], and various extensions [22]. In the context of Bayesian optimization specifically, Kandasamy et al. [14] recently showed that knowing the additive structure of a function ahead of time gives *exponential* reductions in sample complexity.

However, it is challenging in general to reason about the structure of arbitrary black-box functions *a priori.* The core contribution of our work is an efficient mechanism for discovering the underlying structure of the function *while* BayesOpt is executed. Although in this paper we focus on additive structure, we believe this technique could be used more broadly in a wide variety of settings. Rather than fixing a model *a priori*, we make use Bayesian model selection to, at each iteration, sample an additive structure consistent with the data observed so far. We deal with the large number of possible additive structures by using a Metropolis–Hastings scheme to sample from the model posterior, and demonstrate that this mechanism quickly samples additive structures that explain the data well.

We evaluate our method both in-model and on several benchmark optimization problems with varying additive structure. The experiments validate empirically that our approach significantly outperforms both standard BayesOpt and the random-bag-of-models exploration scheme introduced by Kandasamy et al. [14]. On synthetic test functions, we show that our approach discovers the correct structure even for functions in a moderately large number of dimensions and is substantially faster than existing structure-discovery techniques. Finally, we evaluate the efficacy of our proposed algorithm on two real-world applications: a matrix completion task and an astrophysics simulation experiment to estimate physical constants to high accuracy. In both cases, our method converges faster and finds better optimal solutions than prior work.

2 RELATED WORK

The idea that modeling functions can be done more efficiently by exploiting underlying additive structure is well known in general [1]. Generalized additive models (GAMs) are linear models that explicitly model a response variable as a linear combination of univariate functions [13]. These results have been extended to the setting where some component functions depend on more than one input variable, allowing for general additive structure [22]. In the context of Gaussian process regression in machine learning, several papers exist that exploit general forms of additive structure in the literature. Duvenaud et al. [7] introduced an additional generaliztion of GAMs that allows for "higher-order" additive kernels (GAMs corresponding to first-order additive kernels).

Kandasamy et al. [14] demonstrated that Bayesian optimization using additive Gaussian processes achieves lower regret than fully dependent models. They showed that Bayesian optimization using a prior that exploits additive structure in the objective function has *theoretically* significantly lower sample complexity than with a kernel that does not exploit this structure. In their paper, the authors proposed evaluating a number of randomly selected additive structures—which we will call the *bag-of-models* approach—and choosing the structure that explains any observed data the best. The primary goal of our work is to show that this random search is inefficient and can be improved dramatically by performing an *informed* search in model space in tandem with the optimization procedure.

The problem of choosing covariance functions that result in better models than basic kernels has been well studied in general. One approach to this is to compose simple kernels (for example, the RBF kernel, the linear kernel, etc) using simple operations like addition and multiplication [18]. Duvenaud et al. [8] proposed to search over possible compositions by constructing a *grammar* of possible kernel compositions, starting with basic base kernels and producing more complex kernels via composition. This tree is then greedily searched over to construct a kernel with high model evidence. Malkomes et al. [16] extended this idea, replacing the greedy search with Bayesian optimization in model space, defining a novel "kernel kernel" to reason about the similarity between data explanations offered by different kernels and speed up the search. Rather than constructing complex models from simple ones, Wilson & Adams [26] took a different approach, and introduced a highly flexible kernel by using Bochner's theorem to write any stationary kernel as the Fourier transform of a finite measure, and parameterizing a class of stationary kernels by using a Gaussian mixture as the spectral density. Similar to our setting, Gardner et al. [9] considered actively discovering the structure of a function by sequentially querying as few points as possible. However, the techniques in that paper are not used for optimization.

Choosing generally more informative priors in Bayesian optimization has received some attention as well. For example, Swersky et al. [21] considered using information gained from functions related to the objective to guide optimization. This occurs, for example, in hyperparameter tuning when hyperparameters on subsampled data can be used to inform about the validation error on the full dataset.

3 BACKGROUND

Bayesian optimization (BayesOpt) is a technique for optimizing an expensive black-box function [17]: $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. It has become popular in the machine learning community in recent years for its application to hyperparameter tuning [19], but as a strong global optimization algorithm with many theoretical guarantees [3, 20], it has also been successfully applied in many other areas [5, 6, 10, 15]. For a comprehensive overview of BayesOpt see [2]; we give a brief overview of the main components below.

The core idea of BayesOpt is to deal with the fact that evaluating $f(\mathbf{x})$ is expensive by constructing a model of f given samples, and then using this model to decide where to sample next. This new sample reveals new information about the function, and our model can therefore be updated; the process then repeats. Intuitively, if the model of f is perfect and can be optimized cheaply, then the function itself can be optimized cheaply. However, when we are faced with very few and noisy samples the model will typically be far from perfect. It is therefore critical that the model has well-calibrated uncertainty estimates.

Gaussian processes are the most common choices to model f, as they naturally provide uncertainty estimates with their predictions [18]. One can view the use of Gaussian process regression as placing a prior over the objective function f. It is nonparametric, parametrized by the choice of a mean function μ and covariance function k:

$$f \sim \mathcal{GP}\big(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big). \tag{1}$$

Under this assumption, given any finite set of points $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_m]$, the vector of function values $\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_n)]$ is multivariate Gaussian distributed:

$$p(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}\big(\mathbf{f}; \mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big). \tag{2}$$

Typically, Gaussian process regression is performed by assuming a Gaussian noise observation model: for each training point $\mathbf{x}_i$, we assume that we observe the true function value offset by some Gaussian noise. That is, $y_i = f(\mathbf{x}_i) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. With this model, the posterior after conditioning on some dataset $\mathcal{D}$ is itself a Gaussian process, where the predictive distribution $p(f(\mathbf{x}^*)|\mathcal{D}, \mathbf{x}^*)$ for a test point $\mathbf{x}^*$ is Gaussian distributed:

$$p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*) = \mathcal{N}\big(\mu_{f|\mathcal{D}}(\mathbf{x}^*), k_{f|\mathcal{D}}(\mathbf{x}^*, \mathbf{x}^*)\big); \tag{3}$$
$$\mu_{f|\mathcal{D}}(\mathbf{x}) = \mu(\mathbf{x}) + k(\mathbf{x}, \mathbf{X})(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\big(\mathbf{f} - \mu(\mathbf{X})\big);$$
$$\sigma_{f|\mathcal{D}}(\mathbf{x}, \mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - k(\mathbf{x}, \mathbf{X})(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}k(\mathbf{X}, \mathbf{x}).$$

Acquisition functions. After conditioning on data, we use the predictive distribution (3) to evaluate how promising each candidate $\mathbf{x}^*$ is using some *acquisition function.* One of the most popular acquisition functions used in BayesOpt is the *expected improvement*. Let f_{best} be the best function value observed so far. Define $I(f(\mathbf{x}^*))$ as the *improvement* obtained by sampling at $\mathbf{x}^*$:

$$I\big(f(\mathbf{x}^*)\big) = \max\big(0, f_{\text{best}} - f(\mathbf{x}^*)\big) \tag{4}$$

The EI acquisition function evaluates the expectation of the above function over the predictive posterior (3):

$$\begin{aligned}\text{EI}(\mathbf{x}^*) &= \mathbb{E}\left[I\big(f(\mathbf{x}^*)\big)\right]_{p(f(\mathbf{x}^*)|\mathcal{D},\mathbf{x}^*)} \\ &= \int_{-\infty}^{f_{\text{best}}} \big(f_{\text{best}} - f(\mathbf{x}^*)\big) p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*)\, \mathrm{d}f(\mathbf{x}^*)\end{aligned}$$

The expected improvement may be evaluated analytically. Further, the form of the expected improvement has two terms that can naturally be interpreted as encouraging both *exploitation*, i.e.,points that our model suggest have low predictive mean $\mu_{f|\mathcal{D}}(\mathbf{x})$, and *exploration*, *i.e.* points with high posterior variance $\sigma_{f|\mathcal{D}}(\mathbf{x}, \mathbf{x})$. To maximize the acquisition function, it is typically evaluated on a fine grid of candidate points within a pre-defind hyper-cube. Although the evaluation of a single candidate point is very fast, it is important to emphasize that this search scales exponentially with the number of dimensions and can become prohibitively slow in regimes of 10 or more dimensions.

3.1 Model Selection

When using Gaussian process regression to model a function f, the choice of covariance function k is critical, as it allows the data scientist to encode information about the structure of f. However, doing so manually may require expert knowledge about both the function being modeled as well as about kernel methods in general. Bayesian model selection alleviates this need by providing a mechanism to perform inference over the covariance function from a set of candidate kernels.

Suppose we are given a set of observed data $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ and a set of kernel functions to choose from, $\mathcal{M}_1, ..., \mathcal{M}_m$, with corresponding hyperparameters θ_i. The first object of interest in Bayesian model selection is the *model evidence* of a model, $p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_i)$. It represents the likelihood of having drawn the dataset $\mathcal{D}$ from a Gaussian process with kernel $\mathbf{K}_{\mathcal{M}_i}$. Computing the model evidence exactly is in general not analytically tractable for Gaussian processes. One option is to use the MLE hyperparameters:

$$\log p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_i) \approx p(\mathbf{y} \mid \mathbf{X}, \hat{\theta}_i, \mathcal{M}_i). \tag{5}$$

The log marginal likelihood for a model $\mathcal{M}_i$ and hyperparameters θ_i can be computed analytically:

$$\log p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_i, \theta_i) = -\tfrac{1}{2}\mathbf{y}^\top \mathbf{K}_{\mathcal{M}_i}^{-1}\mathbf{y} - \tfrac{1}{2}\log\big((2\pi)^n\,|\mathbf{K}_{\mathcal{M}_i}|\big).$$

Because the log marginal likelihood does not penalize model complexity, the *Bayesian information critereon*

(BIC) is often used to penalize the number of hyperparameters $|\theta_i|$ and better approximate the model evidence:

$$-\tfrac{1}{2}\text{BIC} = \log p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_i, \theta_i) - \tfrac{1}{2}|\theta_i| \log(|\mathcal{D}|). \quad (6)$$

In our experiments, we found that the difference in log marginal likelihood between "correct" additive models and "incorrect" additive models was so large that the penalty term $-\frac{1}{2}|\theta_i| \log\big(|\mathcal{D}|\big)$ did not have a sigificant impact. Given some prior distribution over the possible kernels, $p(\mathcal{M})$, applying Bayes' theorem to the above, and marginalizing out θ_i, results in the *model posterior*,

$$p(\mathcal{M}_i \mid \mathcal{D}) = \frac{p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_i)p(\mathcal{M}_i)}{\sum_j p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_j)p(\mathcal{M}_j)}, \quad (7)$$

a probability distribution over the possible models given the data. The model posterior provides us with a direct measure of how well each model in the set explains the data. It allows us to generalize eq. (2) to compute a predictive distribution marginalized over all possible models:

$$\mathbb{E}\left[p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*, \mathcal{M})\right]_{p(\mathcal{M}|\mathcal{D})} = \sum_i p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*, \mathcal{M}_i)p(\mathcal{M}_i \mid \mathcal{D}).$$

3.2 Additive Structure

Recent work [14] has shown that exploiting known additive structure can significantly accelerate the convergence of Bayesian optimization, both theoretically and empirically. Consider a d-dimensional function $f(x_1, ..., x_d)$. Suppose that $f(\mathbf{x})$ decomposes into some partitioning of the dimensions, P, i.e.

$$f(\mathbf{x}) = \sum_{i=1}^{|P|} f_i\big(\mathbf{x}[P_i]\big) \quad (8)$$

where $\mathbf{x}[P_i]$ denotes $\mathbf{x}$ restricted to the dimensions in P_i. If P is the partition where each $i = 1, \ldots, d$ is its own part, we say the f is *fully additive*. As a clarifying example, to say a 5-dimensional function decomposes into the partitioning $P : \big\{[1, 3, 4], [2], [5]\big\}$ means there exist functions f_1, f_2, f_3 so that:

$$f(\mathbf{x}) = f_1(x_1, x_3, x_4) + f_2(x_2) + f_3(x_5). \quad (9)$$

Intuitively, f is much easier to optimize if this structure is known, *even if the functions $f_1, \ldots, f_{|P|}$ cannot be directly evaluated.* For example, the 5-dimensional optimization problem above could be solved as one 3-dimensional optimization problems and two 1-dimensional problems. In the context of BayesOpt, known additive structure can be readily encoded in a Gaussian process prior. In particular, if a function $f(x)$ decomposes into $\sum_{i=1}^{|P|} f_i(\mathbf{x}[P_i])$, then, given a base kernel $k(x, x')$, the additive kernel $\sum_{i=1}^{|P|} k(\mathbf{x}[P_i], \mathbf{x}'[P_i])$ gives rise to a prior with support over functions with this additive structure.

4 MODEL SELECTION VIA MCMC

The number of possible additive decomposition models in d dimensions is given by the dth Bell number B_d [25], which grows super-exponentially, and for $d = 10$ already reaches $B_{10} = 115\,975$. This makes computing the model posterior in eq. (7) prohibitively expensive, as it involves conditioning a Gaussian process with each of these models to compute the model evidences. However, computing a small (i.e., not super-exponential) number of model evidences is tractable. A natural alternative to computing the model posterior is to sample from it using Metropolis–Hastings, which requires only one additional model evidence computation per sample.

Proposal distribution. As we focus on additive structure throughout this section, a model is uniquely defined by its partitioning of the underlying dimensions, and with a slight abuse of notation we refer to the partitioning and the resulting model both as $\mathcal{M}$. The primary component of Metropolis–Hastings that needs to be specified is the *proposal distribution* $g(\mathcal{M}' \mid \mathcal{M})$. Given an additive structure partitioning $\mathcal{M}$, there are two natural operations that can be performed.

First, an existing element of the partition can be split in two. For example, if $[1, 2, 3] \in \mathcal{M}$, we can form $[1][2, 3], [2][1, 3], [3][1, 2]$ by splitting. In general, a component of size k can be split in 2^{k-1} different ways.

Second, two existing elements of the partition may be merged. For example, $[1, 4]$ and $[2, 3]$ can be merged to form $[1, 2, 3, 4]$. In general, a partition with k components has $\binom{k}{2}$ possible merges.

With these operations in mind, we construct a proposal distribution as illustrated in Figure 1. We first choose whether to split or merge, each with 50% probability. Next, if we split (left sub-tree), we choose a component of the existing partition $\mathcal{M}$ uniformly at random and split it in two (again uniformly at random). If we instead merge (right sub-tree), we choose two components of M uniformly without replacement and merge them. Given this proposal mechanism, sampling from the proposal distribution $g(\mathcal{M}' \mid \mathcal{M})$ is straightforward and efficient.

5 STRUCTURE DISCOVERY AND EXPLOITATION

The goal of our work is to discover the additive model structure underlying the objective function $f(\mathbf{x})$, while simultaneously exploiting it for BayesOpt. At each iteration i of BayesOpt, we will have collected some dataset of function evaluations $\mathcal{D}_i = \{\mathbf{X}_i, \mathbf{y}_i\}$. BayesOpt, as described in section 3, uses this data to update the posterior $p(f \mid \mathcal{D}_i)$ and then to identify the new point $\mathbf{x}^*$, which maximizes

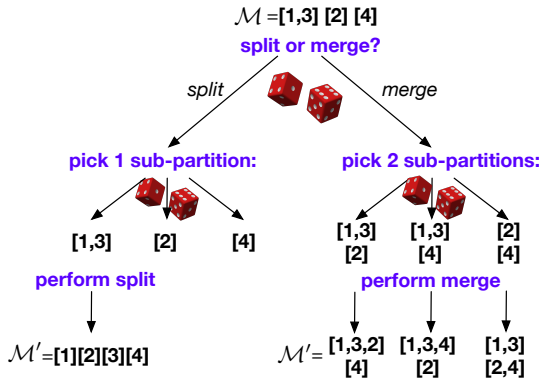

Figure 1: An illustration of the proposal distribution $g(\mathcal{M}' \mid \mathcal{M})$ on a simple input model with three sub-partitions, $\mathcal{M} = [1,3][2][4]$. Splits with dice represent choices performed uniformly at random (without replacement). See text for details.

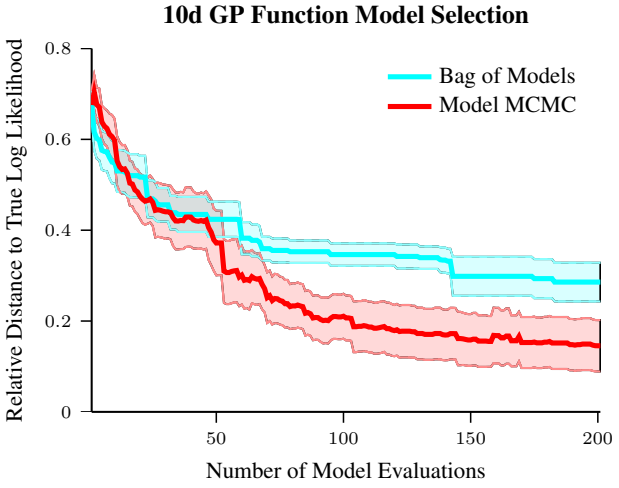

Figure 2: Plot of the relative distance to the true loglikelihood as a function of the number of model evaluations (see text). For MCMC, this means the number of samples drawn in the Markov chain. For bag of models, this means the number of models in the selected bag. Error bars are standard error averaged over 10 runs.

$\text{EI}(\mathbf{x}^*)$ and for which $f(\mathbf{x}^*)$ should be evaluated next.

The expected improvement $\text{EI}(\mathbf{x}^*)$ is a function of $p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*)$. As we are considering multiple models, each providing us with a different posterior over f, we approximately marginalize out the model. We sample k models $\mathcal{M}_1, \ldots, \mathcal{M}_k$ from $p(\mathcal{M} \mid \mathcal{D}_i)$ to obtain

$$p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*) \approx \frac{1}{k} \sum_{j=1}^{k} p(f(\mathbf{x}^*) \mid \mathcal{D}, \mathbf{x}^*, \mathcal{M}_j). \quad (10)$$

MCMC. To obtain these samples, we use the Metropois-Hastings algorithm with the proposal distribution $g(\mathcal{M} \mid \mathcal{M}')$. First observe that the model posterior is proportional to the model evidence $p(\mathbf{y} \mid \mathbf{X}, \mathcal{M})$, by eq. (7), which can be efficiently evaluated for single models. We can therefore sample k models as follows: Given the current model $\mathcal{M}_j$ (initializing $\mathcal{M}_0$ to the final model found in the previous iteration), we sample a proposed model $\mathcal{M}'$ from the proposal distribution $g(\mathcal{M}' \mid \mathcal{M}_j)$. Next, we compute the model evidence for $\mathcal{M}'$ and use this to compute the Metropolis-Hastings acceptance probability:

$$A(\mathcal{M}' \mid \mathcal{M}_j) = \min\left(1, \frac{p(\mathbf{y}_i \mid \mathbf{X}_i, \mathcal{M}') g(\mathcal{M}_j \mid \mathcal{M}')}{p(\mathbf{y}_i \mid \mathbf{X}_i, \mathcal{M}_j) g(\mathcal{M}' \mid \mathcal{M}_j)}\right).$$

Finally, we update the current state to $\mathcal{M}'$ with probability $A(\mathcal{M}' \mid \mathcal{M}_i)$.

Candidate selection. As pointed out at the end of section 3, finding the point $\mathbf{x}_i^*$ to maximize $\text{EI}(\mathbf{x}^*)$ has exponential sample complexity with the number of dimensions and can be slow. In the presence of additive structure, this search can be decomposed and performed for each component individually—leading to exponential speed-ups. To do this, we start with some initial $\mathbf{x}_i^*$ and iteratively optimize EI in each component of the partition. For example, if the first component of the partition is [1,3,5], we first optimize the first, third, and fifth dimensions of $\mathbf{x}_i^*$, holding the other dimensions fixed. If the next component is [4], we optimize EI by varying the 4th dimension (holding other dimensions fixed), and so on.

As each model $\mathcal{M}_i$ has its own additive structure, we identify the best $\mathbf{x}_i^*$ for each of the k models,

$$\mathbf{x}_i^* = \arg\max_{\mathbf{x}} \text{EI}(\mathbf{x}_i \mid \mathcal{M}_i). \quad (11)$$

We consider the resulting points $\mathbf{x}_1, \ldots \mathbf{x}_k$ as candidate points and set $\mathbf{x}^* = \mathbf{x}_i$ for the particular i that maximizes the marginalized acquisition function $\text{EI}(\mathbf{x}_i^*)$ using eq. (10).

6 RESULTS

For all the experiments, we used the Python GP toolbox GPy [12]. We will make our code publicly available at `https://github.com/jrg365/bayesopt_model_mcmc`.

6.1 Model Selection

Here we demonstrate the effectiveness of MCMC search in model space. We sample a random partition P of $\{1, \ldots, d\}$ and sample observations from a Gaussian process with additive structure corresponding to P. In particular, we take $N = 50$ points from a scrambled Halton sequence on $[0, 1]^d$, and sample corresponding function values $f(\mathbf{x})$ from the Gaussian process. We then perform MCMC model search using these N observations. The initial kernel is the fully dependent kernel. Figure 2 shows

plots of the Markov chain length versus model evidence achieved with that chain length for $d = 10$. We compare with the bag-of-models (BOM) proposed by Kandasamy et al. [14]. For each method we plot, as a function of the number of models evaluated, the *relative distance* to the true model. This is given by $1 - \frac{\log p(\mathbf{y}|\mathbf{X},\mathcal{M}_i,\theta_i)}{\log p(\mathbf{y}|\mathbf{X},\mathcal{M}^*,\theta^*)}$, where $\mathcal{M}^*$ is the true model and $\mathcal{M}_i$ is the model evaluated at iteration i.

MCMC consistently finds additive decompositions within 15–17% of the ground-truth log likelihood. In 8 out of 10 runs, MCMC samples the ground-truth model exactly. BOM consistently fails to find models with a relative log likelihood distance of less than 30%. This experiment demonstrates that MCMC is an effective technique for exploring the space of additive structures, at least when the underlying function is well modeled by a Gaussian process. Specifically, we can expect it to outperform the BOM when used as a subroutine in Bayesian optimization when additive structure is critical to the optimization task.

6.2 Optimization

We next incorporate MCMC model search into the BayesOpt framework. In all experiments below, we sample $k = 50$ models using the MCMC techniques discussed above at each iteration. For the Bag of Models (BOM) baseline, we use a bag of 50 models. Thus, the MCMC approach and the bag of models approach perform the same number of MLE optimizations of the log likelihood $\log p(\mathbf{y} \mid \mathbf{X}, \mathcal{M}_i, \theta_i)$. Instead of performing *burn-in*—discarding initial low-likelihood samples—we initialize the Markov chain at each iteration with the final model sampled at the previous iteration. Therefore, the MCMC and BOM approaches add virtually identical amounts of overhead to the baseline BayesOpt algorithm, and the wall-clock time required for each iteration is essentially identical for both methods.

We do note that the amount of overhead added by performing model selection is not inconsequential for cheap functions f. Indeed, considering k models at each iteration results in roughly a factor k increase in the overhead added by BayesOpt. However, in many real world settings (like the cosmological constants experiment and the matrix factorization experiment below), the overhead of running BayesOpt is insignificant compared to evaluating the objective function f) as long as only a relatively small number of optimization iterations are run.

In all experiments, the squared exponential (SE) kernel is used as the base kernel.

6.3 Optimization of Benchmark Functions

We first consider two standard optimization benchmark functions that have additive structure: the Styblinski–Tang function, and the Michalewicz function. The d-dimensional Styblinski–Tang function is defined as

$$\text{Stybtang}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{d} x_i^4 - 16x_i^2 + 5x_i.$$

The global minimum is approximately $-39.166d$ and is attained at point $\mathbf{x}^* \approx (-2.9, \ldots, -2.9)$. We restrict the domain of the function to $[-4, 4]^d$ for the optimization.

The d-dimensional Michalewicz function is defined as

$$\text{Michalewicz}(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i)\sin^{2m}\left(\frac{ix_i}{\pi}\right).$$

Here, m controls the steepness of valleys and ridges. For this experiment, we set $m = 10$. For $d = 10$, the global minimum is approximately -9.66 over the domain $[0, \pi]^d$.

In addition to these two functions, we extend the Styblinski–Tang function, which fully additively decomposes, to a *transformed* Styblinski–Tang function: We sample a random partition P and for each part i of P, we sample a random orthonormal matrix Q_i over the dimensions of part i. If Q is the block diagonal matrix formed by placing each Q_i on a diagonal, then Stybtang(Qx) is no longer fully additive, but instead is additive across the components of P. We use this to investigate performance when the true function is not fully additive across individual dimensions.

We compare against three baselines. The fully dependent model uses a single d-dimensional kernel as the model. This is the approach used by popular BayesOpt packages such as Spearmint [19]. The oracle model uses a sum of squared-exponential kernels whose parts capture the true additive structure of the underlying function. For all experiments, we expect the oracle model to outperform all other methods since it has prior knowledge of the true function structure. Our third baseline is BOM [14]. In all experiments, BayesOpt+Model MCMC significantly outperforms the non-oracle approaches, both in terms of convergence speed and ultimate objective value.

Figure 3 left shows the plot of optimizing the Styblinski–Tang function. Each curve shows the mean cumulative minimum function evaluation up to the current iteration across 10 runs, while the shaded region shows the standard error. It can be seen that our method attains the true global minimum as quickly as the oracle model, while outperforming BOM. The fully dependent model does not converge within 200 iterations.

Figure 3 right shows the plot of optimizing the Michalewicz function. Since the function has many steep valleys, it is much more difficult to model with a Gaussian process than the Styblinski–Tang function. Nevertheless, both the oracle model and our method converge to the global optimum within 300 iterations while BOM has not

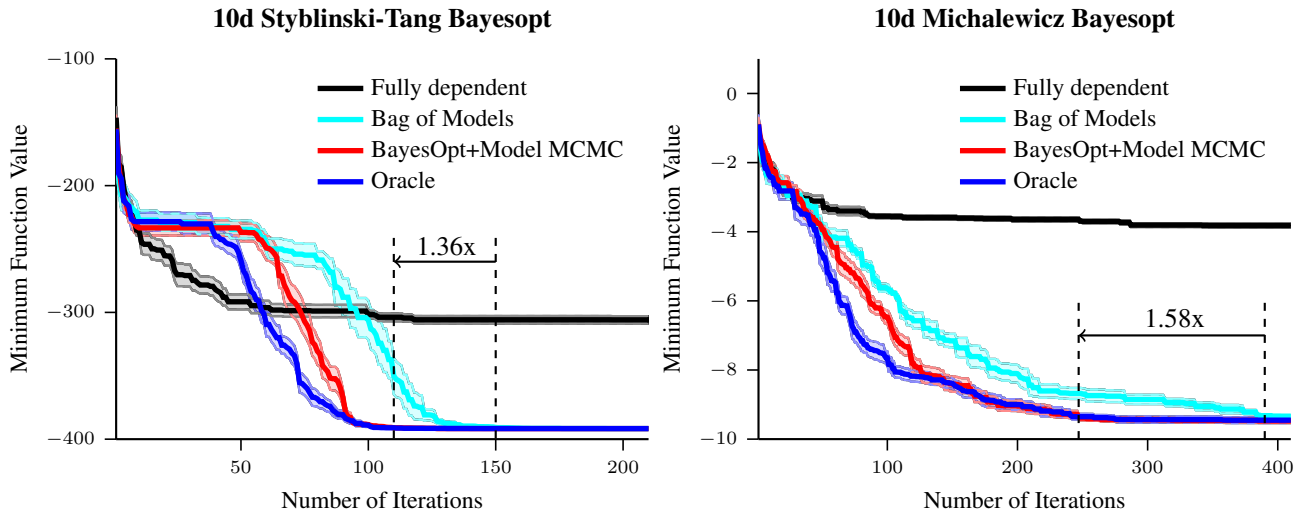

Figure 3: Optimizing the 10d Styblinksi-Tang and transformed Styblinksi-Tang function. Shaded error bars are 2 standard errors over 10 runs. BayesOpt+Model MCMC converges faster than BOM in both cases.

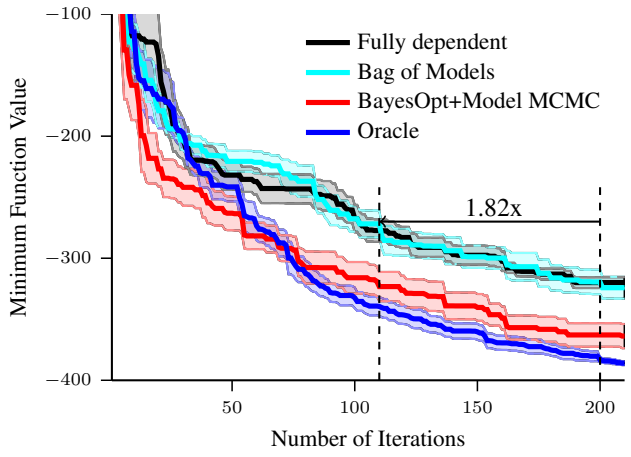

Figure 4: Optimizing the 10d transformed Styblinksi-Tang function. Shaded error bars are 2 standard errors over 10 runs. BayesOopt+Model MCMC converges to the same final solution as BOM much faster, and significantly outperforms it in terms of final objective value

yet converged after 400 iterations. Again, the fully dependent model shows no sign of convergence.

In the transformed Styblinski-Tang experiment (Figure 4), the more difficult model selection problem poses a significant challenge to the BOM method, which does not significantly outperform the baseline fully dependent model. In contrast, Model MCMC successfully discovers appropriate additive structure and performs significantly better. Of particular interest is the first 50 iterations where Model MCMC outperforms the baseline. This happens in cases where the ground truth model has large non-additive components, which makes the EI optimization problem significantly more difficult for the oracle method. The Model MCMC method averages over many additive structures which, while incorrect, have much simpler structure and for which EI can be more efficiently optimized.

6.4 Determining Cosmological Parameters

In this section, we test our method on the task of determining the values of various cosmological constants (e.g., Hubble's constant, the density of baryonic matter in the universe, etc). These cosmological constants are values that are required in standard physical models of the universe, but their exact values must be determined experimentally. To do this, scientists can run simulations with various settings of these cosmological constants and evaluate how well these simulations model experimentally observed data.

We use software released by NASA[1] that, given settings of these cosmological constants, computes via simulation the likelihood of a set of experimental data released by the Sloan Digital Sky Survey. As in [14], we tune the 9 parameters that this software takes as input and produces the negative log likelihood. However, unlike [14], we do not introduce placeholder dimensions that have no effect on the objective function value.

To set ranges for each of the parameters, we take the values set in a parameter file shipped with the software, and optimize over a range of $75\% - 125\%$ of this default value for each parameter. The results of this experiment are in Figure 5. The parameter values shipped with the software achieve an objective function value of 23.7. All three methods find statistically significantly better solutions by 200 iterations, with the BayesOpt+Model MCMC approach performing the best, consistently converging to an optimal negative log likelihood of 22.17. Furthermore, BayesOpt+Model MCMC converges much faster, achieving the global solution found by the bag of models method nearly 40% faster.

[1] https://lambda.gsfc.nasa.gov/toolbox/lrgdr/

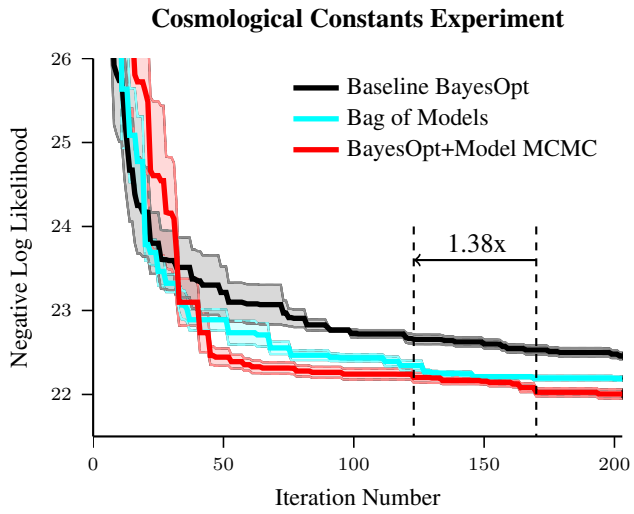

Figure 5: Setting values of various cosmological constants to match experimental data using BayesOpt. Shaded regions correspond to 2 standard errors over 10 runs. BayesOpt+Model MCMC matches BOM 38% faster, and then outperforms it.

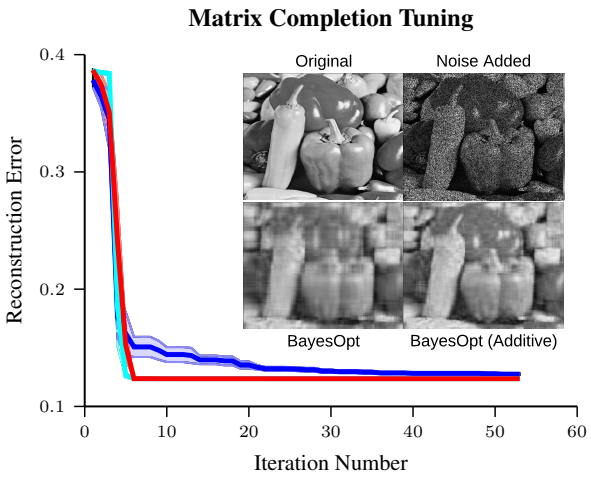

Figure 6: Optimizing the reconstruction error of images with matrix completion. Corner: Example of image reconstruction on the peppers image using hyperparameters found by both the baseline and by BayesOpt with additive structure. The additive BayesOpt parameters produce a significantly sharper image.

6.5 Matrix completion

Many of the experiments in Kandasamy et al. [14] and our paper here have focused on functions with 10 or more dimensions. However, there are also many machine learning algorithms with only a small number (3 or fewer) of hyperparameters. Although with these few dimensions, employing the machinery of MCMC to explore the model space is not necessarily critical (indeed, BOM reduces to evaluating all possible additive decompositions and choosing the best), we seek to demonstrate two things: First, exploiting additive structure can still have a substantial impact on performance. Second, even in a scenario where BOM is optimal (i.e. the best additive structure is chosen each iteration), our sampling approach matches its performance.

We evaluate our method on the task of choosing hyperparameters for a matrix completion algorithm in [4]: a singular-value decomposition-based method for recovering a matrix with missing entries which can be used for image denoising, recommendation, and data interpolation, among other applications. We set the following hyperparameters using standard BayesOpt, our method, and BOM: the regularization trade-off τ, the step-size δ, and the noise-constraint ε. A plot of the average validation reconstruction error on an image reconstruction task as a function of the number of BayesOpt iterations is displayed in figure 6.

Both additive structure mechanisms significantly outperform the standard BayesOpt baseline, both in terms of convergence speed and in terms of final objective function value. To assess whether the hyperparameters made a difference in terms of image reconstruction quality, we use the hyperparameters found by each method to recover the benchmark "peppers" image [23] after noise has destroyed 50% of the pixels. The corrupted image as well as the reconstructions are in figure 6. The image reconstructed using the additive Bayesian optimization methods is significantly sharper than the image reconstructed using the non-additive approach, with less blurring and much sharper edges.

7 CONCLUSION

In this paper we introduced an integrated solution to discover and exploit additive structure while performing BayesOpt. Our algorithm is based on MCMC sampling of model candidates and in practice converges surprisingly fast for plausible and useful model decompositions. Given the drastic speed-ups that can be achieved through exploitation of additive structure, we hope that our algorithm will become a standard component of Bayesian optimization. We hope to extend this work to allow data-driven modeling choices beyond additive structure.

8 ACKNOWLEDGEMENTS

JRG, CG, and KQW are supported in part by the III-1618134, III-1526012, and IIS-1149882 grants from the National Science Foundation, as well as the Bill and Melinda Gates Foundation. RG (WUSTL) is supported by the National Science Foundation (NSF) under award number IIA-1355406.

References

[1] Aronszajn, Nachman. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):

337–404, 1950.

[2] Brochu, E., Cora, V. M., and De Freitas, N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[3] Bull, Adam D. Convergence rates of efficient global optimization algorithms. *The Journal of Machine Learning Research*, 12:2879–2904, 2011.

[4] Cai, Jian-Feng, Candès, Emmanuel J, and Shen, Zuowei. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[5] Calandra, Roberto, Peters, Jan, Seyfarth, Andre, and Deisenroth, Marc P. An experimental evaluation of bayesian optimization on bipedal locomotion. In *International Conference on Robotics and Automation*, 2014.

[6] Carr, Shane, Garnett, Roman, and Lo, Cynthia. BASC: Applying Bayesian Optimization to the Search for Global Minima on Potential Energy Surfaces. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 898–907, 2016.

[7] Duvenaud, David K, Nickisch, Hannes, and Rasmussen, Carl E. Additive gaussian processes. In *NIPS*, 2011.

[8] Duvenaud, David K, Lloyd, James Robert, Grosse, Roger B, Tenenbaum, Joshua B, and Ghahramani, Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *ICML*, 2013.

[9] Gardner, Jacob, Malkomes, Gustavo, Garnett, Roman, Weinberger, Kilian Q, Barbour, Dennis, and Cunningham, John P. Bayesian active model selection with an application to automated audiometry. In *NIPS*, 2015.

[10] Garnett, Roman, Osborne, Michael A, and Roberts, Stephen J. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010.

[11] Garnett, Roman, Osborne, Michael A, and Hennig, Philipp. Active learning of linear embeddings for gaussian processes. In *UAI*, 2014.

[12] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[13] Hastie, Trevor J and Tibshirani, Robert J. *Generalized additive models*, volume 43. CRC press, 1990.

[14] Kandasamy, Kirthevasan, Schneider, Jeff, and Poczos, Barnabas. High dimensional bayesian optimisation and bandits via additive models. In *ICML*. JMLR Workshop and Conference Proceedings, 2015.

[15] Mahendran, Nimalan, Wang, Ziyu, Hamze, Firas, and Freitas, Nando D. Adaptive mcmc with bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 751–760, 2012.

[16] Malkomes, Gustavo, Schaff, Chip, and Garnett, Roman. Bayesian optimization for automated model selection. In *Advances in Neural Information Processing Systems 29*, 2016.

[17] Mockus, J., Tiesis, V., and Zilinskas, A. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

[18] Rasmussen, C. E. and Williams, C.K.I. Gaussian processes for machine learning. MIT Press, 2006.

[19] Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *NIPS 2012*, pp. 2960–2968. 2012.

[20] Srinivas, Niranjan, Krause, Andreas, Kakade, Sham M, and Seeger, Matthias. Gaussian process optimization in the bandit setting: No regret and experimental design. *ICML*, 2010.

[21] Swersky, Kevin, Snoek, Jasper, and Adams, Ryan P. Multi-task bayesian optimization. In *NIPS*, 2013.

[22] Wahba, Grace. *Spline models for observational data*. SIAM, 1990.

[23] Wang, Zheng, Lai, Ming-Jun, Lu, Zhaosong, Fan, Wei, Davulcu, Hasan, and Ye, Jieping. Orthogonal rank-one matrix pursuit for low rank matrix completion. *SIAM Journal on Scientific Computing*, 37(1):A488–A514, 2015.

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

[25] Wilf, Herbert S. *Generatingfunctionology*. A. K. Peters, Ltd., Natick, MA, USA, 2006. ISBN 1568812795.

[26] Wilson, Andrew Gordon and Adams, Ryan Prescott. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, 2013.