Bayesian Optimization Exploiting Monotonicity and Its Application in Machine Learning Hyperparameter Tuning

Wenyi Wang WENYW@CS.UBC.CA
Department of Computer Science, University of British Columbia

William J. Welch WILL@STAT.UBC.CA
Department of Statistics, University of British Columbia

Abstract

We propose an algorithm for a family of optimization problems where the objective can be decomposed as a sum of functions with monotonicity properties. The motivating problem is optimization of hyperparameters of machine learning algorithms, where we argue that the objective, validation error, can be decomposed into two monotonic functions of the hyperparameters. Our proposed algorithm adapts Bayesian optimization methods to incorporate monotonicity constraints. We illustrate the improvement in search efficiency using examples of machine learning hyperparameter tuning applications.

Keywords: Bayesian Optimization, Machine Learning Hyperparameter Tuning, Monotonic Gaussian Process Regression

1. Introduction

Bayesian optimization (BO) has been successfully applied to many global optimization problems (Jones et al., 1998; Martinez-Cantin et al., 2007; Hutter et al., 2011; Snoek et al., 2012). Typically, it makes few assumptions about the objective function, treating it as a black box. When prior knowledge is available, however, it may be possible to improve the efficiency of the optimization search. In particular, function monotonicity has been successfully exploited to improve accuracy in statistical modeling for the analysis of computer experiments (e.g., Golchi et al., 2015). The methods proposed here employ such monotonicity information when tuning a machine learning (ML) algorithm with respect to its hyperparameters. A sequential search adapts the BO framework by decomposing a complex objective function into a sum of functions with monotonicity constraints, which are easier to model. We analyze the applicability to ML hyperparameter tuning problems and provide positive experimental results.

The rest of this paper is organized as follows. In section 2 we provide a brief overview of BO using Gaussian processes (GPs) and previous work on monotonicity constraints. Section 3 describes the basics of the proposed method: objective decomposition and monotonicity. In section 4 we argue that these ideas can be applied to ML hyperparameter tuning and compare the proposed algorithm with a standard BO approach. Finally, in section 5 we make some concluding remarks and discuss future work.

2. Related Work

2.1. Bayesian Optimization

BO is a sequential algorithm for global optimization. At each iteration it builds a statistical model of the objective function based on the evaluations so far; the model guides the location of the next function evaluation. Compared with local search, it tends to use more (often all) available function evaluations to determine the next. This makes BO particular useful when objective evaluations are limited by computational cost. Furthermore, it requires little knowledge of the properties (e.g., convexity) of the objective function.

A typical BO algorithm may be outlined as follows. Let $\mathbf{x} = (x_1, \ldots, x_d)$ denote a vector of d parameters (e.g., ML hyperparameters) in a d-dimensional region of interest $\mathcal{X}$ and $f(\mathbf{x})$ the objective function to minimize (e.g., validation error). The algorithm starts with an initial set of vectors $\mathbf{x}^{(i)}$ for $i = 1, \ldots, n$ sparsely filling $\mathcal{X}$ and their corresponding function evaluations $y^{(i)} = f(\mathbf{x}^{(i)})$. A statistical model $\hat{f}(\mathbf{x})$ to approximate $f(\mathbf{x})$ is trained using the data $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$. Common choices for $\hat{f}(\mathbf{x})$ include GP regression, which we employ exclusively, and random forests. A key advantage of GP regression is that $\hat{f}(\mathbf{x})$ provides a prediction of $f(\mathbf{x})$ at any $\mathbf{x}$ in $\mathcal{X}$ along with a measure of uncertainty. The next $\mathbf{x}$ to be evaluated maximizes an acquisition function $A(\mathbf{x}, \hat{f})$ based on the statistical properties of $\hat{f}(\mathbf{x})$. It balances local search (choose an $\mathbf{x}$ with small predicted $f(\mathbf{x})$) versus global search (choose an $\mathbf{x}$ where the prediction of $f(\mathbf{x})$ has large uncertainty). Expected improvement, probability of improvement, and upper confidence bound (this criterion is minimized) are typically used for the acquisition function. The point $\mathbf{x}$ maximizing $A(\mathbf{x}, \hat{f})$ is appended to D (hence n is incremented), the statistical model $\hat{f}(\mathbf{x})$ is retrained, and the updated $A(\mathbf{x}, \hat{f})$ is used for the search for another point. The algorithm iterates until the acquisition function shows little further gain is to be had or the computational budget is exhausted. For a comprehensive overview see Brochu et al. (2010). In this paper we adapt GP regression and the expected improvement for better accuracy and therefore optimization performance when $f(\mathbf{x})$ can be decomposed as a sum of functions with monotonicity properties.

2.2. Gaussian Processes with Monotonicity Constraints

As discussed in section 2.1, GP regression builds a probabilistic model of a function based on limited evaluations. It is a robust model that can be applied to a large class of functions; it only assumes the target function behaves like a sample path from the GP. The GP itself has parameters, which are trained using the observed data D via maximum likelihood or Bayesian MCMC. The trained GP leads to predictions by considering sample paths conditional on D. For more details see Rasmussen and Williams (2006).

When priori knowledge of the target function is available, performance of the model can be improved. In this paper, we study functions that can be decomposed as a sum of monotonic functions, and model the decomposed functions separately. There are various approaches to GP modeling subject to (approximate) monotonicity constraints (Lin and Dunson, 2014; Riihimäki and Vehtari, 2010; Wang and Berger, 2016). We employ Riihimäki and Vehtari (2010)'s implementation because it is readily accessible.

Riihimäki and Vehtari (2010) incorporate monotonicity by introducing derivative information at v virtual points, $\tilde{\mathbf{x}}^{(i)}$ for $i = 1, \ldots, v$, in $\mathcal{X}$. The objective function is not evaluated at the virtual points, rather they are used to favor a GP $\hat{f}(\mathbf{x})$ with positive (or negative) derivatives at these points in all directions for which monotonicity is assumed. For definiteness, take a target function that increases with respect to x_j. Subject to some differentiability conditions, the derivative of a GP is a related GP, and hence has statistical properties. The contribution to the likelihood from the derivative of $\hat{f}(\mathbf{x})$ with respect to x_j at $\tilde{\mathbf{x}}^{(i)}$ is taken to be $P\left(\frac{\partial \hat{f}}{\partial x_j}(\tilde{\mathbf{x}}^{(i)}) > z\right) = \Phi(z/\nu)$, where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution and $\nu \geq 0$ is a constant scalar. The likelihood increases with a more positive derivative. For the experiments of this paper, we set $\nu = 0.1$ so that monotonicity is only approximate ($\nu = 0$ would strictly enforce monotonicity). The likelihood incorporating all such derivative contributions as well as those from the actual function evaluations is used to train the GP parameters and for prediction.

3. Problem Formulation and Proposed Algorithm

Our focus is an objective function f that can be written as a sum of K functions,

$$f(\mathbf{x}) = \sum_{k=1}^{K} f_k(\mathbf{x}), \tag{1}$$

where K is known, with monotonicity constraints

$$f_k(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n) \leq f_k(x_1, \ldots, x_{j-1}, x'_j, x_{j+1}, \ldots, x_n) \tag{2}$$

or

$$f_k(x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_n) \geq f_k(x_1, \ldots, x_{j-1}, x'_j, x_{j+1}, \ldots, x_n) \tag{3}$$

for some k, j and all feasible $x_j < x'_j$ and $(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$. Whether the function $f_k(\mathbf{x})$ is monotonically non-decreasing or non-increasing with respect to x_j must be known. Our algorithm needs to observe each function $f_k(\mathbf{x})$ at chosen $\mathbf{x}$ values in $\mathcal{X}$. Even if $f_k(\mathbf{x})$ is observed without noise, we assume a GP with an additional white-noise term for computational robustness.

BO (section 2.1) is adapted so that instead of modeling $f(\mathbf{x})$ directly, we model each $f_k(\mathbf{x})$ separately using independent monotonicity-constrained GP regressions (section 2.2). The predictive mean of $\hat{f}(\mathbf{x})$ at any trial $\mathbf{x}$ is then the sum of the predictive means of the $\hat{f}_k(\mathbf{x})$, and because of the assumed independence of the component GPs, the predictive variance of the sum is similarly the sum of the variances. Once the predictive mean and variance of $\hat{f}(\mathbf{x})$ have been computed in this way, the expected improvement of a new evaluation at $\mathbf{x}$ can be computed in the standard way. Algorithm 1 summarizes this procedure.

For all the experiments presented, each component GP is assumed to have zero mean, constant variance for the correlated process, an isotropic squared-exponential correlation function, and constant variance for the white-noise. Monotonicity information is also incorporated if known. The search starts from $n = 4$ initial evaluations since there are three parameters to be estimated for each GP regression: the constant variance of the correlated process, the scale parameter of the squared exponential correlation function, and the constant variance of the white noise.

Algorithm 1 Bayesian Optimization with Monotonicity Information

Input: the objective function decomposed as $\{f_k\}$, the region of interest $\mathcal{X}$, n initial points $\mathbf{x}^{(i)}$, v virtual points $\tilde{\mathbf{x}}^{(i)}$, and the acquisition function $A(\mathbf{x}, \hat{f})$
Output: X and Y, lists for the points where the objective is evaluated and the respective function values
Initialize $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\}$
For all i, k, evaluate $y_{ik} = f_k(\mathbf{x}^{(i)})$, hence $y_i = \sum_k y_{ik}$ for all i
Initialize $Y_k = \{y_{k1}, \ldots, y_{kn}\}$ for all k, and $Y = \{y_1, \ldots, y_n\}$
repeat
  For all k, train the GP $\hat{f}_k$ using X, Y_k and the monotonicity constraints at $\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(v)}$
  $\hat{f}(\mathbf{x}) = \sum_k \hat{f}_k(\mathbf{x})$
  Determine the next evaluation point $\mathbf{x}^{(n+1)} = \arg\max_{x \in \mathcal{X}} A(x, \hat{f})$
  For all k, evaluate $y_{n+1,k} = f_k(\mathbf{x}^{(n+1)})$, hence $y_{n+1} = \sum_k y_{n+1,k}$
  Append $\mathbf{x}^{(n+1)}$ to X, $y_{n+1,k}$ to Y_k for all k, y_{n+1} to Y, and increment n by 1
until maximum number of iterations reached or some stopping criterion is satisfied
return X and Y

4. Application to Machine Learning Hyperparameter Tuning

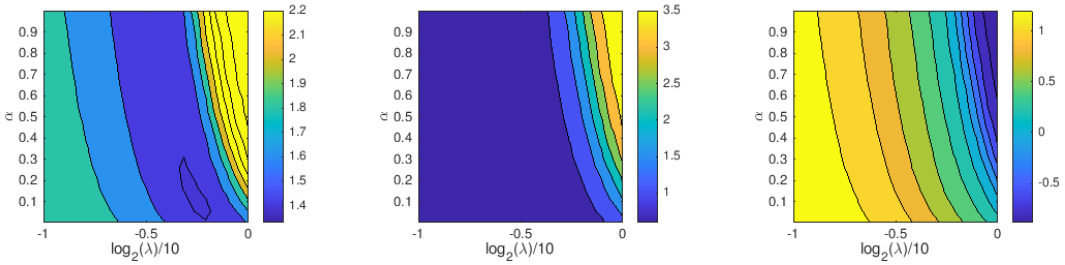
In practice, users minimize the validation error of an ML algorithm with respect to its hyperparameters, as the generalization error is unavailable. Usually it is unreasonable to assume that the validation error is monotonic with respect to any given hyperparameter. Rather than optimizing validation error directly, we decompose it as training error plus the difference between validation and training errors. We claim that these component functions are reasonably assumed to be monotonic with respect to ML model complexity and hence any hyperparameter x_j controlling complexity, i.e., $K = 2$ in (1), and x_j satisfies (2) or (3) for $k = 1, 2$. If the user is not prepared to make such assumptions about behavior with respect to x_j, there is no contribution to the likelihood from x_j at the virtual points.

For many ML algorithms, the training error is monotonically decreasing as model complexity increases. The training error is also monotonic with respect to regularizing parameters, which can also be thought of as hyperparameters. Hence monotonicity of training error with respect to hyperparameters widely exists.

Analysis of monotonicity of the difference between validation and training errors is more subtle, and to the best of our knowledge there are no general results. However the notion that generalizability (inverse of the difference between generalization and training error) is decreasing with respect to effective model complexity leads to the heuristic argument used implicitly by experts in manual ML tuning. They increase or decrease hyperparameters using monotonicity assumptions and relativities between target error, training error, and validation error minus training error (Goodfellow et al., 2016, pp. 424, 425).

4.1. Elastic Net Regularized Linear Regression

Consider the elastic net regularized linear regression algorithm (Zou and Hastie, 2005). With training data X_{tr} and y_{tr}, it finds $\hat{\beta}$ to minimize the loss $l(\beta) = \frac{1}{2|y_{tr}|}\|y_{tr} - \beta X_{tr}\|_2^2 + \lambda\left(\frac{(1-\alpha)}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1\right)$. The hyperparameters λ and α are to be tuned.



(a) Validation error (b) Training error (c) Validation error - training error

Figure 1: Error of elastic net regularized linear regression as a function of the hyperparameters α and λ.

In the following experiment, all elements of X_{tr} and X_v are sampled independently from the standard normal distribution, with 200 examples in each set and 100 features for each example. The underlying linear transformation is defined by a 100×1 vector C with 50 zero elements; the remaining 50 nonzero elements are drawn from a normal distribution with mean zero and standard deviation 0.22. Data y_{tr} and y_v follow independent normal distributions with means CX_{tr} and CX_v, respectively, and standard deviation 1. The parameters for generating the data are chosen to make the optimization problem nontrivial. The hyperparameter ranges of interest are $\alpha \in [0, 1]$, and $\lambda \in 2^{([-10,0])}$.

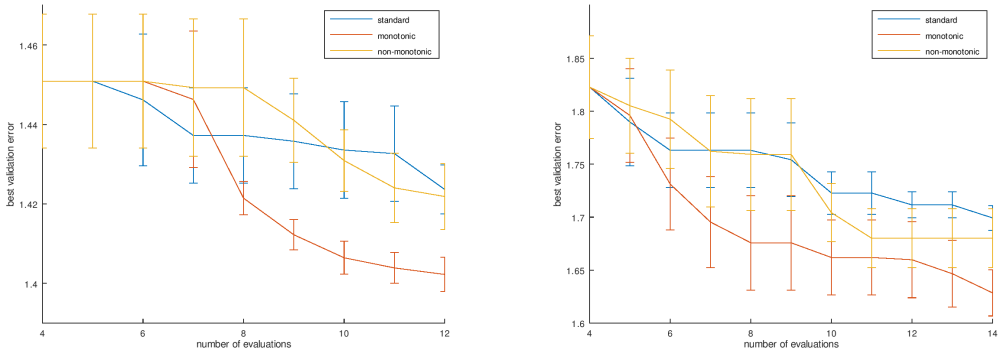
Figure 1(a) shows the validation error as a function of α and λ from exhaustive evaluation. It is clearly not monotonic. To optimize the validation error, however, we decompose it as the sum of the training error and the difference between validation error and training error. Figures 1(b) and 1(c) show that these two components are (at least roughly) monotonic, satisfying our assumptions.

To examine the roles of function decomposition versus monotonicity, we also consider a modification of our algorithm without monotonicity modeling. Thus, the methods compared are (1) standard, (2) decomposition and monotonicity (the proposed algorithm), and (3) decomposition without monotonicity. Each is applied 10 times, starting from different initial function evaluations at four random points. A 10×10 grid of virtual points is used for monotonicity.

Figure 2(a) shows the results from the three algorithms. The mean of the best validation error averaged over the 10 trials is plotted versus the number of evaluations. The error bars show ± 1 standard error of the sample mean. We see that the performance of the algorithm employing decomposition and monotonicity dominates that of the other two statistically. From the experiment we also observe that the standard and decomposition-only algorithms start by evaluating the functions at the corners. In contrast, the algorithm exploiting monotonicity does not always do so. In addition, its GPs usually have less variance, which is reasonable due to incorporating more prior knowledge.

4.2. Convolutional Neural Network (CNN) on CIFAR-10 Dataset

Deep learning has enjoyed wide success in many areas in the past decade, but a challenge is to select the relatively large number of hyperparameters. Some of them define the effective model complexity, where it is reasonable to assume monotonicity as discussed in general. In



(a) Elastic net application (b) CNN application

Figure 2: Best validation error found versus number of evaluations for three algorithms. Ten trials are averaged, and the error bars show ± 1 standard error of the mean.

this work, we tune three hyperparameters: the L_1 regularizer $\alpha_1 \in [0, 1]$, the L_2 regularizer $\alpha_2 \in [0, 1]$, and an integer $n \in [4, 68]$ that controls the number of neurons for a 5-layered CNN. The target problem is the CIFAR-10 image classification dataset (Krizhevsky, 2009). A systemic study of CNN image classification and its application to this data set can be found in Krizhevsky et al. (2012).

We construct CNNs with one input layer, two convolutional layers, a fully connected layer, and a softmax output layer. The two convolutional layers consist of n and $2n$ feature maps with 3×3 size, ReLU activation functions, and 2×2 max pooling. The fully connected layer adapts ReLU activation functions with $n^2/2$ neurons. Each CNN is trained by stochastic gradient decent on L_1 and L_2 regularized cross entropy with constant learning rate 0.02, decay factor 10^{-6}, momentum 0.9 and 30 epochs. Since training the whole data set takes substantial time, for demonstration purposes we randomly select 2000 examples as a training set. We use cross entropy as the performance measure on the validation set since it is used as a part of the objective function for training.

We apply the three algorithms, with virtual points on a $4 \times 4 \times 4$ grid for monotonicity. Figure 2(b) shows experimental result of running the three algorithms 10 times from four initial objective function evaluations. We see that the algorithm imposing monotonicity again has the best performance. We also observe that the algorithm exploiting decomposition and monotonicity makes improvement earlier compared with the other two algorithms.

5. Conclusions

We have proposed a BO algorithm for objective functions that can be decomposed as a sum of functions with monotonicity constraints and argued that many ML hyperparameter tuning problems fall into this family. We provided experimental results for regularized linear regression and a convolutional neural network for image classification. They show our algorithm outperforms the standard BO method based on direct modeling of the validation error. Future work includes adaptive choice of virtual points for monotonicity, using a projection method for monotonic GP regression and design of an acquisition function that uses monotonicity information.

Acknowledgements

This research is supported in part by NSERC Discovery grant RGPIN-2014-04962.

References

Eric Brochu, Vlad M. Cora, and Nando de Freitas. 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.

Shirin Golchi, Derek R. Bingham, Hugh Chipman, and David A Campbell. Monotone emulation of computer experiments. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):370–392, 2015.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. *LION*, 5:507–523, 2011.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Lizhen Lin and David B. Dunson. Bayesian monotone regression using Gaussian process projection. *Biometrika*, 101(2):303–317, 2014.

Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems III*, pages 334–341, 2007.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT press, 2006.

Jaakko Riihimäki and Aki Vehtari. Gaussian processes with monotonicity information. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 645–652, 2010.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Xiaojing Wang and James O. Berger. Estimating shape constrained functions using Gaussian processes. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):1–25, 2016.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.