EXPLORING MULTI-FIDELITY BAYESIAN OPTIMIZATION FOR HYPERPARAMETER TUNING

by

Herbert Wright

Contribution Paper

In

HONOR 3200

ABSTRACT

Hyperparameters are usually set manually, but due to their importance, there has been much work trying to develop algorithms to optimally select such hyperparameters. Bayesian optimization, a Bayesian approach to black-box optimization, is a common approach to hyperparameter optimization. State-of-the-art Bayesian optimization methods take into account function evaluations at different fidelities. This is known as multi-fidelity Bayesian optimization and there are many different approaches to it that have been proposed.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Hyperparameter Optimization

Many machine learning algorithms attempt to construct a model by optimizing some objective function. The behavior of such a machine learning model can be defined through a set of parameters. The algorithm optimizes over some of these parameters in order to construct its model during the learning process. Other parameters, which define the behavior of the algorithm are generally not optimized during this process [1]. These parameters are called *hyperparameters*, and must be selected by some other means. Examples of hyperparameters include the learning rate when training a neural network, the number of decision trees in a random forest, and the kernel used in a support vector machine [2].

The setting of hyperparameters can often be the deciding factor between good results and state-of-the-art results [3]. Usually they are selected manually by rules of thumb [4], but because of their importance, there has been much work into developing algorithms to optimally select such hyperparameters so that the learning process will produce the best model. Because we can view the success of the model produced by an algorithm as a function of its hyperparameters, we can apply optimization techniques to find optimal hyperparameters [4].

A couple of the most basic optimization algorithms that can be used in hyperparameter optimization are grid search and random search, however these approaches can become inefficient if it is computationally expensive to evaluate the function being optimized (if the learning process is computationally expensive in the case of hyperparameter optimization) [5]. One state-of-the-art optimization algorithm that performs well with costly function evaluations is *Bayesian optimization* [5]–[7]. Other types of algorithms can also be used, such as methods which maintain a population of possible solutions [5].

1.2 Bayesian Methods

In statistics and probability there are two common interpretations of the concept of probability; frequentist and Bayesian. The frequentist perspective views probability as the relative frequency of a certain event over a theoretically infinite amount of trials. The Bayesian perspective instead views probability as a degree of belief [1], [8]. Such a belief is updated when new information is introduced to you. A common example of the difference in these perspectives is to imagine flipping a fair coin. Before you flip the coin, both perspectives would say there is a 50% chance of heads, however after the coin is flipped, but before you look at it, the Bayesian perspective would still say there is a 50% probability of heads whereas the frequentist perspective would not assign a probability, as the coin has already been flipped.

This idea is quantified through Bayes theorem, a fundamental concept in probability. The theorem states that, if Ω is the sample space, $H, D \in 2^{\Omega}$ are events, and $P : \Omega \to \mathbb{R}$ is a probability measure, then

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)},$$

where $P(H|D)$ denotes the conditionaly probability of H given D and is defined as $P(H|D) = \frac{P(H \cap D)}{P(D)}$ (and similarly for $P(D|H)$) [9]. While the theorem is still valid under a frequentist view of probability, the Bayesian perspective lends a more philosophically impactful interpretation. The Bayesian interpretation of the theorem is that $P(H)$ is the *prior* probability (the prior probability ascribed to some hypothesis H), $P(D|H)$ is the *likelihood* (how likely observed data, D, is, given some hypothesis H), and $P(H|D)$ is the *posterior* probability (the adjusted belief about some hypothesis H, given data D). The idea is that your prior belief should be updated to a posterior belief based on new evidence.

The process of estimating some parameter, θ, with this method would start with assigning a prior distribution over possible values this parameter could take ($P(\theta)$). Often, this prior is very spread out [1]. Then, collecting informative data and calculating the likelihood of seeing such data given each possible value ($P(D|\theta)$). The posterior distribution would be calculated by multiplying the prior by the likelihood for each θ, then scaling all values so that $P(\theta|D)$ is a valid probability distribution ($P(\Omega) = 1$ where Ω is the sample space) [8]. Often, the set of values a parameter can take is continuous, so the equation for the posterior probability density function becomes

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int_{\mathbb{R}^n} p(D|\theta)p(\theta)d\theta},$$

[8], [9] where the posterior ($p(\theta|D)$) and prior ($p(\theta)$) are probability density functions over θ and $p(D|\theta)$ gives the relative likelihood of observed data given θ as the parameter value. Bayesian methods like this can be used to estimate parameters of a machine learning model, such as the coefficients in linear regression [1].

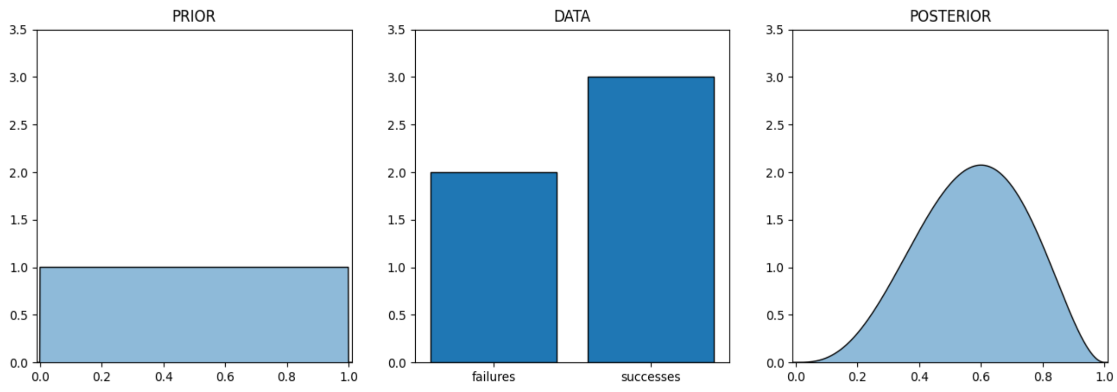

Figure 1: *Example of Bayesian updating from prior (left) to posterior (right) from observed data (middle)*

An example of Bayesian updating is shown in the above figure, where the parameter being estimated is the probability of a success in a Bernoulli trial (a Bernoulli trial is like a coin flip between success and failure, where the probability is not necessarily 50%). In the example, the prior distribution chosen is a uniform distribution over $[0, 1]$ (left). Then, five trials are run, with three successes and two failures (middle). Using this data, D, the posterior distribution is calculated (right). Notice that if the probability of success, θ, is 0 or 1, then $P(D|\theta) = 0$, because it would be impossible to get a mix of successes and failures. Thus, the posterior distribution takes the value of 0 at both 0 and 1.

2 BACKGROUND

2.1 Bayesian Optimization

One common black-box optimization algorithm is Bayesian optimization (BO). BO uses a probabilistic surrogate model of functions to model the objective function $f : \mathcal{X} \to \mathbb{R}$ (the function being optimized) [10]. The goal is to find $x^* = \underset{x \in \mathcal{X}}{\arg\min} f(x)$ [11]. The algorithm begins by placing a prior over different possible functions that f could be. Then, the algorithm iteratively selects a point $x_i \in \mathcal{X}$ based on some *acquisition function* (or a set of x_i's), then calculates $f(x)$, which is used to update the surrogate model into a posterior distribution over possible functions f could be, given the known function values [11].

While there are different possible surrogate models for BO, such as Bayesian neural networks, the most commonly used is a Gaussian process (GP) [6], [10]. A GP is a set of random variables such that for any finite subset of them, their joint distribution is a Gaussian distribution [12]. A GP, $f \sim \mathcal{GP}(m, k)$ is defined in terms of a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a mean function $m : \mathcal{X} \to \mathbb{R}$ [6]. Given a finite set of points, $X \in \mathcal{X}^n$, a GP provides a distribution for $f(X)$ that takes the following form: $f(X) \sim \mathcal{N}(m(X), K(X, X))$ (where $f(X), m(X)$ denotes the vector with f, m applied to each element of X and $K(X, X)$ is the covariance matrix where $K_{i,j} = k(X_i, X_j)$) [12]. Once the values of f for certain x are known, a posterior GP can be calculated. If $\mathcal{D} = (X, y)$ are data/function evaluations such that for all i, $f(X_i) = y_i$, then the posterior GP, given by $\mathcal{GP}(m_\mathcal{D}, k_\mathcal{D})$, has the following closed form:

$$m_\mathcal{D}(x) = m(x) + K(X, x)^T K(X, X)^{-1} y$$

$$k_\mathcal{D}(x, x') = k(x, x') - K(X, x)^T K(X, X)^{-1} K(X, x')$$

[5], [11], [12]. This follows from conditional probability in multivariate Gaussians.

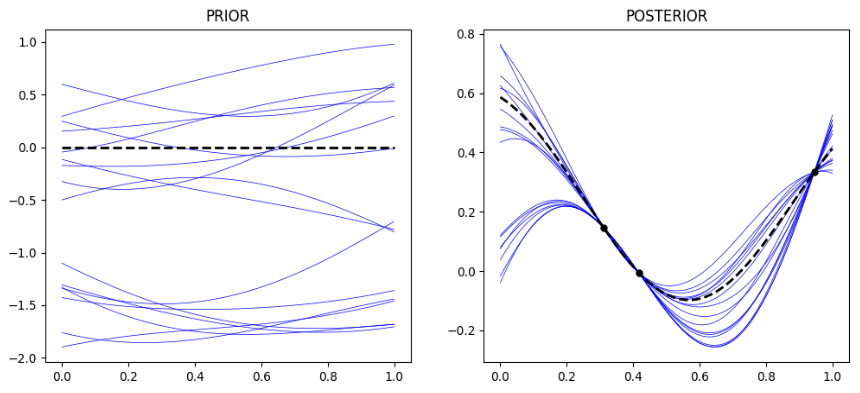

Figure 2: *16 draws from a GP prior (left) and posterior (right) calculated from a Bayesian update from three function evaluations. The black dashed line is the mean of the respective GP.*

The figure above shows a GP before and after being updated with function values. GP's can also be computed with an assumption of Gaussian noise at function evaluations [12]. One example of Gaussian processes being used is in estimating pollution levels in Salt Lake City with noisy sensor data [13].

Another important part of BO is the selection of the acquisition function. An acquisition function uses the posterior surrogate model in order to select new points to evaluate the objective function at. One common function is expected improvement, where the x to query next is found by $x_{n+1} = \arg\max_{x \in \mathcal{X}} E[\max(f_{\min} - Y, 0)]$, where Y is the current distribution of $f(x)$ and $f_{\min} = \min\{f(x_1), f(x_2), ..., f(x_n)\}$ [14] ($E[\cdot]$ denotes expected value). Another approach is select x by maximizing (or minimizing) an upper-confidence bound (or lower-confidence bound) (UCB) [15]. A more recent acquisition function is the max-value entropy search, where x is selected with $x_{n+1} = \arg\max_{x \in \mathcal{X}} I(Y, f^*)$, where f^* is the global minimum of a function drawn from the surrogate model and $I(\cdot, \cdot)$ denotes mutual information (a measure of how much one of the random variables tells you about the other) [16].

2.2 Bayesian Neural Networks

Artificial Neural Networks (ANNs) are a family of machine learning models. The simplest kind of ANN is called a multilayer perceptron (MLP) and is parameterized by weights, W_i, and biases, b_i. An MLP constructs a function $f : \mathbb{R}^n \to \mathbb{R}^m$ through a series of intermediary vectors $h_i \in \mathbb{R}^p$ that can be defined like so:

$$h_{i+1} = \phi(W_i h_i + b_i),$$

where $h_0 = x$ is the input, $h_q = f(x)$ is the output, and $\phi : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function [1], [9]. The number of layers, the size of those layers, the choice of an optimization function, and the initialization of weights are examples of hyperparameters. During the training process, there are other hyperparameters, such as choice of optimization algorithm and number of epochs of training [2]. The number of epochs when training an ANN is simply the number of times the optimization algorithm loops through the dataset during the optimization process.

Usually, the parameters of an ANN, denoted by θ (the weights and biases), are specific values, however in something called a *Bayesian neural network* (BNN), θ is modeled by a probability distribution [9]. In a BNN, a prior distribution is placed over θ, and a posterior distribution is calculated for θ given some data using Bayes theorem. Because some of the calculations involved are difficult, often Markov chain Monte Carlo or Variational methods are used to train BNNs in practice [9].

2.3 Multi-Fidelity Bayesian Optimization

With most hyperparameter optimization algorithms, including BO, a function evaluation means computing the entire machine learning algorithm [1]. This can be very costly. In order to combat this cost, variants of BO have been introduced which allow for less expensive approximations of the objective function to be queried in order to efficiently find optimal values. These approximations are known as low-fidelity function evaluations [17]. BO approaches that support these evaluations are known as *multi-fidelity Bayesian optimization* algorithms [7].

2.4 The Bandit Problem

The bandit problem asks the player to find, given a set of unknown probability distributions, $\{D_1, D_2, ..., D_k\}$, the one with the highest expected value. The player is allowed to draw $r_i \sim D_j$ from one of the distributions at each turn; the value drawn is called the reward [18]. Bandit algorithms are concerned with finding optimal ways to select which distribution to draw from in order to maximize the reward. Because rewards are drawn from unknown distributions, they can model noisy function evaluations like in *Gaussian process bandit optimisation with multi-fidelity evaluations* by Kandasamy *et al.* [19].

3 CURRENT METHODS

3.1 Hyperband

In *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*, a 2018 journal article, hyperparameter optimization is phrased as an infinitely armed bandit problem [20]. The article introduces a novel algorithm called the hyperband algorithm. hyperband iteratively calls another algorithm called successive halving with different parameters each time based on a finite amount of resources. The successive halving algorithm starts with a batch of canditate x values, then evaluates them at each fidelity, throwing away poor performing x values as the fidelity increases [21]. Hyperband performs a geometric search over different runs of successive halving, where the proportion of canditate x values dropped is varied (along with the corresponding number of starting x values) [20].

3.2 MF-GP-UCB

In *Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations*, a 2016 paper, The MF-GP-UCB algorithm is introduced [19]. MF-GP-UCB stands for multi-fidelity, Gaussian process, and upper confidence bound respectfully. MF-GP-UCB assumes that the objective function at different fidelities takes the form of an infinitely armed bandit drawn from a GP. The algorithm then models each of these different fidelities with seperate GPs. Then, the algorithm selects the next point to query with a modified UCB acquisition function that first attempts ot upperbound the acquisition function with high probability for each fidelity, then selects the next x value to query the objective function based on the tightest of these bounds. The fidelity at which to query is selected by finding the lowest fidelity where the GP has a standard deviation at the chosen x that is above some threshold determined by the level of fidelity and iteration the algorithm is on [19].

3.3 MF-MES

In *Multi-fidelity Bayesian Optimization with Max-value Entropy Search and its Parallelization*, a 2020 paper in ICML, the MF-MES algorithm is introduced [22]. The algorithm models every fidelity with one GP, where the kernel used takes different fidelities into account. A variant of max value entropy is used as the acquisition function, where the x and fidelity to query next is selected based on the mutual information of the optimal value at the highest fidelity and the distribution of the GP at x with the specified fidelity, adjusted by the cost of that fidelity [22].

3.4 BMBO-DARN

In *Batch Multi-Fidelity Bayesian Optimization with Deep Auto-Regressive Networks*, a conference paper published in NeurIPS 2021, the BMBO-DARN algorithm is proposed [23]. BMBO-DARN uses a series of BNNs instead of a GP as a surrogate model for BO. This is done so that more complex relationships between different fidelities can be expressed. The series of BNNs model distributions of the objective function at different fidelities. Each BNN in the sequence takes as input the output of all previous BNNs as well as the original input. The paper also proposes a batch acquisition function, which allows for a batch of x values with which to evaluate the objective function to be proposed (instead of a single x value). BMBO-DARN shows improvement over various other approaches to multi-fidelity optimization [23].

3.5 Other Notable Approaches

Another notable multi-fidelity BO algorithm is MF-PES, which models different fidelities with a convolved multi-ouput GP, and uses a modified predicive entropy search for its acquisition function [24]. On the other hand, DNN-MFBO models different fidelities with ANNs that have a probabalistic final layer [10]. In [7], the taKG acquisition function is introduced, which takes into account how an observation at a certain fidelity affects all the posterior at lower fidelites at that value. A different algorithm, called MF-MI-Greedy, explores low fidelity evaluations, before picking an x value to query at a high fidelity [25].

4 CONCLUSION

Hyperparameters exist in almost every machine learning algorithm [1]. Being able to optimized hyperparameters allow the construction of better machine learning models. One state-of-the-art approach is known as Bayesian optimization, and is instantiated in many different ways [5]. One flavor that has gained popularity is multi-fidelity Bayesian optimization. In order to understand the methods used in these algorithms, many difficult math concepts must be understood. This essay has provided an overview of a few of these concepts and related them to current multi-fidelity hyperparameter optimization algorithms.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] G. Luo, “A review of automatic selection methods for machine learning algorithms and hyper-parameter values,” *Network Modeling Analysis in Health Informatics and Bioinformatics*, vol. 5, no. 1, pp. 1–16, 2016.

[3] F. Hutter, J. Lücke, and L. Schmidt-Thieme, “Beyond manual tuning of hyperparameters,” *KI-Künstliche Intelligenz*, vol. 29, no. 4, pp. 329–337, 2015.

[4] M. Claesen and B. De Moor, “Hyperparameter search in machine learning,” *arXiv preprint arXiv:1502.02127*, 2015.

[5] M. Feurer and F. Hutter, “Hyperparameter optimization,” in *Automated machine learning*, Springer, Cham, 2019, pp. 3–33.

[6] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *Advances in neural information processing systems*, vol. 25, 2012.

[7] J. Wu, S. Toscano-Palmerin, P. I. Frazier, and A. G. Wilson, “Practical multi-fidelity bayesian optimization for hyperparameter tuning,” in *Uncertainty in artificial intelligence*, 2020, pp. 788–798.

[8] M. E. Glickman and D. A. van Dyk, “Basic bayesian methods,” *Topics in Biostatistics*, pp. 319–338, 2007.

[9] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, “Hands-on bayesian neural networks–a tutorial for deep learning users,” *arXiv preprint arXiv:2007.06823*, 2020.

[10] S. Li, W. Xing, R. Kirby, and S. Zhe, “Multi-fidelity bayesian optimization via deep

neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8521–8531, 2020.

[11] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[12] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*, 2003, pp. 63–71.

[13] K. E. Kelly, W. W. Xing, T. Sayahi, L. Mitchell, T. Becnel, P.-E. Gaillardon, M. Meyer, and R. T. Whitaker, "Community-based measurements reveal unseen differences during air pollution episodes," *Environmental science & technology*, vol. 55, no. 1, pp. 120–128, 2020.

[14] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[15] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[16] Z. Wang and S. Jegelka, "Max-value entropy search for efficient bayesian optimization," in *International conference on machine learning*, 2017, pp. 3627–3635.

[17] D. Huang, T. T. Allen, W. I. Notz, and R. A. Miller, "Sequential kriging optimization using multiple-fidelity evaluations," *Structural and Multidisciplinary Optimization*, vol. 32, no. 5, pp. 369–382, 2006.

[18] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv*

preprint arXiv:1402.6028, 2014.

[19] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Póczos, "Gaussian process bandit optimisation with multi-fidelity evaluations," *Advances in neural information processing systems*, vol. 29, 2016.

[20] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[21] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial intelligence and statistics*, 2016, pp. 240–248.

[22] S. Takeno, H. Fukuoka, Y. Tsukada, T. Koyama, M. Shiga, I. Takeuchi, and M. Karasuyama, "Multi-fidelity bayesian optimization with max-value entropy search and its parallelization," in *International conference on machine learning*, 2020, pp. 9334–9345.

[23] S. Li, R. Kirby, and S. Zhe, "Batch multi-fidelity bayesian optimization with deep auto-regressive networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[24] Y. Zhang, T. N. Hoang, B. K. H. Low, and M. Kankanhalli, "Information-based multi-fidelity bayesian optimization," in *NIPS workshop on bayesian optimization*, 2017.

[25] J. Song, Y. Chen, and Y. Yue, "A general framework for multi-fidelity bayesian optimization with gaussian processes," in *The 22nd international conference on artificial intelligence and statistics*, 2019, pp. 3158–3167.