An Overview and Comparison on Bayesian Optimization Techniques

Rochita Das
Texas A&M University

Ananya Roy Chowdhury
Texas A&M University

E. Christi Thompson
Texas A&M University

Abstract

Bayesian optimization is an optimization approach that utilizes Bayes Theorem to tune hyper-parameters in machine learning efficiently and effectively. In this overview, we describe how Bayesian optimization is used to find extrema of objective functions. Additionally, we compare combinations of acquisition functions with tree-structured Parzen estimators and Gaussian Process regression.

1 INTRODUCTION

Bayesian Optimization (BayesOpt) has been widely used for the hyperparameter tuning in the Machine Learning world. Bayesian optimization is a sequential design strategy for global optimization of black-box functions that does not assume any functional forms. It is usually used to optimize functions which are expensive to evaluate.

BayesOpt originated with the work of Kushner (1964), Zilinskas (1975); Mockus (1975), but received substantially more attention after that work was popularized by Jones et al. (1998). After that, innovations developed in that same literature include multi-fidelity optimization, multi-objective optimization, and a study of convergence rates.

We can start with the basic overview of hyperparameter optimization methods, which are generally of 4 types: Manual Search, Random Search, Grid Search, and Bayesian Optimization. Bayesian Optimization differs from Random Search and Grid Search as it improves the search speed by using past performances, whereas the other two methods are uniform (or independent) of past evaluations. In that sense, Bayesian Optimization is like Manual Search, where the performance of the past hyperparameter affects the future decision. But Bayesian Optimization has a vast range of application compared to Manual Search. Thus, Bayesian Optimization is a much more efficient method.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

2 BAYESIAN OPTIMIZATION OVERVIEW

The ability to optimize expensive black-box derivative-free functions makes BayesOpt extremely versatile. Recently it has become extremely popular for tuning hyperparameters in machine learning algorithms, especially deep neural networks. Over a longer period, BayesOpt has been used extensively in robotics, sensor networks, for designing engineering systems. BayesOpt has also been used to choose laboratory experiments in materials and drug design, in calibration of environmental models, particle physics and in reinforcement learning.

In this article, we first introduce the typical form that Bayesian optimization algorithms. This form involves two primary components: a method for modeling the objective function, such as Tree-structured Parzen estimator (TPE), Random Forest (RF) Regression, Gaussian Process (GP) regression; and an acquisition function for deciding where to sample, which is often Expected Improvement (EI). We describe these two components in detail in Sections 3, 4 and 5. In Section 6, we compare the three surrogate models with a toy example. Next in Section 7, we briefly discuss about the problems falling outside the strict set of assumptions for BayesOpt, which we call Exotic Bayesian optimization problems. Finally we conclude with a discussion of future research directions in Section 8.

Algorithm: Basic pseudo-code for BayesOpt

```
1. Place a Gaussian process prior on $f$.
2. Observe $f$ at $n_0$ points according to an initial
   space-filling experimental design. Set $n = n_0$.
3. **While** $n \leq N$ **do**
```

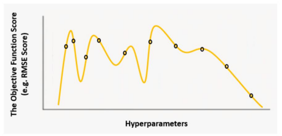
```
        (a) Update the posterior probability distribution
            on $f$ using all available data.
        (b) Let $x_n$ be a maximizer of the acquisition
            function (EI) over $x$, where the acquisition
            function is computed using the current pos-
            terior distribution.
        (c) Observe $y_n = f(x_n)$.
        (d) Increment n.
4. end while
5. Return a solution: either the point evaluated with
   the largest $f(x)$, or the point with the largest pos-
   terior mean.
```

2.1 Surrogate Model

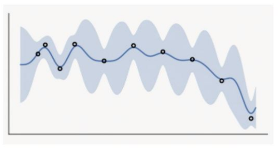
A surrogate model is the probability representation of the objective function, which maps hyperparameters to a probability of a score on the objective function. Mathematically it can be represented as,
Pr(Objective function score | Hyperparameters).

The true objective function is a fixed function. Let us assume it actually looks like Fig 1(a), but we do not know about it at the beginning of the hyperparameter tuning. If there are unlimited resources, we would compute every single point of the objective function so that we can know its actual shape. Now consider, we only have 10 samples from the true objective function, represented as black circles in Fig 1(a).

Using these 10 samples, we need to build a surrogate model to approximate the true objective function. In Fig 1(b) the surrogate model is represented as the blue line. The blue shade represents the deviation from the true objective function.



(a) Objective Function



(b) Surrogate Model

Figure 1: Figure:1

2.2 Acquisition Function

Acquisition function is a technique by which the posterior is used to select the next sample from the search space. Acquisition function is maximized to select the next choice of hyperparameter.

More formally, the objective function f will be sampled at $x_t = \text{argmax}_x u(x|D_{1:t-1})$, where u is the acquisition function and $D_{1:t-1} = \big((x_1, y_1), \cdots, (x_{t-1}, y_{t-1})\big)$ are $(t-1)$ samples drawn from f.

Now considering the previous example, we have 10 samples of the objective function and next the question could be how to decide which parameter to try as the 11^{th} sample. We need to build an acquisition function. The next hyperparameter of choice would be where the acquisition function is maximized. In Fig 2, the green shade is the acquisition function and the red straight line is where it is maximized. Therefore the corresponding hyperparameter and its objective function score, represented as a red circle, is used as the 11^{th} sample to update the surrogate model.

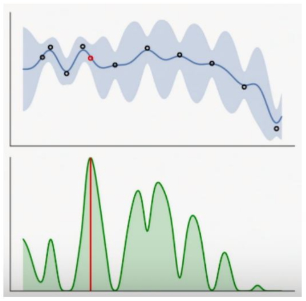


Figure 2: Maximized acquisition function

We can repeat the above steps until the maximum time or maximum iteration is reached. And then we have an accurate approximation of the true objective function and can easily find the global optima from the past evaluated samples. This completes Bayesian Optimization process.

3 TREE-STRUCTURED PARZEN ESTIMATOR (TPE)

3.1 Methodology

One choice of surrogate function is the tree-structured Parzen estimator (TPE), which utilizes Bayes rule to model the probability distribution of hyperparameters x given the score on the objective function y. Instead of constructing the surrogate model $p(y|x)$, TPE uses:

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

to model $p(x|y)$ and $p(y)$.

To begin, the algorithm is initialized by sampling the response surface through random search. Then, the observations are split into two groups based on those that performed the best (within some threshold) and the rest, where y^* is the splitting value. The chosen y^* is some quantile γ of the observed y values such that

$p(y < y^*) = \gamma$. Thus, TPE defines:

$$p(x|y) = \begin{cases} \ell(x), & \text{if } y < y^* \\ g(x), & \text{if } y \geq y^*, \end{cases}$$

where $\ell(x)$ is the density for the values of the objective function less than y^*, and $g(x)$ is the density for the values of the objective function at or greater than y^* (Bergstra et al. 2011).

Essentially, $\ell(x)$ is the density of the best observations, and $g(x)$ is the density of the poor observations, which both depend on the observed y values (Onishi et al. 2020). These densities are constructed using adaptive Parzen estimators, which are non-parametric density estimators. If the variables are categorical instead of continuous, then a categorical distribution is used to define $\ell(x)$ and $g(x)$ (Garrido-Merchán et al. 2018).

3.2 Expected Improvement(EI) with TPE

Expected Improvement (EI) is a type of acquisition function that is built on top of a surrogate model. In the case of TPE, its optimization is facilitated utilizing Bayes Rule. The formula for EI in combination with TPE is derived in Bergstra et al. (2011) as follows:

$$\begin{aligned} EI_{y^*}(x) &= \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy \\ &= \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} dy \qquad (1) \end{aligned}$$

The deconstruction of $p(y|x)$ can be rewritten in terms of the definition of $p(x|y)$ in Section 3.1. With $p(y < y^*) = \gamma$ (and $p(y \geq y^*) = 1 - \gamma$), we can write

$$\begin{aligned} p(x) &= \int_R p(x|y)p(y)dy \\ &= \gamma\ell(x) + (1-\gamma)g(x) \qquad (2) \end{aligned}$$

as a weighted mixture of the two distributions based on the observed y values. Similarly, we can define:

$$\begin{aligned} \int_{-\infty}^{y^*} (y^* - y)p(x|y)p(y) &= \ell(x) \int_{-\infty}^{y^*} (y^* - y)p(y)dy \\ &= \gamma y^* \ell(x) - \ell(x) \int_{-\infty}^{y^*} p(y)dy. \qquad (3) \end{aligned}$$

Putting together (2) and (3), we can rewrite (1) as:

$$\begin{aligned} EI_{y^*}(x) &= \frac{\gamma y^* \ell(x) - \ell(x) \int_{-\infty}^{y^*} p(y)dy}{\gamma\ell(x) + (1-\gamma)g(x)} \\ &= \left(\frac{\gamma\ell(x) + g(x)(1-\gamma)}{\gamma y^* \ell(x) - \ell(x) \int_{-\infty}^{y} {}^* p(y)dy} \right)^{-1} \\ &\propto \left(\gamma + \frac{g(x)}{\ell(x)}(1-\gamma) \right)^{-1}, \end{aligned}$$

where the last step is achieved after dividing the numerator and denominator by $\ell(x)$. For every iteration, this algorithm will return x^*, which is the value with the largest EI. Thus, maximizing the EI is equivalent to maximizing the ratio $\frac{\ell(x)}{g(x)}$ (Bergstra et al. 2013). Then the draw values of the hyper-parameters will be more likely under $\ell(x)$, the density of our best observations, than $g(x)$, the density of our poor observations.

The benefit of EI is that updating the hyper-parameters requires the model to search over the sample space while also making use those areas that perform the best.

4 RANDOM FOREST (RF) REGRESSION

4.1 Methodology

Sequential Model-based Algorithm Configuration (SMAC) is a tool for optimizing algorithm parameters. In particular, the models are random forests that are modified to yield uncertainty estimates. Due to their automated feature selection and robustness, they provide quality estimates efficiently (Feurer et al. 2014).

Random forests are collections of regression trees that have target algorithm performance values at their leaves (Hutter et al. 2010). At the top of each tree is the collection of all possible configurations. The tree then successively splits into further branches by grouping similar observations into nodes.

To begin SMAC, the algorithm is intialized at a predefined start point, which gives other algorithms that intialize with a random grid search a slight disadvantage (Ilievski et al. 2017). A user-specified number of regression trees are built on random samples of the training dataset consulting of the hyper-parameters and score pairs (θ_i, o_i) for $i = 1, ...n$. From this data, a new configuration θ is used an input to learn the model, and predictions are made on the training dataset.

Within each tree, the prediction performance is computed for the combinations of the given parameter configuration and each instance. Then, the predictions are combined with the objective function, and the means and variances across trees is calculated. The RF's predictive predictive mean and variance for the new configuration θ is denoted as μ_θ and σ_θ^2 (Hutter et al. 2010).

4.2 Expected Improvement (EI) with RF Regression

In order to quantify how desirable the input θ is, the expected improvement of the model's predictive distribution for the configuration θ is calculated over the best configuration seen so far. This best configuration is called the incumbent and denoted as θ_{inc}. A configuration with a large $EI(\theta)$ will have low predictive cost and and high predictive uncertainty to balance exploration and exploitation.

In Hutter et al (2010), the RF predictions were based on a log-transformed cost metric of θ. Thus when calculating EI, it is defined as:

$$EI(\theta) = f_{min}\phi(\nu) - e^{\frac{1}{2}\sigma_\theta^2 + mu_\theta}\phi(\nu - \sigma_\theta) \ , \qquad (4)$$

where f_{min} is the empirical mean performance of θ, ϕ is the cumulative distribution function of the standard normal density, and $\nu := \frac{ln(f_{min}) - \mu_\theta}{\sigma_\theta}$ is the log-transformed cost metric adjusted for the mean and variance of the RF. Previously, $fmin = \mu(\theta_{inc}) + \sigma(\theta_{inc})$ was used, but setting it to the empirical mean performance of θ_{inc} is proven to perform better.

To speed up computation time, Hutter et al. (2010) suggests to perform a multi-start local search to identify all configurations with locally maximal EI. Specifically, they calculate the EI for the configurations in the previous runs, pick the ten with the maximal EI, and then start a local search at them.

5 GAUSSIAN PROCESS (GP) REGRESSION

5.1 Methodology

GP regression is a Bayesian statistical approach for modeling functions. Let us first describe GP regression, by focusing on the functional value of f at a finite collection of points $x_1, \cdots, x_k \in R^d$. Then whenever the quantity $f(x_1), \cdots, f(x_k)$ is unknown in Bayesian statistics, we suppose that it was drawn randomly from some prior probability distribution. GP regression takes this prior distribution to be multivariate normal, with a particular mean vector and covariance matrix. This prior distribution is what is known as the Gaussian Process. The following definition is taken from Rasmussen and Williams (2006).

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(x)$ and the covariance function mean function $k(x, x')$ of a real process $f(x)$ as

$$\begin{aligned} m(x) &= \mathrm{E}[f(x)] \\ k(x, x') &= \mathrm{E}[(f(x) - m(x))(f(x') - m(x')] \end{aligned}$$

and will write the Gaussian process as

$$f(x) \sim GP(m(x), k(x, x'))$$

We construct the mean vector by evaluating a mean function $_0$ at each x_i and we construct the covariance matrix by evaluating a covariance function or kernel Σ_0 at each pair of points x_i, x_j . The kernel is chosen so that points x_i, x_j have a large positive correlation if they are close to each other, cementing the ideology that they should have more similar function values than points that are far apart. The kernel should also have the property that the resulting covariance matrix is positive semi-definite, regardless of the collection of points chosen.

In real life we observe, $y = f(x) + \epsilon$. Using a GP prior on f, we have,

$$f(x_{1:k}) \sim N(\mu_0(x_{1:k}), \Sigma_0(x_{1:k}, x_{1:k}))$$

where

$$\begin{aligned} f(x_{1:k}) &= [f(x_1), ..., f(x_k)], \\ \mu_0(x_{1:k}) &= [\mu_0(x1), \cdots, \mu_0(x_k)] \\ \Sigma_0(x_{1:k}, x_{1:k}) &= \begin{pmatrix} \Sigma_0(x_1, x_1) & \cdots & \Sigma_0(x_1, x_k) \\ \cdots & \cdots & \cdots \\ \Sigma_0(x_k, x_1) & \cdots & \Sigma_0(x_k, x_k) \end{pmatrix} \end{aligned}$$

Suppose we observe $f(x_{1:n})$ without noise for some n and we wish to infer the value of $f(x)$ at some new point x. Then the conditional distribution of $f(x)$ (key predictive equations for Gaussian process regression) given these observations is given by:

$$f(x)|f(x_{1:n}) \sim N(\mu_n(x), \sigma_n^2(x)), \qquad (5)$$

$$\begin{aligned} \text{where} \quad \mu_n(x) &= \Sigma_0(x, x_{1:k})\Sigma_0(x_{1:k}, x_{1:k})^{-1} \\ &\quad (f(x_{1:k}) - \mu_0(x_{1:k})) + \mu_0(x) \\ \sigma_n^2(x) &= \Sigma_0(x, x) - \Sigma_0(x, x_{1:k})\Sigma_0(x_{1:k}, x_{1:k})^{-1} \\ &\quad \Sigma_0(x_{1:k}, x) \end{aligned}$$

This conditional distribution is also called the posterior probability distribution.The posterior mean $\mu_n(x)$ is a weighted average between the prior $_0(x)$ and an estimate based on the data $f(x_{1:n})$, with a weight that depends on the kernel. The posterior variance $\sigma_n^2(x)$ is equal to the prior covariance $\Sigma_0(x, x)$ less a term that corresponds to the variance removed by observing $f(x_{1:n})$.

5.2 Expected Improvement (EI) with GP Regression

We define the expected improvement as,

$$EI_n(x) := E_n[(f(x) - f_n^*)^+]$$

Here, $E_n[\cdot] = E[\cdot|x_{1:n}, y_{1:n}]$ indicates the expectation taken under the posterior distribution given evaluations of f at $x_1, \cdots, x_n$. This posterior distribution is given by (5): $f(x)$ given $x_{1:n}, y_{1:n}$ is normally distributed with mean $\mu_n(x)$ and variance $\sigma_n^2(x)$. The expected improvement can be evaluated in closed form as:

$$EI_n(x) := [\Delta_n(x)]^+ + \sigma_n(x)\psi(\frac{\Delta_n(x)}{\sigma_n(x)}) - |\Delta_n(x)|\Phi(\frac{\Delta_n(x)}{\sigma_n(x)})$$

where $\Delta_n(x) := \mu_n(x) - f_n^*$ is the expected difference in quality between the proposed point x and the previous best.

The expected improvement algorithm then evaluates at the point with the largest expected improvement,

$$x_{n+1} = \text{argmax} \quad EI_n(x) \tag{6}$$

breaking ties arbitrarily.

Implementations use a variety of approaches for solving (6). $EI_n(x)$ is inexpensive to evaluate and allows easy evaluation of first- and second order derivatives. Implementations of the expected improvement algorithm can then use a continuous first- or second-order optimization method to solve (6). For example, one technique that has worked well for the author is to calculate first derivatives and use the quasi-Newton method L-BFGS-B (Liu and Nocedal, 1989).

6 TOY EXAMPLE FOR COMPARISON

To compare the performance of these algorithms, we use two toy examples. For this application, we found the minimizer of two different objective functions based on the three types of Bayesian Optimization methods discussed previously: TPE, RF regression, and GP regression. The first objective function is to find the minimizer of:

$$\widehat{f}_1(x) = \text{argmin} \min_{x \in [0,1]} (2x-1)^2 \sin(5\pi x + 4)^3 \tag{7}$$

The other objective function is of the form:

$$\widehat{f}_2(x) = \text{argmin} \min_{x \in [0,1]} (6x-2)^2 \sin(12x - 4) \tag{8}$$

For this, we used the `Python hyperopt` package for TPE, and we implemented RF and GP regression from scratch in R and Python respectively, though there is software available for both. We started with a random sample of 100 observations for values of x ranging from 0 to 1, and the corresponding y values were initially fit from noisy versions of the objective functions above. We ran each algorithm for 200 iterations.

Table 1: Results from three algorithms and the actual minimizer of each objective function. The column for **EX. 1** is the minimizer of (7), and the column for **EX. 2** is the minimizer of (8).

ALGORITHM	**EX. 1**	**EX. 2**
TPE	0.041	0.765
RF regression	0.046	0.748
GP regression	0.212	0.999
Actual	0.040	0.757

From the results of Table 1, we see that both TPE and RF regression provided minimizers that were approximately equivalent to the true value. This is not surprising since tree-based algorithms tend to provide more flexibility, hence providing better estimates (Ilievski et al. 2017).

7 EXOTIC BAYESIAN OPTIMIZATION

Above we described methodology for solving the standard Bayesian optimization problem. This problems assume a feasible set in which membership is easy to evaluate, such as a hyperrectangle or simplex; a lack of derivative information; and noise-free evaluations. But there could be situations where one or even more of these assumptions are broken. We call these as Exotic problems. These exotic Bayesian optimization problems include those with parallel evaluations, multi-fidelity evaluations, multiple information sources, random environmental conditions, multi-task objectives, and derivative observations. Here, for space constraint we only describe Parallel Evaluation, Multi-Fidelity and Multi-Information Source Evaluations.

7.1 Parallel Evaluation

Performing evaluations in parallel using multiple computing resources allow obtaining multiple function evaluations in the time that would ordinarily be required to obtain just one with sequential evaluations. For this reason, parallel function evaluations is a conceptually appealing way to solve optimization problems in less time. EI can be extended in a straightforward way to allow parallel function evaluations.

EI: $\mathrm{EI}_n(x^{(1:q)}) = \mathrm{E}_n\left[[\max_{i=1,\cdots,q} f(x^{(i)}) - f_n^*]^+\right]$

where $x^{(1:q)} = (x^{(1)}, \ldots, x^{(q)})$ is a collection of points at which we are proposing to evaluate.

Parallel EI then proposes to evaluate the set of points that jointly maximize this criteria. This approach can also be used asynchronously, where we hold fixed those $x^{(i)}$ currently being evaluated and we allocate our idle computational resources by optimizing over their corresponding $x^{(j)}$.

The Constant Liar approximation: Parallel EI and other parallel acquisition functions are more challenging to optimize than their original sequential versions. One innovation is the Constant Liar approximation to the parallel EI acquisition function (Ginsbourger et al., 2010), which chooses $x^{(i)}$ sequentially by assuming that $f(x^{(j)})$ for $j < i$ have been already observed, and have values equal to a constant (usually the expected value of $f(x^{(j)})$ under the posterior. This substantially speeds up computation.

Expanding on this, Wang et al. (2016) showed that infinitesimal perturbation analysis can produce random stochastic gradients that are unbiased estimates of $\nabla EI_n(x^{(1:q)})$, which can then be used in multistart stochastic gradient ascent to optimize. This method has been used to implement the parallel EI procedure for as many as $q = 128$ parallel evaluations.

7.2 Multi-Fidelity and Multi-Information Source Evaluations

In multi-fidelity optimization, rather than a single objective f, we have a collection of information sources $f(x; s)$ indexed by s. Here, s controls the fidelity, with lower s giving higher fidelity, and $f(x; 0)$ corresponding to the original objective. Increasing the fidelity (decreasing s) gives a more accurate estimate of $f(x; 0)$, but at a higher cost $c(s)$. For example, x might describe the design of an engineering system, and s the size of a mesh used in solving a partial differential equation that models the system. Or, s might describe the time horizon used in a steady-state simulation. Authors have also recently considered optimization of neural networks, where s indexes the number of iterations or amount of data used in training a machine learning algorithm.

Accuracy is modeled by supposing that $f(x; s)$ is equal to $f(x; 0)$ and is observed with noise whose variance $\lambda(s)$ increases with s, or by supposing that $f(x; s)$ provides deterministic evaluations with $f(x; s+1) - f(x; s)$ modeled by a mean 0 Gaussian process that varies with x. Both settings can be modeled via a Gaussian process on f, including both x and s as part of the modeled domain.

Now the goal is to $\max_x f(x; 0)$ by observing $f(x; s)$ at a sequence of points and fidelities $(x_n; s_n)$ with total cost $\sum_{n=1}^{N} c(s_n) \leq \mathrm{B}$ (budget).

In the more general problem of multi-information source optimization, we relax the assumption that the $f(\cdot; s)$ are ordered by s in terms of accuracy and cost. Instead, we simply have a function f taking a design input x and an information source input s, with $f(x; 0)$ being the objective, and $f(x; s)$ for $s \neq 0$ being observable with different biases relative to the objective, different amounts of noise, and different costs.

EI: EI is difficult to apply directly in these problems because evaluating $f(x; s)$ for $s \neq 0$ never provides an improvement in the best objective function value seen, $\max\{f(x_n; 0) : s_n = 0\}$. Thus, a direct translation of EI to this setting causes EI $= 0$ for $s \neq 0$, leading to measurement of only the highest fidelity. For this reason, the EI-based method uses EI to select x_n assuming that $f(x; 0)$ will be observed (even if it will not), and (even if it will not), and uses a separate procedure to select s.

8 DISCUSSION

In this paper we provided a brief overview of some of the commonly used Bayesian optimization techniques like the Tree Parzen Estimator, the Random Forest Regression and the Gaussian Process Regression techniques. We also reviewed some of the exotic Bayesian optimization techniques like Parallel Evaluations,Multi-Fidelity, and Multi-Information Source Evaluations. As seen from the small comparison we made, TPE and RF provides the most accurate estimation of the minimizers with TPE performing slightly better than the RF. In cases when Gaussianity is preserved, any of the exotic Bayesian algorithms may also be used. A vast amount of resources as available online to implement any of these algorithms with ease.

Many research directions present themselves in this exciting field. First, there is substantial room for developing a deeper theoretical understanding of Bayesian optimization. So far multi-step optimal algorithms are extremely limited. Even in the asymptotic regime, we know very little about rates of convergence of Bayesian optimization algorithms. Second, we can look for new acquisition functions that may provide substantial better result in high dimensional set up. Third, developing Bayesian optimization methods that work well in high dimensions is of great practical and theoretical interest. Directions for research include developing statistical methods that identify and leverage structure present in high-dimensional objectives arising in practice.

References

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl (2011). Algorithms for Hyper-Parameter Optimization. In J. Shawe-Taylor and R. S. Zemel and P. L. Bartlett and F. Pereira and K. Q. Weinberger (ed.), *Advances in Neural Information Processing Systems 24,* 2546–2554.

J. Bergstra, D. Yamins, and D.D. Cox (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *Proceedings of the 30th International Conference on Machine Learning,* 115–123.

M. Feurer, J.T. Springenburg, and F. Hutter (2014). Using Meta-Learning to Initialize Bayesian Optimization of Hyperparameters. *Proceedings of the 2014 International Conference on Meta-Learning and Algorithm Selection,* 3–10.

P.I. Frazier (2018). A Tutorial on Bayesian Optimization. *arXiv e-prints*, 1807.02811.

E.C. Garrido-Merchán and D. Hernández-Lobato (2018). Dealing with Categorical and Integer-values Variables in Bayesian Optimization with Gaussian Processes. *arXiv e-prints*, 1805.03463.

F. Hutter, H.H. Hoos, K. Leyton-Brown (2010). Sequential Model-Based Optimization for Algorithm Configuration *Proceedings of the 5th International Conference on Learning and Intelligent Optimization,* 507–523.

I. Ilievski, T. Akhtar, J. Feng and C.A. Shoemaker (2017). Efficient Hyperparameter Optimization of Deep Learning Algorithms Using Deterministic RBF Surrogates *arXiv e-prints*, 1607.08316v2.

D.R. Jones, M. Schonlau W.J. Welch (1998). Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 455–492

H.J. Kushner (1964). A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering, 86(1)*, 97–106.

J. Mockus (1975). On the Bayes Methods for Seeking the Extremal Point. *IFAC Proceedings Volumes*, 428–431

M. Onishi, Y. Ozaki, Y. Tanigaki, and S. Watanabe (2020). Multiobjective Tree-structured Parzen Estimator for Computationally Expensive Optimization Problems. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference,* 533–541.

C.E. Rasmussen and C.K.I. Williams (2006) Gaussian Processes for Machine Learning *MIT Press, 2006, ISBN 026218253X*

A. Zilinskas (1975). One-step Bayesian method of the search for extremum of an one-dimensional function. *Cybernetics 1*, 139–144.