PRACTICAL BAYESIAN OPTIMIZATION OF MACHINE LEARNING ALGORITHMS

Anonymous Authors[1]

Abstract

The performance of most of the machine learning algorithms are heavily dependent upon the tuning of hyperparameters, regularization terms and optimization parameters. Since, there is no straightforward way of choosing these parameters, it is often considered as a "black art". We need to develop automated procedures to perform this "black art" by keeping human intervention to minimal. In this work, we use Bayesian framework where we model machine learning algorithm's performance as a sample from Gaussian Process(GP). We show that the above mentioned framework finds minimum of multimodal functions consistently and demonstrate it's use to logistic regression to tune hyperparameters.

1. Introduction

Machine learning algorithms are rarely parameter-free. As a practitioner, it is one's responsibility to decide on these parameters. These parameters can directly affect the algorithms' performance. There is no easy or straightforward way of determining how to choose the parameters to increase the performance of the algorithms. Since these choices can significantly affect the performance, it is highly desired that we find an automated way to determine these choices.

We can consider the tuning of hyperparameters as a optimization process itself. But unlike lower level objective functions which has well defined function mathematical expression and its derivatives, we don't have such well defined function in case of hyperparameter tuning. In case of hyper parameter tuning, the objective function can be considered as accuracy(maximize the objective function in this case) or error(minimize the objective function in this case). As such, it is highly expensive in order to determine the objective function (accuracy or error). We can conclude that the process of tuning of hyperparameters is an optimization process itself but with expensive objective function calculations and not easily calculable gradients or other derivatives of the objective function. In this context, our goal is to find the points which can possibly give us best results. To solve this, Bayesian optimization provides an elegant approach and outperforms several other global optimization problems. In Bayesian optimization, the process works by assuming that the objective function is sampled from Gaussian Process(GP) and maintains a posterior distributions for this process as observations are made. Based on the information got from the previous samples, the next point of interest is chosen by optimizing acquisition functions(to be discussed in later section).

In our work, the contribution is the identification of good practices for Bayesian Optimization. We examine the importance of fully Bayesian treatment of GP kernel parameters to get better results. We also examine the choice of covariance functions of GP. We show that the developed method can be used to get set of hyperparameters for a logistic regression problem.

2. Background

2.1. Gaussian Process

To understand the Gaussian process, we need to first introduce the concept of stochastic processes. A stochastic process is defined as a sequence of random variables where the sequence can be considered as interpretation of time. Another way to look the stochastic process is that it is a distribution of functions. We define the Gaussian Process (GP) using the first definition and later use second definition wherever necessary. The Gaussian process is defined as a stochastic process in which every finite collection of random variables has a joint Gaussian (Normal) distribution. A GP is fully characterised by the mean function and covariance function.

$$f \sim GP(m, k)\ ; m(x)\ ; k(x, x') \tag{1}$$

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2.2. Posterior Gaussian Process

The Gaussian process can be viewed as a distribution of functions. Our goal is to try to find a GP which represents the target function as close as possible from given set of observations. Let f be the known function values from the observation data (D), we get the posterior Gaussian process as

$$\begin{aligned} f|D &\sim GP(m_D, k_D) \\ m_D(x) &= m(x) + \Sigma(X,x)^T \Sigma^{-1}(f-m) \\ k_D(x,x') &= k(x,x') - \Sigma(X,x)^T \Sigma^{-1} \Sigma(X,x') \end{aligned} \tag{2}$$

where $\Sigma(X,x)$ is the covariance between every observed case and x, Σ is the covariance between observed data. These are the update equations to determine the posterior Gaussian Process. One important observation here is that if we see the covariance function of the posterior, we see that we subtract an always positive term from the prior covariance. It implies that with more data, our covariance becomes low enough and GP closes to the target function.

2.3. Bayesian Optimization

Bayesian Optimization is a generic method for optimizing a black-box function. A black-box function can be considered as a function without any easy to evaluate explicit mathematical expression. The only thing we can do with black-box function is to ask it to evaluate at a certain input and it returns the function value.

The two main parts of the algorithm is Surrogate Model and Acquisition functions. In our case, Gaussian process is the surrogate model which fits the data and can be used to determine the uncertainty of the target function value $f(x)$. The Acquisition functions gets information from surrogate model about which are the areas of interest to conduct function evaluations. We require our acquisition functions to be able to provide us areas where the target function $f(x)$ can be low. The acquisition functions are in the form such that they are of high values in areas which are not quite explored and have a good probability of lower $f(x)$ values and low values in areas where we have enough information and the $f(x)$ values are not lower.

The acquisition functions carry on the following two methods. They try to exploit areas which already proved to be good solution values and also try to explore the regions with high uncertainty which are not explored enough. These acquisition function values mainly depend on the previous observations and Gaussian process hyperparameters.

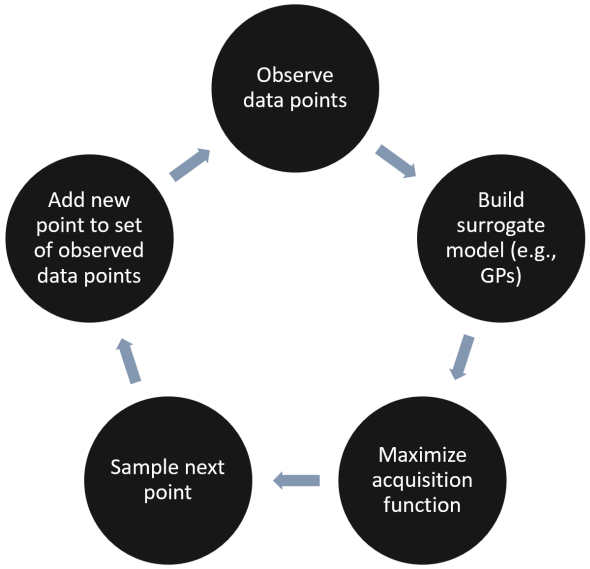

Figure 1. Bayesian Optimization

2.4. Acquisition functions

We assume that our target function $f(x)$ is drawn from a GP prior and our observations are of the form $\{x_n, y_n\}_{n=1..N}$. Usually, since the observations are not perfect and can contain amount of noise we model the noise as AWGN whose variance is ν. Hence $y_n \sim \mathcal{N}(f(x_n), \nu)$. Our acquisition functions are of form $a : X \rightarrow \mathbb{R}^+$. As we previously discussed, acquisition function value depends upon the previous observations and GP hyperparameters. Assume that θ represents the hyperparameters of the GP. We denote this dependence of acquisition functions by $a(x; \{x_n, y_n\}, \theta)$. The next point is determined from the acquisition functions by maximizing it. We use the following denotations in the proceeding: $x_{best} = argmin_{x_n} f(x_n)$, $\Phi(.)$ denotes the cumulative distribution function of the standard normal, and $\phi(.)$ denotes the standard normal probability density function. Following are some of the mostly used acquisition functions.

2.4.1. Probability of Improvement (PI)

We need values which return function values which are lower than our existing best. So, intuitive way to get those values is to find the probability of function returning a lower value than the current best. The basic idea behind PI is shown in the $Figure$ 2 for maximizing f(x).

$$\begin{aligned} a_{PI}(x; \{x_n, y_n\}, \theta) &= \Phi(\gamma(x)) \\ \gamma(x) &= \frac{f(x_{best}) - \mu(x; \{x_n, y_n\}, \theta)}{\sigma(x; \{x_n, y_n\}, \theta)} \end{aligned} \tag{3}$$

2.4.2. Expected Improvement (EI)

In PI, we get the probability to improve but we never factor in the magnitude of improvement. We can think of the improvement at each point to be a random variable. In EI, we try to find the expected improvement so that we go to the next point which gives highest expected improvement.

$$a_{EI}(x; \{x_n, y_n\}, \theta) = \sigma(x; \{x_n, y_n\}, \theta)(\gamma(x)\Phi(\gamma(x))) \tag{4}$$

2.4.3. Lower Confidence bound (LCB)

The confidence bound is a technique in which we can directly leverage between exploitation and exploration using a weighted sum of variance and mean. Since it is a weighted sum, there is an extra hyperparameter κ which comes into play in LCB. This parameter directly dictates the weightage if we want to go to the areas with already proven solutions or to go to the areas with high uncertainity which may provide better solutions.

$$\begin{aligned} a_{LCB}(x; \{x_n, y_n\}, \theta) =& \mu(x; \{x_n, y_n\}, \theta) \\ & - \kappa\sigma(x; \{x_n, y_n\}, \theta) \end{aligned} \tag{5}$$

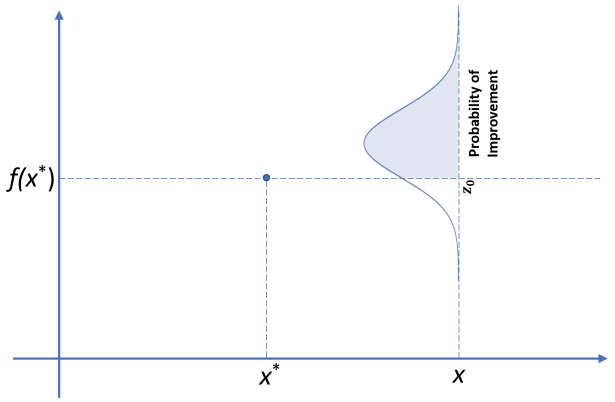

Figure 2. Probability of Improvement to maximize f(x)
x^* is the current best value; At x , we see the shaded are is the probability of improvement

In our proposed algorithm, we use expected improvement(EI) as it is shown to be much better than probability of improvement(PI) and also we don't need another tuning parameter such as in Lower confidence bound. From the experiments conducted, we see EI performs in a desired manner. In the experiment section, we show a full comparison of various acquisition function performance.

3. Method

Although Bayesian optimization is an elegant framework, many factors stopped it from becoming a widely used technique for optimizing hyperparameters. The major hindrances are the choice of covariance functions and it's hyperparameters. In the following section, we try to address this issue.

The major factor in which Gaussian Process can express a lot of functions is solely because of the covariance function. The variety of functions can be expressed due to different covariance functions. The default choice for the Gaussian Process regression is automatic relevance determination (ARD) squared exponential kernel.

$$\begin{aligned} K_{SE} &= \theta_0 exp\{\frac{-1}{2} r^2(x, x')\} \\ r(x, x') &= \sum_{d=1}^{D} (x_d - x'_d)^2 / \theta_d^2 \end{aligned} \tag{6}$$

But the ARD squared is a very smooth function and is not much applicable to the functions which we want to minimize. Hence we propose the usage of ARD Matern 5/2 kernel.

$$\begin{aligned} K_{52}(x, x') =& \theta_0(1 + \sqrt{5r^2(x, x')} + \frac{5}{3} r^2(x, x')) * \\ & exp(-\sqrt{5r^2(x, x')}) \end{aligned} \tag{7}$$

Consider, the dimension of the input vector x_i to be D. From the above, we see that $r(x, x')$ has D $\theta_{1:D}$ parameters. In addition to those D parameters, we have θ_0, observation noise ν, and constant mean m. So, in total we have $D + 3$ hyperparameters. The usual approach to determine these hyperparameters is the MAP estimate using marginal likelihood. However, we propose using a full Monte-Carlo estimate through full Bayesian treatment of hyperparameters. In other words, we assume that the hyperparameters come from a distribution. Then we try to find the *integrated acquisition function* marginalizing over the entire distribution (Hereby we refer this method as MCMC). We assume that θ consists of all the $D + 3$ hyperparameters discussed above.

$$\begin{aligned} &\hat{a}(x; \{x_n, y_n\}_{n=1}^N) = \\ &\int a(x; \{x_n, y_n\}_{n=1}^N, \theta) p(\theta | \{x_n, y_n\}_{n=1}^N) d\theta \end{aligned} \tag{8}$$

Since the complexity of getting the MAP(optimization) estimate and Monte Carlo estimate is dominated by other calculations (moreover by the function evaluation itself), we can safely use our MCMC method.

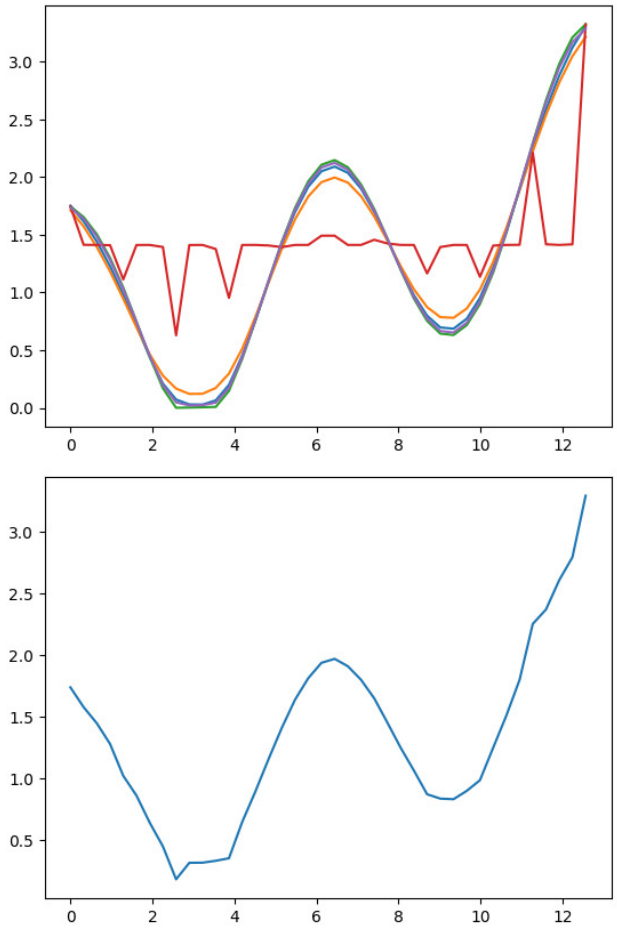

Figure 3. (Top) This fig shows expected improvement(EI) under various hyperparameters
(Bottom) This fig shows the integrated acquisition function defined in the our method

4. Experiment and observations

In this section, we analyse and compare the introduced algorithm in this paper with those of existing. We refer to our method as "MCMC EI" and refer to the method which uses LCB instead of expected improvement as "MCMC LCB". We refer to the method in which MAP of GP hyperparameters are used as "OPT EI" and "OPT LCB". We will explain the experiment in the proceeding. The hyperparameters of our Gaussian Process prior are drawn as follows for our method "MCMC EI". θ is drawn from a Gamma distribution. ν the noise component of observations is drawn from Half-cauchy distribution. We compare the different approaches we mentioned in the previous sections on a *1-D Griewank function* defined over $x \in \mathbb{R}\ where\ 0 \leq x \leq 4\pi$ and plotted the performance in the following $Figure4$. We also applied our method to find the l_2 regularization parameter for Ridge regression between $0\ and\ 2$ applied on NHANES dataset.

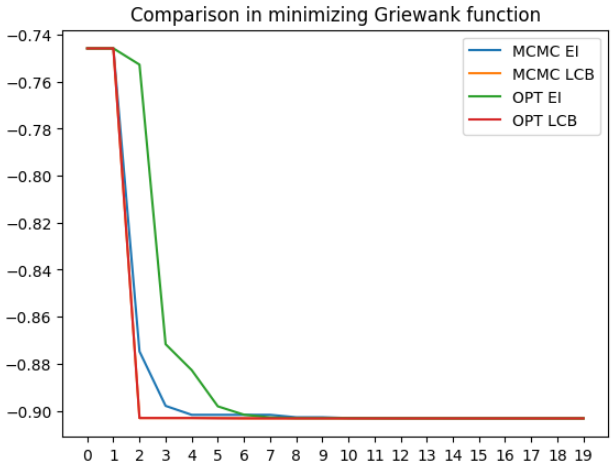

Figure 4. Minimum value vs Function evaluations

From the experiments we conducted, LCB acquisition functions performed better, but it needs its own hyperparameter κ. Our proposed method "MCMC EI" is almost on par with the best performing algorithms. Among the previously discussed acquisition functions, PI behaved very poorly and does not converge to the minimum most of the times. Important practical hurdles and observations about our proposed method are discussed in the following paragraph.

In our experiments, we see that the *integrated Expected Improvement acquisition function* is usually a multimodal(consists of many maxima) function. So, whenever we try to approach the problem of optimizing it, it heavily depends upon the initial point chosen in the optimization algorithm. When we tried to use global optimization algorithms, the runtime increased exponentially and is not a feasible option. So, we need a method to efficiently find the correct maxima of acquition function to be used as the next point. In our experiment, we tried to randomize the initial point for optimization problem so that it indeed converges to various different maximas instead of always getting stuck at one place.

5. Conclusion

In our work, we presented Bayesian methods to find the best hyperparameters of a ML model. We introduced various existing methods of using Bayesian Optimization and compared them. There is still a lot to decide on the Gaussian Process prior to be used. Also, we can try to find the maximizer of the acquisition function efficiently instead of global gradient algorithms to get the correct value everytime.

Github repository URL:

https://github.com/sandeepkumar47/ECE50024_final_project

References:

1. Gaussian Processes - https://mlg.eng.cam.ac.uk/pub/pdf/Ras04.pdf
2. Based on the code from pymc package examples.
3. https://ekamperi.github.io/machine%20learning/2021/06/11/acquisition-functions.html
4. https://ekamperi.github.io/machine%20learning/2021/05/08/bayesian-optimization.html#acquisition-function