Practical Bayesian Optimization of Machine Learning Algorithms

Madeleine Yuh [1]

1. Introduction

Machine learning requires the tuning of model hyperparameters, regularization terms, and optimization parameters tend to require expert experience or brute force search which can be time-consuming. The method proposed by Snoek et al. (2012) considers automatic tuning using the framework of Bayesian optimization. In other words, Snoek et al. (2012) models the learning algorithm's generalization performance as a sample from a Gaussian process. This allows for the effective use of information from previous experiments to make better optimal choices of parameters for the model. Additionally, the method leads to better, optimal tuning of parameters that can match that of expert-level parameter tuning performance, making the implementation of machine learning algorithms more accessible.

2. Related Work

The work done in Snoek et al. (2012) implements a new algorithm that takes into account the variable cost or duration of learning experiments and leverages multiple cores of parallel experimentation. Snoek et al. (2012) proposes to use Bayesian optimization. Bayesian optimization is used to find the minimum of the function $f(x)$ on some bounded $X \in \mathbb{R}^d$ What separates this method from other procedures is that this procedure exploits a constructed probabilistic model of $f(x)$ to make decisions on where in X to next evaluate $f(x)$. This allows for this method to use all available information from previous evaluations of $f(x)$ instead of relying on a local gradient or hessian approximations. This leads to a procedure that can be potentially used to find the minimum of non-convex functions with fewer evaluations. This paper will first lay the foundation for understanding Gaussian processes that are used to formulate a Bayesian framework for regression (Rasmussen, 2004; Rints, 2019; Salvatier et al., 2016). Then, the Bayesian optimization techniques from Snoek et al. (2012); Xia et al. (2020) will be re-implemented. Specifically, a comparison will be made between the acquisition function chosen to dictate the next inputs to evaluate, and the method of managing the hyperparameters will be explored.

[1]School of Mechanical Engineering PhD program, Purdue University, West Lafayette, IN, USA. Correspondence to: Madeleine Yuh <myuh@purdue.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

3. Method

A tutorial on how Gaussian Processes (GP) can be used for formulating Bayesian frameworks for regression is presented by Rasmussen (2004). This is re-implemented in Sections 3.1-3.3. Sections 3.4 and 3.5 re-implements the methods used by Snoek et al. (2012) for Bayesian optimization with Gaussian process priors.

3.1. Gaussian Process

In this demonstration, we consider the GP given by:

$$f \sim \mathcal{GP}(m, k), \tag{1}$$

where

$$m(x) = \sin(x), \tag{2}$$

and

$$k(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right). \tag{3}$$

To generate samples from function f at a finite number of n locations, using values of x, a vector of means and a covariance matrix can be evaluated using Equations 1, 2, and 3:

$$\mu_i = m(x_i) = \frac{1}{4}x_i^2, i = 1, ..., n \text{ and}$$
$$\Sigma_{ij} = k(x_i, x_j) = \exp\left(-\frac{1}{2}(x_i - x_j)^2\right), i, j = 1, ..., n. \tag{4}$$

From this, a random vector can be generated from the normal Gaussian distribution. As such, the vector coordinates will be the values of the function $f(x)$ for the corresponding x's in:

$$\mathrm{f} \sim \mathcal{N}(\mu, \Sigma). \tag{5}$$

This is demonstrated in Figure 1. From the figure, it can be observed that the sampled functions generated from the distribution do follow a similar trend to the mean function.

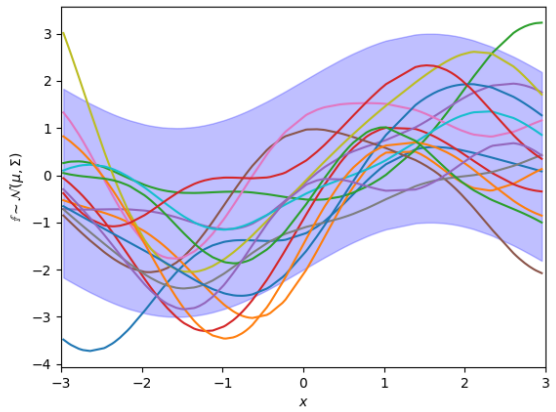

Figure 1. Fifteen functions generated from the GP in Equation 1. The generated dots are from Equation 5. The shaded area is the 95% confidence interval from 100 functions.

The confidence interval shows the sinusoidal property of the original function.

So we have defined distributions over functions using Gaussian Processes. The GP will then be utilized for Bayesian inference as a prior which does not depend on training data but rather it specifies some function properties. These function properties can include details such as how the function in Figure 1 is smooth and similar to a sinusoidal. The prior can be updated using rules derived from the training data. A flowchart diagram is provided to visually represent the methodology used to make these predictions or new samples based on the Gaussian process prior and previously observed data points using the posterior distribution.

3.2. Posterior Gaussian Process

Next, we will implement the posterior GP so that predictions can be generated from unseen cases. First, we write the joint distribution in Equation 6:

$$\begin{bmatrix} f \\ f* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} \Sigma, \sigma_* \\ \Sigma_*^T, \Sigma_{**} \end{bmatrix} \right), \tag{6}$$

where f is the set of known values of the training cases and f_* is the set of corresponding function values to the test set inputs X_*. Equation 4 is still true for the training means μ and test means μ_*. For the covariances, Σ, Σ_* and Σ_{**} are the covariance for the training set, training–test, and test set, respectively. The values of training set f are known, so the condition distribution of f_* give f can be written as what is shown in Equation 7.

$$f_*|f \sim \mathcal{N}(\mu_* + \Sigma_*^T \Sigma^{-1}(f - \mu), \Sigma_{**} - \sigma_*^T \Sigma^{-1} \Sigma_*), \tag{7}$$

represents the posterior distribution of a certain set of test cases. The corresponding posterior process is shown in Equation 8.

$$\begin{aligned} f|\mathcal{D} &\sim \mathcal{GP}(m_\mathcal{D}, k_)\mathcal{D})), \\ m_\mathcal{D} &= m(x) + \Sigma(X, x)^T \Sigma^{-1}(f - m) \\ k_\mathcal{D} &= k * (x, x') - \Sigma(X, x)^T \Sigma^{-1} \Sigma(X, x'), \end{aligned} \tag{8}$$

where $\Sigma(X, x)$ represents the vector of covariances between every training case and x. These key equations are used for the GP predictions. It can be observed that the posterior variance $k_\mathcal{D}(x, x))$ and prior variance $k(x, x)$ minus a positive term depending on the training inputs are equal. The posterior variance is smaller than the prior variance because the data provides additional information. Figure 3 depicts the distribution of posterior and prior data resulting from implementing predictions from the posterior GP. Figure 4 shows ten functions sampled from the posterior distribution in Figure 3.

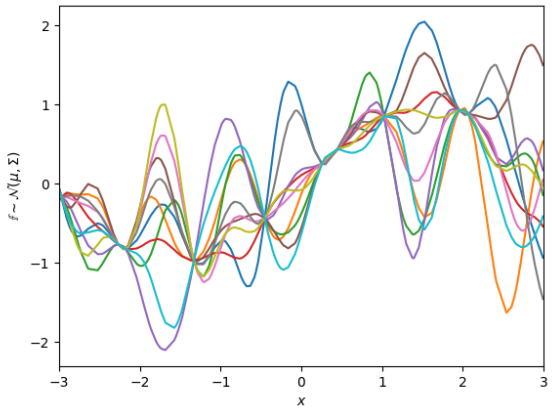

Figure 4. Ten functions generated from the posterior GP in Equation 8.

The predictions above were made while assuming that observations f of the training data are from a noiseless distribution. However, noise may exist in the training outputs as is it common for noise to exist in observations.

3.3. Noisy Observations

Suppose we assume the most common assumption of additive identically independently distributed (i.i.d.) Gaussian noise in the outputs. In that case, this can be accounted for by giving $f(x)$ extra covariances with itself with a magnitude of the noise variance, as shown in Equation 9.

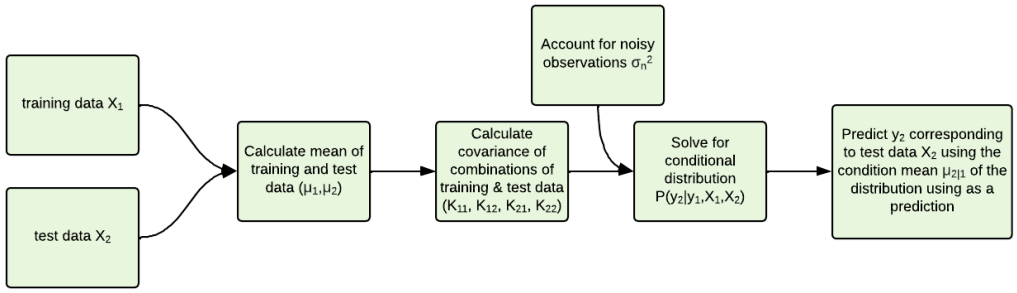

Figure 2. Flowchart showing code design process for making predictions using posterior Gaussian Process. The inputs are the training and test data X_1 and X_2, respectively. The output is the predicted output y_2 based on the posterior distribution.

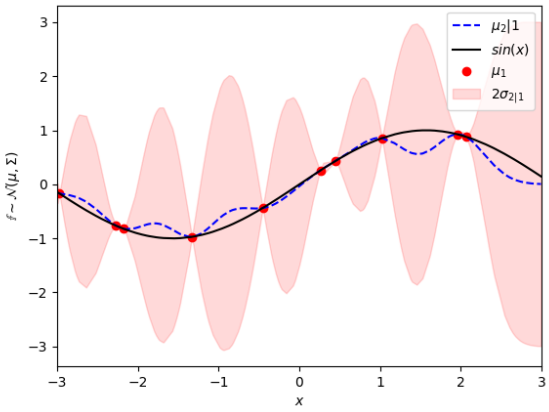

Figure 3. Distribution of posterior and prior data using Equation 8. The blue dashed line represents the true function. The black data points represent the observations from the training data. The red line is the conditional mean of the test data given the training data. The shaded area is the 95% confidence interval of predictions generated from ten test functions.

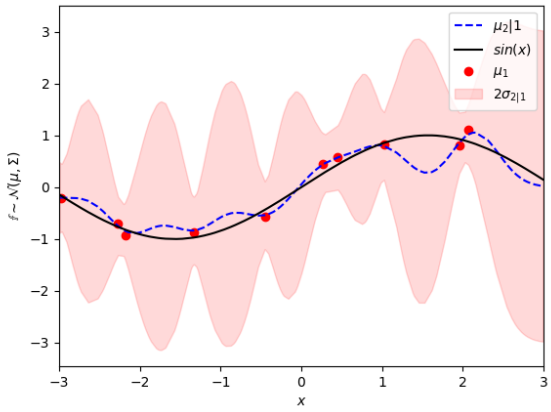

Figure 5. Distribution of posterior and prior data using Equation 9 applied to Equation 8. The blue dashed line represents the true function. The black data points represent the noisy observations from the training data. The red line is the conditional mean of the test data given the training data. The shaded area is the 95% confidence interval of predictions generated from ten test functions.

$$y(x) = f(x) + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$
$$f \sim \mathcal{GP}(m, k)\ , y \sim \mathcal{GP}(m, k + \sigma_n^2 \delta_{ii'}), \tag{9}$$

Figure 5 depicts the distribution of posterior and prior data resulting from implementing predictions from the noisy posterior GP. Figure 6 shows ten functions sampled from the noisy posterior distribution in Figure 5. The noise is generated using $\sigma_n = 1.0$.

The predictions above were made while assuming that observations f of the training data are from a noiseless distribution. However, noise may exist in the training outputs as is it common for noise to exist in observations.

3.4. Acquisition Function

So far, function $f(x)$ is drawn from a GP prior. The observations are denoted as $\{x_n, y_n\}_{n=1}^N$ where $y_n \sim \mathcal{N}(f(x_n), \nu)$, and ν denotes the noise variance of the function observations. The acquisition function, $a : \mathcal{X} \rightarrow \mathbb{R}^+$, is the posterior over functions induced by the data and prior. The acquisition function determines the next point in $\mathcal{X}$ that should be evaluated using a proxy optimization $x_{next} = \arg\max_x(x)$. Several acquisitions have been proposed, but overall, these functions depend on previous observations and the GP hyper-parameters such that $a(x;\{x_n, y_n\}, \theta)$. Using a Gaussian process prior, the acquisition function depends on the model's predictive mean function $\mu(x;\{x_n, y_n\}, \theta)$ and predictive variance function $\sigma^2(x;\{x_n, y_n\}, \theta)$. We will refer to the current best value x as $x_{best} = \arg\min_{x_n} f(x_n)$ where $\Phi(\cdot)$ represents the cumulative distribution function

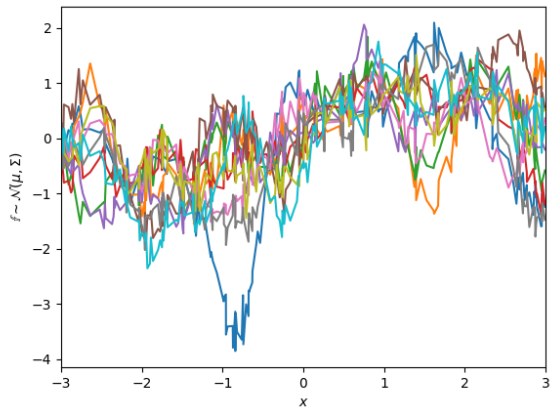

Figure 6. Ten functions generated from the posterior GP in Equation 9.

of the standard normal, and $\phi(\cdot)$ represents the standard normal density function (Snoek et al., 2012). Three acquisition functions are explored in this re-implementation: the probability of improvement (PI), expected improvement (EI), and GP lower confidence bound (LCB). The probability of improvement strategy maximizes the probability of improving over the best current value. The acquisition function can be analytically computed as:

$$a_{PI}(x;\{x_n, y_n\}, \theta) = \Phi(\gamma(x)). \tag{10}$$

If instead the expected improvement is maximized over the current best, the acquisition function then has the closed form:

$$a_{EI}(x;\{x_n, y_n\}, \theta) = \sigma(x;\{x_n, y_n\}, \theta) - (\gamma(x)\Phi(\gamma(x)) + \mathcal{N}(\gamma(x);, 0, 1)). \tag{11}$$

For both the probability improvement and expectation improvement methods, $\gamma(x)$ is denoted as:

$$\gamma(x) = \frac{f(x_{best}) - \mu(x;\{x_n, y_n\}, \theta)}{\sigma(x;\{x_n, y_n\}, \theta)} \tag{12}$$

The last method exploits the lower confidence bounds such that the acquisition function minimizes the regret over the optimization. The upper confidence bound is used when considering maximization. The acquisition function is then computed as:

$$a_{LCB}(x;\{x_n, y_n\}, \theta) = \mu(x;\{x_n, y_n\}, \theta) - \kappa\sigma(x;\{x_n, y_n\}, \theta) \tag{13}$$

where κ is a tunable variable that balances exploitation and exploration.

3.5. Practical Considerations of Bayesian Optimization of Hyperparameters

In machine learning, there are a few limitations to using Bayesian optimization to optimize hyperparameters. The first limitation is choosing an appropriate covariance function. This is due to the fact that the ability of the GP to represent a rich function distribution is dependent on the covariance function. Usually, the automatic relevance determination (ARD) squared exponential kernel, as shown in Equation 14, is used as the default for GP regression, however, the sample functions of the squared exponential covariance are unrealistically smooth for practical optimization problems (Snoek et al., 2012).

$$K_{SE}(x, x') = \theta_0 \exp\{\frac{1}{2}r^2(x, x')\} \tag{14}$$

As a result, Snoek et al. (2012) suggests using the ARD Matérn 5/2 kernel:

$$K_{M52}(x, x') = \theta_0(1 + \sqrt{(5r^2(x, x')} + \frac{5}{3}r^2(x, x')) \exp\{\sqrt{5r^2(x, x')}\}. \tag{15}$$

The ARD Matérn 5/2 kernel results in twice differentiable functions without needing the smoothness of the squared exponential (Snoek et al., 2012).

In addition to choosing an appropriate covariance function, another limitation is managing the hyperparameters and the mean function. Snoek et al. (2012) suggests using $D + 3$ GP hyperparameters: D length scales $\theta_{1:D}$, the covariance amplitude θ_0, the observation noise ν, and constant mean m. Commonly, a point estimate of the parameters is used by optimizing the marginal likelihood under the GP:

$$p(y|\{x_n\}_{n=1}^N, \theta, \nu, m) = \mathcal{N}(y|m1, \Sigma_\theta + \nu I). \tag{16}$$

In Equation 16, $y = [y_1, y_2, ..., y_n]^T$ and Σ_θ is the covariance matrix the results from N input points under hyperparameters θ. Alternatively, a fully-Bayesian treatment of hyperparameters is desirable and can be implemented. This involved computing the integrated acquisition function:

$$\hat{a}(x;\{x_n, y_n\}) = \int a(x;\{x_n, y_n\}, \theta)p(\theta|\{x_n, y_n\}_{n=1}^N)d\theta. \tag{17}$$

In Equation 17, $a(x)$ depends on θ and all observations. For both the PI and EI acquisition functions, this expectation is able to account for the uncertainty of the parameters,

therefore, the acquisition functions arising from the samples from the posterior over GP hyperparameters can be blended together. We then have a Monte Carlo estimate of the integrated expected improvement (Snoek et al., 2012). According to Snoek et al. (2012), the optimization and Markov chain Monte Carlo (MCMC) estimation are both computationally affected mainly by the cubic cost of solving an N-dimensional linear system, so the fully-Bayesian treatment is sensible.

Next, the Bayesian optimization is implemented on a set of low dimensional data in Section 4.

4. Experiment

A comparison is made between the PI, EI, and LCB acquisition functions using a MAP estimator in Equation 16. Additionally, the comparison also includes MCMC method implemented with the EI acquisition function. In this paper, the chosen unknown objective function is:

$$y = x^2 + \sin(x) \tag{18}$$

Furthermore, the ARD Matérn kernel is used for the covariance function. The main goal is to find the global minimum associated with x_{best}. Therefore, the solution to the objective function is approximated $x_{best} \approx -0.45018$. In all implementations of Bayesian optimization, a set of 15 initial points is randomly chosen to represent observations, as seen in Figure 9

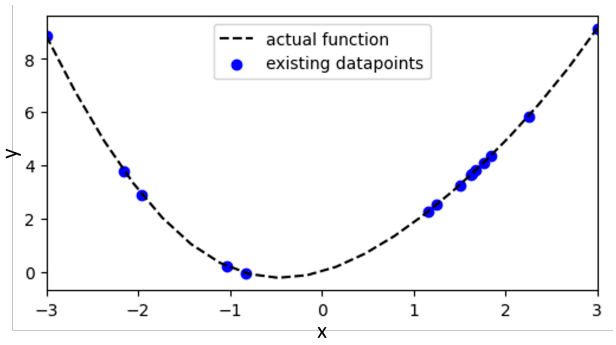

Figure 7. 15 initialized data points representing existing observations. The actual function is the unknown objective function

The next x is chosen based on the acquisition function used, as seen in Figure

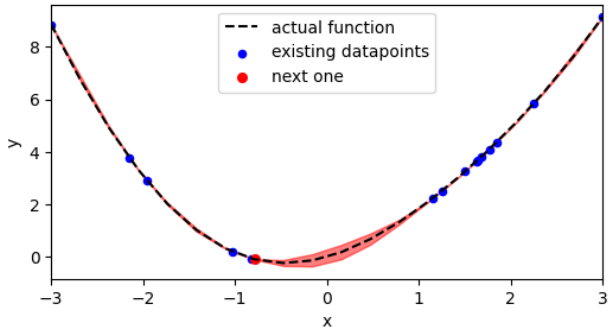

Figure 8. The next x in the first iteration represented by the red dot, is chosen using the PI acquisition function and MAP estimator. The red highlights represent the 95% confidence interval of the model's predictive mean function

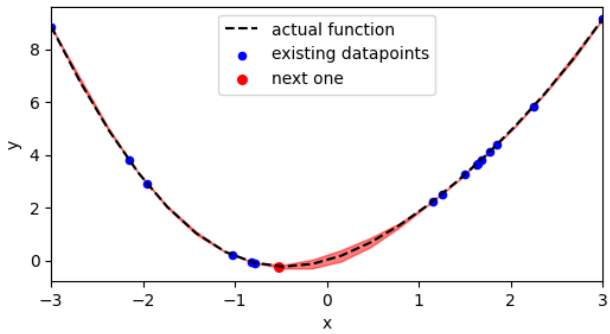

Figure 9. The next x in the second iteration, represented by the red dot, is chosen using the PI acquisition function and MAP estimator. The red highlights represent the 95% confidence interval of the model's predictive mean function

This process repeats until the optimization converges or reaches the maximum iterations, which in this case is 15 iterations.

4.1. MAP estimator

Overall, the PI, EI, and LCB acquisition functions converged to similar x_{best} values, however, the EI acquisition function was able to converge closest to the actual global minimum. The resulting estimated x_{best} is shown in Table 1.

Table 1. Comparison of acquisition functions using MAP

$a(x)$	x_{best}	Iterations	Computation Time
PI	-0.49958	5	5m 5.7s
EI	-0.45104	14	40m 49.5s
LCB	-0.44991	15	117m 35.0s

Out of the three methods, the PI acquisition function was

the fastest, however, the value it converged to was the furthest from the actual x_{best}. The implementation with EI acquisition converged the closest to the actual x_{best} value and took less time than the implementation with LCB acquisition function. Lastly, the LCB acquisition function lead to the optimization needing the maximum number of iterations, so it may have converged close to the actual x_{best}, however, the 15 iterations took over double the amount of time the EI acquisition function required. Therefore, the EI acquisition function leads to the best result while being less computationally expensive.

4.2. Markov Chain Monte Carlo Implementation

Compared to using a MAP estimator optimizing the marginal likelihood under the GP, the MCMC estimator took much longer to complete one iteration of finding the next x to evaluate. This makes sense because of the multiple evaluations and blending of acquisition functions used in this method. For that reason, the maximum number of iterations was set to 5. The results are shown in Table 2.

Table 2. Comparison of acquisition functions using MCMC

$a(x)$	x_{best}	Iterations	Computation Time
PI	-0.22073	3	5m 50.5s
EI	-0.44411	5	31m 4.5s
LCB	-0.45002	5	53m 23.9s

The results show that when using the MCMC estimator, PI acquisition function leads to faster convergence, however the estimated x_{best} is not close to the actual x_{best} compared to the other two functions. Both the EI and LCB acquisition functions reached the maximum number of iterations, and much like that of the MAP estimator results, the optimization with an EI acquisition function was faster. The resulting x_{best} values when using the EI and LCB acquisition functions are within 1.3% and 0.04% of the true x_{best} value, respectively. Although the LCB acquisition function led to a closer x_{best} value, the EI acquisition function would be the better method to use as it is able to converge close to the true x_{best} value and takes less time.

It should be noted that although the MCMC estimation method with PI and LCB acquisition functions takes longer to complete compared to that of the MAP estimator, the MCMC estimator is able to estimate an x_{best} closer to the true value within five iterations. This is likely because of the MCMC estimator's capability of blending acquisition functions.

The code used can be accessed at bit.ly/3Lv5tLS.

5. Conclusion

There are a few limitations to this work. Firstly, depending on the dimension of the data and context of optimization, these methods may be very computationally expensive. Snoek et al. (2012) suggests using Monte Carlo acquisition for parallelizing Bayesian optimization and demonstrated its use empirically. Due to equipment limitations, this was not implemented, however, if computationally available, parallelizing the optimization would likely allow for the optimization to converge to a solution earlier.

Another limitation is that while the MCMC method did work for a convex objective function, the code created did not work all the time for non-convex equations. On the other hand, the MAP estimator which was used for optimizing the marginal likelihood under the GP was able to converge to the x_{best} for a non-convex function such as $y = \sin(x)$. This may have to do with balancing exploration and exploitation to avoid getting stuck in local solutions. One of the advantages of using the method proposed by Snoek et al. (2012) is that it should be used to find the solutions for non-convex functions with fewer evaluations, and so this is more of a limitation with the code created for the re-implementation.

Overall, this paper re-implements the practical Bayesian optimization of machine learning algorithms from Snoek et al. (2012). First, we went through posterior Gaussian processes so that predictions could be generated from unseen cases. In addition, the effect of noisy observations on the distribution of posterior and prior data was shown. Once a foundation on the posterior GP was provided, we moved on to discussing different acquisition functions used and the difference between the PI, EI, and LCB approaches when optimizing the estimated x_{best}. Practical considerations of the Bayesian optimization of hyperparameters were briefly discussed, providing reasoning for using ARD Matérn kernel covariance function and the difference between optimizing the marginal likelihood under the GP and the MCMC estimate. It was shown that the EI acquisition function performed best in terms of accuracy and computational expense. Furthermore, while the MAP estimator is sufficient for Bayesian optimization, the MCMC estimator opens more opportunities for better computational power usage through parallelization, which would be beneficial when optimizing hyperparameters of higher dimensional contexts.

Acknowledgements

Thank you to Dr. Qi Guo, Wei Chen, and Ruqi Bai for their help throughout the semester. I would also like to acknowledge the other students in ECE500 that chose the same project as me because discussing the paper with them during and outside of office hours and class hours really helped with my understanding.

References

Rasmussen, C. E. Gaussian Processes in Machine Learning. In Bousquet, O., von Luxburg, U., and Rätsch, G. (eds.), *Advanced Lectures on Machine Learning*, volume 3176, pp. 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-23122-6 978-3-540-28650-9. doi: 10.1007/978-3-540-28650-9_4. URL http://link.springer.com/10.1007/978-3-540-28650-9_4. Series Title: Lecture Notes in Computer Science.

Rints, P. Gaussian processes (1/3) - From scratch, January 2019. URL https://peterroelants.github.io/posts/gaussian-process-tutorial/.

Salvatier, J., Fonnesbeck, C., and Wiecki, T. Gaussian Process Regression — PyMC3 3.1rc3 documentation, 2016. URL https://pymc3-testing.readthedocs.io/en/rtd-docs/notebooks/GP-introduction.html.

Snoek, J., Larochelle, H., and Adams, R. P. Practical Bayesian Optimization of Machine Learning Algorithms, August 2012. URL http://arxiv.org/abs/1206.2944. arXiv:1206.2944 [cs, stat].

Xia, Y., Qian, L., and Gu, J. Bayesian Optimization with pymc3, July 2020. URL https://github.com/AM207-Study-Group/Bayesian-Optimization-with-pymc3/blob/145fcc6604876f571747985501d3338e4c0f458a/AM207_Final%20Project_%20Bayesian%20Optimization%20of%20ML%20Algorithms.ipynb. original-date: 2018-12-12T03:25:59Z.