Practical bayesian optimization of machine learning algorithms

Wenlong Chen, Tudor Paraschivescu, Can Xu
Department of Engineering, Cambridge University

UNIVERSITY OF CAMBRIDGE

Introduction

The performance of machine learning algorithms highly depends on the tuning of hyperparameters. Unfortunately, hyperparameter tuning is a complicated process that involves expert knowledge, rules of thumb and even brute force search. We introduce an automatic approach to hyperparameter tuning through the framework of Bayesian optimization [4], in which we use Gaussian Processes(GP) to model the function described by the hyperparameters and thus find its minima.

Bayesian Optimisation with GP Priors

Integrated Expected Improvement: Suppose we have N observations $\{\mathbf{x}_n, y_n\}_{n=1}^N$, where $y_n \sim \mathcal{N}(f(\mathbf{x}_n), \nu)$ and ν is the variance of noise, and assume the function $f(x)$ is drawn from a Gaussian Process prior, then many popular acquisition functions $a(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)$, determining the next point we should evaluate($\mathbf{x}_{next} = \text{argmax}_{\mathbf{x}}\, a(\mathbf{x})$), solely depend on the posterior mean function $\mu(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)$ and posterior variance function $\sigma^2(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)$. In this work, the acquisition function we focus on is Expected Improvement(EI). With GP prior, it has the form:

$$a_{EI}(x; \{\mathbf{x}_n, y_n\}, \theta) = \sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}); 0, 1)), \quad (1)$$

where $\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{best}) - \mu(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}{\sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}$, $\mathbf{x}_{best} = \text{argmin}_{\mathbf{x}_n} f(\mathbf{x}_n)$ and Φ is the cumulative distribution function of standard Gaussian distribution. Instead of using a point estimate of the hyperparameters of GP by maximizing the marginal likelihood given the observations, we consider a fully Bayesian treatment of hyperparameters by marginalizing over hyperparameters and compute the integrated acquisition function:

$$\hat{a}(\mathbf{x}; \{\mathbf{x}_n, y_n\}) = E_{P(\theta|\{\mathbf{x}_n, y_n\})}[a(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)] \approx \frac{1}{M}\sum_{m=1}^{M} a(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta_m) \quad (2)$$

where θ_m's are samples drawn from $P(\theta|\{\mathbf{x}_n, y_n\})$ by a slice sampler[3].

Expected Improvement per Second: The duration of evaluating $f(x)$ in different regions of parameter space tends to vary significantly and in practice we are interested in finding optima of the function as quickly as possible in terms of wallclock time. To this end, we model $ln(c(\mathbf{x}))$ with another GP alongside $f(\mathbf{x})$, where $c(\mathbf{x})$ is the duration function, and we assume they are independent. Then we can easily compute the predicted expected inverse duration and use it to compute the expected improvement per second as a function of x. This new criterion prefers to acquire points that are not only likely to be good but that are also likely to be evaluated quickly.

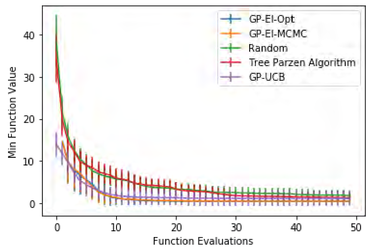
Parallelized Bayesian Optimization: In order to take advantage of multi-core computing, we want to be able to choose the next point to evaluate even with some pending evaluations. The goal can be achieved by using Monte-Carlo method to estimate the expected acquisiton function over all possible results from pending function evaluations. The next point to evaluate can be chosen according to this estimate of the expected acquistion function. Suppose N evaluations have completed, yielding data $\{\mathbf{x}_n, y_n\}_{n=1}^N$, and J evaluations are pending at locations $\{\mathbf{x}_j\}_{j=1}^J$, then the estimate of the expected acquistion is:

$$\begin{aligned}\hat{a}(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta, \{\mathbf{x}_j\}) &= E_{P(\{y_j\}|\{\mathbf{x}_j\}, \{\mathbf{x}_n, y_n\}, \theta)}[a(\mathbf{x}; \{\mathbf{x}_n, y_n\} \cup \{\mathbf{x}_j, y_j\}, \theta)] \\ &\approx \frac{1}{M}\sum_{m=1}^{M} a(\mathbf{x}; \{\mathbf{x}_n, y_n\} \cup \{\mathbf{x}_j, y_j^{(m)}\}, \theta)\end{aligned} \quad (3)$$

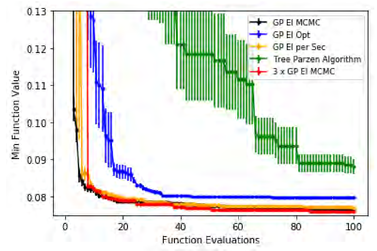
where $\{y_j^m\}$ are drawn from $P(\{y_j\}|\{\mathbf{x}_j\}, \{\mathbf{x}_n, y_n\}, \theta)$, which is a J-dimensional Gaussian distribution. Again, the integrated acquisition function can be obtained by averaging over expected acquisition functions with θ drawn from $P(\theta|\{\mathbf{x}_n, y_n\})$.

Empirical Results

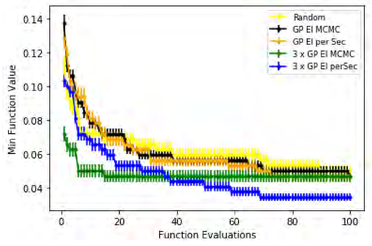
We refer to the method of using optimized point estimate of hyperparameters as "GP EI Opt", integrated expected improvement as "GP EI MCMC", EI per second as "GP EI perSec",and J times parallelized GP EI MCMC as "J x GP EI MCMC". We also compare these methods with Random Grid Search(Random) and Tree Parzen Algorithm(TPA)[1].
In Figure 1 we recreate the experimental results from [4] by comparing various strategies of optimization over the same grid for multiple models. In Figure 1d we fix the acquisition function to be integrated expected improvement and compare various GP covariance functions.



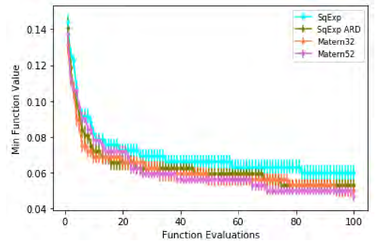
(a) Brainin-Hoo function.



(b) Logistic regression.



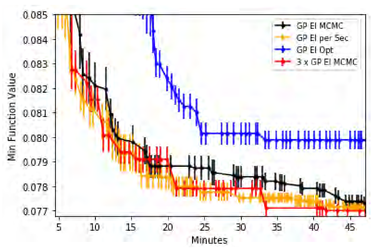
(c) Latent structured SVM.



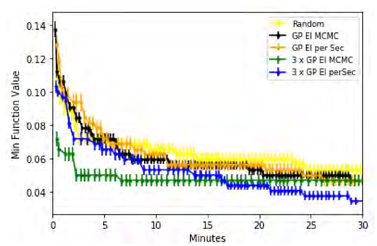
(d) Latent structured SVM for different GP kernels.

Figure 1: Minimum function value vs Function evaluation number for multiple models.

The results from [4] have been recreated for the Brainin-Hoo and logistic regression models. For latent structured SVM, the parallelized strategies achieve best performance. The other acquisition functions just return slightly better results than random grid search, but this is most likely because we're using a different implementation of the model and a different dataset from the original paper. In all cases, Bayesian Optimization outperforms Random Grid Search and TPA.



(a) Logistic regression.



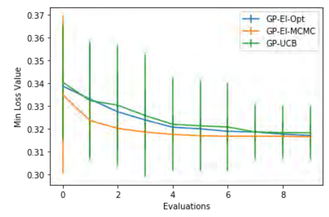
(b) Latent structured SVM.

Figure 2: Minimum function value vs wallclock time for multiple models.

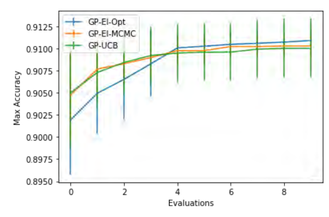
Figure 2 compares the models using wallclock time. We see similar results to [4]; although EI per second is less efficient in function evaluation it outperforms standard EI when considering wallclock time.

Extensions

One of our extensions is testing various hyperparameters optimization methods on a simple CNN model. The CNN model was built to perform classification of hand written digits in MNIST data set. The model contains a convolutional layer with a 3x3 kernel, and a dense layer of dimension 10 with softmax activation function. The hyperparameter optimization was run for both loss value(cross-entropy) and validation set accuracy. The integrated expected improvement acquisition achieved best results for loss value while using expected improvement acquisition with optimized point estimate of GP hyperparameters achieved best result for validation set accuracy.



(a) Loss of CNN.



(b) Accuracy of CNN.

Figure 3: Minimum loss value and maximum accuracy vs Function evaluation number for a CNN model.

Possible Future Work: Another interesting problem is to investigate the performance on hyperparameter tuning of Bayesian optimisation with other acquisition function such as Predictive Entropy Searh(PES) proposed in [2].

Conclusion

We explored various methods for doing Bayesian optimization for hyperparameter selection of machine learning algorithms spanning various areas of the field. Bayesian Optimization proved to be more efficient than standard methods such as Random Grid Search or the Tree Parsen Algorithm both in terms of number of function evaluations and wallclock time. We also explored multiple acquisition functions and GP kernels and found that their choice highly impacts the performance of the search.

References

[1] James S Bergstra et al. "Algorithms for hyperparameter optimization". In: *In Advances in Neural Information Processing Systems 25*. 2011.

[2] Jose Miguel Hernandez-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. "Predictive Entropy Search for Efficient Global Optimization of Black-box Functions". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*. 2014, pp. 918–926.

[3] Iain Murray and Ryan Prescott Adams. "Slice sampling covariance hyperparameters of latent Gaussian models". In: *In Advances in Neural Information Processing Systems 24*. 2010, pp. 1723–1731.

[4] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems*. 2012, pp. 2951–2959.