Efficient Hyper-parameter Optimization for NLP Applications

Lidan Wang[1], **Minwei Feng**[1], **Bowen Zhou**[1], **Bing Xiang**[1], **Sridhar Mahadevan**[2,1]
[1]IBM Watson, T. J. Watson Research Center, NY, USA
[2]College of Information and Computer Sciences, University of Massachusetts Amherst, MA, USA
{wangli,mfeng,zhou,bingxia}@us.ibm.com
mahadeva@cs.umass.edu

Abstract

Hyper-parameter optimization is an important problem in natural language processing (NLP) and machine learning. Recently, a group of studies has focused on using sequential Bayesian Optimization to solve this problem, which aims to reduce the number of iterations and trials required during the optimization process. In this paper, we explore this problem from a different angle, and propose a multi-stage hyper-parameter optimization that breaks the problem into multiple stages with increasingly amounts of data. Early stage provides fast estimates of good candidates which are used to initialize later stages for better performance and speed. We demonstrate the utility of this new algorithm by evaluating its speed and accuracy against state-of-the-art Bayesian Optimization algorithms on classification and prediction tasks.

1 Introduction

Hyper-parameter optimization has been receiving an increasingly amount of attention in the NLP and machine learning communities (Thornton et al., 2013; Komer et al., 2014; Bergstra et al., 2011; Bardenet et al., 2013; Zheng et al., 2013). The performance of learning algorithms depend on the correct instantiations of their hyper-parameters, ranging from algorithms such as logistic regression and support vector machines, to more complex model families such as boosted regression trees and neural networks. While hyper-parameter settings often make the difference between mediocre and state-of-the-art performance (Hutter et al., 2014), it is typically very time-consuming to find an optimal setting due to the complexity of model classes, *and* the amount of training data available for tuning. The issue is particularly important in large-scale problems where the size of the data can be so large that even a quadratic running time is prohibitively large.

Recently several sequential Bayesian Optimization methods have been proposed for hyper-parameter search (Snoek et al., 2012; Eggensperger et al., 2015; Brochu et al., 2010; Hutter et al., 2011; Eggensperger et al., 2014). The common theme is to perform a set of iterative hyper-parameter optimizations, where in each round, these methods fit a hyper-parameter response surface using a probabilistic regression function such as Gaussian Process (Snoek et al., 2012) or tree-based models (Hutter et al., 2011), where the response surface maps each hyper-parameter setting to an approximated accuracy. The learned regression model is then used as a cheap surrogate of the response surface to quickly explore the search space and identify promising hyper-parameter candidates to evaluate next in order to enhance validation accuracy.

While these methods have enjoyed great success compared to conventional random search (Bergstra et al., 2012; Bengio et al., 2013) and grid search algorithms by significantly reducing the number of iterations and trials required during the process, the focus and starting point of these work have largely been on dealing with many dimensions of hyper-parameters, *rather than* scaling to large amount of data as typical in many NLP tasks, where the efficiency bottleneck stems from the size of the training data in addition to hyper-parameter dimensions. For example, as dataset size grows, even simple models (with few hyper-parameters) such as logistic regression can require more training time per iteration in these algorithms, leading to increased overall time complexity.

In this work, we introduce a multi-stage Bayesian Optimization framework for efficient hyper-parameter optimization, and empirically study the impact of the multi-stage algorithm on hyper-parameter tuning. Unlike the previous approaches, the multi-stage approach considers hyper-parameter optimization in successive stages with increasingly amounts of training data. The first stage uses a small subset of training data, applies sequential optimization to quickly identify an initial set of promising hyper-parameter settings, and these promising candidates are then used to initialize Bayesian Optimization on later stages with full training dataset to enable the expensive stages operate with better prior knowledge and converge to optimal solution faster.

The key intuition behind the proposed approach is that both dataset size and search space of hyper-parameter can be large, and applying the Bayesian Optimization algorithm on the data can be both expensive and unnecessary, since many evaluated candidates may not even be within range of best final settings. We note our approach is orthogonal and complementary to parallel Bayesian Optimization (Snoek et al., 2012) and multi-task learning (Yogatama et al., 2014; Swersky et al., 2012), because the improved efficiency per iteration, as achieved by our algorithm, is a basic building block of the other algorithms, thus can directly help the efficiency of multiple parallel runs (Snoek et al., 2012), as well as runs across different datasets (Yogatama et al., 2014; Swersky et al., 2012).

2 Methodology

The new multi-stage Bayesian Optimization is a generalization of the standard Bayesian Optimization for hyper-parameter learning (Snoek et al., 2012; Feurer et al., 2015). It is designed to scale standard Bayesian Optimization to large amounts of training data. Before delving into the details, we first describe hyper-parameter optimization and give a quick overview on the standard Bayesian Optimization solution for it.

2.1 Hyper-parameter Optimization

Let $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_m\}$ denote the hyper-parameters of a machine learning algorithm, and let $\{\Lambda_1, \ldots, \Lambda_m\}$ denote their respective domains. When trained with $\boldsymbol{\lambda}$ on training data T_{train}, the validation accuracy on T_{valid} is denoted as $L(\boldsymbol{\lambda}, \mathrm{T}_{train}, \mathrm{T}_{valid})$. The goal of hyper-parameter optimization is to find a hyper-parameter setting $\boldsymbol{\lambda}^*$ such that the validation accuracy L is maximized. Current state-of-the-art methods have focused on using model-based Bayesian Optimization (Snoek et al., 2012; Hutter et al., 2011) to solve this problem due to its ability to identify good solutions within a small number of iterations as compared to conventional methods such as grid search.

2.2 Bayesian Optimization for Hyper-parameter Learning

Model-based Bayesian Optimization (Brochu et al., 2010) starts with an initial set of hyper-parameter settings $\boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_n$, where each setting denotes a set of assignments to all hyper-parameters. These initial settings are then evaluated on the validation data and their accuracies are recorded. The algorithm then proceeds in rounds to iteratively fit a probabilistic regression model V to the recorded accuracies. A new hyper-parameter configuration is then suggested by the regression model V with the help of acquisition function (Brochu et al., 2010). Then the accuracy of the new setting is evaluated on validation data, which leads to the next iteration. A common acquisition function is the expected improvement, EI (Brochu et al., 2010), over best validation accuracy seen so far L^*:

$$a(\boldsymbol{\lambda}, V) = \int_{-\infty}^{\infty} max(L - L^*, 0) p_V(L|\boldsymbol{\lambda}) d_L$$

where $p_V(L|\boldsymbol{\lambda})$ denotes the probability of accuracy L given configuration $\boldsymbol{\lambda}$, which is encoded by the probabilistic regression model V. The acquisition function is used to identify the next candidate (the one with the highest expected improvement over current best L^*). More details of acquisition functions can be found in (Brochu et al., 2010).

The most common probabilistic regression model V is the Gaussian Process prior (Snoek et al., 2012), which is a convenient and powerful prior distribution on functions. For the purpose of our experiments, we also use Gaussian Process prior as the regression model. However, we would like to note the fact that the proposed multi-stage Bayesian Optimization is agnostic of the regression model used, and can easily handle other instantiations of the regression model.

Algorithm 1: Multi-stage Bayesian Optimization for Hyper-parameter Tuning

```
Input: Loss function $L$, number of stages $S$,
        iterations per stage
        $\boldsymbol{Y} = \langle Y_1, \ldots, Y_S \rangle$, training data per
        stage $T_{train} = \langle T^1_{train}, \ldots, T^S_{train} \rangle$,
        validation data $T_{valid}$, initialization
        $\boldsymbol{\lambda}_{1:k}$
Output: hyper-parameter $\boldsymbol{\lambda}^*$
for stage s=1 to $S$ do
    for i=1 to $k$ do
        $L_i$ = Evaluate L($\boldsymbol{\lambda}_i, T^s_{train}, T_{valid}$)
    end
    for j=k+1 to $Y_s$ do
        $V$: regression model on $\langle \boldsymbol{\lambda}_i, L_i \rangle_{i=1}^{j-1}$
        $\boldsymbol{\lambda}_j = \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} a(\boldsymbol{\lambda}, V)$
        $L_j$ = Evaluate L($\boldsymbol{\lambda}_j, T^s_{train}, T_{valid}$)
    end
    reset $\boldsymbol{\lambda}_{1:k}$=best $k$ configs $\in \langle \boldsymbol{\lambda}_1, \ldots \boldsymbol{\lambda}_{Y_s} \rangle$
    based on validation accuracy $\boldsymbol{L}$
end
return $\boldsymbol{\lambda}^* = \arg\max_{\boldsymbol{\lambda}_j \in \{\boldsymbol{\lambda}^{Y_1}, \ldots, \boldsymbol{\lambda}^{Y_S}\}} L_j$
```

2.3 Multi-stage Bayesian Optimization for Hyper-parameter Tuning

The multi-stage algorithm as shown in Algorithm 1 is an extension of the standard Bayesian Optimization (Section 2.2) to enable speed on large-scale datasets. It proceeds in multiple stages of Bayesian Optimization with increasingly amounts of training data $|T^1_{train}| \leq \ldots, \leq |T^S_{train}|$. During each stage s, the k best configurations (based on validation accuracy) passed from the previous stage[1] are first evaluated on the current stage's training data T^s_{train}, and then the standard Bayesian Optimization algorithm are initialized with these k settings and applied for $Y_s - k$ iterations on T^s_{train} (discounting the k evaluations done earlier in the stage), where Y_s is the total number of iterations for stage s. Then the top k configurations based on validation accuracy are used to initialize the next stage's run.

We note after the initial stage, rather than only considering candidates passed from the previous stage, the algorithm expands from these points on larger data. Continued exploration using larger data allows the algorithm to eliminate any potential sensitivity the hyper-parameters may have with respect to dataset size. After running all S stages the algorithm terminates, and outputs the configuration with the highest validation accuracy from all hyper-parameters explored by all stages (including the initialization points explored by the first stage).

	Hyper-parameters
SVM	bias, cost parameter, and regularization parameter
Boosted regression trees	feature sampling rate, data sampling rate, learning rate, # trees, # leaves, and minimum # instance per leaf

Table 1: Hyper-parameters used in SVM and boosted regression trees.

This multi-stage algorithm subsumes the standard Bayesian optimization algorithm as a special case when the total number of stages $S = 1$. In our case, for datasets used at stages $1, \ldots, S-1$, we use random sampling of full training data to get subsets of data required at these initial stages, while stage S has full data. For the number of top configurations k used to initialize each following stage, we know the larger k is, the better results in the next stage since Bayesian Optimization relies on good initial knowledge to fit good regression models (Feurer et al., 2015). However, larger k value also leads to high computation cost at the next stage, since these initial settings will have to be evaluated first. In practice, the number of stages S and the value of k depend on the quantity of the data and the quality of stage-wise model. In our experiments, we empirically choose their values to be $S = 2$ and $k = 3$ which result in a good balance between accuracy and speed on the given datasets.

3 Experiment

We empirically evaluate the algorithm on two tasks: classification and question answering. For classification we use the Yelp dataset (Yelp, 2014) which is a customer review dataset. Each review contains a star/rating (1-5) for a business, and the task is to predict the rating based on the textual information in the review. The training data contains half-million feature vectors, and unique unigrams are used as features (after standard stop-word removal and stemming (Manning et al., 2008)). For

[1] A special case is the initial stage. We adopt the convention that a Sobol sequence is used to initialize the first stage (Snoek et al., 2012). The value k for the first stage is the number of points in the Sobol sequence.

question answering (QA), the task is to identify correct answers for a given question. We use a commercial QA dataset containing about 3800 unique training questions and a total of 900, 000 feature vectors. Each feature vector corresponds to an answer candidate for a given question, the vector consists of a binary label (1=correct, 0=incorrect) and values from standard unigram/bigram, syntactic, and linguistic features used in typical QA applications (Voorhees et al., 2011). Both QA and Yelp datasets contain independent training, validation, and test data, from which the machine learning models are built, accuracies are evaluated, and test results are reported, respectively.

We evaluate our multi-stage method against two methods: 1) state-of-the-art Bayesian Optimization for hyper-parameter learning (Snoek et al., 2012)), and 2) the same Bayesian Optimization but only applied on a small subset of data for speed. For experiments, we consider learning hyper-parameters for two machine learning algorithms: SVM implementation for classification (Fan et al., 2008) and boosted regression trees for question answering (Ganjisaffar et al., 2011) as shown in Table 1.

3.1 Accuracy vs time

Figures 1 and 2 compare the test accuracy of our proposed multi-stage Bayesian optimization as a function of tuning time for QA and Yelp, respectively. The state-of-the-art Bayesian optimization (Snoek et al., 2012) is applied on full training data, and the fast variant of Bayesian Optimization is applied with 30% of training data (randomly sampled from full dataset). The top-1 and classification accuracies on test data are reported on the y-axis for QA and Yelp, respectively, and the tuning time is reported on the x-axis. For fairness of comparison, the multi-stage method uses the same 30% training data at the initial stage, and full training data at the subsequent stage.

From these figures, while in general both of the comparison methods produce more effective results when given more time, the multi-stage method consistently achieves higher test accuracy than the other two methods across all optimization time values. For example, best test accuracy is achieved by the multi-stage algorithm at time (45 min) for the QA task, while both the full Bayesian Optimization and the subset variant can only achieve a fraction of the best value at the same time value. We also note in general the multi-stage algorithm approaches the upper bound more rapidly as more time is given. This shows that the new algorithm is superior across a wide range of time values.

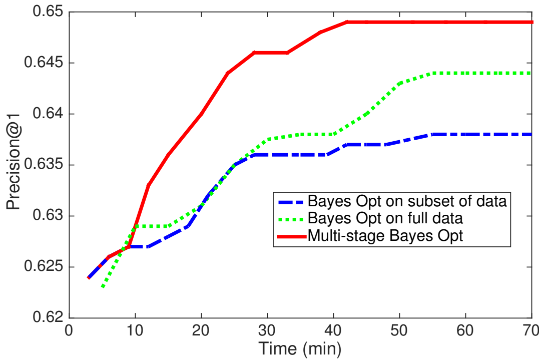

Figure 1: QA task: test accuracy vs tuning time.

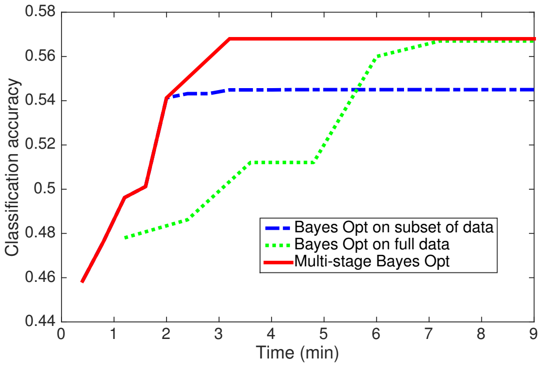

Figure 2: Yelp classification: test accuracy vs tuning time.

3.2 Expected accuracy and cost per iteration

To investigate the average accuracy and cost per iteration achieved by different methods across different time points, we compare their mean expected accuracy (according to precision@1 for QA and classification accuracy for Yelp) in Table 2, and their average speed in Table 3. In terms of average accuracy, we see that the state-of-the-art Bayesian optimization on full training data and the multi-stage algorithm achieve similar test accuracy, and they both outperform the subset variant of Bayesian Optimization. However,

	QA	**Yelp**
Bayes opt on small subset	0.633	0.530
Bayes opt on full data	0.639	0.543
Multi-stage algorithm	0.641	0.542

Table 2: Average test accuracy for QA (precision@1) and Yelp dataset (classif. accuracy).

	QA	Yelp
Bayes opt on small subset	3.2 min	0.4 min
Bayes opt on full data	7.8 min	1.2 min
Multi-stage algorithm	6 min	0.6 min

Table 3: Average time (min) per iteration.

in terms of time per iteration, the full Bayesian Optimization is the most expensive, taking more than twice amount of time over subset variant algorithm, while the multi-stage is 23% and 50% faster than standard Bayesian Optimization on QA and Yelp (Table 3), respectively, while maintaining the same accuracy as full Bayesian Optimization. This demonstrates the multi-stage approach achieves a good balance between the two baselines and can simultaneously delivers good speedup and accuracy.

4 Conclusion

We introduced a multi-stage optimization algorithm for hyper-parameter optimization. The proposed algorithm breaks the problem into multiple stages with increasingly amounts of data for efficient optimization. We demonstrated its improved performance as compared to the state-of-the-art Bayesian optimization algorithm and fast variants of Bayesian optimization on sentiment classification and QA tasks.

References

Remi Bardenet, Matyas Brendel, Balazs Kegls, and Michele Sebag. 2013. Collaborative hyperparameter tuning. In *ICML 2013: Proceedings of the 30th International Conference on Machine Learning.*

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. In *Pattern Analysis and Machine Intelligence,* Volume:35 , Issue: 8, 2013.

James Bergstra, Remi Bardenet, Yoshua Bengio, and Balazs Kegl. 2011. Algorithms for Hyper-Parameter Optimization. In *NIPS 2011: Advances in Neural Information Processing Systems.*

James Bergstra, and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. In *The Journal of Machine Learning Research,* Volume 13 Issue 1, January 2012.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. 2010. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. In *arXiv:1012.2599v1, Dec 12, 2010.*

Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown 2014. Surrogate Benchmarks for Hyperparameter Optimization. In *Meta-Learning and Algorithm Selection Workshop, 2014.*

Katharina Eggensperger, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2015. Efficient Benchmarking of Hyperparameter Optimizers via Surrogates. In *AAAI 2015: Twenty-ninth AAAI Conference.*

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. In *Journal of Machine Learning Research: 2008, 1871-1874.*

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. 2015. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In *AAAI 2015: Twenty-ninth AAAI Conference.*

Yasser Ganjisaffar, Rich Caruana, and Cristina Lopes. 2011. Bagging Gradient-Boosted Trees for High Precision, Low Variance Ranking Models. In *SIGIR 2011: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information.*

Frank Hutter, Holger H. Hoos and Kevin Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *LION4, 2011.*

Frank Hutter, Holger Hoos and Kevin Leyton-brown. 2014. An Efficient Approach for Assessing Hyperparameter Importance. In *ICML 2014: Proceedings of the 31st International Conference on Machine Learning*

Brent Komer, James Bergstra, and Chris Eliasmith. 2014. Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. In *SCIPY 2014: Proceedings of the 13th Python In Science Conference.*

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. Introduction to Information Retrieval. In *Cambridge University Press, 2008.*

Jasper Snoek, Hugo Larochelle, and Ryan Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *NIPS 2012: Advances in Neural Information Processing Systems.*

Kevin Swersky, Jasper Snoek, and Ryan Adams. 2013. Multi-Task Bayesian Optimization. In *NIPS 2013: Advances in Neural Information Processing Systems.*

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *KDD 2013: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining.*

Ellen M. Voorhees. 2011. The TREC question answering track. In *Natural Language Engineering,*Volume 7 Issue 4, December 2001

Yelp Academic Challenge Dataset. `http://www.yelp.com/dataset_challenge`.

Dani Yogatama, and Gideon Mann. 2014. Efficient Transfer Learning Method for Automatic Hyperparameter Tuning. In *AISTATS 2014: International Conference on Artificial Intelligence and Statistics* .

Alice X. Zheng, and Mikhail Bilenkoh. 2013. Lazy Paired Hyper-Parameter Tuning. In *IJCAI 2013: Twenty-third International Joint Conference on Artificial Intelligence* .