Automated Machine Learning for Soft Voting in an Ensemble of Tree-based Classifiers

Jungtaek Kim JTKIM@POSTECH.AC.KR and **Seungjin Choi** SEUNGJIN@POSTECH.AC.KR
Pohang University of Science and Technology, Pohang 37673, Republic of Korea

Abstract

We elucidate our automated machine learning system, referred to as *mlg.postech*, which ranked second in *AutoML Challenge 2018* that was held as the data competition at PAKDD-2018. Our system focuses on an automated construction of ensemble classifiers, employing the Bayesian optimization to determine weights in the soft voting classifier as well as hyperparamters involving base classifiers. As base classifiers, we use three tree-based models, including gradient boosting, extra-trees and random forests classifiers. Our implementation uses tree-based classifiers provided by `scikit-learn`, as well as our own Bayesian optimization package for sequential optimization of hyperparameters in the ensemble of tree-based classifiers. Empirical results on the AutoML Challenge 2018 datasets are presented to demonstrate the useful behavior of our system.

Keywords: Automated machine learning, Bayesian optimization, ensemble classifiers, tree-based classifiers, voting classifiers.

1. Introduction

Automated machine learning attempts to find automatically the optimal machine learning model without human intervention (Guyon et al., 2015). It usually includes algorithm selection and hyperparameter optimization, as well as model parameter learning. Note that $\mathcal{A}$ and Λ are an algorithm space and a hyperparameter space, respectively. Given a training dataset $\mathcal{D}_{\text{train}} = \{\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}\}$ and a validation dataset $\mathcal{D}_{\text{val}} = \{\mathbf{X}_{\text{val}}, \mathbf{y}_{\text{val}}\}$, an automated machine learning system finds the optimal algorithm vector $\mathbf{A}^* \in \mathcal{A}$ and hyperparameter vector $\boldsymbol{\lambda}^* \in \Lambda$:

$$(\mathbf{A}^*, \boldsymbol{\lambda}^*) = \text{AutoML}(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{A}, \Lambda) \tag{1}$$

where AutoML is an automated machine learning system over $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{A}$, and Λ (see Section 3 for the details).

The previous works (Hutter et al., 2011; Snoek et al., 2012; Feurer et al., 2015) have employed Bayesian optimization in hyperparameter optimization and automated machine learning, because Bayesian optimization is a global optimization method for black-box function. Especially, most participants of the previous AutoML Challenge (Guyon et al., 2016; Kim et al., 2016) utilized Bayesian optimization in automated machine learning system. Moreover, a hyperparameter optimization company, SigOpt uses Bayesian optimization in their optimization service (Martinez-Cantin et al., 2018).

Our automated machine learning system[1], which is referred to as *mlg.postech*[2], finds an appropriate machine learning model using Bayesian optimization. In particular, our system focuses on tree-based classifiers: gradient boosting classifier, extra-trees classifier, and random forests classifier, because they usually outperform for various datasets. We build a soft majority voting model, an ensemble of the tree-based classifiers, the hyperparameters of which are automatically tuned by Bayesian optimization. In practice, some machine learning models such as tree-based classifiers in our system are implemented by a Python machine learning package, `scikit-learn` (Pedregosa et al., 2011). Moreover, their hyperparameters are optimized by our own Bayesian optimization package, which is included in our system repository. Finally, our system, *mlg.postech* took second place of *AutoML Challenge 2018*, which was held as the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2018) data competition.

First, we introduce an ensemble method used in our system and Bayesian optimization which optimizes the hyperparameters in Section 2. Next, we present our automated machine learning system, *mlg.postech* in Section 3 and the AutoML Challenge 2018 result in Section 4.

2. Background

We will briefly introduce majority voting and Bayesian optimization in the subsequent section.

2.1. Majority Voting

Majority voting is an ensemble method to construct a classifier using a majority vote of k base classifiers. It has two types: hard voting and soft voting. For a hard voting, each base classifier has one vote (i.e. $w_j = 1$) if uniform weight is given, and $w_j \in \mathbb{N} \geq 1$ votes if occurrence of base classifier j is given. Class assignment of instance i for $1 \leq i \leq n$ is

$$c_i = \arg\max \sum_{j=1}^{k} w_j \mathbf{c}_i^{(j)}$$

where n is the number of instances, arg max returns an index of maximum value in given vector, $w_j \in \mathbb{N} \geq 1$ is a weight of base classifier j, and $\mathbf{c}_i^{(j)}$ is a class assignment of base classifier j (i.e. one-hot vector).

Each base classifier of soft *voting classifier* contributes class probabilities with given weights for all classes. Class assignment of soft voting classifier is

$$c_i = \arg\max \sum_{j=1}^{k} w_j \mathbf{p}_i^{(j)}$$

1. https://github.com/jungtaekkim/automl-challenge-2018
2. *narnars0* in AutoML Challenge 2018 leaderboard (https://competitions.codalab.org/competitions/17767#results). To represent our system precisely, we solely use *mlg.postech* as the name of our system and our team in this paper.

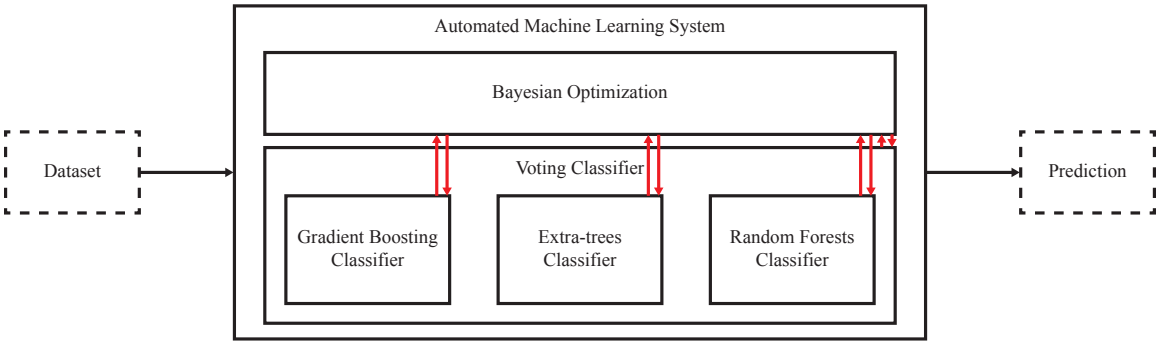

Figure 1: Our automated machine learning system, *mlg.postech*. Voting classifier which is constructed by three tree-based classifiers: gradient boosting classifier, extra-trees classifier, and random forests classifier produces predictions, where voting classifier and tree-based classifiers are iteratively optimized by Bayesian optimization for time budget (red arrows).

where $w_j \in \mathbb{R} \geq 0$ is a weight of base classifier j, and $\mathbf{p}_i^{(j)}$ is a class probability vector of base classifier j. In our system, we employ the soft majority voting classifier as an ensemble method.

2.2. Bayesian Optimization

Bayesian optimization is a useful method to find global minimum or maximum for black-box function (Brochu et al., 2010). It improves the current best solution as iterating the following steps: modeling a surrogate function and acquiring a next point that has maximum value of acquisition function. Because a target function is black-box, we model a surrogate function using a regression method that also produces an uncertainty, such as Gaussian process regression (Jones et al., 1998), random forests regression (Hutter et al., 2011), and Bayesian neural networks (Springenberg et al., 2016). Next, an acquisition function instead of an original target function is optimized. Generally, the acquisition function balances function estimate and uncertainty estimate. For example, one of the widely used acquisition functions, Gaussian process upper confidence bound (GP-UCB) (Srinivas et al., 2010) for a minimization case is

$$a_{\mathrm{UCB}}(\mathbf{x}) = -\mu(\mathbf{x}) + \kappa\sigma(\mathbf{x})$$

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ are posterior mean and posterior standard deviation functions over $\mathbf{x}$, respectively. κ is a balancing hyperparameter for posterior mean function and posterior standard deviation function. In our system, Gaussian process regression and GP-UCB are used as surrogate model and acquisition function.

3. Our Automated Machine Learning System, *mlg.postech*

We implement our automated machine learning system, written in Python, with a machine learning package, `scikit-learn` and our Bayesian optimization package. As shown in Fig. 1, a soft voting classifier as an ensemble model, which is constructed by three tree-based classifiers: gradient boosting classifier, extra-trees classifier, and random forests classifier is

Table 1: Datasets of feedback phase in AutoML Challenge 2018. Train. #, Valid. #, Test #, Feature #, Chrono., and Budget stand for training dataset size, validation dataset size, test dataset size, the number of features, chronological order, and time budget, respectively. Time budget shows in seconds.

Dataset	**Train. #**	**Valid. #**	**Test #**	**Feature #**	**Chrono.**	**Budget**
ada	4,147	415	41,471	48	False	600
arcene	100	100	700	10,000	False	600
gina	3,153	315	31,532	970	False	600
guillermo	20,000	5,000	5,000	4,296	False	1,200
rl	31,406	24,803	24,803	22	True	1,200

used to predict labels of input dataset. Because we specify our system to a single ensemble model, $\mathbf{A}^*$ in Eq. (1) is fixed.

As we explained before, our system has some components (see Fig. 1), which are controlled by six hyperparameters (now, we can think Λ in Eq. (1) as a six-dimensional space): (i) extra-trees classifier weight/gradient boosting classifier weight for voting classifier, (ii) random forests classifier weight/gradient boosting classifier weight for voting classifier, (iii) the number of estimators for gradient boosting classifier, (iv) the number of estimators for extra-trees classifier, (v) the number of estimators for random forests classifier, and (vi) maximum depth of gradient boosting classifier. Each hyperparameter is constrained and searched in the pre-defined range, which is set by one of four conditions: (i) large dataset size and large feature dimensions (we do not pass Bayesian optimization step for this condition and fit with the whole training dataset directly, due to time budget limitation.), (ii) small dataset size and large feature dimensions, (iii) dataset which has a chronological order, and (iv) otherwise (see our open repository to check the detailed conditions and pre-defined ranges).

Bayesian optimization with the acquisition function, GP-UCB optimizes the hyperparameters of our system. To optimize our system without validation and test datasets, we first split training dataset to training and validation datasets for Bayesian optimization, $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$ in Eq. (1) (e.g. 0.60 of training dataset as training dataset for Bayesian optimization and 0.40 of training dataset as validation dataset for Bayesian optimization). Bayesian optimization in our system iteratively finds the model that shows the best performance measure for the given time budget. Finally, if the system leaves time to be able to run one loop (including time margin), our system fits the final machine learning model with the whole training dataset and the hyperparameters which produce the current best model.

4. Challenge Result

AutoML Challenge 2018 is a competition for automating a machine learning pipeline, including feature transformation, algorithm selection, hyperparameter optimization, and model parameter learning, where dataset and machine learning task are given. We will describe the details and result of AutoML Challenge 2018 in the following section.

Table 2: AutoML Challenge 2018 result. Our system, *mlg.postech* took second place in the challenge. A normalized area under the ROC curve (AUC) score (upper cell in each row) is computed for each dataset, and a dataset rank (lower cell in each row) is determined by numerical order of the normalized AUC score. Finally, an overall rank is determined by the average rank of five datasets.

Place	Team	Set 1	Set 2	Set 3	Set 4	Set 5	Average
1	aad_freiburg	0.5533 (3)	0.2839 (4)	0.3932 (1)	0.2635 (1)	0.6766 (5)	2.8
2	**mlg.postech**	**0.5418 (5)**	**0.2894 (2)**	**0.3665 (2)**	**0.2005 (9)**	**0.6922 (1)**	**3.8**
	wlWangl	0.5655 (2)	0.4851 (1)	0.2829 (5)	-0.0886 (16)	0.6840 (3)	5.4
3	thanhdng	0.5131 (6)	0.2256 (8)	0.2605 (7)	0.2603 (2)	0.6777 (4)	5.4
	Malik	0.5085 (7)	0.2297 (7)	0.2670 (6)	0.2413 (5)	0.6853 (2)	5.4

4.1. Challenge Details

AutoML Challenge 2018 has two phases: feedback phase and AutoML challenge phase. In the feedback phase, five datasets for binary classification (see Table 1) are provided, and each dataset has training/validation/test datasets. All instances in training dataset are labeled, and performance measure for validation dataset is posted in the leaderboard of the feedback phase, after submitting a code or prediction file. In the AutoML challenge phase, the automated machine learning system that has submitted in the feedback phase solves five blind datasets for binary classification. Finally, performance measures for those blind datasets are used to determine challenge winners. More precisely, the performance measures of all participants for a dataset determine a dataset rank, and an overall rank is determined by average of all dataset ranks (see Table 2).

A normalized area under the ROC curve (AUC) metric is used to measure performance of the submitted automated machine learning system. The normalized AUC metric can be written as

$$\text{Normalized AUC} = 2 \cdot \text{AUC} - 1 \tag{2}$$

where AUC is computed as a standard AUC metric. Since AUC is bounded in $[0.0, 1.0]$, the normalized AUC is transformed into $[-1.0, 1.0]$. Moreover, because every dataset has its own time budget, the submitted automated machine learning system should finish each task within the given time budget. Based on the announcement of the challenge organizers, each submission is executed in the Ubuntu machine which has (i) 2 cores, (ii) 8GB memory, and (iii) 40GB SSD.

4.2. AutoML Challenge 2018

As shown in Table 2, our automated machine learning system took second place in AutoML Challenge 2018. Our system showed the normalized AUC scores which took 5th, 2nd, 2nd,

9th, and 1st in Set 1, Set 2, Set 3, Set 4, and Set 5, respectively. Additionally, in the feedback phase, our system ranked first, but was tied with the team, wlWangl.

5. Conclusion

Our automated machine learning system, *mlg.postech*, using a soft majority voting classifier with gradient boosting classifier, extra-trees classifier, and random forests classifier took second place in AutoML Challenge 2018. We could show that Bayesian optimization is effective to optimize the hyperparameters of the voting classifier with the tree-based classifiers. Especially, we could validate our system could train and test blind datasets without human intervention.

References

E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010. arXiv preprint arXiv:1012.2599.

M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2962–2970, Montreal, Quebec, Canada, 2015.

I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macià, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. Design of the 2015 ChaLearn AutoML Challenge. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Killarney, Ireland, 2015.

I. Guyon, I. Chaabane, H. J. Escalante, S. Escalera, D. Jajetic, J. R. Lloyd, N. Macia, B. Ray, L. Romaszko, M. Sebag, A. Statnikov, S. Treguer, and E. Viegas. A brief review of the ChaLearn AutoML Challenge. In *International Conference on Machine Learning Workshop on Automatic Machine Learning*, New York, New York, USA, 2016.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization*, pages 507–523, Rome, Italy, 2011.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

J. Kim, J. Jeong, and S. Choi. AutoML Challenge: AutoML framework using random space partitioning optimizer. In *International Conference on Machine Learning Workshop on Automatic Machine Learning*, New York, New York, USA, 2016.

R. Martinez-Cantin, K. Tee, and M. McCourt. Practical Bayesian optimization in the presence of outliers. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Playa Blanca, Lanzarote, Canary Islands, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, volume 25, pages 2951–2959, Lake Tahoe, Nevada, USA, 2012.

J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 29, pages 4134–4142, Barcelona, Spain, 2016.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1015–1022, Haifa, Israel, 2010.