Bayesian Optimization and Automated Machine Learning

Jungtaek Kim (jtkim@postech.ac.kr)
Advisor: Prof. Seungjin Choi

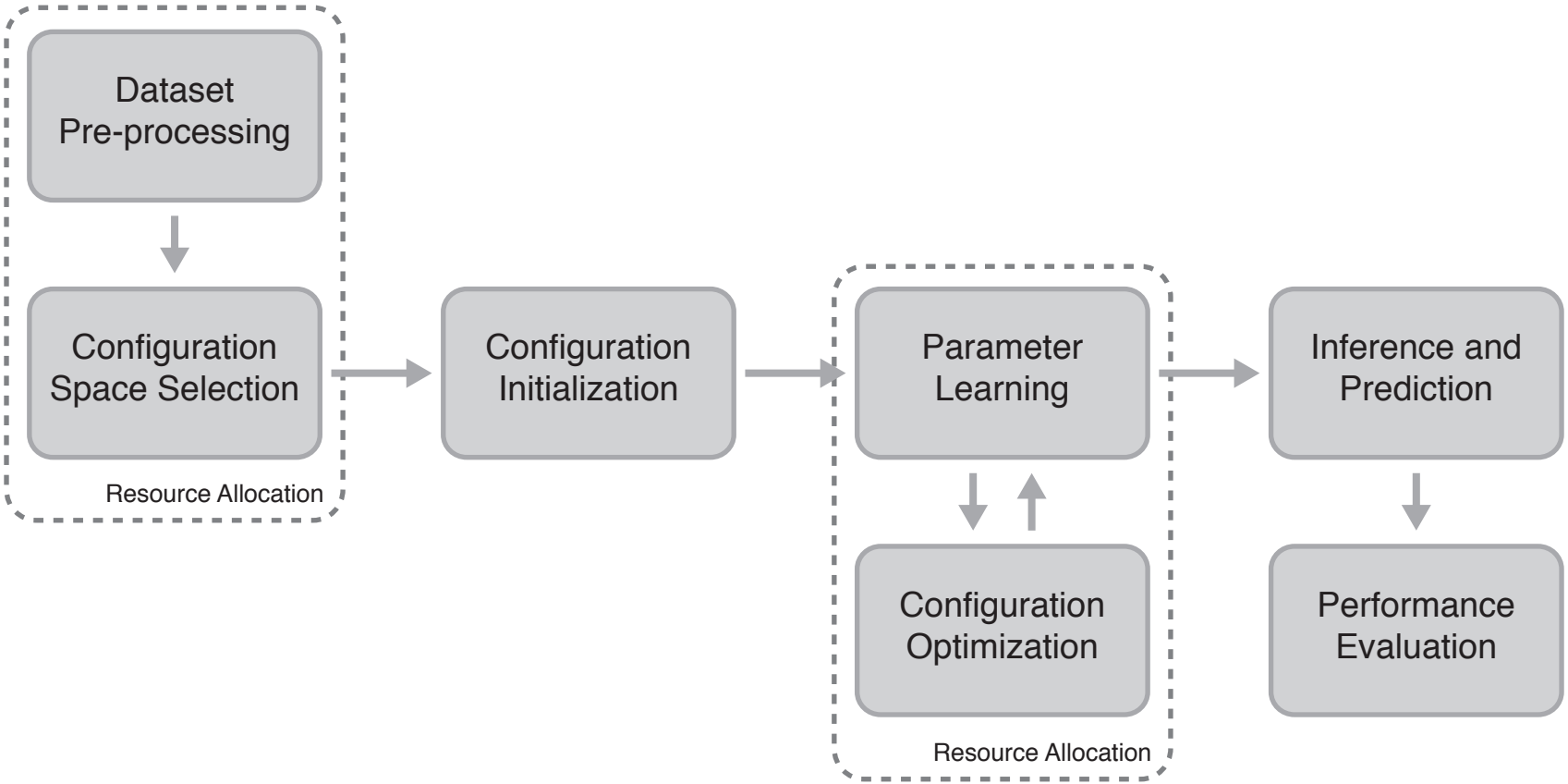
Machine Learning Group,
Department of Computer Science and Engineering, POSTECH,
77 Cheongam-ro, Nam-gu, Pohang 37673,
Gyeongsangbuk-do, Republic of Korea

December 13, 2018

Table of Contents

Overview

Automated Machine Learning

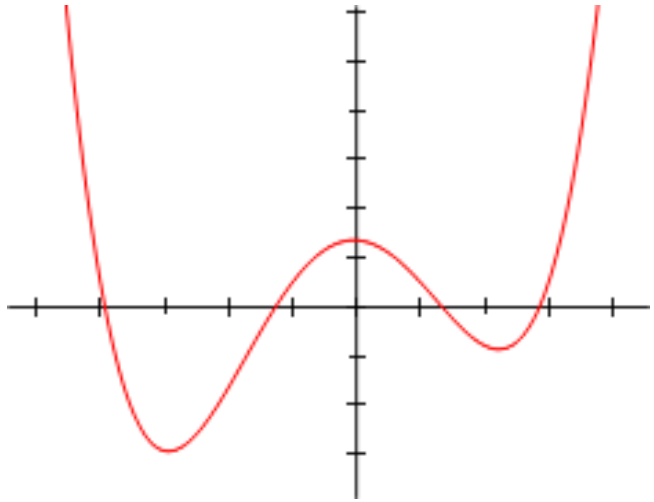


Automated Machine Learning

- Data pre-processing
- Configuration space selection
  - Specify a searching space $\mathcal{S}$ (e.g., a Cartesian space of kernel type, penalty parameter, kernel coefficient, and degree)
- Configuration initialization
  - Random initialization
  - Low discrepancy initialization
  - Learn to initialize [Kim et al., 2017]
- Configuration optimization
  - Random search
  - Grid search
  - Bayesian optimization
  - Neural architecture search
- Resource allocation

Bayesian Optimization

Global Optimization



From Wikipedia (`https://en.wikipedia.org/wiki/Local_optimum`)

- A method to find global minimum or maximum of given target function:

$$\boldsymbol{x}^* = \arg\min \mathcal{L}(\boldsymbol{x}),$$

or

$$\boldsymbol{x}^* = \arg\max \mathcal{L}(\boldsymbol{x}).$$

Target Functions in Bayesian Optimization

- Usually an expensive black-box function.
- Unknown functional forms or local geometric features such as saddle points, global optima, and local optima.
- Uncertain function continuity.
- High-dimensional and mixed-variable domain space.

Bayesian Approach

- In Bayesian inference, given a prior knowledge for parameters, $p(\boldsymbol{\theta}|\boldsymbol{\lambda})$ and a likelihood over dataset, conditional to parameters, $p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\lambda})$, the posterior distribution:

$$p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\lambda}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\lambda})p(\boldsymbol{\theta}|\boldsymbol{\lambda})}{p(\mathcal{D}|\boldsymbol{\lambda})} = \frac{p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\lambda})p(\boldsymbol{\theta}|\boldsymbol{\lambda})}{\int p(\mathcal{D}|\boldsymbol{\theta}, \boldsymbol{\lambda})p(\boldsymbol{\theta}|\boldsymbol{\lambda})\mathrm{d}\boldsymbol{\theta}}$$

  where $\boldsymbol{\theta}$ is a vector of parameters, $\mathcal{D}$ is an observed dataset, and $\boldsymbol{\lambda}$ is a vector of hyperparameters.
- Produce an uncertainty as well as a prediction.

Bayesian Optimization

- A powerful strategy for finding the extrema of objective functions that are expensive to evaluate,
  - where one does not have a closed-form expression for the objective function,
  - but where one can obtain observations at sampled values.
- Since we do not know a target function, optimize an acquisition function, instead of the target function.
- Compute acquisition function using outputs of Bayesian regression model.

Bayesian Optimization

Algorithm 1 Bayesian Optimization

Input: Initial data $\mathcal{D}_{1:k} = \{(\boldsymbol{x}_i, y_i)_{1:k}\}$ and time budget T.

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a function $f^*(\boldsymbol{x}|\mathcal{D}_{1:k+t-1})$ considered as an objective function.
3: Find $\boldsymbol{x}_{k+t}$ that maximizes an acquisition function, $\boldsymbol{x}_{k+t} = \arg\max_{\boldsymbol{x}} a(\boldsymbol{x}|\mathcal{D}_{1:k+t-1})$.
4: Sample the true objective function, $y_{k+t} = f(\boldsymbol{x}_{k+t}) + \epsilon_{k+t}$.
5: Update on $\mathcal{D}_{1:k+t} = \{\mathcal{D}_{1:k+t-1}, (\boldsymbol{x}_t, y_t)\}$.
6: **end for**

Surrogate Function

- Replace a true function with a surrogate function.
- To balance exploration and exploitation, predict a function estimate and its uncertainty estimate.
- Gaussian process regression, random forest regression [Hutter et al., 2011], and Bayesian neural network [Springenberg et al., 2016] are used.

Background: Gaussian Process

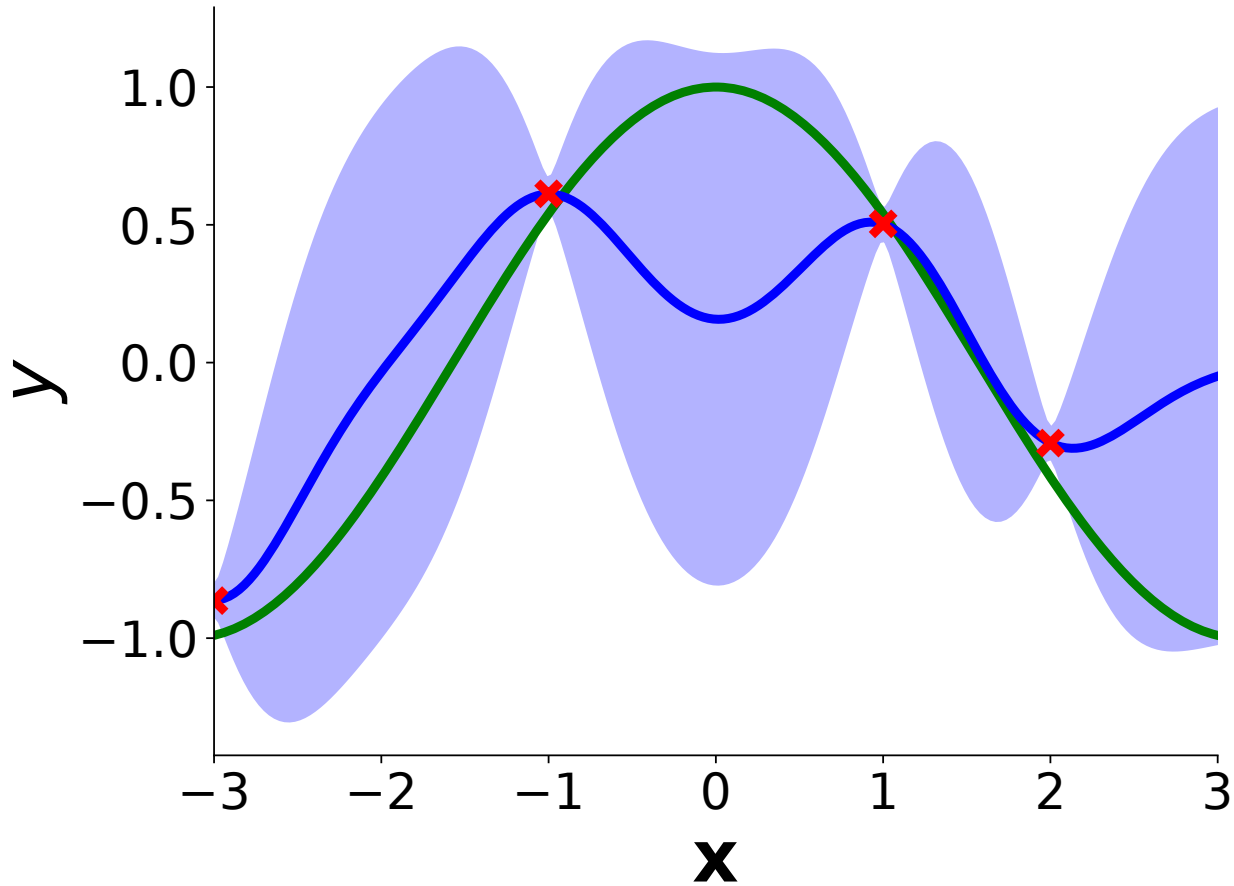
- A collection of random variables, any finite number of which have a joint Gaussian distribution [Rasmussen and Williams, 2006].
- Generally, Gaussian process (GP):

$$f \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

where

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{D}[f(\boldsymbol{x})] \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{D}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))]. \end{aligned}$$

Background: Gaussian Process Regression



Background: Gaussian Process Regression

- One of basic covariance functions, the squared-exponential covariance function in one dimension:

$$k\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left(\boldsymbol{x} - \boldsymbol{x}'\right)^2\right) + \sigma_n^2 \delta_{\boldsymbol{x}\boldsymbol{x}'},$$

  where σ_f is the signal standard deviation, l is the length scale and σ_n is the noise standard deviation. [Rasmussen and Williams, 2006]

- Posterior mean function $\mu(\cdot)$ and covariance function $\Sigma(\cdot)$:

$$\boldsymbol{\mu}^* = K(\boldsymbol{X}^*, \boldsymbol{X})(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I})^{-1}\boldsymbol{y},$$
$$\boldsymbol{\Sigma}^* = K(\boldsymbol{X}^*, \boldsymbol{X}^*) - K(\boldsymbol{X}^*, \boldsymbol{X})(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I})^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*).$$

Background: Gaussian Process Regression

- If non-zero mean prior is given, posterior mean and covariance functions:

$$\boldsymbol{\mu}^* = K(\boldsymbol{X}^*, \boldsymbol{X})(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})(\boldsymbol{y} - \mu(\boldsymbol{X})) + \mu(\boldsymbol{X})$$
$$\boldsymbol{\Sigma}^* = K(\boldsymbol{X}^*, \boldsymbol{X}^*) + K(\boldsymbol{X}^*, \boldsymbol{X})(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})^{-1} K(\boldsymbol{X}, \boldsymbol{X}^*).$$

Acquisition Function

- A function that acquires a next point to evaluate for an expensive black-box function.
- Traditionally, the probability of improvement (PI) [Kushner, 1964], the expected improvement (EI) [Moćkus et al., 1978], and GP upper confidence bound (GP-UCB) [Srinivas et al., 2010] are used.
- Several functions such as entropy search [Hennig and Schuler, 2012] and a combination of existing functions [Kim and Choi, 2018c] have been recently proposed.

Traditional Acquisition Functions (Minimization Case)

- PI [Kushner, 1964]

$$a_{\mathrm{PI}}(\boldsymbol{x}|\mathcal{D},\boldsymbol{\lambda}) = \Phi(Z),$$

- EI [Moćkus et al., 1978]

$$a_{\mathrm{EI}}(\boldsymbol{x}|\mathcal{D},\boldsymbol{\lambda}) = \begin{cases} (f(\boldsymbol{x}^\dagger)-\mu(\boldsymbol{x}))\Phi(Z)+\sigma(\boldsymbol{x})\phi(Z) & \text{if } \sigma(\boldsymbol{x})>0 \\ 0 & \text{if } \sigma(\boldsymbol{x})=0 \end{cases},$$

- GP-UCB [Srinivas et al., 2010]

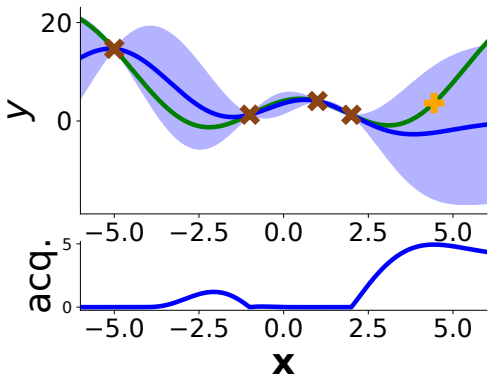
$$a_{\mathrm{UCB}}(\boldsymbol{x}|\mathcal{D},\boldsymbol{\lambda}) = -\mu(\boldsymbol{x}) + \beta\sigma(\boldsymbol{x}),$$

where

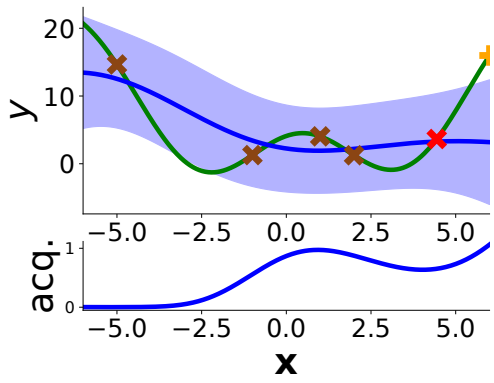
$$Z = \begin{cases} \frac{f(\boldsymbol{x}^\dagger)-\mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})} & \text{if } \sigma(\boldsymbol{x})>0 \\ 0 & \text{if } \sigma(\boldsymbol{x})=0 \end{cases}$$

$$\mu(\boldsymbol{x}) \coloneqq \mu(\boldsymbol{x}|\mathcal{D},\boldsymbol{\lambda}),\ \sigma(\boldsymbol{x}) \coloneqq \sigma(\boldsymbol{x}|\mathcal{D},\boldsymbol{\lambda}).$$

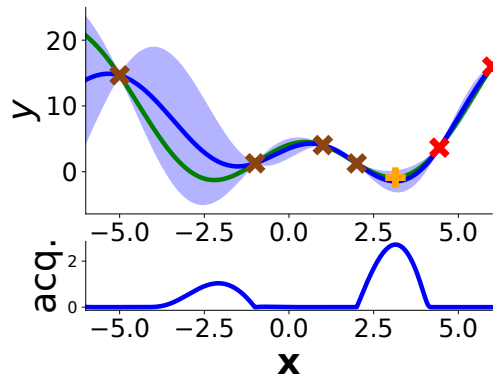
Synthetic Examples



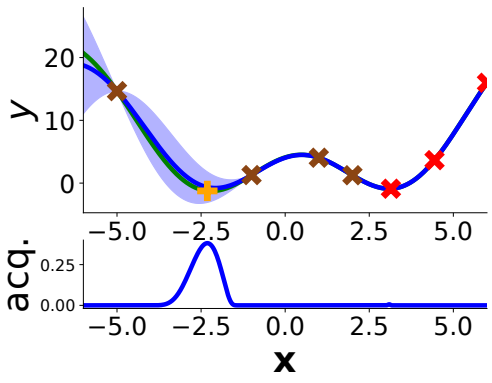
(a) Iteration 1



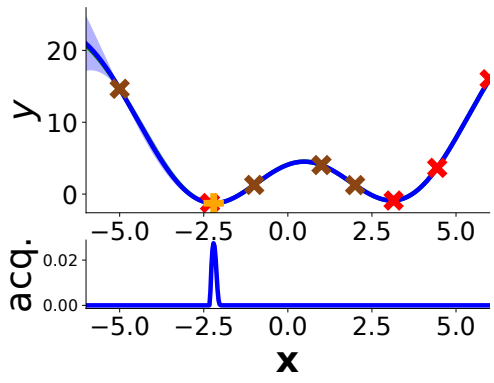
(b) Iteration 2



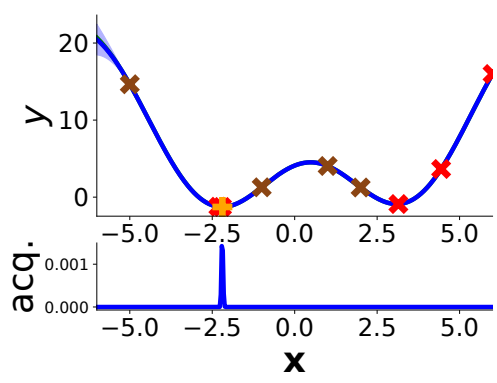
(c) Iteration 3



(d) Iteration 4



(e) Iteration 5



(f) Iteration 6

Figure 1: $y = 4.0\cos(\boldsymbol{x}) + 0.1\boldsymbol{x} + 2.0\sin(\boldsymbol{x}) + 0.4(\boldsymbol{x} - 0.5)^2$. EI is used to optimize.

bayeso

- Simple, but essential Bayesian optimization package.
- Written in `Python`.
- Licensed under the MIT license.
- `https://github.com/jungtaekkim/bayeso`

Automated Machine Learning

Automated Machine Learning

- Attempt to find automatically the optimal machine learning model without human intervention.
- Usually include feature transformation, algorithm selection, and hyperparameter optimization.
- Given a training dataset $\mathcal{D}_{\text{train}}$ and a validation dataset $\mathcal{D}_{\text{val}}$, the optimal hyperparameter vector $\boldsymbol{\lambda}^*$ for an automated machine learning system:

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

where AutoML is an automated machine learning system and $\lambda \in \Lambda$.

Issues on Automated Machine Learning

- High-dimensional searching space
- Categorical variables
- Uncertainty estimation
- Performance evaluation without re-training
- Efficient automated machine learning
- Automated machine learning for neural networks
- Parallelism for automated machine learning
- Initialization for automated machine learning
- Resource allocation for automated machine learning

Previous Works

- Bayesian optimization and hyperparameter optimization
  - `GPyOpt` [The GPyOpt authors, 2016]
  - `SMAC` [Hutter et al., 2011]
  - `BayesOpt` [Martinez-Cantin, 2014]
  - `bayeso`
  - SigOpt API [Martinez-Cantin et al., 2018]
- Automated machine learning framework
  - `auto-sklearn` [Feurer et al., 2015]
  - `Auto-WEKA` [Thornton et al., 2013]
  - Our previous work [Kim et al., 2016]

Background: Soft Majority Voting

- An ensemble method to construct a classifier using a majority vote of k base classifiers.
- Class assignment of soft majority voting classifier:

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

for $1 \leq i \leq n$ where n is the number of instances, arg max returns an index of maximum value in given vector, $w_j \in \mathbb{R} \geq 0$ is a weight of base classifier j, and $\mathbf{p}_i^{(j)}$ is a class probability vector of base classifier j.

Our AutoML System [Kim and Choi, 2018a]

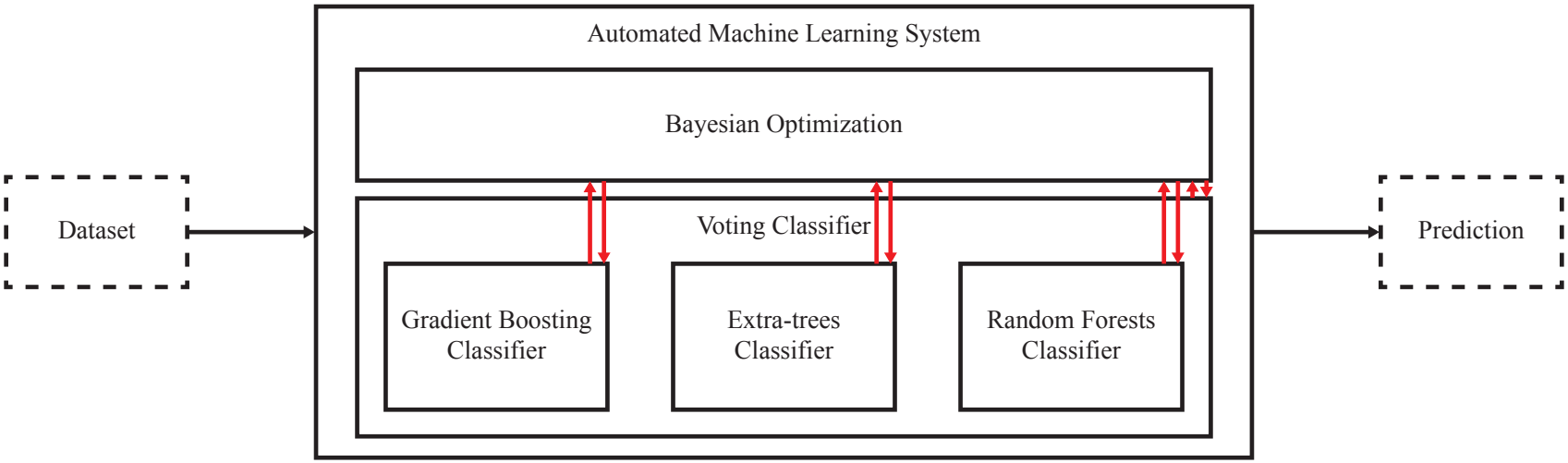


Figure 2: Our automated machine learning system. Voting classifier constructed by three tree-based classifiers: gradient boosting, extra-trees, and random forests classifiers produces predictions, where voting classifier and tree-based classifiers are iteratively optimized by Bayesian optimization for the given time budget.

Our AutoML System [Kim and Choi, 2018b]

- Written in `Python`.
- Use `scikit-learn` and our own Bayesian optimization package.
- Split training dataset to training (0.6) and validation (0.4) sets for Bayesian optimization.
- Optimize six hyperparameters:
  1. extra-trees classifier weight/gradient boosting classifier weight for voting classifier,
  2. random forests classifier weight/gradient boosting classifier weight for voting classifier,
  3. the number of estimators for gradient boosting classifier,
  4. the number of estimators for extra-trees classifier,
  5. the number of estimators for random forests classifier,
  6. maximum depth of gradient boosting classifier.
- Use GP-UCB.

AutoML Challenge 2018

- Two phases: feedback phase and AutoML challenge phase.
- In the feedback phase, provide five datasets for binary classification.
- Given training/validation/test datasets, after submitting a code or prediction file, validation measure is posted in the leaderboard.
- In the AutoML challenge phase, determine challenge winners, comparing a normalized area under the ROC curve (AUC) metric for blind datasets:

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

AutoML Challenge 2018

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

Figure 3: 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.

AutoML Challenge 2018 Result

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

Figure 4: AutoML Challenge 2018 result. 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.

Future Works

References

References I

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.

P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

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.

J. Kim and S. Choi. Automated machine learning for soft voting in an ensemble of tree-based classifiers. In *International Workshop on Automatic Machine Learning (AutoML)*, Stockholm, Sweden, 2018a.

J. Kim and S. Choi. Automated machine learning for soft voting in an ensemble of tree-based classifiers, 2018b. `https://github.com/jungtaekkim/automl-challenge-2018`.

J. Kim and S. Choi. Clustering-guided GP-UCB for Bayesian optimization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, Alberta, Canada, 2018c.

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 (AutoML)*, New York, New York, USA, 2016.

References II

J. Kim, S. Kim, and S. Choi. Learning to warm-start bayesian hyperparameter optimization. *arXiv e-prints*, arXiv:1710.06219, 2017.

H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1): 97–106, 1964.

R. Martinez-Cantin. BayesOpt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15:3735–3739, 2014.

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)*, Lanzarote, Spain, 2018.

J. Moćkus, V. Tiesis, and A. Źilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

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.

References III

The GPyOpt authors. GPyOpt: A Bayesian optimization framework in Python, 2016. `https://github.com/SheffieldML/GPyOpt`.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 847–855, Chicago, Illinois, USA, 2013.