Bayesian Optimization and Its Applications

Jungtaek Kim (jtkim@postech.ac.kr)

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

June 28, 2019

Table of Contents

Bayesian Optimization

Global Optimization

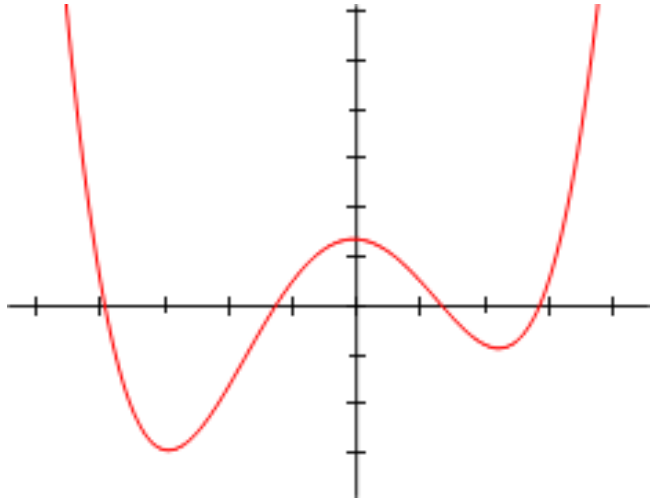


Figure 1: From Wikipedia.

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

$$\mathbf{x}^* = \arg\min \mathcal{L}(\mathbf{x}), \tag{1}$$

or

$$\mathbf{x}^* = \arg\max \mathcal{L}(\mathbf{x}). \tag{2}$$

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, $p(\boldsymbol{\theta} \mid \boldsymbol{\lambda})$ and a likelihood over dataset, conditional to parameters, $p(\mathcal{D} \mid \boldsymbol{\theta}, \boldsymbol{\lambda})$, the posterior distribution:

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

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

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a function $f^*(\mathbf{x} \mid \mathcal{D}_{1:k+t-1})$ considered as an objective function.
3: Find $\mathbf{x}_{k+t}$ that maximizes an acquisition function, $\mathbf{x}_{k+t} = \arg\max_{\mathbf{x}} a(\mathbf{x} \mid \mathcal{D}_{1:k+t-1})$.
4: Sample the true objective function, $y_{k+t} = f(\mathbf{x}_{k+t}) + \epsilon_{k+t}$.
5: Update on $\mathcal{D}_{1:k+t} = \{\mathcal{D}_{1:k+t-1}, (\mathbf{x}_t, y_t)\}$.
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 forests regression [Hutter et al., 2011], and Bayesian neural network [Springenberg et al., 2016] have been 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):

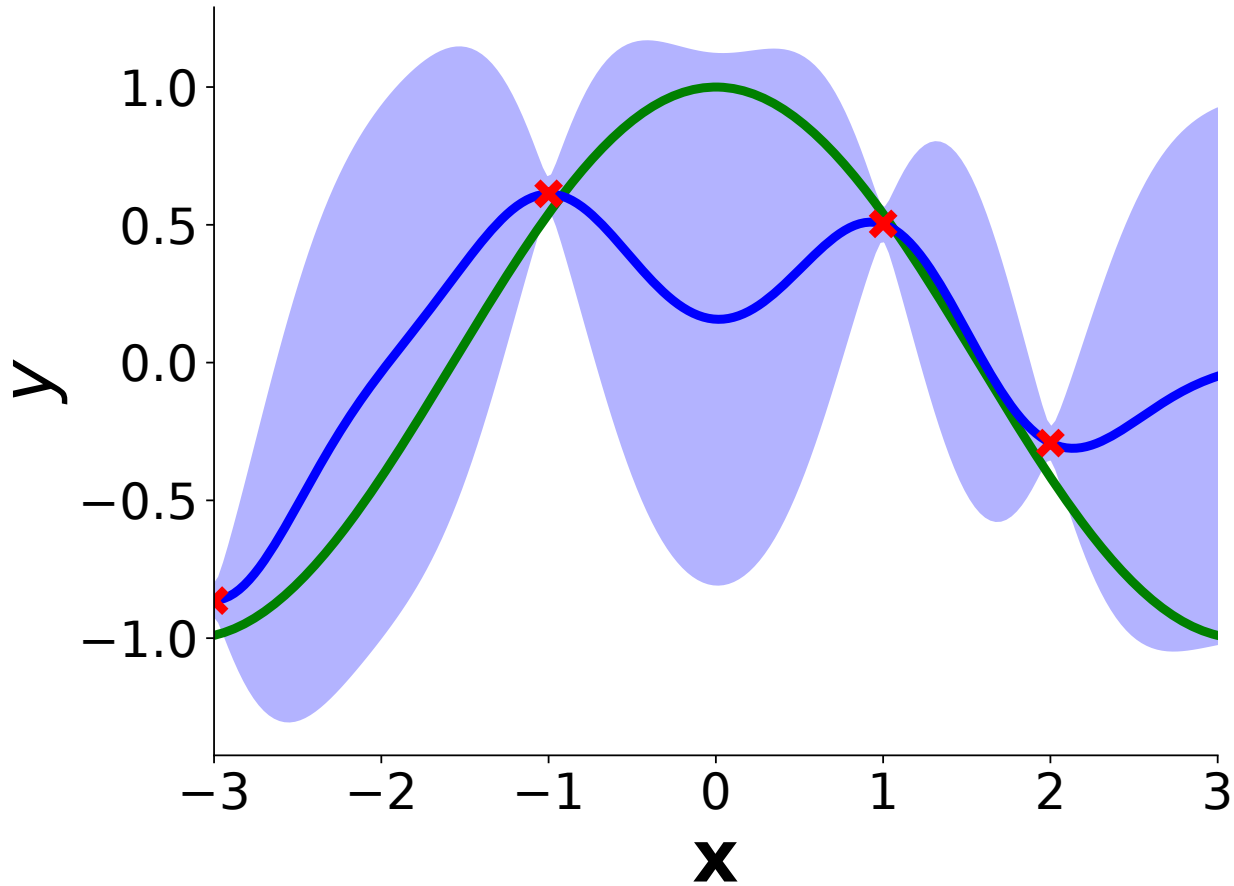
$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3}$$

where

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{4}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{5}$$

Background: Gaussian Process Regression



Background: Gaussian Process Regression

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

$$k\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left(x - x'\right)^2\right) + \sigma_n^2 \delta_{xx'}, \tag{6}$$

where σ_f is a signal level, l is a length scale and σ_n is a noise level [Rasmussen and Williams, 2006].

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

$$\mu(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y}, \tag{7}$$

$$\Sigma(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} K(\mathbf{X}, \mathbf{X}^*). \tag{8}$$

Background: Gaussian Process Regression

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

$$\mu(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})(\mathbf{y} - \mu_p(\mathbf{X})) + \mu_p(\mathbf{X}) \quad (9)$$

$$\Sigma(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X}^*) + K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} K(\mathbf{X}, \mathbf{X}^*) \quad (10)$$

where $\mu_p(\cdot)$ is a prior mean function.

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] have been 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]: $a_{\mathrm{PI}}(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}) = \Phi(z)$,
- EI [Moćkus et al., 1978]:

$$a_{\mathrm{EI}}(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}) = \begin{cases} (f(\mathbf{x}^\dagger) - \mu(\mathbf{x}))\Phi(z) + \sigma(\mathbf{x})\phi(z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}, \quad (11)$$

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

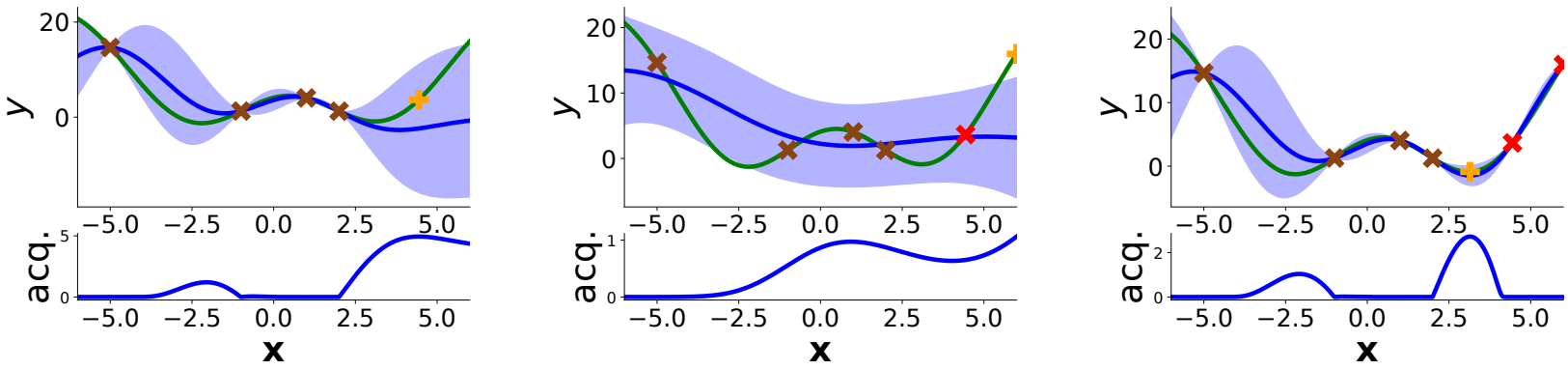
$$a_{\mathrm{UCB}}(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}) = -\mu(\mathbf{x}) + \beta\sigma(\mathbf{x}), \quad (12)$$

where

$$z = \begin{cases} \frac{f(\mathbf{x}^\dagger) - \mu(\mathbf{x})}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}, \quad (13)$$

$$\mathbf{x}^\dagger = \underset{(\mathbf{x}, y) \in \mathcal{D}}{\arg\min}\, y, \ \mu(\mathbf{x}) := \mu(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}), \ \sigma(\mathbf{x}) := \sigma(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}). \quad (14)$$

Synthetic Examples



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

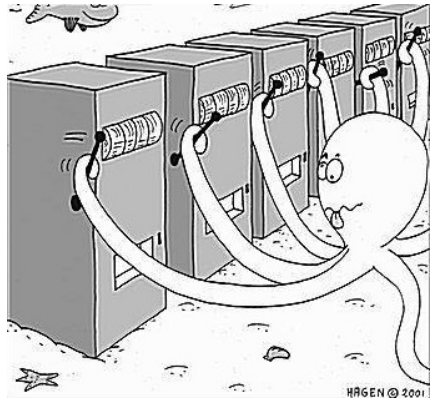
(d) Iteration 4 (e) Iteration 5 (f) Iteration 6

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

Acquisition Function Optimization

- We should find a global optimum of acquisition function.
- But, in practice, either global optimizer or local optimizer is used.
- Multi-started local optimizers can be a good option for acquisition function optimization.
- Analyses on these selections are provided in [Kim and Choi, 2019].

Relationship to Multi-Armed Bandit Problem



- Each machine returns a reward $\hat{r}_a \sim p_{\theta_a}(r_a)$ where $a \in \{1, \ldots, K\}$.
- Minimize a cumulative regret $T\mu^* - \sum_{t=1}^{T} \hat{r}_{a_t}$ where $\mu^* = \max_{a \in \{1,\ldots,K\}} \mu_a$.

- Bayesian optimization can be considered as infinite bandits with dependent arms.

Relationship to Thompson Sampling

- Thompson sampling is usually applied in multi-armed bandit problems.
- For the case of a beta-Bernoulli bandit, Thompson sampling is defined as

Algorithm 2 Thompson Sampling for a Beta-Bernoulli Bandit

for $t = 1, 2, \ldots$ **do**
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_k \sim \mathrm{beta}(\alpha_k, \beta_k)$
  end for
  $x_t \leftarrow \arg\max_k \hat{\theta}_k$
  Apply x_t and observe r_t
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$
end for

- After sampling the possibilities, it chooses a maximizer of those sampled values.

Issues on Bayesian Optimization

- Exploration and exploitation trade-off
  - Characteristics of Bayesian optimization
- Convergence guarantee
  - [Kawaguchi et al., 2015]
- Optimization with a constraint
  - [Gelbart et al., 2014] and [Gardner et al., 2014]
- Surrogate function modeling
  - GP regression is widely used.
  - Random forests [Hutter et al., 2011] and Mondrian forests [Kim et al., 2016]

Issues on Bayesian Optimization (cont'd)

- Acquisition function optimization
  - [Wilson et al., 2018] and [Kim and Choi, 2019]
- Auxiliary optimization elimination
  - [Wang et al., 2014]
- Batch Bayesian optimization
  - [González et al., 2016] and [Wang et al., 2018]
- Known target value, but unknown optimizer
  - [Nguyen and Osborne, 2019]

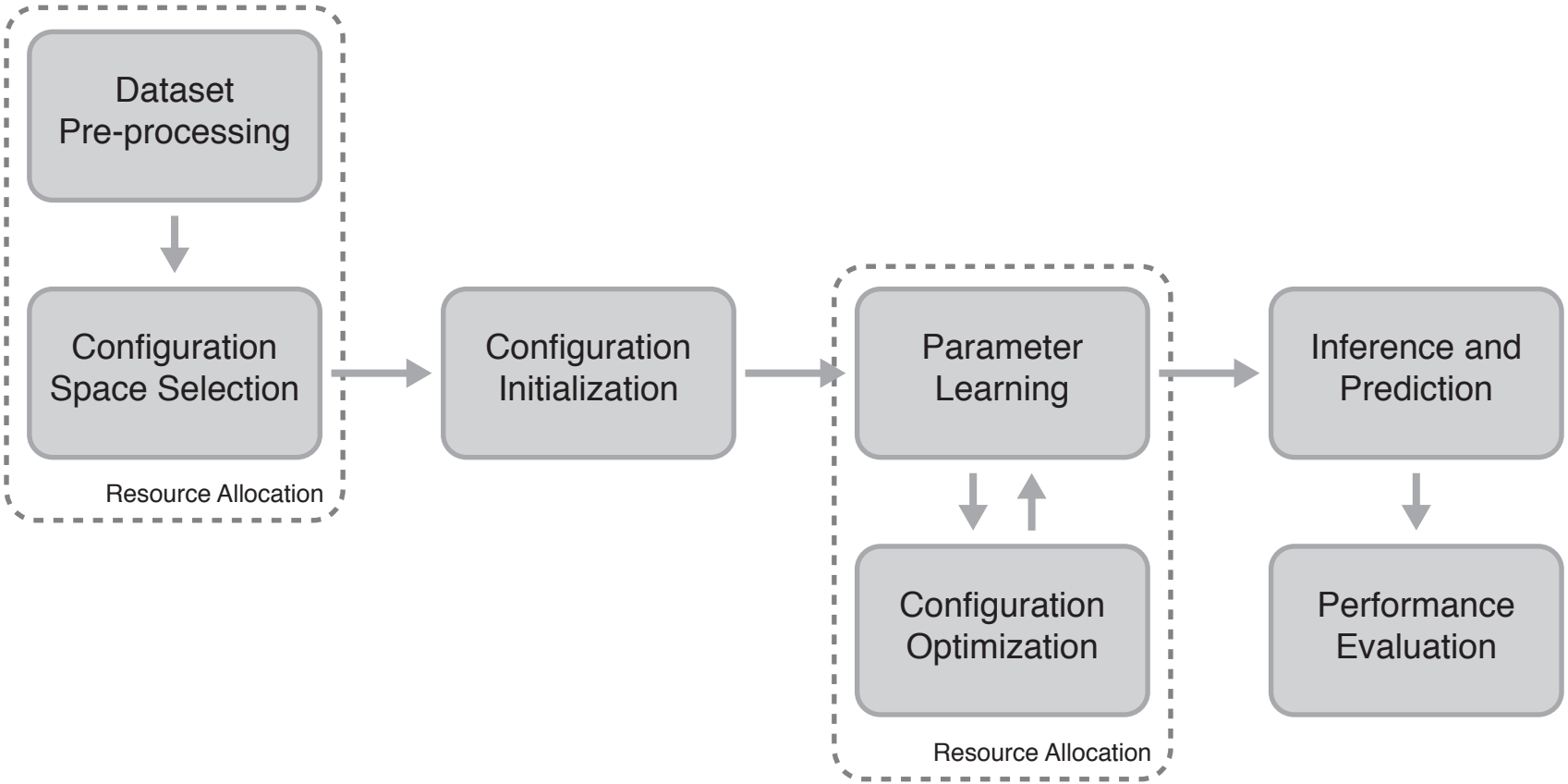
bayeso

- Simple, but essential Bayesian optimization package [Kim and Choi, 2017].
- Written in `Python`.
- Licensed under the MIT license.

- `https://github.com/jungtaekkim/bayeso`

Automated Machine Learning

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

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]

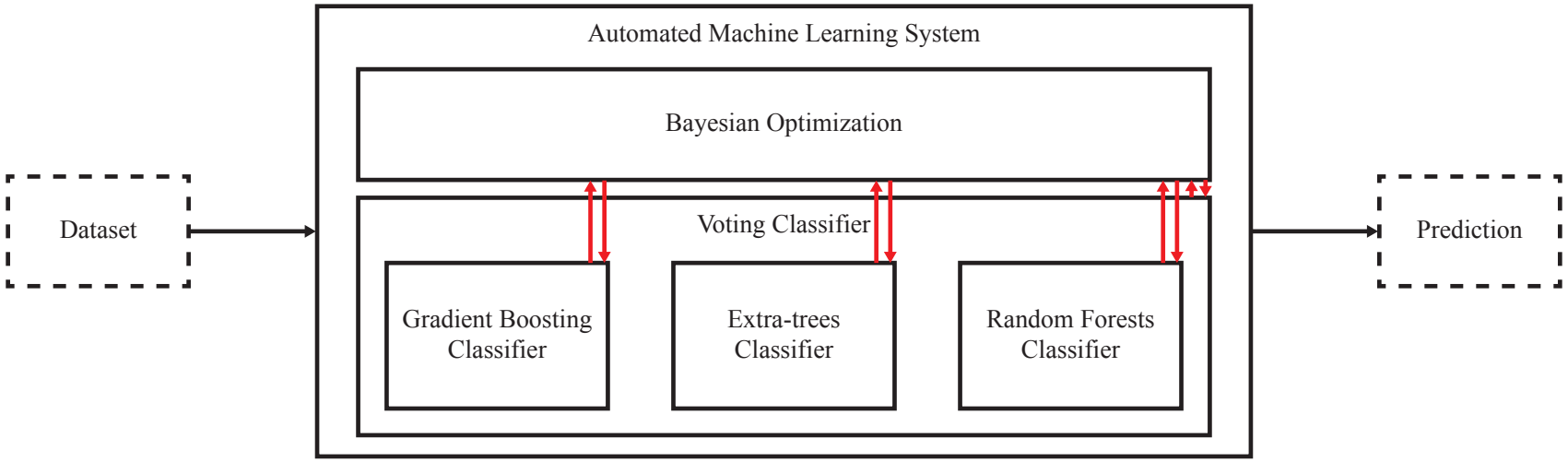


Figure 3: Our automated machine learning system. Voting classifier constructed by three tree-based classifiers: gradient boosting, extra-trees, and random forests classifiers produces predictions, where voting classifier and tree-based classifiers are iteratively optimized by Bayesian optimization for 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 4: 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 5: 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.

On Local Optimizers of Acquisition Functions in Bayesian Optimization

Jungtaek Kim and Seungjin Choi

https://arxiv.org/abs/1901.08350

Motivation

- To provide a theoretical guarantee for acquisition function optimization, in this paper we provide the bounds for instantaneous regret difference.
- We bound an instantaneous regret difference between single local optimizer and global optimizer with some probability.
- Moreover, we bound an instantaneous regret difference between multi-started local optimizers and global optimizer with some probability.

Main Results

- Instantaneous regret at iteration t is defined as

$$r_t = f(\mathbf{x}_t^*) - f(\mathbf{x}^\dagger) \tag{15}$$

where $\mathbf{x}_t^*$ is an acquired point at iteration t and $\mathbf{x}^\dagger = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.

Main Results

Theorem 1. *Given $\delta_l \in [0, 1)$ and $\epsilon_l, \epsilon_1, \epsilon_2 > 0$, an instantaneous regret difference with single local optimizer at iteration t, $r_{t,g}$ and $r_{t,l}$ is less than ϵ_1 with a probability at least $1 - \delta_l$:*

$$\mathbb{P}\left[\|r_{t,g} - r_{t,l}\|_2 < \epsilon_l\right] \geq 1 - \delta_l \tag{13}$$

where

$$\delta_l = \frac{\gamma}{\epsilon_1}(1 - \beta_g) + \frac{M_{local}}{\epsilon_2} \tag{14}$$

and $\epsilon_l = \epsilon_1 \epsilon_2$.

- γ is a maximum distance on a compact space.
- β_g is a probability that local solution would be a global optimum under Assumption 1.
- M_{local} is a Lipschitz constant for local solution sets.

Main Results

Theorem 2. *Given $\delta_m \in [0, 1)$ and $\epsilon_m, \epsilon_2, \epsilon_3 > 0$, an instantaneous regret difference with N multi-started local optimizers at iteration t is less than ϵ_m with a probability at least $1 - \delta_m$:*

$$\mathbb{P}\left[\|r_{t,g} - r_{t,m}\|_2 < \epsilon_m\right] \geq 1 - \delta_m \tag{15}$$

where

$$\delta_m = \frac{\gamma}{\epsilon_3}\left(1 - \beta_g\right)^N + \frac{M_{local}}{\epsilon_2} \tag{16}$$

and $\epsilon_m = \epsilon_2 \epsilon_3$.

- γ is a maximum distance on a compact space.
- β_g is a probability that local solution would be a global optimum under Assumption 1.
- M_{local} is a Lipschitz constant for local solution sets.

Bayesian Optimization over Sets

Jungtaek Kim, Michael McCourt, Tackgeun You,
Saehoon Kim, and Seungjin Choi

https://arxiv.org/abs/1905.09780

Motivation

- Classic Bayesian optimization assumes that a search region $\mathcal{X} \subset \mathbb{R}^d$ is defined and that the function over $\mathbf{x} \in \mathcal{X}$ can only produce a scalar output.
- Unlike the classic Bayesian optimization, we propose a Bayesian optimization method which optimizes a function that can take sets as inputs:

$$y = f(\mathbf{X}) + \epsilon \tag{16}$$

  where $\mathbf{X} \in \{\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \mid \mathbf{x}_i \in \mathbb{R}^d\}$ for a fixed positive integer m.
- Finding optimal initial points for k-means clustering is a motivating example.

Proposed Method

- Denote that a set of m vectors is $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.
- If we define a set kernel

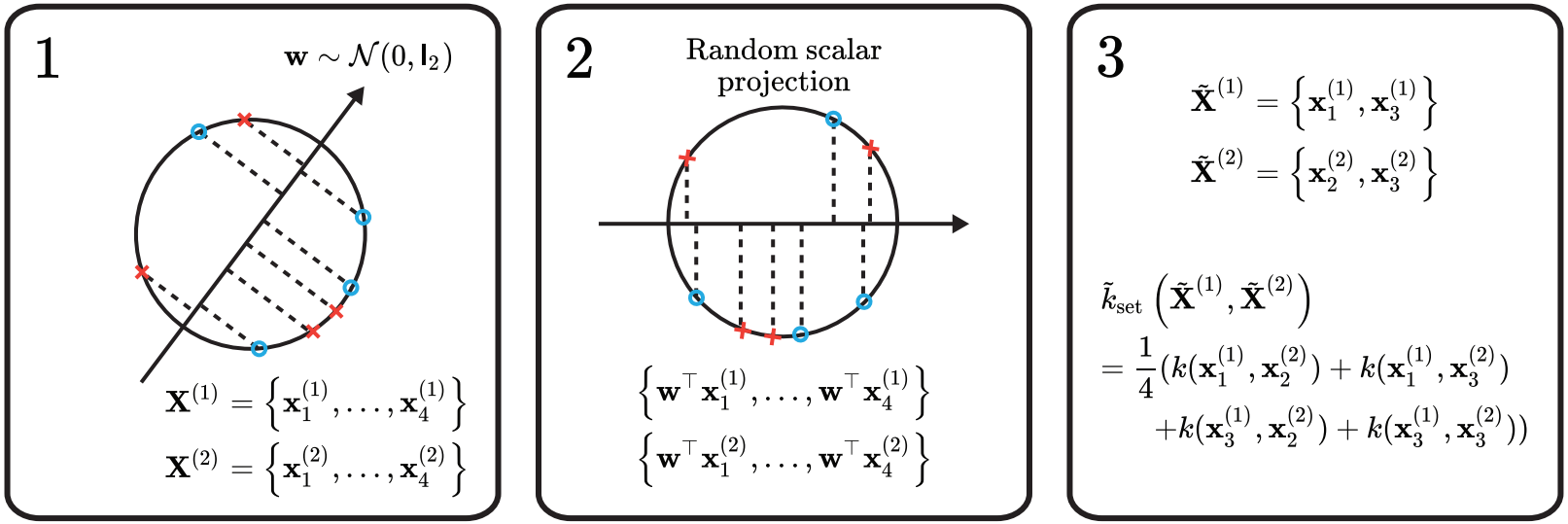
$$k_{\text{set}}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = \frac{1}{|\mathbf{X}^{(1)}||\mathbf{X}^{(2)}|} \sum_{i=1}^{|\mathbf{X}^{(1)}|} \sum_{j=1}^{|\mathbf{X}^{(2)}|} k(\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(2)}), \quad (17)$$

  a stochastic process can be directly employed.
- To reduce the computational complexity $\mathcal{O}(n^2 m^2 d)$, we propose an approximation of the set kernel:

$$\tilde{k}_{\text{set}}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}; \pi, \mathbf{w}, L) = k_{\text{set}}(\tilde{\mathbf{X}}^{(1)}, \tilde{\mathbf{X}}^{(2)}) \quad (18)$$

  where π is a permutation function, $\mathbf{w} \in \mathbb{R}^d$ is a vector for random scalar projection, and L is the number of subsamples.

Approximation of the Set Kernel



- Use a randomly generated vector $\mathbf{w}$ to impose an arbitrary ordering of the elements of all sets $\mathbf{X}^{(i)}$.
- Randomly permute the indices $[1, \ldots, m]$ via a function π.
- Sample L vectors from $\mathbf{X}^{(i)}$.

Approximation of the Set Kernel

Property 1. *The approximation satisfies pairwise symmetry:*

$$\tilde{k}_{set}\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}; \mathbf{w}, \pi, L\right) = \tilde{k}_{set}\left(\mathbf{X}^{(j)}, \mathbf{X}^{(i)}; \mathbf{w}, \pi, L\right). \tag{8}$$

Because $\tilde{\mathbf{X}}^{(i)}$ is uniquely defined given $\mathbf{w}, \pi, L$, this simplifies to $k_{set}(\tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{X}}^{(j)}) = k_{set}(\tilde{\mathbf{X}}^{(j)}, \tilde{\mathbf{X}}^{(i)})$, which is true because k_{set} is symmetric.

Property 2. *The "ordering" of the elements in the sets $\mathbf{X}^{(i)}, \mathbf{X}^{(j)}$ should not matter when computing $\tilde{k}_{set}$. Indeed, because (6) enforces ordering based on $\mathbf{w}$, and not whatever arbitrary indexing is imposed in defining the elements of the set, the kernel will be permutation invariant.*

Property 3. *The kernel approximation (5) reduces to computing k_{set} on a lower cardinality version of the data (with L elements selected from m). Because k_{set} is positive-definite on these L-element sets, we know that $\tilde{k}_{set}$ is also positive-definite.*

Property 4. *Since the approximation method aims to choose subsets of input sets, the computational cost becomes lower than the original formulation.*

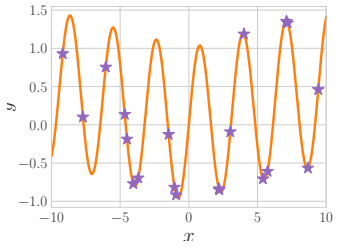
Approximation of the Set Kernel

Theorem 1. *Suppose that we are given two sets* $\mathbf{X}, \mathbf{Y} \in \mathcal{X}_{set}$ *and* $L \in \mathbb{Z}_+$. *Suppose, furthermore, that* $\mathbf{w}$ *and* π *can be generated randomly as defined in Section 3.1 to form subsets* $\tilde{\mathbf{X}}$ *and* $\tilde{\mathbf{Y}}$. *The value of* $\tilde{k}_{set}(\mathbf{X}, \mathbf{Y}; \mathbf{w}, \pi, L)$ *is an unbiased estimator of the value of* $k_{set}(\mathbf{X}, \mathbf{Y})$.

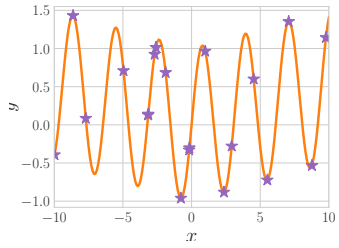
Theorem 2. *Suppose the same conditions as in Theorem 1. Suppose, furthermore, that* $k(\mathbf{x}, \mathbf{x}') \geq 0$ *for all* $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$. *The variance of* $\tilde{k}_{set}(\mathbf{X}, \mathbf{Y}; \mathbf{w}, \pi, L)$ *is bounded by a function of* m, L *and* $k_{set}(\mathbf{X}, \mathbf{Y})$*:*

$$Var\left[\tilde{k}_{set}(\mathbf{X}, \mathbf{Y}; \mathbf{w}, \pi, L)\right] \leq \left(\frac{m^4}{L^4} - 1\right) k_{set}(\mathbf{X}, \mathbf{Y})^2. \tag{10}$$

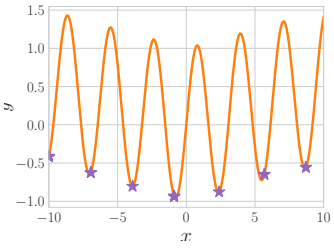
Experiments



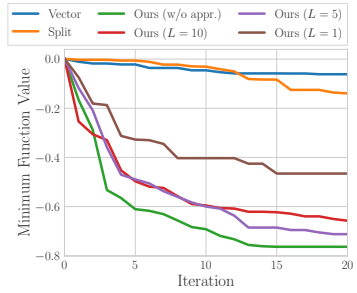
(a) Split



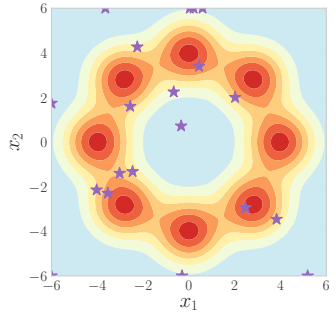
(b) Vector



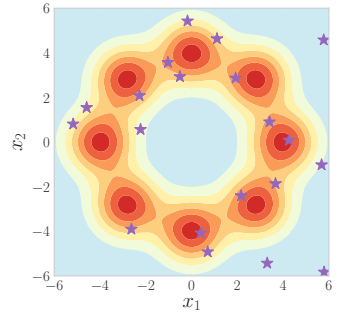
(c) Ours



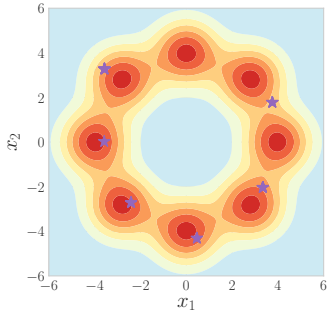
(d) Synthetic 1



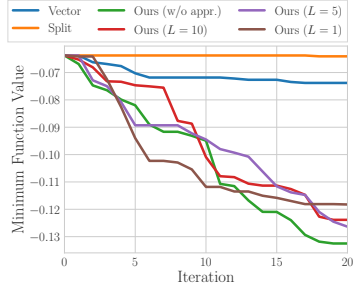
(e) Split



(f) Vector

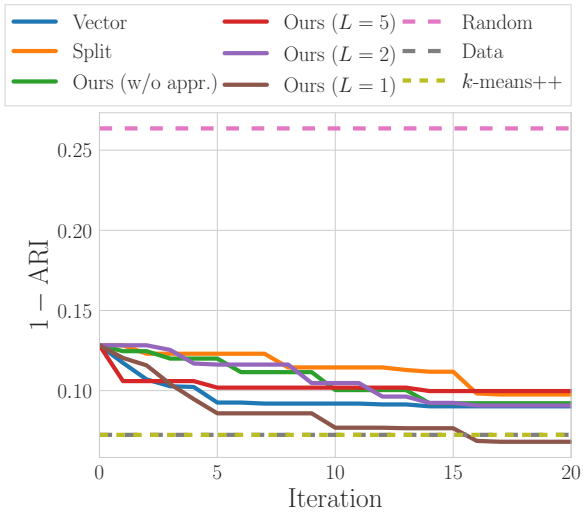


(g) Ours

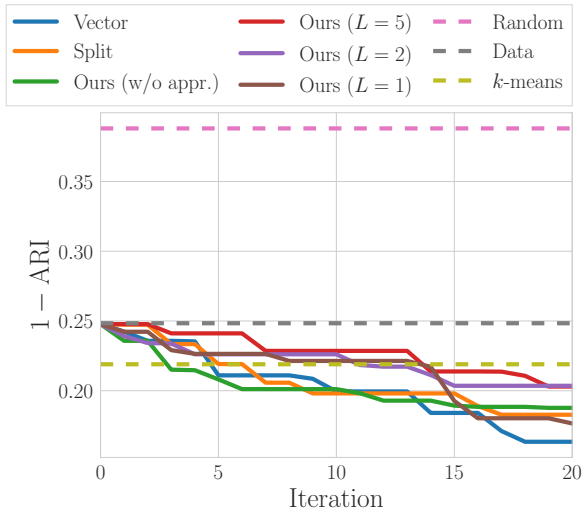


(h) Synthetic 2

Experiments



(i) k-means



(j) MoG

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.

J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 937–945, Beijing, China, 2014.

M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 250–259, Quebec City, Quebec, Canada, 2014.

J. González, Z. Dai, P. Hennig, and N. Lawrence. Batch Bayesian optimization via local penalization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 648–657, Cadiz, Spain, 2016.

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.

K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez. Bayesian optimization with exponential convergence. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 2809–2817, Montreal, Quebec, Canada, 2015.

J. Kim and S. Choi. bayeso: A Bayesian optimization framework in Python. `https://github.com/jungtaekkim/bayeso`, 2017.

J. Kim and S. Choi. Automated machine learning for soft voting in an ensemble of tree-based classifiers. In *International Conference on Machine Learning 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`.

References II

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 and S. Choi. On local optimizers of acquisition functions in Bayesian optimization. *arXiv e-prints*, arXiv:1901.08350, 2019.

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.

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.

V. Nguyen and M. A. Osborne. Knowing the what but not the where in Bayesian optimization. *arXiv e-prints*, arXiv:1905.02685, 2019.

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 (NeurIPS)*, 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.

Z. Wang, B. Shakibi, L. Jin, and N. de Freitas. Bayesian multi-scale optimistic optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1005–1014, Reykjavik, Iceland, 2014.

Z. Wang, C. Gehring, P. Kohli, and S. Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 745–754, Lanzarote, Spain, 2018.

J. T. Wilson, F. Hutter, and M. P. Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 9906–9917, Montreal, Quebec, Canada, 2018.