Team Optuna Developers' Method for Black-Box Optimization Challenge 2020

Masashi Shibata[1]*, **Toshihiko Yanase**[2*], **Hideaki Imamura**[2+], **Masahiro Nomura**[1+]
Takeru Ohta[2+], **Shotaro Sano**[2+], **Hiroyuki Vincent Yamazaki**[2+]
[1]CyberAgent, Inc.
[2]Preferred Networks, Inc.
[1]`{shibata_masashi,nomura_masahiro}@cyberagent.co.jp`
[2]`{yanase, mamu, tohta, sano, vincent}@preferred.jp`

Abstract

In this paper, we present our approach to the NeurIPS 2020 Black-Box Optimization Challenge. The proposed method is based on TuRBO, a batch Bayesian optimization algorithm which restricts the search space to certain trust regions. We made several improvements to overcome the following problems: (1) unknown smoothness of the objective function, (2) mixture of categorical and real-valued variables, and (3) small evaluation budgets available for surrogate model initialization. Our method won the 5th place in the challenge.

1 Introduction

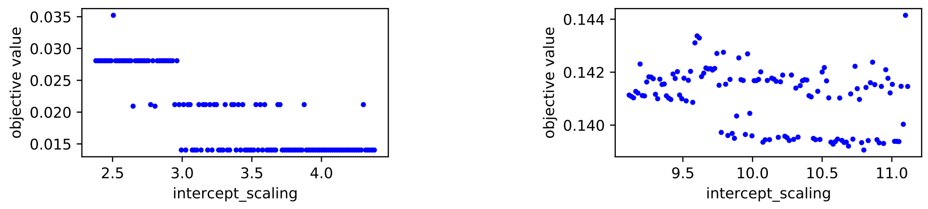
In this paper, we present our approach for the NeurIPS 2020 Black-Box Optimization (BBO) Challenge.[2] The goal of the BBO Challenge is to perform efficient hyperparameter optimization (HPO) [7] over a large number of different problems. The evaluation of this competition is conducted based on the *bayesmark* framework.[3] Nine types of ML models, six datasets, and four evaluation metrics are built into the framework. However, the BBO Challenge focuses on the development of optimization algorithms that improve the average performance for unknown, held-out problems of bayesmark that are not observable.

HPO is generally treated as BBO problems [2]. Objective functions in HPO are formulated as a black box functions, where gradients and smoothness parameters are unavailable. What makes these problems challenging is that the evaluation of these objective function oftentimes are costly [2]. Therefore, the problem amounts to finding a minimizer (or maximizer) of a black-box function with the minimum number of iterative function evaluations.

Without placing any assumptions on the objective functions, these problems are ill-posed [23]. In BBO, it is often assumed that the objective functions follow Gaussian processes (GPs) and impose a smoothness [18]. With these assumptions, an algorithmic framework for solving black-box optimization problems is called Bayesian optimization [17, 20]. Bayesian optimization shows attractive performance in various fields including HPO [22, 21, 15, 24, 8, 10, 27].

The BBO Challenge employs a *batch evaluation* setting, in which multiple points must be sampled simultaneously. Extending Bayesian optimization to such a parallel environment has been actively studied [26, 13, 28, 19, 5, 4, 12, 9, 11, 25, 6]. Among them, TuRBO, recently proposed by Eriksson et al. [6], is one of the most promising batch Bayesian optimization methods. TuRBO enables scalable

[*]The authors with * and + contributed equally and are ordered alphabetically
[2]`https://bbochallenge.com`
[3]`https://github.com/uber/bayesmark`



(a) Misclassification rate (b) Negative log-likelihood

Figure 1: Distribution of objective values around the best-found solutions. The model has two hyperparameters 'C' and 'intercept scaling' and we sweep the latter one. The vertical axis shows the misclassification rate (**Left**) and negative log-likelihood (**Right**), and the horizontal axis shows the parameter values.

global optimization by Thompson sampling within trust regions. While TuRBO was originally proposed for high-dimensional optimization problems, we confirmed with preliminary experiments that it can be used efficiently even for low-dimensional problems, such as the BBO Challenge.

In this work, we extend TuRBO for the problems in the BBO Challenge. Specifically, we focus on the following three characteristics of the BBO Challenge: (1) unknown smoothness of the objective function, (2) mixture of categorical variables and real-valued variables, (3) small evaluation budgets available for surrogate model initialization. For the first issue, (1) we combine multiple kernels when performing Thompson sampling, and (2) a restart strategy for the trust region update that is used when the optimization is stale. For the second issue, we (3) introduce an exhaustive search when the search space is finite and has a small cardinality, and (4) do not apply the trust region update for categorical and logit variables in the beginning of optimization. For the third issue, (5) we employ Quasi Monte Carlo sampling based on Sobol sequences instead of Latin hypercube sampling, which was used by the original TuRBO. Our proposed method combines these techniques with TuRBO. This resulted in a 5th place in the BBO Challenge in NeurIPS 2020. Our source code can be found at `https://github.com/optuna/bboc-optuna-developers`.

2 TuRBO

TuRBO is a powerful method for batch Bayesian optimization that uses Gaussian process (GP) models and Thompson sampling. In TuRBO, selecting the next solution to evaluate based on the *trust region* (TR) enables efficient scalable global optimization. TuRBO is a method that can maintain multiple (local) GP models at the same time, and TuRBO using m GP models is called TuRBO-m. Since our work is based on TuRBO-1, we will only describe TuRBO-1 and will refer to TuRBO-1 as TuRBO hereinafter for simplicity.

TuRBO can be described using the following steps: (1) Initialization using Latin hypercube sampling [16]. (2) Selection of solutions to evaluate next using Thompson sampling within a TR. (3) Update of GP models based on newly obtained data. Return to step (2).

As the solution to be evaluated next, TuRBO selects the minimum realization value of the GP posterior for a TR:

$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x} \in \mathrm{TR}} f_{\mathbf{x}}^{(i)}, \tag{1}$$

where $f_{\mathbf{x}}^{(i)}$ is the i-th realization of the GP posterior function for TR: $f_{\mathbf{x}}^{(i)} \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. A hyperrectangle is used as the TR, and the best solution so far is set at the center of the TR in the noise-free case. If the best of the observed values so far is updated several times consecutively, the length of TR is increased. On the other hand, if it fails several times consecutively, the length is reduced. The TR update is restarted when the length becomes smaller than a pre-speficied threshold.

3 Proposed Method

3.1 Dealing with Unknown Smoothness

Selection from Multiple Kernels. The approximation performance of the surrogate model clearly depends on the type of a kernel function used for the GP. Since it is difficult to select a kernel function suitable for the objective function before optimization, a pre-specified kernel is used in general. This means that if practitioners select the wrong kernel for the objective function, the performance of optimization degrades significantly.

To address the issue, we try to perform a more *optimistic* exploration. Specifically, the solution is selected using Thompson sampling with multiple kernels as follows:

$$\hat{\mathbf{x}} \in \mathop{\arg\min}_{\mathbf{x} \in \mathrm{TR}} \left\{ f_{\mathbf{x},k}^{(i)} \mid k \in \mathcal{K} \right\}, \tag{2}$$

where $f_{\mathbf{x},k}^{(i)}$ is the i-th realization of the GP posterior function with the kernel function k for TR: $f_{\mathbf{x},k}^{(i)} \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, $\mathcal{K}$ is a pre-defined set of kernels.

To solve (2) which minimizes the value of $f_{\mathbf{x},k}^{(i)}$ on TR for multiple kernels, we make the following implementation tweaks. First, we independently sample a set number of $\mathbf{x}$ for each kernel. For each kernel, we create a list of computed values of $f_{\mathbf{x},k}^{(i)}$. These lists, computed for all kernels, are concatenated to form a single list, in which we find the minimum value of $f_{\mathbf{x},k}^{(i)}$. Let $\mathbf{x}$, which achieves such a minimum, be an approximate solution to (2).

Stagnation Driven Trust Region Restart Strategy. In the optimization for nonsmooth objective functions as shown in Fig. 1 (a), the newly evaluated solution often has the same evaluation value as the previously evaluated solutions. In those cases, the local GP model may be trapped in a certain region, which leads to performance degradation. Therefore, in the proposed method, in addition to the restart condition of the original TuRBO, the TR update is restarted when all the solutions in the same batch have the same evaluation value.

3.2 Dealing with Categorical or Discrete Variables

Masking Length for the Trust Region. One important characteristic of the BBO Challenge is the existence of categorical variables, including Boolean variables, which are not handled by the original TuRBO. To address this situation, the proposed method does not apply the TR to categorical variables up to 10 iterations to avoid insufficient exploration at early search phases. That is, the length of the TR is set to the maximum value for the categorical variables. In addition, we apply the same treatment to logit variables (continuous variables with logit transformation). This is because logit variables should be searched at the edges with higher resolution (as shown in the Appendix A).

Exhaustive Search. We consider the situation where the search space is a finite set and the evaluation budget is larger than or equal to the cardinality of the set. In this case, if observations are be noise-free, optimal solution can be found with an exhaustive search. Thus, we do an exhaustive search if the cardinality of the search space is smaller than or equal to the evaluation budget.

3.3 Dealing with Small Evaluation Budgets

Initialization with Sobol Sequences. It is known that the design of initialization affects the performance in problems with small evaluation budgets [3]. Kucherenko et al. reported that Sobol sequences can provide better uniformity properties than Latin hypercube sampling [14]. Therefore, we employ Sobol sequences instead of Latin hypercube sampling, which is used by the original TuRBO.

4 Performance Evaluation

4.1 Experimental Settings

The experimental settings are as follows, as specified in the BBO Challenge:

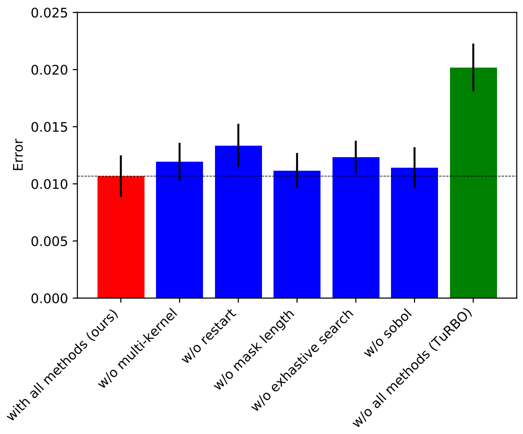


Figure 2: Result of the ablation study. The vertical axis represents the average of the normalized mean error with cross validation. The leftmost, rightmost, and middle bars correspond to our method, the original TuRBO-1, and our method without each technique described in Section 3, respectively.

- There are 16 iterations with a batch size of 8, i.e., $16 \times 8 = 128$ evaluations in total.
- The meta-data of problems such as kind of model and dataset are unavailable to the optimizer.

4.2 Ablation Study with the Bayesmark Framework

We evaluated the performance of our method using the default suite of the bayesmark framework. We ran the benchmark both with the original TuRBO and with the proposed method. We also conducted an ablation study where each technique described in the section 3 was dropped from our method one by one. Each task in the default suite was repeated 10 times, and all tasks were aggregated as the normalized mean score with the `bayesmark-agg` command. To estimate the mean performance accurately, we further repeated this process 30 times. That is, each task was evaluated 300 times in total. For this section and also during the competition, our performance evaluations were based on the cross validation (CV) score instead of the generalization score. More details on this choice are described in Appendix B.

The results are shown in Fig. 2. The rightmost bar corresponds to the result of the original TuRBO, and the leftmost bar corresponds to that of the proposed method. The difference of the values are relatively large compared with the error bar, and it shows that the proposed method clearly outperformed the original TuRBO in this experiment.

The other bars show the result of the ablation study. The performance was degraded by dropping each technique in our method, which means that each technique contributes to the performance improvements from the original TuRBO. Above all, the *restart* and *multi-kernel*, the countermeasures for the unknown smoothness of the objective function, show large impacts on the performance. This implies that escaping the local minima or flat points and handling a variety of smoothness are possible improvement areas of the existing TuRBO. The performance was markedly improved by *exhaustive search* as well. With a discrete and finite search space, TuRBO may suggest duplicated points, unless explicitly handled. While the exhaustive search is a trivial technique to avoid duplication, it is a simple and effective extension that is applicable to practical HPO as demonstrated in this experiment.

4.3 Evaluation in the Black-Box Optimization Challenge

Our method was evaluated in the black-box optimization challenge in NeurIPS 2020. During the submission period of the challenge, the submissions were evaluated on the held-out problems and the scores were publicly made available on the competition site leaderboard (public leaderboard). After the submission period, the evaluation ran again on different held-out problems from the public leaderboard, and its scores were announced as the final result of the competition. The participants could not access the held-out problems for the public nor the final leaderboards.

The public leaderboard score of our solution was 96.939, ranked in the 9th position at the best, while the public score was 95.491 for the original TuRBO. Note that the leaderboard scores were presented as $100 - 100 \times error$, where $error$ was the normalized mean error calculated by the `bayesmark-agg` command.

Our final leaderboard score was 91.806, ranked in the 5th position.

5 Conclusion

In this paper, we presented our approach to the NeurIPS 2020 Black-Box Optimization Challenge. We developed a method that improves upon TuRBO, to more efficient handle real problems, such as those in the BBO Challenge. The proposed method showed better performance than TuRBO on the bayesmark framework where various hyperparameter optimization problems are implemented, and the final result was 5th place on the final leaderboard of the competition. We plan to utilize the insight gained in this competition to improve Optuna [1] , which is a hyperparameter optimization library maintained by us.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631. ACM, 2019.

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[3] Jakob Bossek, Carola Doerr, and Pascal Kerschke. Initial design strategies and their effects on sequential model-based optimization: An exploratory case study based on bbob. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 778–786, New York, NY, USA, 2020. Association for Computing Machinery.

[4] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[5] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

[6] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 5496–5507, 2019.

[7] Matthias Feurer and Frank Hutter. Hyperparameter Optimization. In *Automated Machine Learning*, pages 3–33. 2019.

[8] Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pages 937–945, 2014.

[9] David Ginsbourger, Janis Janusevskis, and Rodolphe Le Riche. Dealing with asynchronicity in parallel gaussian process based global optimization. 2011.

[10] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

[11] Janis Janusevskis, Rodolphe Le Riche, David Ginsbourger, and Ramunas Girdziusas. Expected improvements for the asynchronous parallel global optimization of expensive functions: Potentials and challenges. In *International Conference on Learning and Intelligent Optimization*, pages 413–418. Springer, 2012.

[12] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

[13] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

[14] Sergei Kucherenko, Daniel Albrecht, and Andrea Saltelli. Exploring multi-dimensional spaces: A comparison of latin hypercube and quasi monte carlo sampling techniques. *arXiv preprint arXiv:1505.02350*, 2015.

[15] Nimalan Mahendran, Ziyu Wang, Firas Hamze, and Nando De Freitas. Adaptive mcmc with bayesian optimization. In *Artificial Intelligence and Statistics*, pages 751–760, 2012.

[16] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[17] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, pages 117–129, 1978.

[18] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pages 3330–3338, 2015.

[20] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *In Proceedings of the IEEE*, (1), 2015.

[21] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

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

[23] 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 27th International Conference on Machine Learning*, 2010.

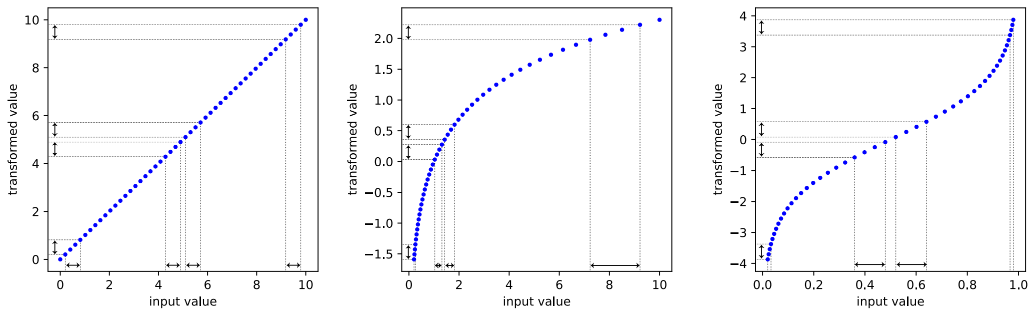
[24] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[25] Jialei Wang, Scott C Clark, Eric Liu, and Peter I Frazier. Parallel bayesian global optimization of expensive functions. *arXiv preprint arXiv:1602.05149*, 2016.

[26] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754, 2018.

[27] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pages 3627–3635, 2017.

[28] Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 3126–3134, 2016.



(a) Linear (b) Log (c) Logit

Figure 3: Warping parameters. The horizontal axis shows the input values and the vertical axis shows the transformed values.

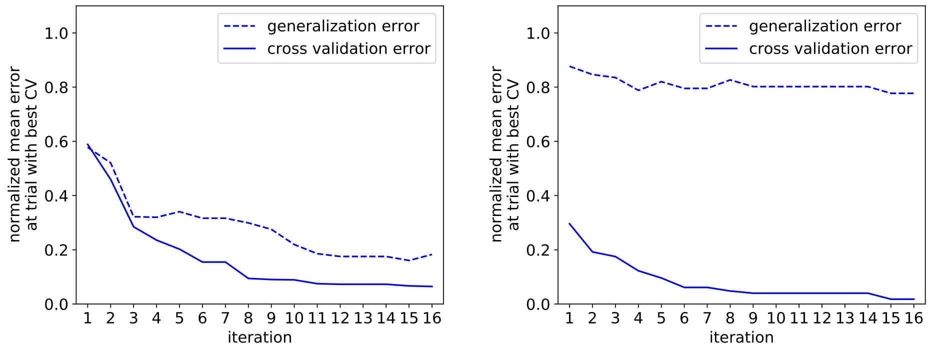


Figure 4: Experimental results of optimizing the *negative log-likelihood* (nll) metric for the *digits* dataset with the *AdaBoost* (**left**) and *kNN* (**right**) classifiers. The CV and generalization scores are consistent in the left figure, while the generalization score scarcely follows the CV score in the right figure.

A Input Warping in Bayesmark

Bayesmark provides annotations for parameters transformation the input space in order for the optimizers to take advantage of a priori knowledge. Fig. 3 describes linear, log and logit transformations. The horizontal axis shows the original parameter values and the vertical axis shows the transformed values. We plot 50 points evenly on the vertical axis.

The linear warp is an identity transformation as shown in Fig. 3 (a), and the transformed values also arrange evenly. Fig. 3 (b) shows the log warp. The smaller the original values are, the larger the interval between the transformed values will be. Since optimizers explore solutions in the transformed space, this warp enables them to explore the region around the lower edge with a higher resolution. In the logit warp shown in Fig. 3 (c), the intervals around both edges of the ranges are larger than the intervals around the center of the input space. This shows that the logit warp expands the regions near both edges while the log warp expands only the region near lower edge.

B Analysis on the Gap Between CV and Generalization Scores

One of the challenges in the evaluation based on the bayesmark framework is that generalization scores are often extremely noisy. For example, Figure 4 (right) shows the transition of the CV and generalization scores when optimizing the *kNN* model running on the *digits* dataset for the metric of *nll*. The generalization score scarcely improves despite that fact that the CV error decreases along the number of iteration.

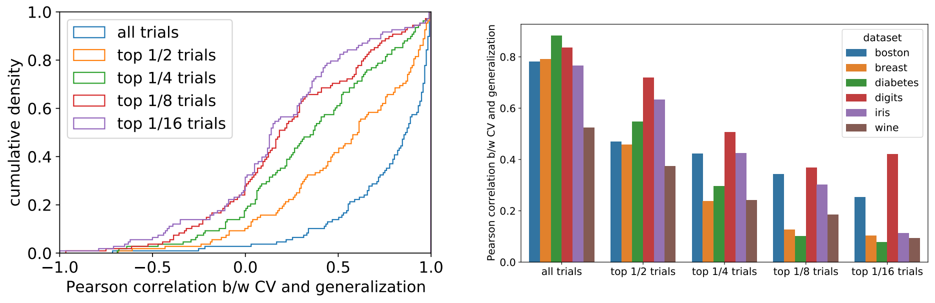


Figure 5: Statistics of Pearson correlation between the CV and generalization scores. **Left:** cumulative distribution of number of trials as a function of correlation, where each line represents the subset of the all trials. While the scores are well correlated with each other with all trials, they are relatively inconsistent only with well-performing trials. **Right:** average Pearson correlation for each dataset and subset of trials. Decreasing trend differs among datasets, where *boston* and *digits* show relatively high correlation even with top-performing trials.

Such inconsistency stands out especially with well-optimized parameters. Figure 5 (left) shows the cumulative distribution of the number of trials in a single run of the default suite, where the horizontal axis corresponds to the Pearson correlation between the CV and generalization errors. The parameters are suggested by random sampling and other experimental settings follow the evaluation in Section 4. Each line shows the statistics among *all trials* and those only with *top* $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, *and* $\frac{1}{16}$ *trials*. With all trials, the distribution tends toward the Pearson correlation of 1.0, while it's not the case only with top performing trials. This implies that: (1) during the initial stage of the optimization, the CV and generalization scores tend to be consistent; and (2) they, on the other hand, turn into poorly correlated scores with well-optimized parameters.

The decreasing trend of the correlation depends on the dataset's characteristics. Figure 5 (right) shows the average Pearson correlation along each dataset in the default suite and subset of trials. Among the datasets, *boston* and *digits* show relatively high correlation with top-performing trials.

The size of the dataset is one of possible causes of the inconsistency, where the number of rows is less than 600 except for the *digits* dataset. In those cases, the evaluation may be sensitive to the randomness in the experimental process. E.g., the randomness in data split can result in difference in the datasets' distribution between the CV and the test data. Note that those gaps are not necessarily the case with the competition leaderboards because the leaderboard problems are totally inaccessible and the datasets' sizes might be larger.

Because large noises exist with the default suite and no other prior information is available about the competition datasets, we omitted the generalization scores from our evaluation process during the competition. This is to isolate the performance evaluation from the noises, under the assumption that the improvement in a closed experiment results in that of an open setting.