A quantile-based approach for hyperparameter transfer learning

David Salinas, Huibin Shen, Valerio Perrone
Amazon
Berlin, Germany
`david.salinas.pro@gmail.com, {huibishe, vperrone}@amazon.com`

Abstract

Bayesian optimization (BO) is a popular methodology to tune the hyperparameters of expensive black-box functions. Despite its success, standard BO focuses on a single task at a time and is not designed to leverage information from related functions, such as the performance metric of the same algorithm tuned across multiple datasets. In this work, we introduce a novel approach to achieve transfer learning across different *datasets* as well as different *metrics*. The main idea is to reparametrize raw metrics as quantiles via the probability integral transform, and learn a mapping from hyperparameters to metric quantiles. We introduce two methods to leverage this estimation: a pure random search biased toward sampling lower quantiles, and a Gaussian process using such quantile estimate as a prior. We show that these strategies can combine the estimation of multiple metrics such as runtime and accuracy, steering the optimization toward cheaper hyperparameters for the same level of accuracy. Experiments on an extensive set of hyperparameter tuning tasks demonstrate significant improvements over several baselines.

1 Introduction

Tuning complex machine learning models such as deep neural networks can be a daunting task. Object detection or language understanding models often rely on deep neural networks with many tunable hyperparameters, and automatic hyperparameter optimization (HPO) techniques such as Bayesian optimization (BO) are critical to extract the best accuracy and save time. BO addresses the black-box optimization problem by placing a probabilistic model on the function to minimize, such as the mapping of neural network hyperparameters to a validation loss, and determine which hyperparameters to evaluate next by trading off exploration and exploitation. While traditional BO focuses on each problem in isolation, recent years have seen a surge of interest in *transfer learning* for HPO. In this context, transfer learning aims to leverage evaluations from previous, related *tasks* (e.g., the same neural network tuned on multiple datasets) to further speed up the hyperparameter search. Given evaluations from previous tasks $\{(x_i^l, y_i^l)\}_{i=1}^{n_l}$, where $y_i^l \in \mathbb{R}$ is the i-th evaluation of hyperparameter $x_i^l \in \mathbb{R}^d$ on black-box f^l, one aims to solve $\operatorname{argmin}_{x\in\mathbb{R}^d} f(x)$ for a new task f.

A variety of methods have been developed to this end. The most common approach is to model tasks jointly or via a conditional independence structure, which has been been explored through multi-output GPs [24], weighted combination of GPs [20, 7], and neural networks, either fully Bayesian [23] or hybrid [22, 17, 13]. A different line of research has focused on the setting where tasks come over time as a sequence and models need to be updated online as new problems accrue. A way to achieve this is to fit a sequence of surrogate models to the residuals relative to predictions of the previously fitted model [9, 18]. Specifically, the GP over the new task is centered on the predictive mean of the previously learned GP. Finally, rather than fitting a surrogate model to all past data, some transfer can be achieved by warm-starting BO with the solutions to the previous BO problems [8, 26]. A central challenge of all these approaches is that different black-boxes typically have different scales, varying noise levels, and possibly contain outliers, making it hard to learn a joint model. The Gaussian Copula Process (GCP) [25] can be used to alleviate scale issues on a single task at the extra cost of estimating the CDF of the data. Using GCP for HPO was proposed in [2] to handle potentially non-Gaussian data, albeit only for the single-task case.

In this work, we show how the probability integral transform can be used to map evaluations to the quantile space, where fitting a joint model becomes easier as scale issues vanish. Using this map we fit a global quantile parametric model, and then show how such estimation can be used to transfer information to new tasks. To this end, we propose two strategies: a quantile random search and a quantile-based Gaussian Process. We show that these strategies can jointly model several metrics with potentially different scales, such as validation error and compute time, without requiring processing. Extensive experiments demonstrate that significant speed-ups can be achieved.

2 Quantile based HPO

Quantile estimation We start by introducing quantile estimation, the building block of the HPO strategies we propose. Let $\bar{F}^l$ denote the empirical cumulative distribution function on the set of evaluations of task $l : \bar{F}^l(t) = \frac{1}{n_l}\sum_i \mathbb{1}_{y_i^l \leq t}$. This function maps each evaluation to a quantile value $u^l = \bar{F}^l(y^l) \in [0,1]$. We use the available evaluations $\{x_i^l, u_i^l\}_{i,l}$ to fit a parametric quantile estimate $u_w(x)$. The quantile estimate is regressed with a multi-layer-perceptron (MLP) whose parameters w are obtained by minimizing the $\ell 2$-error $\sum_{i,l}(u_w(x_i^l) - u_i^l)^2$. It is important to note that the quantile estimate $u_w(x)$ is made independent of the dataset. Given a new task, we hope that $u_w(x)$ is a good approximation of $F(y(x))$ where F is the CDF of y. It should be clear that if $u_w(x) = F(y(x))$, then the problem of finding the minimum of f would be solved as a parameter x minimizing $f(x)$ also minimizes the quantile $F(y(x))$. However, the estimation of $u_w(x)$ will not be perfect and will yield some approximation error. For instance, a naive estimate always predicting the median, e.g. $u(x) = 0.5$, would have a mean absolute error (MAE) of 0.25[1]. For similar tasks, we would expect the MAE to be lower than 0.25 for effective transfer. We next show how this improvement can be leveraged to design two new HPO strategies.

Quantile random search Given a quantile estimation for the new task $u_w(x)$, we would now like to design a HPO strategy. In particular, we want to design a random strategy with an exploration-exploitation trade-off. Assume we are given N hyperparameter configurations $x_1, \ldots, x_N$ to sample from. We can predict quantiles with $u_w(x)$ and pick the hyperparameter configurations with the minimum quantile score, but there would be no exploration. Instead, an exploration-exploitation trade-off can be obtained by setting a probability density on the quantile domain $[0,1]$ so that lower quantiles are sampled with larger probability. For this, we use a beta-distribution with $\alpha = 1$ which yields the following pdf: $p_\beta(u) = \beta(1-u)^{\beta-1}$. The parameter $\beta \geq 1$ controls the exploration-exploitation trade-off. Note that the density should decrease towards 1 to ensure that lower quantiles are favored and larger values of β corresponds to more exploitation as the probability of picking smaller quantiles increases, which can be seen in Fig 1 in the appendix. We call such a method quantile random-search (quantile-RS) and present pseudo-code in Algorithm 1. The steps inside the loop perform inverse transform sampling [5] from the density defined on quantiles with the Beta distribution. Indeed, one has $F_\beta(u) = \int_0^u p_\beta(t)\, dt = 1 - (1-u)^\beta$, and consequently, $F_\beta^{-1}(q) = 1 - (1-q)^{1/\beta}$ for $q \in [0,1]$.

Algorithm 1: Quantile random search (quantile-RS)

```
Estimate $u_w(x)$ on hold-out evaluations of other tasks and a given $\beta$.
while Has budget do
    Sample $N$ hyperparameters $x_1, \ldots, x_N$ and sort them according to $u_w(x_i)$.
    Sample $q \sim \mathcal{U}([0,1])$.
    Return $x_{\lfloor F_\beta^{-1}(q) \times N \rfloor} = x_{\lfloor (1-(1-q)^{1/\beta}) \times N \rfloor}$.
end while
```

[1]The average mean absolute error of a median predictor is given by $\int_0^1 |q - \frac{1}{2}| dq = 2\int_0^{\frac{1}{2}} |q| dq = \frac{1}{4}$.

Quantile-based Gaussian Process While the quantile random search approach can re-use information from previous tasks, it does not exploit the evaluations from the current task as each draw is independent of the observed evaluations. This can become an issue if the new black-box significantly differs from previous tasks. We now show that the quantile approach can be combined with a GP to both learn from previous tasks while adapting to the current task.

Instead of modeling observations directly as a GP, we model them as a Gaussian Copula Process (GCP) [25]. Consider the map $\psi = \Phi^{-1} \circ F$ where Φ is the standard normal CDF, recalling that F is the CDF of y. Then we can define a new random variable z and model the function $z = \psi(y) \sim \text{GP}(\mu(x), k(x))$. As ψ is monotonically increasing and mapping into the line, we can alternatively view this modeling as a warped GP [21] with a non-parametric warping. One advantage of this transformation is that z follows a Normal distribution, which may not be the case for $y = f(x)$. In the specific case of HPO, y may represent accuracy scores in $[0, 1]$ of a classifier where a Gaussian cannot be used. Another advantage is that we can transfer the information from other tasks extracted in the parametric quantile estimate u_w by using the prior mean function $m(x) = \Phi^{-1}(u_w(x))$. Using the probability-integral-transform is key here as it maps evaluations of each task to a standard Normal distribution, meaning that evaluations of different tasks are mapped to a similar space. Finally, we fit the GP using L-BFGS for marginal likelihood optimization, and use the Expected Improvement [16] to sample new hyperparameters. The pseudo-code is given in Algorithm 2.

Algorithm 2: Quantile-based GP (quantile-GP)

```
Estimate $u_w(x)$ on the available evaluations from related tasks.
Sample $x$ via quantile random search as described in the previous section.
while Has budget do
    Deduct the prior means by $\psi(x) - m(x)$ and then fit the GP.
    Return the parameter $x$ maximizing the Expected Improvement in the space of $\psi(x) + m(x)$.
    Evaluate $f(x)$ and collect the result.
end while
```

Let us now make a few remarks on ψ. When no observations are available, F is not defined. Therefore, we warm-start the optimization on the new task by sampling a hyperparameter configuration based on quantile random-search, as described above. Finally, as ψ will be infinite when applied to the minimum or maximum of y, we apply the following truncated estimator for the CDF from [14]:

$$\tilde{F}(t) = \begin{cases} \delta_m & \text{if } \bar{F}(t) < \delta_m \\ \bar{F}(t) & \text{if } \delta_m \leq \bar{F}(t) \leq 1 - \delta_m \\ 1 - \delta_m & \text{if } \bar{F}(t) > 1 - \delta_m \end{cases}$$

where m is the number of observations and $\delta_m = \frac{1}{4m^{1/4}\sqrt{\pi \log m}}$ strikes a bias-variance trade-off.

Optimizing multiple metrics In addition to optimizing the accuracy of a black-box function, it is often desirable to optimize its runtime or memory consumption. Indeed, given two hyperparameters with the same expected error, the one requiring less resources is preferable. For tasks where runtime is available, we use both runtime and error metrics by averaging their quantiles, e.g., we set $u(x) = \frac{1}{2}(u^{\text{error}}(x) + u^{\text{time}}(x))$, where $u^{\text{error}}(x)$ represents the error quantile and $u^{\text{time}}(x)$ represents the time quantile. This allows us to seamlessly optimize for time and error when running HPO, so that the cheaper hyperparameter is favored when two hyperparameters lead to similar expected error.

3 Experiments

We considered the problem of tuning three algorithms: `XGBoost` [4], a 2-layer feed-forward neural network (`FCNET`) [11], and the RNN-based time series prediction model proposed in [19] (`DeepAR`). To speed up experiments we used a variation of the lookup table approach advocated in [6], reducing the BO problem to selecting the next hyperparameter configurations from a fixed set which has been evaluated in advance (as a result, no extrapolation error is introduced). These hyperparameter evaluations were obtained by querying each algorithm at hyperparameters sampled uniformly at random

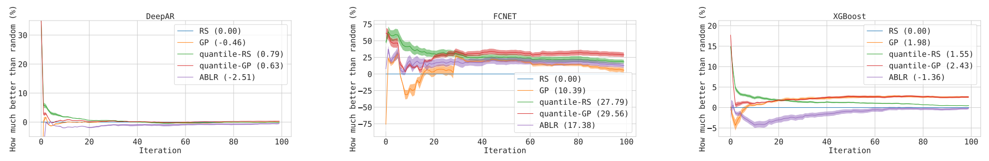

Figure 1: Relative improvement over random search across iterations.

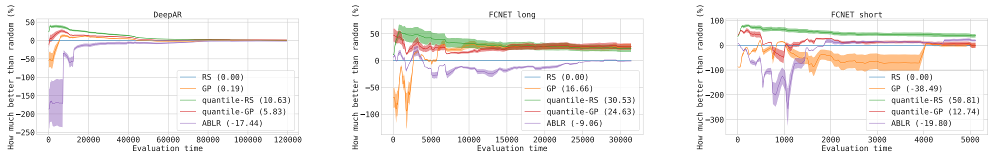

Figure 2: Relative improvement over random search against run time. For FCNET we separate `naval propulsion` and `parkinsons telemonitoring` from `slice localization` and `protein structure` as the run time for the first two datasets was considerably lower.

from their search space. We tuned `XGBoost` on 10 `libsvm` datasets[2] [3] to minimize $1-$AUC. Then, we tuned `FCNET` on 4 datasets from [11], namely {`slice localization, protein structure, naval propulsion, parkinsons telemonitoring`}, to minimize the test mean squared error. As for `DeepAR`, the evaluations were collected on the data provided by GluonTS [1], consisting of 6 datasets from the M4-competition [15] and 5 datasets used in [12], and the goal is to minimize the quantile loss [10]. Additionally, for `DeepAR` and `FCNET` the runtime to evaluate each hyperparameter configuration was available, and we ran additional experiments exploiting this metric. More details on the HPO problems and search spaces are in Table 1 and Table 2 of the appendix.

We assessed the transfer learning capabilities in a leave-one-task-out setting: we sequentially left out one dataset and then aggregated the results for each algorithm. The performance of each method was first averaged over 30 replicates for one dataset in a task, and the relative improvements over random search computed on every iteration for that dataset. The relative improvement for an optimizer (`opt`) is defined by $(y_{random} - y_{opt})/y_{random}$, which is upper bounded by 100% as we shift y to ensure $y \in \mathbb{R}^+$. By computing the relative improvements, we are able to aggregate results across all datasets for each algorithm. Each curve and shaded area in the plots respectively corresponds to the mean improvement over random search and one standard error across datasets. Finally, for all quantile-based methods we set $\beta = 2$, and learn the mapping to quantiles via a 3-layer MLP with 50 units per layer, optimized by ADAM with early-stopping.

Results We benchmarked our methods against random search, standard GP-based BO, and ABLR [17], a recently proposed approach to transfer learning consisting of a shared neural network across tasks on top of which lies a Bayesian linear regression layer for each task. Figure 1 shows that the quantile-based approaches consistently speed up convergence to a good optimum. Specifically, quantile random search tends to outperform competitors in the beginning of the BO, while quantile GP catches up as it adapts to the current task with an increasing number of observations, eventually reaching a better optimum. Detailed results for each dataset are in Table 3 of the appendix.

We then studied the ability of the quantile-based approaches to transfer information from multiple metrics. Figure 2 reports results against run time, leveraging both error and time metrics for quantile-based BO. Here, performance for each optimizer was first averaged over 30 replicates within each dataset and then further aggregated across datasets. As tunings for different datasets and optimizers require different time for a fixed number of iterations, to aggregate results we average only up to the time taken by the fastest dataset and optimizer. Table 4 in the appendix includes the full results for each dataset. Quantile random search and quantile GP outperform competing methods by a large margin, successfully tapping into multiple metrics to accelerate the hyperparameter search.

[2] {`a6a, australian, german.numer, heart, ijcnn1, madelon, skin nonskin, spambase, svmguide1, w6a` }.

References

[1] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic Time Series Modeling in Python. *arXiv preprint arXiv:1906.05264*, 2019.

[2] Alec Anderson, Sebastien Dubois, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Sample, estimate, tune: Scaling bayesian auto-tuning of data science pipelines. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 361–372. IEEE, 2017.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[4] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. 2016.

[5] Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

[6] K Eggensperger, F Hutter, HH Hoos, and K Leyton-brown. Efficient benchmarking of hyperparameter optimizers via surrogates background: hyperparameter optimization. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 1114–1120, 2012.

[7] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for Bayesian optimization using ranking-weighted Gaussian process ensembles. In *ICML 2018 AutoML Workshop*, July 2018.

[8] Matthias Feurer, T Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[9] 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.

[10] Tim Januschowski, David Arpin, David Salinas, Valentin Flunkert, Jan Gasthaus, Lorenzo Stella, and Paul Vazquez. Now available in amazon sagemaker: Deepar algorithm for more accurate time series forecasting. *https://aws.amazon.com/blogs/machine-learning/now-available-in-amazon-sagemaker-deepar-algorithm-for-more-accurate-time-series-forecasting/*, 2018.

[11] Aaron Klein and Frank Hutter. Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv preprint arXiv:1905.04970*, 2019.

[12] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015, 2017.

[13] Ho Chung Leon Law, Peilin Zhao, Junzhou Huang, and Dino Sejdinovic. Hyperparameter learning via distributional transfer. Technical report, preprint arXiv:1810.06305, 2018.

[14] Han Liu, John Lafferty, and Larry Wasserman. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. 10:2295–2328, 2009.

[15] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[16] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2, 1978.

[17] Valerio Perrone, Rodolphe Jenatton, Matthias Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[18] Matthias Poloczek, Jialei Wang, and Peter I Frazier. Warm starting Bayesian optimization. In *Winter Simulation Conference (WSC), 2016*, pages 770–781. IEEE, 2016.

[19] David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017.

[20] Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. Scalable hyperparameter optimization with products of Gaussian process experts. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 33–48. Springer, 2016.

[21] Edward Snelson, Zoubin Ghahramani, and Carl E. Rasmussen. Warped gaussian processes. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 337–344. MIT Press, 2004.

[22] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2171–2180, 2015.

[23] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4134–4142, 2016.

[24] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2004–2012, 2013.

[25] Andrew Gordon Wilson and Zoubin Ghahramani. Copula processes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, pages 2460–2468, USA, 2010. Curran Associates Inc.

[26] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.