Hyperparameter Learning via Distributional Transfer

Ho Chung Leon Law University of Oxford
Peilin Zhao Tencent AI Lab
Junzhou Huang Tencent AI Lab
Dino Sejdinovic University of Oxford

Abstract

Bayesian optimisation is a popular technique for hyperparameter learning but typically requires initial 'exploration' even in cases where potentially similar prior tasks have been solved. We propose to transfer information across tasks using kernel embeddings of distributions of training datasets used in those tasks. The resulting method has a faster convergence compared to existing baselines, in some cases requiring only a few evaluations of the target objective.

1 Introduction

Hyperparameter selection is an essential part of training a machine learning model and a judicious choice of values of hyperparameters such as regularisation or kernel parameters is what often makes a difference between an effective and a useless model. To tackle the challenge in a more principled way, the machine learning community has been increasingly focusing on Bayesian optimisation (BO) [13], a sequential strategy to select hyperparameters θ based on previous past evaluations of model performance. Using a Gaussian process (GP) [12] prior to build a representation of the underlying accuracy f as a function of the hyperparameters, and using an acquisition function $\alpha(\theta; f)$ to trade off exploration and exploitation, given the posterior of f, has been shown to give superior performance compared to traditional methods [13] such as grid search or random search. However, even in this setup, BO still suffers from the so called 'cold start' problem [11, 15]. Namely, in order to begin fitting a GP model, one needs to have initial evaluations of f at a range of hyperparameter choices. Hence, prior research considered transferring knowledge about optimal parameters from previously 'solved' tasks, as in [15, 3, 14, 11]. However, to consider the similarity across tasks, initial random evaluations of the model at hand are still required. This might be prohibitive: evaluations of f can be computationally costly and our goal may be to select hyperparameters and deploy our model as soon as possible. We note that treating f as a black-box function, as is often the case in BO, is ignoring the highly structured nature of hyperparameter learning – it corresponds to training specific models on specific datasets. We make steps towards utilizing such structure in order to borrow strength across different tasks and datasets.

Contribution We assume a scenario where a number of tasks have already been 'solved' and propose a new BO algorithm, making use of the mean embeddings of the joint distribution of the training data [10, 2]. In particular, we propose a GP model that can jointly model all tasks at once, by considering extended domain of inputs to model accuracy f: joint distribution of the training data $\mathcal{P}_{XY}$, sample size of the training data s and hyperparameters θ. Through utilising all seen evaluations from all tasks and meta-information, we can optimise the marginal likelihood to *learn* a meaningful similarity between tasks. Experimentally, our methodology performs favourably already at initialisation and has a faster convergence compared to existing baselines.

Related work The idea of transferring information from different tasks in the context of BO is not new, and it has mainly been studied in the setting of multi-task BO [15, 3, 14, 11]. In multi-task BO, there is a set of tasks that we wish to solve jointly or we wish to solve a target task given some 'solved' source task. We follow the same setup, but take a different approach. Currently, correlation across tasks is only captured through evaluations of f. However, in terms of

2nd Workshop on Meta-Learning at NeurIPS 2018, Montréal, Canada.

hyperparameter search for a machine learning model, this ignores additional information available: datasets used in training. Further, since task similarity is captured through evaluations of f only, this implies that we need to observe *sufficient evaluations from the target task* first in order to learn these task correlations. However, this is unnecessary in our proposed methodology, which can yield good initial hyperparameter candidates without having seen any evaluations from our target task, since we draw information from the meta-features corresponding to the training data distribution.

The use of such meta-information has in fact been explored before, but the current literature either uses hand-crafted manual features [4] or defines them in an unsupervised way [6]. These strategies are not optimal, as while different tasks can have very different meta-information, their fs can in fact be highly correlated. In this case, using such meta-information (to define similarity) can have an adverse effect on exploration – highlighting the importance of using evaluations of f to *learn* the similarity between two tasks. Furthermore, having obtained a similarity across tasks, current literature suggests to initialise with the best θ from the solved tasks, but this neglects the non-optimal θs that can also provide useful information about the target task. Our methodology can be seen as a combination of these two frameworks, as we use *learnt* embeddings of the joint distribution of the training data, while implicitly capturing correlation across tasks. The most similar in spirit to ours is the work of [7], who consider an additional input to be the sample size s, but do not consider different tasks corresponding to different training data distributions.

2 Background

Let f^{target} be the target task objective we would like to optimise, i.e. we want to find $\theta^*_{target} = \text{argmax}_{\theta\in\Theta} f^{target}(\theta)$. Assume that there are n (potentially) related source tasks $f^i,\ i = 1,\dots n$. For each source task, we assume that we have $\{\theta^i_k, t^i_k\}_{k=1}^{N_i}$ from past runs, where t^i_k denotes a noisy evaluation of $f^i(\theta^i_k)$ and N_i denotes the number of evaluations of f^i from task i. Here, we assume that $f^i\ \forall\ i$ is the (underlying) accuracy of a trained machine learning model with training data $D_i = \{\mathbf{x}^i_l, y^i_l\}_{l=1}^{s_i}$, where $\mathbf{x}^i_l \in \mathbb{R}^p$ are the covariates, y^i_l are the labels and s_i is the size of the data. Under this setting, we have $(f^i, D_i = \{\mathbf{x}^i_l, y^i_l\}_{l=1}^{s_i}, \{\theta^i_k, t^i_k\}_{k=1}^{N_i})$ for $i = 1,\dots n$, where D_i denotes a dataset of source task i. Our strategy now is to measure the similarity between datasets (as a representation of the task itself), in order to find θ^*_{target}.

Assumptions To compare between different datasets, we make the assumption that $\mathbf{x}^i_l \in \mathcal{X}$ and $y^i_l \in \mathcal{Y}$ for all i, l, and that throughout the supervised learning model class M is the same. For example, one could imagine a data stream setting, where models have to be constantly retrained. This assumption implies that the source of differences of f^i across i and f^{target} is in the data D_i and D_{target} *only*, i.e. it is the dataset that affects the location of θ^*_i. In particular, we assume $D_i = \{\mathbf{x}^i_l, y^i_l\}_{l=1}^{s_i} \sim \mathcal{P}^i_{XY}$, where $\mathcal{P}^i_{XY}$ is the underlying joint distribution of the data for source task i. Further, as sample size is closely related to model complexity choice which is in turn closely related to hyperparameter choice [7], we will also encode this information.

With these assumptions, we consider $f(\theta, \mathcal{P}_{XY}, s)$, where f is a function on hyperparameters θ, joint distribution of the underlying data $\mathcal{P}_{XY}$ and sample size s. Here, f could be the negative of the empirical risk, i.e. $f(\theta, \mathcal{P}_{XY}, s) = -\frac{1}{s}\sum_{l=1}^{s} L(h_\theta(\mathbf{x}_l), y_l))$, where L is the loss function and h_θ is the model. In this form, we can also recover f^i and f^{target} from $f^i(\theta) = f(\theta, \mathcal{P}^i_{XY}, s_i)$ and $f^{target}(\theta) = f(\theta, \mathcal{P}^{target}_{XY}, s_{target})$. Now by constructing an appropriate covariance function for these inputs, we can use a GP to model f. Intuitively, similarly to assuming that f varies smoothly as a function of θ in standard BO, this model also assumes smoothness of f across $\mathcal{P}_{XY}$ as well as across s following [7]. Intuitively, if two distributions and sample sizes are similar (in some distance that we will learn), their corresponding f will also be similar. In this source and target task setup, this would mean we can utilise information from *all* previous source datasets evaluations $\{\theta^i_k, t^i_k\}_{k=1}^{N_i}$.

3 Methodology

In order to model the quantities $\theta, \mathcal{P}_{XY}$ and s in a GP, we need a corresponding covariance function C on these quantities. Assuming a separable kernel for the inputs, we have:

$$C(\{\theta_1, \mathcal{P}^1_{XY}, s_1\}, \{\theta_2, \mathcal{P}^2_{XY}, s_2\}) = \nu k_\theta(\theta_1, \theta_2) k_p(\psi(D_1), \psi(D_2)) k_s(s_1, s_2) \tag{1}$$

where ν is a constant, k_θ and k_p is the standard Matérn-$3/2$ kernel in BO and k_s is the sample size kernel found in [7]. These are standard choices in the literature to facilitate fair comparisons, hence we do not investigate them and instead focus on modelling the dataset representation $\psi(D)$.

To specify $\psi(D)$, a feature map on joint distributions, estimated through samples D, we will follow an approach similar to [2] who consider transfer learning, and make use of kernel mean embeddings in order to compute feature maps of distributions (cf. [10] for an overview). In particular, we will consider feature maps of covariates and labels seperately, denoting them by $\phi_1(\mathbf{x}) \in \mathbb{R}^p$ and $\phi_2(y) \in \mathbb{R}^q$. Given these two feature maps, to meaningfully embed a joint distribution $\mathcal{P}^i_{XY}$, we use the cross covariance operator $\mathcal{C}^i_{XY}$ (see [5] for review), estimated by D_i with: $\hat{\mathcal{C}}^i_{XY} = \frac{1}{s_i}\sum_{\ell=1}^{s_i} \phi_1(\mathbf{x}^i_\ell) \otimes \phi_2(y^i_\ell)$, i.e. in the case of finite-dimensional features, the cross covariance operator is just a tensor product of the feature maps $\Phi^i_1(\mathbf{x}) = [\phi_1(\mathbf{x}^i_1), \ldots, \phi_1(\mathbf{x}^i_{s_i})] \in \mathbb{R}^{p\times s_i}$, $\Phi^i_2(y) = [\phi_2(y^i_1), \ldots, \phi_2(y^i_{s_i})] \in \mathbb{R}^{q\times s_i}$, $\hat{\mathcal{C}}^i_{XY} = \frac{1}{s_i}\Phi^i_1(\mathbf{x})\Phi^i_2(y)^\top \in \mathbb{R}^{p\times q}$. Given $\hat{\mathcal{C}}^i_{XY}$, we can flatten it to obtain $\psi(D_i) \in \mathbb{R}^{pq}$, an estimator of the representation of the joint distribution $\mathcal{P}^i_{XY}$. As $\psi(D_i) \in \mathbb{R}^{pq}$, we can use a kernel $k_p(s,t) : \mathbb{R}^{pq} \times \mathbb{R}^{pq} \to \mathbb{R}$ to measure similarity between $\mathcal{P}^i_{XY}$ and $\mathcal{P}^j_{XY}$ say. Note that while we have discussed the embedding of the joint distribution $\mathcal{P}_{XY}$, it is straightforward to also embed the marginal or conditional distributions.

An important choice is the form of $\phi_1(\mathbf{x})$ and $\phi_2(y)$, as these define the features of the distribution $\mathcal{P}_{XY}$ we would like to capture. For example $\phi_1(\mathbf{x}) = \mathbf{x}$ would be capturing the mean of the marginal distribution $\mathcal{P}_X$. In practice (keeping sample size s the same), if we know that $\mathcal{P}^i_{XY} \approx \mathcal{P}^j_{XY}$, we would expect that $\theta^*_i \approx \theta^*_j$, however generally the converse does not hold and the exact relationship is unknown. Hence, we opt for a flexible representation for $\phi_1(\mathbf{x})$ and $\phi_2(y)$ using small neural networks, which we optimise as part of the marginal likelihood maximisation. This can be thought of as a learning to learn setup [1], where a smaller robust model is used to optimise over bigger models. As we now have f on inputs $(\theta, \mathcal{P}_{XY}, s)$, we can fit a GP (with the standard normal noise model) on all observations $\{\{(\theta^i_k, \mathcal{P}^i_{XY}, s_i), t^i_k\}_{k=1}^{N_i}\}_{i=1}^n$ (along with any observations on the target), optimising any unknown parameters through the marginal likelihood. In order to propose the next θ^{target} to evaluate, we let $f^{target}(\theta) = f(\theta, \mathcal{P}^{target}_{XY}, s_{target})$ and maximise the acquisition function $\alpha(\theta; f^{target})$. Here, we use the expected improvement [9], however other options are readily applicable.

4 Experiments

We term the proposed method distBO and use a two layer neural network (of size 20 hidden units and 10 output units with RELU activation) for $\phi_1(\mathbf{x})$, while using an RBF network with 4 landmark points for $\phi_2(y)$. In general, results are fairly robust to sensible settings of these networks. For baselines, we will consider random search (RS), Bayesian optimisation (noneBO), multi-task Bayesian optimisation (multiBO) [15] (learns a similarity between tasks through evaluations of f *only*) and Bayesian optimisation with all source samples (allBO). The last baseline essentially ignores the distribution of the dataset, and assumes all seen samples come from the same (target) task. For noneBO and multiBO, we will initialise with 5 iterations of random search. We repeat each experiment 30 times and ignore k_s as samples sizes do not differ across tasks here. Throughout for both source and target tasks, we will run 30 iterations for each algorithm. To obtain $\{\theta^i_k, t^i_k\}_{k=1}^{30}$ for source task i, we will use noneBO. For additional experiments, please see [8].

Toy dataset To demonstrate our algorithm, we create 1-dimensional distribution of datasets from the following generative process:

$$\mu^i \sim \mathcal{N}(\gamma^i, 1) \quad X^i_l|\mu^i \sim \mathcal{N}(\mu^i, 1) \quad Y^i_l|X^i_l \sim \mathcal{N}(X^i_l, 1)$$

Given γ^i, we can simulate a μ^i as a characteristic varying across task, before sampling $D_i = \{x^i_l, y^i_l\}_{l=1}^{s_i}$ from $X^i_l, Y^i_l|\mu^i$. We consider a simple form of f given by $f(\theta; D_i) = \exp\left(-\frac{(\theta - \frac{1}{s}\sum_l x^i_l)^2}{2}\right)$, and here $\theta \in [-8, 8]$ plays the role of a hyperparameter that we would like to learn and 'labels' y^i_l are just nuisance variables. The optimal hyperparameter is the sample mean of the $\{x^i_l\}$ and hence it is varying together with the underlying mean of the data μ^i. We now perform an experiment with $n = 15$, and $s_i = 400$, and generate 3 source task with $\gamma^i = 0$,

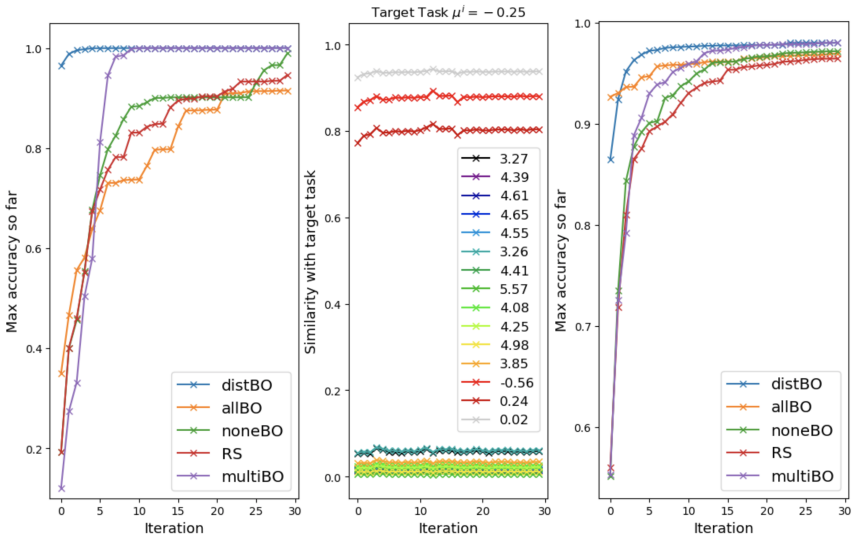

Figure 1: **Left, middle**: Target toy task with 15 source datasets, with max accuracy f seen so far (left) and similarity with the target task D_{target} (right). Legend represents the various sampled μ^i for the source tasks (target task $\mu^i = -0.25$). **Right**: Target patient task with 30 source datasets.

and 12 source task with $\gamma^i = 4$. In addition, we generate an additional target dataset with $\gamma^i = 0$. The idea is that only 3 of the source tasks should be helpful to solve our target task, and distBO which can learn this should be able to reach the target optimum in only a few shots, as shown in the left of Figure 1. Here, distBO is able to take advantage of the correct source tasks, allowing it to outperform other baselines. In particular this is evident in the middle graph of Figure 1, which shows the similarity measure $k_p(\psi(D_i), \psi(D_{target})) \in [0, 1]$ for $i = 1, \dots 15$. The feature representation has correctly learned to put high similarity on the three source datasets from the same process and low similarity on source datasets that are not from the same process. We also observe that while the multi-task BO also achieve the target optimum quickly, it is unable to few-shot the optimum, as it does not make use of meta-information, hence needing intialisations from the target task to even begin learning the similarity across tasks.

Real dataset The Parkinson's disease telemonitoring dataset[1] consists of voice measurements using a telemonitoring device for 42 patients with Parkinson disease (200 recordings each). The label is the clinician's Parkinson disease symptom score for each recording. Following [2], we can treat each patient as a separate task. For the model, we employ RBF kernel ridge regression (with hyper-parameters α, γ), with f as the coefficient of determination (R^2) $\in [0, 1]$. While this problem does not gain from BO, as f is not expensive, this allows for a comprehensive benchmark comparisons. Here, we take a particular patient to be our target task, and randomly choose $n = 30$ other patients as source tasks. We show the result in the right of Figure 1 for a particular patient to highlight the behaviour. In fact across all patients being the target, distBO outperforms (or is equivalent to) all baselines in terms of faster convergence to the optimum. Here, it is clear that by encoding the distributional meta-information, we can learn useful similarities between tasks.

5 Conclusion

We demonstrated that it is possible to borrow strength between multiple hyperparameter learning tasks by making use of the similarity between training datasets used in those tasks. This helped us to develop a method which finds a favourable setting of hyperparameters in only a few evaluations of the target objective. We argue that the model performance should not be treated as a black box function as it corresponds to specific known models and specific known datasets and that its careful consideration as a function of all its inputs, and not just of its hyperparameters, can lead to useful algorithms.

[1] http://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[2] Gilles Blanchard, Aniket Anand Deshmukh, Urun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *arXiv preprint arXiv:1711.07910*, 2017.

[3] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable meta-learning for bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018.

[4] Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. 2015.

[5] Arthur Gretton. Notes on mean embeddings and covariance operators. 2015.

[6] Jungtaek Kim, Saehoon Kim, and Seungjin Choi. Learning to transfer initializations for bayesian hyperparameter optimization. *arXiv preprint arXiv:1710.06219*, 2017.

[7] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016.

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

[9] J Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

[10] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, Bernhard Schölkopf, et al. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141, 2017.

[11] Matthias Poloczek, Jialei Wang, and Peter I Frazier. Warm starting bayesian optimization. In *Proceedings of the 2016 Winter Simulation Conference*, pages 770–781. IEEE Press, 2016.

[12] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[13] 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.

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

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