A COPULA APPROACH FOR HYPERPARAMETER TRANSFER LEARNING

Anonymous authors
Paper under double-blind review

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 tuning performance metrics of the same algorithm 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 regress the mapping from hyperparameter to metric quantiles with a semi-parametric Gaussian Copula distribution, which provides robustness against different scales or outliers that can occur in different tasks. We introduce two methods to leverage this estimation: a Thompson sampling strategy as well as a Gaussian Copula 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 state-of-the-art methods.

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 find the good hyperparameters in short time. BO addresses the black-box optimization problem by placing a probabilistic model on the function to minimize (e.g., the mapping of neural network hyperparameters to a validation loss), and determine which hyperparameters to evaluate next by trading off exploration and exploitation through acquisition function. While traditional BO focuses on each problem in isolation, recent years have seen a surge of interest in *transfer learning* for HPO. The key idea is to exploit evaluations from previous, related *tasks* (e.g., the same neural network tuned on multiple datasets) to further speed up the hyperparameter search.

A central challenge of hyperparameter transfer learning is that different tasks typically have different scales, varying noise levels and, and possibly contain outliers, making it hard to learn a joint model. In this work, we show how semi-parametric Gaussian Copula can be leveraged as learning a joint prior across datasets becomes easier as scale issues vanish. We then demonstrate how such prior estimate can be used to transfer information across tasks and metrics. We propose two HPO strategies: a Copula Thompson Sampling and a Gaussian Copula Process. We show that these approaches can jointly model several metrics with potentially different scales, such as validation error and compute time, without requiring processing. We demonstrate significant speed-ups over a number of baselines in extensive experiments.

The paper is organized as follows. Section 2 reviews related work on transfer learning for HPO. Section 3 introduces copula regression, the building block for the HPO strategies we propose in Section 4. Specifically, we show how Copula regression can be applied to design two HPO strategies, one based on Thompson sampling and an alternative GP-based approach. Experimental results are given in Section 5 where we evaluate both approaches against state-of-the-art methods on three algorithms. Finally, Section 6 outlines conclusions and further developments.

2 RELATED WORK

A variety of methods have been developed to induce transfer learning in HPO. The most common approach is to model tasks jointly or via a conditional independence structure, which has been been explored through multi-output GPs (Swersky et al., 2013), weighted combination of GPs (Schilling et al., 2016; Feurer et al., 2018), and neural networks, either fully Bayesian (Springenberg et al., 2016) or hybrid (Snoek et al., 2015; Perrone et al., 2018; Law et al., 2018). 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 (Golovin et al., 2017; Poloczek et al., 2016). 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 (Feurer et al., 2015; Wistuba et al., 2015b).

A key challenge for joint models is that different black-boxes can exhibit heterogeneous scale and noise levels. To address this, some methods have instead focused on search-space level, aiming to prune it to focus on regions of the hyperparameter space where good configurations are likely to lie. An example is Wistuba et al. (2015a), where related tasks are used to learn a promising search space during HPO, defining task similarity in terms of the distance of the respective data set meta-features. A more recent alternative that does not require meta-features was introduced in Perrone et al. (2019), where a restricted search space in the form of a low-volume hyper-rectangle or hyper-ellipsoid is learned from the optimal hyperparameters of related tasks. The Gaussian Copula Process (GCP) (Wilson & Ghahramani, 2010) can also 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 Anderson et al. (2017) to handle potentially non-Gaussian data, albeit only considering non-parametric homoskedastic priors for the single-task and single metric case.

3 GAUSSIAN COPULA REGRESSION

For each task denote with $f^j : \mathbb{R}^p \to \mathbb{R}$ the error function one wishes to minimize, and with $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ the evaluations available for an arbitrary task. Given the evaluations on M tasks $\mathcal{D}^M = \bigcup_{1 \leq j \leq M} \{x_i^j, y_i^j\}_{i=1}^{N_j}$, we are interested in speeding up the optimization of an arbitrary new task f, namely in finding $\arg\min_{x \in \mathbb{R}^p} f(x)$ in the least number of evaluations. One possible approach to speed-up the optimization of f is to build a surrogate model $\hat{f}(x)$. While using a Gaussian process is possible, scaling such an approach to the large number of evaluations obtained in transfer learning is challenging. Instead, we propose to fit a parametric estimate of $\hat{f}_\theta(x)$ distribution which can be latter used in HPO strategies as a prior of a Gaussian Copula Process. A key requirement here is to learn a joint model, e.g., we would like to find θ which fits well on all observed tasks f^j. We show how this can be achieved with a semi-parametric Gaussian Copula in two steps: first we map all evaluations to quantiles with the empirical CDF, and then we fit a parametric Gaussian distribution on quantiles mapped through the Gaussian inverse CDF.

First, observe that as every y_i comes from the same distribution for a given task, the probability integral transform results in $u_i = F(y_i)$, where F is the cumulative distribution function of y. We then model the CDF of $(u_1, \ldots, u_N)$ with a Gaussian Copula:

$$C(u_1, \ldots, u_N) = \phi_{\mu,\Sigma}(\Phi^{-1}(F(y_1)), \ldots, \Phi^{-1}(F(y_N))),$$

where Φ is the standard normal CDF and $\phi_{\mu,\Sigma}$ is the CDF of a normal distribution parametrized by μ and Σ. Assuming F to be invertible, we define the change of variable $z = \Phi^{-1} \circ F(y) = \psi(y)$ and $g = \psi \circ f$.[1] We regress the marginal distribution of $P(z|x)$ with a Gaussian distribution whose mean and variance are two deterministic parametric functions given by

$$P(z|x) \sim \mathcal{N}(\mu_\theta(x), \sigma_\theta(x)) = \mathcal{N}(w_\mu^T h_{w_h}(x) + b_\mu, \Psi(w_\sigma^T h_{w_h}(x) + b_\sigma)),$$

where $h_{w_h}(x) \in \mathbb{R}^d$ is the output of a multi-layer perceptron (MLP) with d hidden nodes and free-parameters $w_h, w_\mu \in \mathbb{R}^d, b_\mu \in \mathbb{R}, w_\sigma \in \mathbb{R}^d, b_\sigma \in \mathbb{R}$ are projection parameters and $\Psi(t) = \log(1 +$

[1]Note that if z is regressed perfectly, then finding the minimum of f is solved as a parameter x minimizing $\psi(f(x))$ also minimizes $f(x)$ since ψ is monotonically increasing.

$\exp t)$ is an activation mapping to positive numbers. The parameters $\theta = \{w_h, w_\mu, b_\mu, w_\sigma, b_\sigma\}$ together with the parameters in MLP are learned by minimizing the Gaussian negative log-likelihood on the available evaluations $\mathcal{D}^{\mathcal{M}} = \bigcup_{1 \leq j \leq M} \{x_i^j, z_i^j\}_{i=1}^{N_j}$, e.g., by minimizing

$$\sum_{(x,z) \in \mathcal{D}^{\mathcal{M}}} \frac{1}{2} \log 2\pi\sigma(x)^2 + \frac{1}{2} \left(\frac{z - \mu(x)}{\sigma(x)} \right)^2 + \psi'(\psi^{-1}(z)), \tag{1}$$

with SGD. Here, the term $\psi'(\psi^{-1}(z))$ accounts for the change of variable $z = \psi(y)$.

Note that the transformation ψ requires F, which needs to be estimated. Rather than using a parametric or density estimation approach, we use the empirical CDF $\tilde{F}(t) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{y_i \leq t}$. While this estimator has the advantage of being non-parametric, it leads to infinite value when evaluating ψ at the minimum of maximum of y. To avoid this issue, we use the Winsorized cut-off estimator:

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

where N is the number of observations of y and choosing $\delta_N = \frac{1}{4N^{1/4}\sqrt{\pi \log N}}$ strikes a bias-variance trade-off (Liu et al., 2009). This approach is semi-parametric in that the CDF is estimated with a non-parametric estimator and the Gaussian Copula is estimated with a parametric approach.

The benefit of using a non-parametric estimator for the CDF is that it allows us to map the observations of each task to comparable distributions as $z^j \sim \mathcal{N}(0, 1)$ for all tasks j. This is critical to allow the joint learning of the parametric estimates μ_θ and σ_θ which shares their parameter θ across all tasks. As our experiments will show, one can regress a parametric estimate that has a standard error lower than 1, meaning that information can be leveraged from the evaluations obtained on the other tasks. A trivial predictor for z would predict 0 and yield a standard error of 1. In the next section we show how this estimator can be leveraged to design two novel HPO strategies.

4 COPULA BASED HPO

4.1 COPULA THOMPSON SAMPLING

Given the predictive distribution $P(z|x) \sim \mathcal{N}(\mu_\theta(x), \sigma_\theta(x))$, it is straightforward to derive a Thompson sampling strategy (Thompson, 1933) exploiting knowledge from previous tasks. Given N candidate hyperparameter configurations $x_1, \ldots, x_N$, we sample from each predictive distribution $\tilde{z}_i \sim \mathcal{N}(\mu_\theta(x_i), \sigma_\theta(x_i))$ and then evaluate $f(x_i)$ where $i = \arg\min_i \tilde{z}_i$. Pseudo-code is given in Algorithm 1.

Algorithm 1 Copula random search (CRS)

Learn the parameters θ of $\mu_\theta(x)$ and $\sigma_\theta(x)$ on hold-out evaluations $\mathcal{D}^M$ by minimizing equation 1.
while Has budget **do**
  Sample N candidate hyperparameters $x_1, \ldots, x_N$ from the search space
  Draw $\tilde{z}_i \sim \mathcal{N}(\mu_\theta(x_i), \sigma_\theta(x_i))$ for $i = 1, \ldots, N$
  Evaluate $f(x_i)$ where $i = \arg\min_i \tilde{z}_i$
end while

While this 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 Gaussian Copula regression can be combined with a GP to both learn from previous tasks while adapting to the current task.

4.2 GAUSSIAN COPULA PROCESS

Instead of modeling observations with a GP, we model them as a Gaussian Copula Process (GCP) (Wilson & Ghahramani, 2010). Observations are mapped through the bijection $\psi = \Phi^{-1} \circ F$, where

we recall that Φ is the standard normal CDF and that F is the CDF of y. As ψ is monotonically increasing and mapping into the line, we can alternatively view this modeling as a warped GP (Snelson et al., 2004) with a non-parametric warping. One advantage of this transformation is that $z = \psi(y)$ 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. Furthermore, we can use the information gained on other tasks with μ_θ and σ_θ by using them as prior mean and variance. To do so, the following residual is modeled with a GP:

$$\begin{aligned} r(x) &= \frac{g(x) - \mu_\theta(x)}{\sigma_\theta(x)} \\ &\sim \text{GP}(m(x), k(x, x')), \end{aligned}$$

where $g = \psi \circ f$. We use a Matérn-5/2 covariance kernel and automatic relevance determination hyperparameters, optimized by type II maximum likelihood to determine GP hyperparameters (Rasmussen & Williams, 2006). Fitting the GP gives the predictive distribution of the residual surrogate

$$\hat{r}(x) \sim \mathcal{N}(\mu_r(x), \sigma_r(x)).$$

Because μ_θ and σ_θ are deterministic functions, the predictive distribution of the surrogate $\hat{g}$ is then given by:

$$\begin{aligned} \hat{g}(x) &= \hat{r}(x)\sigma_\theta(x) + \mu_\theta(x) \\ &\sim \mathcal{N}(\mu_g(x), \sigma_g(x)) \\ &\sim \mathcal{N}(\mu_r(x)\sigma_\theta(x) + \mu_\theta(x), \sigma_r(x)\sigma_\theta(x)) \end{aligned}$$

Using this predictive distribution, we can select the hyperparameter configuration maximizing the Expected Improvement (EI) (Mockus et al., 1978) of $g(x)$. The EI can then be defined in closed form as

$$\begin{aligned} \text{EI}(x) &= \mathbf{E}[\max(0, g(x_{\min}) - \hat{g}(x))] \\ &= \sigma^2(x)(v(x)\Phi(v(x)) + \phi(v(x))), \quad \text{where } v(x) := \frac{\mu_g(x) - g(x_{\min})}{\sigma_g^2(x)}, \end{aligned}$$

with Φ and ϕ being the CDF and PDF of the standard normal, respectively. When no observations are available, the empirical CDF $\tilde{F}$ is not defined. Therefore, we warm-start the optimization on the new task by sampling a set of $N_0 = 5$ hyperparameter configurations via Thompson sampling, as described above. Pseudo-code is given in Algorithm 2.

Algorithm 2 Gaussian Copula process (CGP)

Learn the parameters θ of $\mu_\theta(x)$ and $\sigma_\theta(x)$ on hold-out evaluations $\mathcal{D}^M$ by minimizing equation 1.
Sample an initial set of evaluations $\mathcal{D} = \{(x_i, f(x_i))\}_{i=1}^{N_0}$ via Algorithm 1.
while Has budget **do**
  Fit the GP surrogate $\hat{r}$ to the observations $\{(x, \frac{\psi(y) - \mu_\theta(x)}{\sigma_\theta(x)}) \mid (x, y) \in \mathcal{D}\}$
  Sample N candidate hyperparameters $x_1, \ldots, x_N$ from the search space
  Compute the hyperparameter x_i where $i = \arg\max_i \text{EI}(x_i)$
  Evaluate $y_i = f(x_i)$ and update $\mathcal{D} = \mathcal{D} \cup \{(x_i, y_i))\}$.
end while

4.3 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. For instance, given two hyperparameters with the same expected error, the one requiring fewer resources is preferable. For tasks where runtime is available, we use both runtime and error metrics by averaging in the transformed space, e.g., we set $z(x) = \frac{1}{2}(z^{\text{error}}(x) + z^{\text{time}}(x))$, where $z^{\text{error}}(x) = \psi(f^{\text{error}}(x))$ and $z^{\text{time}}(x) = \psi(f^{\text{time}}(x))$ denotes the transformed error and time observations respectively. 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.

tasks	# datasets	# hyperparameters	# evaluations per dataset	available metrics
DeepAR	11	6	~ 220	quantile loss, time
FCNET	4	9	62208	MSE, time
XGBoost	10	9	5000	1-AUC

Table 1: A summary of the three HPO problems we consider.

5 EXPERIMENTS

We considered the problem of tuning three algorithms on multiple datasets: `XGBoost` (Chen & Guestrin, 2016), a 2-layer feed-forward neural network (`FCNET`) (Klein & Hutter, 2019), and the RNN-based time series prediction model proposed in Salinas et al. (2017) (`DeepAR`). We tuned `XGBoost` on 10 `libsvm` datasets (Chang & Lin, 2011) to minimize $1-$AUC, and `FCNET` on 4 datasets from Klein & Hutter (2019) to minimize the test mean squared error. As for `DeepAR`, the evaluations were collected on the data provided by GluonTS (Alexandrov et al., 2019), consisting of 6 datasets from the M4-competition (Makridakis et al., 2018) and 5 datasets used in Lai et al. (2017), and the goal is to minimize the quantile loss. 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 setup are in Table 1, and the search spaces of the three problems is in Table 4 of the appendix.

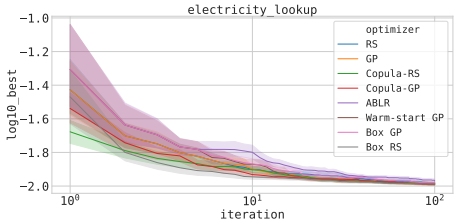
To speed up experiments we used a lookup table approach advocated in Eggensperger et al. (2012) with some notable differences. In their work, instead of evaluating hyperparameters directly on a blackbox, an extrapolation model is built on pre-generated evaluations to predict their performance, thus saving a significant amount of computational time. However, the extrapolation model can introduce noise and lead to inconsistencies compared to using real blackbox evaluations. As a result, in this work we reduced the BO problem to selecting the next hyperparameter configurations from a fixed set that has been evaluated in advance, so that no extrapolation error is introduced. These evaluations were obtained by querying each algorithm at hyperparameters sampled uniformly at random from their search space.

We compare against a number of baselines. We consider random search and GP-based BO as two of the most popular HPO methods. As a transfer learning baseline, we consider warm-start GP (Feurer et al., 2015), using task meta-features to find the most similar task and its 100 best-performing evaluations to warm start the GP on the target task. As meta-features we took the number of observations, number of features, a landmark feature based on Naive Bayes for `FCNET` and `XGBoost`, and also included the class imbalance for the latter. For `DeepAR` we used 7 meta-features available in GluonTS (Alexandrov et al., 2019), namely the log number of times series, log number of observations, mean time series length, log of time series max, log of time series mean, time series min, and mean absolute time series. We also compare against two recently published transfer learning methods for HPO: ABLR (Perrone et al., 2018) and a search space-based transfer learning method (Perrone et al., 2019). ABLR is a transfer learning approach consisting of a shared neural network across tasks on top of which lies a Bayesian linear regression layer per task. Perrone et al. (2019) transfers information by fitting a bounding box to contain the best hyperparameters from each previous task, and applies random search (`Box RS`) or GP-based BO (`Box GP`) in the learned search space.

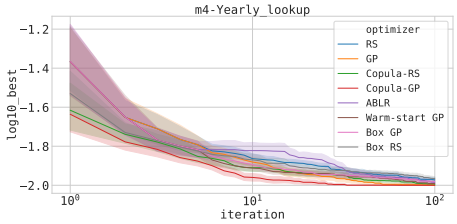
We assess the transfer learning capabilities of these methods in a leave-one-task-out setting: we sequentially leave out one dataset and then aggregated the results for each algorithm. The performance of each method is first averaged over 30 replicates for one dataset in a task, and the relative improvements 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%. Notice that all the metrics y are in $\mathbb{R}^+$. By computing the relative improvements, we can aggregate results across all datasets for each algorithm. Finally, for all copula-based methods, we learn the mapping to copulas via a 3-layer MLP with 50 units per layer, optimized by ADAM with early-stopping.

task	dataset	ABLR	Box GP	Box RS	CGP	CRS	GP	Warm-start GP
DeepAR	electricity	-2.7	-2.1	**0.8**	**0.8**	**0.8**	0.2	-1.8
	exchange-rate	0.5	-0.2	2.1	**3.3**	2.9	0.6	0.3
	m4-Daily	-1.2	-1.0	0.1	**1.4**	1.1	-0.1	-0.8
	m4-Hourly	-18.0	-12.1	-2.8	**3.0**	-0.8	-1.3	-11.7
	m4-Monthly	-0.7	0.2	0.6	**1.0**	0.6	0.2	0.4
	m4-Quarterly	-0.4	-0.1	0.6	**1.0**	0.8	0.1	0.2
	m4-Weekly	-3.8	-3.2	0.5	**0.6**	0.4	0.1	-3.6
	m4-Yearly	-0.1	0.2	0.3	**1.1**	0.7	0.5	0.3
	solar	-0.3	0.3	0.8	**1.1**	0.5	0.3	0.5
	traffic	-1.3	-0.7	**0.5**	**0.5**	0.0	0.2	-0.3
	wiki-rolling	0.1	0.1	0.2	**0.5**	0.3	0.1	0.2
FCNet	naval	57.1	70.6	80.6	**82.3**	78.9	-64.3	44.1
	parkinsons	14.1	27.1	27.3	**38.5**	29.4	20.2	37.2
	protein	0.3	6.8	5.8	**8.3**	6.6	3.3	6.3
	slice	1.8	37.1	48.3	**58.7**	53.3	21.4	23.9
XGBoost	a6a	-0.1	0.1	0.1	**0.2**	-0.1	**0.2**	0.0
	australian	-1.2	1.4	3.1	**3.7**	0.4	1.7	-0.1
	german.numer	-1.3	0.2	**1.3**	0.7	0.5	-0.3	1.3
	heart	1.5	1.5	2.3	**5.1**	0.9	2.8	0.8
	ijcnn1	-8.1	3.1	5.0	**5.8**	3.0	4.1	3.2
	madelon	2.4	3.5	1.7	4.2	-0.4	**4.4**	2.1
	spambase	-2.9	0.9	0.0	1.0	-0.9	**1.8**	-2.2
	svmguide1	-3.2	0.8	**1.7**	1.2	-0.3	1.1	0.3
	w6a	0.8	1.1	-2.5	3.4	0.5	**3.8**	1.6

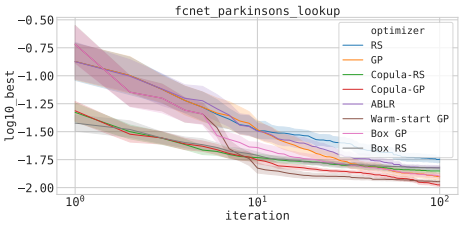
Table 2: Relative improvements over random search averaged over all the iterations. The best methods are highlighted in bold.



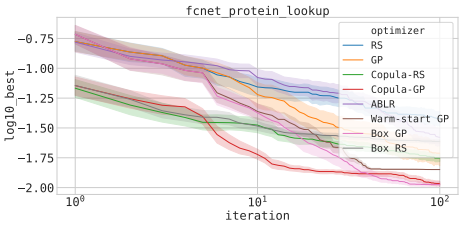
(a) Comparison on `electricity` for `DeepAR`.



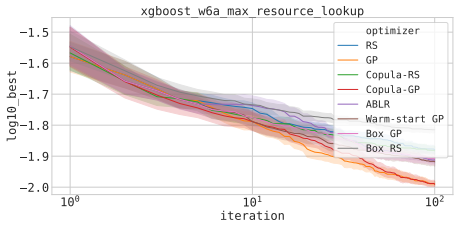
(b) Comparison on `m4-Yearly` for `DeepAR`.



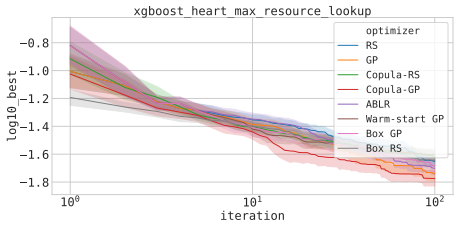
(c) Comparison on `parkinsons` for `FCNET`.



(d) Comparison on `protein` for `FCNET`.



(e) Comparison on `w6a` for `XGBoost`.

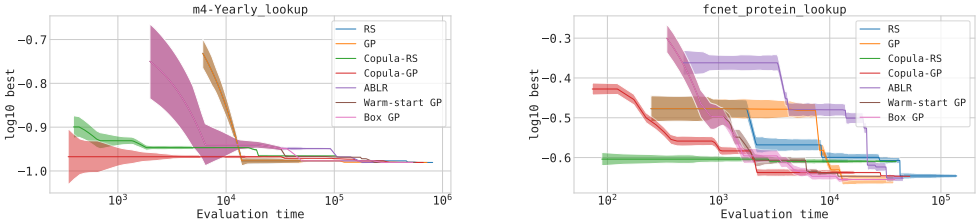


(f) Comparison on `heart` for `XGBoost`.

Figure 1: Result using only error information against iterations.

5.1 RESULTS

Results using only error information We start by studying the setting where only error metrics are used to learn the copula transformation. Within each task, we first aggregate 30 replicates for



(a) Comparison on dataset `m4-Yearly` over time. (b) Comparison on dataset `protein` over time.

Figure 2: Two example tasks using both error and time information against time.

each method to compute the relative improvement over random search at every iteration, and then average the results across all iterations. The results are reported in Table 2, showing that `CGP` is the best method for almost every task except `XGBoost`. In `XGBoost`, there are several tasks on which methods without transfer learning perform quite well. This is not surprising as we have already seen in the ablation study on the copula estimators (Table 6 in the appendix), some tasks in `XGBoost` have relatively high test errors, implying that the transferred prior will not help. In those tasks, `CGP` is usually the runner up method after standard `GP`.

We also show results on two example datasets from each algorithm in Figure 1, reporting confidence intervals obtained via bootstrap. Note that the performance of the methods in the examples for `DeepAR` and `XGBoost` exhibit quite high variations, especially in the beginning of the BO. We conjecture this is due to an insufficient number of evaluations in the lookup tables. Nevertheless, the general trend is that `CRS` and `CGP` outperform all baselines, especially in the beginning of the BO. It can also be observed that `CGP` performs at least on par with the best method at the end of the BO. `Box RS` is also competitive at the beginning, but as expected fails to keep its advantage toward the end of the BO.

Results using both error and time information We then studied the ability of the copula-based approaches to transfer information from multiple metrics. We show two example tasks in `DeepAR` and `FCNET` in Figure 2, where we fix the total number of iterations and plot performance against time with 2 standard error. To obtain distributions over seeds for one method, we only consider time-range where 20 seeds are available which explains why methods can start and end at different time. With ability to leverage training time information, the copula-based approaches have a clear advantage over all baselines, especially at the beginning of the optimization.

We also report aggregate performance over all the tasks in Table 5 in the appendix. Due to the different methods finishing the optimization at different times, we only compare them up to the time taken by the fastest method. For each method we first compute an average over 30 replicates, then compute the relative improvement over random search, and finally average the results across all time points. The copula based methods converge to a good hyperparameter configuration significantly faster than all the considered baselines. Note that we obtain similar results as Hyperband-style method (Li et al., 2016) where optimization can start much earlier than standard HPO methods but only requiring one machine.

5.2 Ablation Study

To give more insight into the components of our method, we perform a detailed ablation study to investigate the choice of the MLP and compare the copula estimation to simple standardization.

Choice of copula estimators For copula-based methods, we use an MLP to estimate the output. We first compare to other possible options, including a linear model and a k-nearest neighbor estimator in a leave-one-out setting: we sequentially take the hyperparameter evaluations of one dataset as test set and use all evaluations from other datasets as a training set. We report the RMSE in Table 6 of the appendix when predicting error of the blackbox. From this table, it is clear that MLP tends to be the best performing estimator among the three. In addition, a low RMSE indicates that the task is close to the prior that we learned on all the other tasks, and we should thus expect trans-

fer learning methods to perform well. As we will see later, FCNET has the lowest RMSE among the three algorithms, and all transfer learning methods indeed perform much better than single-task approaches.

Homoskedastic and Heteroskedastic noise The proposed Copula estimator (MLP) uses heteroskedastic noise. We now compare it to a homoskedastic version where we fix the noise estimate to 1. The results in Table 3 show that heteroskedasticity tends to help on most datasets.

Copula transformation and standardization In our method, we convert the metrics to Gaussian in two steps: first we apply the probability integral transform, followed by a Copula transform using the inverse CDF of a Gaussian. To demonstrate the usefulness of such transformation, we compare it to a simple standardization of the metrics. Results are included in Table 3. Note that the relative improvement metric is not lower bounded, so that when random search has very small value the scale of relative improvement can be arbitrary large (such as for the Protein dataset in FCNET). It is clear that standardization performs significantly worse than the Copula transformation, indicating that it is not able to address the problem of varying scale and noise levels across tasks.

task	dataset	CRS S	CRS Ho	CRS He	CGP S	CGP Ho	CGP He
DeepAR	electricity	-13.2	0.3	**0.8**	-15.3	0.4	**0.8**
	exchange-rate	-127.2	1.8	2.9	-130.6	3.0	**3.3**
	m4-Daily	-58.0	1.0	1.1	-107.3	**1.4**	**1.4**
	m4-Hourly	-98.6	-0.8	-0.8	-94.7	0.7	**3.0**
	m4-Monthly	-24.2	0.3	0.6	-19.2	0.9	**1.0**
	m4-Quarterly	-15.6	0.5	0.8	-11.8	0.8	**1.0**
	m4-Weekly	-96.1	0.2	0.4	-81.9	0.3	**0.6**
	m4-Yearly	-14.0	0.4	0.7	-13.6	0.8	**1.1**
	solar	-14.1	0.4	0.5	-8.7	0.8	**1.1**
	traffic	-17.3	0.3	0.0	-7.4	**0.7**	0.5
	wiki-rolling	-4.5	0.3	0.3	-4.6	0.4	**0.5**
FCNet	naval	-20602.7	72.0	78.9	-4368.4	81.7	**82.3**
	parkinsons	-78.6	27.8	29.6	-96.1	**42.1**	38.8
	protein	-18.8	5.4	6.6	-9.6	8.1	**8.3**
	slice	-870.6	46.3	53.5	14.2	58.5	**58.9**
XGBoost	a6a	-0.7	0.0	-0.1	-0.3	**0.2**	**0.2**
	australian	-50.8	0.2	0.4	-53.4	3.4	**3.7**
	german.numer	-12.2	0.6	0.5	-12.7	0.6	**0.7**
	heart	-70.1	-0.1	0.9	-129.0	3.9	**5.1**
	ijcnn1	-38.8	1.7	3.0	-11.7	**5.9**	5.8
	madelon	-37.9	-0.4	-0.4	-16.2	**4.9**	4.2
	spambase	-30.2	0.6	-0.9	-16.9	**2.6**	1.0
	svmguide1	-28.3	0.8	-0.3	-17.0	1.2	1.2
	w6a	0.6	0.9	0.5	2.7	**4.0**	3.4

Table 3: Relative improvements over random search. CRS stands for Copula-RS and CGP stands for Copula-GP and CRS S stands for Copula-RS using standardization instead of copula transformation, same for CGP S. Ho and He stand for Homoskedastic and Heteroskedastic noise.

6 CONCLUSIONS

We introduced a new class of methods to accelerate hyperparameter optimization by exploiting evaluations from previous tasks. The key idea was to leverage a semi-parametric Gaussian Copula prior, using it to account for the different scale and noise levels across tasks. Experiments showed that we considerably outperform standard approaches to BO, and are more robust to heterogeneous tasks compared to a number of multi-task models for transfer learning recently proposed in the literature. Finally, we showed that our approach can seamlessly combine multiple metrics, such as accuracy and runtime, further speeding up the search of good hyperparameter configurations.

A number of directions for future work are open. First, we could combine our Copula-based HPO strategies with Hyperband-style optimizers (Li et al., 2016). In addition, we could generalize our approach to deal with settings in which related problems are not limited to the same algorithm run over different datasets. This would allow for different hyperparameter dimensions across tasks, or perform transfer learning across different black-boxes.

REFERENCES

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.

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)*, pp. 361–372. IEEE, 2017.

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.

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

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*, pp. 1114–1120, 2012.

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.

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.

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*, pp. 1487–1495, 2017.

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

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.

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

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. Technical report, preprint arXiv:1603.06560, 2016.

Han Liu, John Lafferty, and Larry Wasserman. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. 10:2295–2328, 2009. ISSN 1532-4435. doi: 10.1016/0006-291X(91)91267-G.

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. doi: 10.1016/j.ijforecast.2018.

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

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

Valerio Perrone, Huibin Shen, , Matthias Seeger, Cédric Archambeau, and Rodolphe Jenatton. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

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

Carl Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

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

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*, pp. 33–48. Springer, 2016.

Edward Snelson, Zoubin Ghahramani, and Carl E. Rasmussen. Warped gaussian processes. In S. Thrun, L. K. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 337–344. MIT Press, 2004.

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)*, pp. 2171–2180, 2015.

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

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

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933. ISSN 00063444.

Andrew Gordon Wilson and Zoubin Ghahramani. Copula processes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NeurIPS 2010, pp. 2460–2468, USA, 2010. Curran Associates Inc.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In *Machine Learning and Knowledge Discovery in Databases*, pp. 104–119. Springer, 2015a.

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*, pp. 1–10. IEEE, 2015b.

A APPENDIX

tasks	hyperparameter	search space	type	scale
DeepAR	# layers	$[1, 5]$	integer	linear
	# cells	$[10, 120]$	integer	linear
	learning rate	$[10^{-4}, 0.1]$	continuous	log10
	dropout rate	$[10^{-2}, 0.5]$	continuous	log10
	context_length_ratio	$[10^{-1}, 4]$	continuous	log10
	# bathes per epoch	$[10, 10^{4}]$	integer	log10
XGBoost	num_round	$[2, 2^{9}]$	integer	log2
	eta	$[0, 1]$	continuous	linear
	gamma	$[2^{-20}, 2^{6}]$	continuous	log2
	min_child_weight	$[2^{-8}, 2^{6}]$	continuous	log2
	max_depth	$[2, 2^{7}]$	integer	log2
	subsample	$[0.5, 1]$	continuous	linear
	colsample_bytree	$[0.3, 1]$	continuous	linear
	lambda	$[2^{-10}, 2^{8}]$	continuous	log2
	alpha	$[2^{-20}, 2^{8}]$	continuous	log2
FCNET	initial_lr	$\{0.005, 0.001, 0.05, 0.01, 0.05, 0.1\}$	categorical	-
	batch_size	$\{8, 16, 32, 64\}$	categorical	-
	lr_schedule	{cosine, fix}	categorical	-
	activation layer 1	{relu, tanh}	categorical	-
	activation layer 2	{relu, tanh}	categorical	-
	size layer 1	$\{16, 32, 64, 128, 256, 512\}$	categorical	-
	size layer 2	$\{16, 32, 64, 128, 256, 512\}$	categorical	-
	dropout layer 1	$\{0.0, 0.3, 0.6\}$	categorical	-
	dropout layer 2	$\{0.0, 0.3, 0.6\}$	categorical	-

Table 4: A summary of the search spaces for the three algorithms.

task	dataset	ABLR	Box GP	CGP	CRS	GP	Warm-start GP
DeepAR	electricity	-19.5	-17.0	8.4	9.5	-0.3	-17.0
	exchange-rate	7.4	6.3	39.8	41.1	-0.6	4.5
	m4-Daily	-16.8	-15.0	19.9	19.6	-1.7	-13.6
	m4-Hourly	-92.5	-88.4	27.7	28.3	3.4	-87.1
	m4-Monthly	3.8	6.8	5.3	4.6	-0.6	6.7
	m4-Quarterly	-10.2	-9.5	0.5	-1.2	-0.6	-9.2
	m4-Weekly	-29.1	-27.2	0.5	0.6	-0.5	-26.7
	m4-Yearly	-3.7	-3.0	1.8	1.4	-0.1	-2.8
	solar	1.5	3.9	5.2	4.7	0.1	4.0
	traffic	-43.1	-40.9	6.7	6.4	0.3	-39.8
	wiki-rolling	-1.1	-0.9	2.0	2.5	0.1	-0.7
FCNet	naval	-80.6	69.4	89.9	88.4	-150.0	12.9
	parkinsons	-38.5	30.3	53.5	46.8	22.3	49.6
	protein	-6.9	10.1	11.9	11.6	4.1	11.5
	slice	-14.5	6.1	81.4	81.4	22.4	-4.0

Table 5: Relative improvements over random search averaged over time.

More details on prior MLP architecture. The MLP used to regress μ_θ and σ_θ consists in 3 layers of 50 nodes with dropout layer set to 0.5. The learning rate is set to 0.01, batch size to 64 and we optimize over 100 gradients updates 3 times lowering the learning rate by 10 each time.

task	dataset	KNN_10	KNN_20	KNN_5	Linear	MLP
DeepAR	SP500-returns	1.249	1.249	1.249	1.090	1.140
	dominick-100k	0.815	0.815	0.815	0.832	0.737
	electricity	0.831	0.831	0.831	0.801	0.740
	exchange-rate	0.842	0.842	0.842	0.783	0.780
	m4-Daily	0.804	0.804	0.804	0.792	0.776
	m4-Hourly	0.960	0.960	0.960	0.948	0.884
	m4-Monthly	0.783	0.783	0.783	0.762	0.750
	m4-Quarterly	0.868	0.868	0.868	0.792	0.773
	m4-Weekly	0.776	0.776	0.776	0.754	0.733
	m4-Yearly	0.844	0.844	0.844	0.785	0.759
	parts-nips16	1.271	1.271	1.271	1.160	1.145
	solar	0.963	0.963	0.963	0.875	0.812
	traffic	0.885	0.885	0.885	0.850	0.829
	wiki-rolling	0.904	0.904	0.904	0.868	0.826
FCNet	naval	0.509	0.509	0.509	0.602	0.491
	parkinsons	0.571	0.571	0.571	0.736	0.571
	protein	0.505	0.505	0.505	0.607	0.497
	slice	0.564	0.564	0.564	0.559	0.555
XGBoost	a6a	1.091	1.091	1.091	1.067	1.040
	australian	0.827	0.827	0.827	0.873	0.758
	german.numer	0.900	0.900	0.900	0.891	0.820
	heart	0.818	0.818	0.818	0.793	0.702
	ijcnn1	0.951	0.951	0.951	0.936	0.917
	madelon	0.908	0.908	0.908	0.887	0.834
	skin	1.057	1.057	1.057	0.971	0.971
	spambase	0.931	0.931	0.931	0.950	0.818
	svmguide1	0.849	0.849	0.849	0.912	0.798
	w6a	1.039	1.039	1.039	1.054	1.003

Table 6: RMSE comparison for prior estimators when predicting error of blackbox function given its parameter.