Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Information Systems WSK&I Group

Learning Across Tasks with Surrogate Model Ensembles for Algorithm and Hyperparameter Optimization

Master Thesis

Georgiana Manolache
(0876359)

Supervisor:
Dr. Joaquin Vanschoren

Assessment Committee:
Dr. Joaquin Vanschoren
Dr. Nikolay Yakovets
Dr. Bert de Vries

Eindhoven, August 2019

Learning Across Tasks with Surrogate Model Ensembles for Algorithm and Hyperparameter Optimization

Abstract

Algorithm and hyperparameter optimization is a well studied topic in machine learning. Recent work relies on Bayesian optimization especially for solving the hyperparameter optimization problem for a given machine learning algorithm. Suppose that prior tasks are available on which the algorithm and hyperparameter optimization has already been dealt with either by domain experts or through extensive cross-validation trials. In this paper, we propose to leverage outcomes from these tasks together with Bayesian optimization procedures to warm-start algorithm and hyperparameter optimization for a new task. Specifically, we construct surrogate model ensembles to transfer configurations across similar tasks, unlike many other methods that start searching from scratch every time. In our wide range of experiments, we show that our approach quickly identifies near-optimal models, outperforming the current state-of-the-art.

Keywords: Algorithm Selection, Hyperparameter Optimization, Meta-learning, Bayesian Optimization

1. Introduction

Building a machine learning model often requires a number of iterative, complex and time-consuming computational steps, such as data pre-processing, algorithm selection and hyperparameter tuning (Figure 1). Machine learning includes many algorithms to choose from. These algorithms require further tuning of hyperparameters to reach optimal performance. With a classification problem, for instance, one might wonder whether configuring a Support Vector Machine or a Random Forest classifier yields better performance on a given data set. Because of the interdependency between the algorithms and their hyperparameters, the choices are exponentially numerous. Good configurations can be learned by searching through a small hyperparameter space or by randomly sampling when the hyperparameter space is larger. However, these approaches are very time-consuming and often not feasible, given a large volume of data and a learning algorithm with a high-dimensional hyperparameter space.

Automating the process of choosing the best algorithm and hyperparameters has been an ongoing machine learning research topic over the past years. The most recent work relies on the concept of meta-learning. Machine learning experts use their previous knowledge from similar machine learning problems to resolve a new problem. Similarly, meta-learning can be described as the process of learning from previous information gathered during applying various algorithm configurations on different kinds of data sets and, thus, reducing the time needed to configure new tasks (Vanschoren, 2018). For instance, in systems where new data is continuously acquired, the underlying machine learning models need to be retrained and updated. Although sometimes the models may change significantly, usually

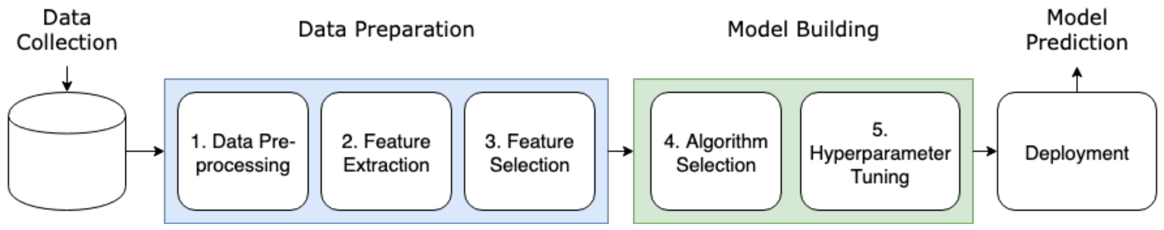

Figure 1: Typical machine learning pipeline

they remain similar to those of previous optimizations and, thus, can be configured more efficiently (Feurer et al., 2018). In practice, we can learn from meta-data that describes previous optimizations, typically algorithm configurations (hyperparameter settings used to train these models), corresponding model evaluations (such as accuracy measures and training time), as well as mensurable proprieties of a optimization, also known as meta-features (such as number of features, classes or missing values).

Bayesian optimization, a technique for solving black-box optimization problems with expensive evaluation functions, has recently become popular for learning models from meta-data. Bayesian optimization constructs a probabilistic model for an expensive function and then uses this model to make decisions about where to evaluate the function next, while trading off exploration and exploitation (Snoek et al., 2012). Several Bayesian optimization methods with meta-data have been developed (Wistuba et al., 2016, 2018; Feurer et al., 2018); however, the majority treats the hyperparameter optimization problem exclusively, as combined algorithm and hyperparameter space is challenging to search.

In this paper, we propose to not only identify hyperparameters for a given machine learning problem, but also to find a better learning algorithm to use based on evaluation meta-data. The question that we intend to answer is: *How can we optimally learn from algorithm evaluations on prior tasks to find optimal algorithms and hyperparameters configurations for new tasks?* First, we discuss some preliminaries (section 2). Following successful previous work for learning from model evaluations (sections 3), we further formulate the problem of algorithm and hyperparameter optimization (section 4). For our methodology, we use ideas from Bayesian optimization to explore the algorithm and hyperparameters space, using a surrogate model ensemble from prior model evaluations (section 5). We perform an extensive empirical analysis using a diverse collection of data sets and experiment settings to demonstrate that the proposed solution outperforms previous state-of-the-art methods (section 6), and finally, discuss and summarize the achievements of this study (section 7).

2. Preliminaries

We briefly review the general Bayesian optimization framework and the meta-learning approach before discussing related work.

Algorithm 1 Sequential Model-based Bayesian Optimization

```
Require:
    Search space $\mathcal{X}$, observation history $\mathcal{D}$, number of iterations $n$, acquisition function $\alpha$,
    objective function $f$, surrogate model $\mathcal{S}$
Ensure:
    Optimal input value
1: for $i \leftarrow 1, n$ do
2:     Fit surrogate model $\mathcal{S}$ on $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{i}$
3:     Select next hyperparameter configuration $x_{new} \leftarrow \arg\max_{x \in \mathcal{X}} \alpha(\mathcal{S})$
4:     Evaluate $y_{new} \leftarrow f(x_{new})$
5:     Update observation history $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_{new}, y_{new})\})$
   return $x_* \leftarrow \arg\min_{(x,y) \in \mathcal{D}}$ y
```

2.1 Bayesian Optimization

Fundamentally, Bayesian optimization is a sequential model-based approach to solving a problem, in particular finding a global minimizer (or maximizer) of an unknown (black-box) function. Formally, Bayesian optimization aims to solve the following objective:

$$x_* = \arg\min_{x \in \mathcal{X}} f(x) \tag{1}$$

where x is any arbitrary query point in the domain space $\mathcal{X}$, often a compact subset of $\mathbb{R}^n$, but can contain categorical or conditional inputs. In the context of machine learning, the function f can be a performance evaluation metric (e.g. accuracy) for an algorithm (e.g. Random Forest) with tunable hyperparameters (e.g. number of trees in the forest) on a given data set. Bayesian optimization is particularly useful when these evaluations of f are costly.

An overview of the generic procedure is shown in Algorithm 1. Bayesian optimization proceeds iteratively by first fitting a probabilistic model $\mathcal{S}$, also referred to as surrogate model, to a set of initial observations $\mathcal{D}$. Observations are tuples (x_i, y_i), where x_i is a function input and y_i is the corresponding output at iteration i. Then, using an acquisition function α on the resulting surrogate model, a new promising input x_{new} is selected (step 3). Finally, the selected configuration is evaluated using an objective function f (step 4) and the observation history is updated (step 5). These steps are repeated until n iterations have been completed or some other stopping condition has been satisfied.

Bayesian optimization uses surrogate models to represent the assumptions over the expensive function f. While there are several different surrogate models available, we use the Gaussian process (GP), which is a common choice. The GP is a non-parametric model that is fully characterized by its prior distribution determined by a mean function $\mu(x)\colon \mathcal{X} \rightarrow \mathbb{R}$ and a positive definite covariance function $k\colon \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. Then, at any finite set of points $x_i \in \mathcal{X}$, $i = \{0, .., n\}$ a GP induces a multivariate posterior Gaussian distribution $\mathcal{N}$ with mean $\mu(x)$ and variance $\sigma^2(x)$:

$$\mathcal{S} \sim \mathcal{N}(\mu(x), \sigma^2(x)) \tag{2}$$

The mean and variance depend on the GP kernel, which has some hyperparameters that can be defined before the model is fit. The posterior mean and variance evaluated at any

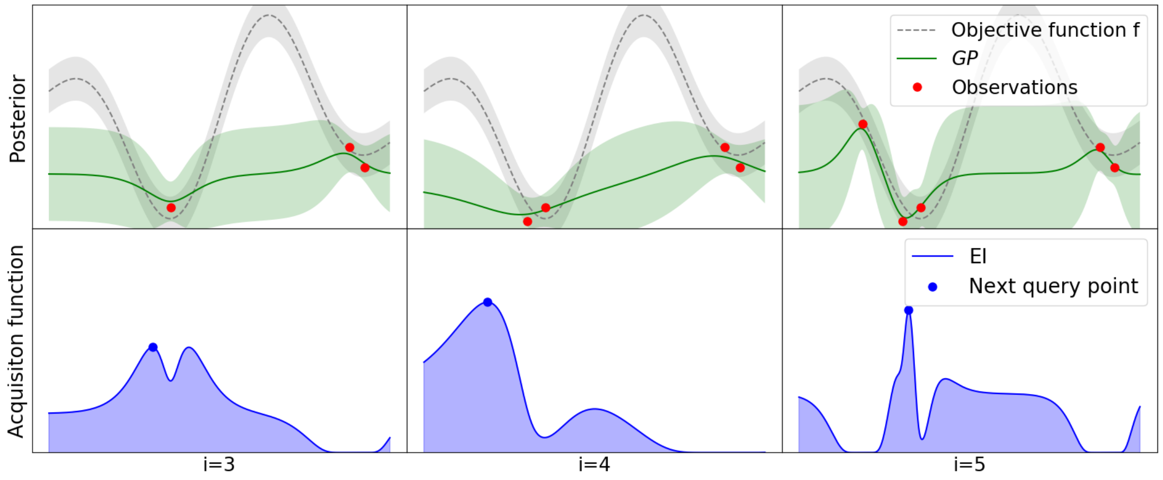

Figure 2: An example of the Bayesian optimization procedure over three iterations. The plots show the mean and confidence intervals estimated with the Gaussian process (GP) approximation of the objective function. Although the objective function is shown, in practice it is unknown. The figure also shows the EI acquisition functions in the lower plots. The acquisition is high where the model predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration).

observed point x_i represents the model's prediction and uncertainty respectively, in the objective function at the point x_i, as shown in Figure 2. These posterior functions are used to select the next input to consider.

For selecting the next promising input acquisition functions designed to trade off exploitation and exploration of the search space areas are typically used. Exploitation means sampling where the surrogate model predicts high values and exploration means sampling at locations where the prediction uncertainty is high. The goal is to maximize the acquisition function, thus, maximize the exploitation and exploration values to determine the next sampling point (in Figure 2, the next query point in the lower plots). The expected improvement (EI) (Jones et al., 1998), a frequent choice for Bayesian optimization, is used in our experiments as an acquisition function. Assuming y_* is the best function evaluation observed by far, namely $y_* \in \min_{y \in \mathcal{D}}$, the EI is defined as:

$$\begin{aligned} \alpha_{\mathrm{EI}}(\mathcal{S}) &=_{y \sim \mathcal{S}} [\max(0, y_* - y)] \\ &= \sigma(\theta) z \phi(z) + \sigma(\theta) z \Phi(z) \end{aligned} \tag{3}$$

where $z = \frac{y_* - \mu(\theta)}{\sigma(\theta)}$. $\phi(\cdot)$ and $\Phi(\cdot)$ denote standard normal density and distribution functions, and μ and σ are the expected value and standard deviation of the surrogate model. After having observed n points, samples can be drawn from GP posterior using Thomson sampling to obtain the next query point x_{new}. For completeness, a thorough introduction to Bayesian optimization is given by Shahriari et al. (2016).

2.2 Learning from Model Evaluations

While there are several meta-learning techniques, this study focuses on learning techniques from evaluation measures alone. These techniques can be used to recommend generally useful configurations or configuration spaces, as well as transferring configurations from similar tasks. Evaluations assess an algorithm's configurations on a task (e.g. binary classification, regression, clustering) according to an evaluation measure (e.g. accuracy, logarithmic loss, F1-score) and method (e.g. cross-validation). Given a set of prior machine learning tasks $t_j \in T$, as well as a set of algorithms fully defined by their hyperparameter configurations $\theta_i \in \Theta$ and the set of corresponding scalar evaluations $y_i = P(\theta_i, t_j) \in \mathbf{P}$ indicating the performance of configuration θ_i on task t_j, the meta-learner is trained on $\mathbf{P} \cup \mathbf{P}_{new}$. $\mathbf{P}_{new}$ is the set of evaluations on a new task t_{new} and it is either learned by the meta-learner or warm-started (Vanschoren, 2018). In the following section we detail several learning techniques from literature which recommend configurations from similar tasks.

3. Related Work

Recommending configurations based on evaluation measures from prior tasks $t_j \in T$, requires additional information on how similar a task t_j is to a new task t_{new}. One way to measure similarity between t_j and t_{new} is using relative landmarks (RL) (Fürnkranz and Petrak, 2001). RL determine the performance difference of a configuration relative to another configuration given a task. Task similarity can then be defined as the strength of the correlation between the relative landmarks obtained on two tasks. Leite et al. (2012) adopt this approach, proposing Active Testing, a strategy which iteratively selects an algorithm whose performance exceeds previous evaluated algorithms on similar tasks. Tasks are deemed similar if the RL of evaluated tasks correlate strongly. After learning task similarity, Leite et al. (2012) find configurations that performed well on the similar tasks and then try them on the new task. In contrast to Bayesian optimization approaches, Active Testing requires a discretized set of algorithms configurations, limiting the choices for the hyperparameters.

Another similarity measure uses the cross-task performance of surrogate models. If a previous task's surrogate model built on all its available evaluations can generate accurate predictions on t_{new}, then those tasks are implicitly similar. This is usually done in combination with Bayesian optimization to warm-start the process of finding the next configurations to be tried for t_{new}. In the following, we discuss only warm-starting techniques from evaluation meta-data.

Wistuba et al. (2016) develop a two-stage knowledge transfer, first fitting GPs to each prior as well as the new task and then combining them. Predictions from each individually fitted GP are weighted using a Nadaraya-Watson kernel (Nadaraya, 1964) by defining a distance metric across tasks. The kernel combines mean predictions of past models with means of the new model, but ignores variances, making the resulting combined model not a GP anymore. Wistuba et al. (2018) further extend previous work proposing to transfer knowledge from meta-data in the acquisition function rather than the surrogate model. Here, an acquisition function is the weighted average of expected improvement on new task evaluations and predicted improvements on all prior task evaluations.

Our work is immediately related to (Feurer et al., 2018) which proposes to combine GPs fitted to each prior task as well as a new task into a single GP as a weighted sum of predictions of each prior task. Predictions from each model are weighted using Lacoste agnostic Bayesian ensemble (Lacoste et al., 2014). The ensemble model remains GP and unlike Wistuba et al. (2016, 2018) the solution allows us to directly use established techniques for Bayesian optimization, such as parallelization via Monte Carlo integration. We adopt this strategy as it scales better than previous warm-starting solutions, but we weight models using relative landmarks (RL).

Warm-started multi-task models are another way to capture similarity between a new task and a prior task. Earlier work (Swersky et al., 2013) starts from a set of tasks assumed a priori to be similar and combines all previous tasks and the new task into a multi-task GP. Information between each past and new task is transferred by building a joint task representation to warm-start Bayesian optimization. However, learning a joint GP scales more poorly than single-task GPs. Springenberg et al. (2016) also assume that the tasks are related, but learn task similarity during optimization using Bayesian Neural Networks. Finally, Perrone et al. (2017) build a multi-task neural network to learn a joint representation of the hyperparameter space but the surrogate models themselves are not multi-task, thus scaling more efficiently.

A complementary way to leverage prior task model evaluation is to recommend which configurations to not be chosen (Vanschoren, 2018). After training surrogate models on prior tasks, regions where the configurations are predicted to be poor are discarded to speed up the search. Wistuba et al. (2015) adopt this approach, ranking configurations using evaluations from past and new tasks.

Apart from Leite et al. (2012), all the above methods aim to solve only the hyperparameter optimization problem, while we aim to include the selection of the algorithm as well. Auto-sklearn (Feurer et al., 2015), a system based on scikit-learn (Pedregosa et al., 2011), does both algorithm and hyperparameter optimization. It improves previous solution Auto-WEKA (Thornton et al., 2013) by identifying configurations from past tasks that perform well on the task and by constructing ensembles from the models evaluated during the optimization. However, Auto-sklearn meta-learns using a set of meta-features instead.

4. Problem Definition

In this section the problem of algorithm selection and hyperparameter tuning is formally defined to introduce our main contribution.

4.1 Algorithm Selection and Hyperparameter Optimization Problem

In machine learning, the goal of algorithm selection as part of the building process is to determine an algorithm $A^* \in \mathcal{A}$ with optimal generalization performance (Thornton et al., 2013). Generalization performance on an algorithm is estimated by training a mathematical model on some sample data D^{train} and measuring its performance on different sample data D^{valid} to provide an evaluation of the model fit on the training data. Determining optimal generalization performance is then the task of finding an algorithm A^* that leads to a model that minimizes the loss on D^{valid}:

$$A^* = \arg\min_{A \in \mathcal{A}} \mathcal{L}(A(D^{train}), D^{valid}) \tag{4}$$

which, for instance, is the misclassification rate for a classification problem, or the Mean Squared Error for a regression problem.

Most machine learning algorithms are further defined by their own hyperparameter configurations $\theta_i \in \Theta_A$, where Θ_A is the hyperparameter space for algorithm A. Unlike the algorithm space, the hyperparameter space is often continuous and highly-dimensional, leading to mixed-type optimization problems. However, we can define the problem of optimizing the hyperparameters similarly:

$$\theta_* = \arg\min_{\theta_i \in \Theta_A} \mathcal{L}(A_{\theta_i}(D^{train}), D^{valid}) \tag{5}$$

Thus, given a set of algorithms $A^1, A^2, ..., A^q \in \mathcal{A}$ and corresponding hyperparameter spaces $\Theta_{A^1}, \Theta_{A^2}, ..., \Theta_{A^q}$, we define model selection in Equation 6. For the remainder of this paper we will consider the problem of algorithm and hyperparameter optimization for a classification task as a concrete example; hence, f returns the misclasification rate. Also, we will refer to the algorithm and hyperparameter optimization problem collectively as *model selection*. Our goal is to find a minimizer for f.

$$\begin{aligned} A^*_{\theta_*} &= \arg\min_{A^k \in \mathcal{A}, \theta_i \in \Theta_{A^k}} \mathcal{L}(A^k_{\theta_i}(D^{train}), D^{valid}) \\ &= \arg\min_{A^k \in \mathcal{A}, \theta_i \in \Theta_{A^k}} f(A^k, \theta_i) \end{aligned} \tag{6}$$

4.2 Bayesian Optimization with Gaussian Process Ensembles

In principle, there are various approaches to tackle Equation 6. A natural choice is of course the Bayesian optimization framework. We consider the approach presented by Feurer et al. (2018) which improve the Bayesian optimization by combining models from past and current task into a GP ensemble before finding the next promising configuration to be tried. In this section we describe the process of building an ensemble model in detail.

Consider $j-1$ prior tasks and define $\mathcal{D}_p = \{(x_i^p, y_i^p)\}_{i=1}^{n_p}$ p=$[1, j-1]$ with n_p function evaluations made for each of those tasks. Firstly, GPs are fitted to the observations from all $j-1$ tasks, one surrogate model per prior task, and referred to as *base models.* These models remain fixed, meaning we do not obtain new observations for them. The current task to be solved is t_j. Then, a GP is refit after each new function evaluation from t_j in the Bayesian optimization framework and called *target model.* Denote base and the target models by $S_1, S_2, ..., S_{j-1}, S_j$.

The goal is to minimize the objective function using the base and the target models in a weighted combination of the predictions of each model:

$$\overline{\mathcal{S}} = \sum_{i=1}^{j} w_i \mathcal{S}_i \tag{7}$$

In simpler terms: the target surrogate model is a weighted combination of the prior and target models. A model of this form is preferred as it does not change the usual Bayesian optimization structure. It maintains the distributional proprieties of a GP and can be used with standard acquisition functions. Furthermore, predicting on each model $\mathcal{S}_i$, $i = [1, j]$ individually incurs $\mathcal{O}(n^3 j)$ computational time, which scales better than holding all prior tasks into a single GP which is $\mathcal{O}(n^3 j^3)$:

$$\overline{\mathcal{S}} \sim \mathcal{N}\big(\sum_{i=1}^{j} w_i \mu(\theta), \sum_{i=1}^{j} w_i^2 \sigma_i^2(\theta)\big) \tag{8}$$

Feurer et al. (2018) weight each model according to estimates of their generalization performance using the ranking loss. Given $n_j > 1$ target function evaluations, the loss is defined as the number of misranked pairs:

$$\mathcal{L}(f, \mathcal{D}_j) = \sum_{i=1}^{n_j} \sum_{k=1}^{n_j} 1(f(x_i^j) < f(x_k^j) \oplus (y_i^j < y_k^j)) \tag{9}$$

where $\oplus$ is the exclusive-or operator and $f(x_i^j)$, $f(x_k^j)$ are sampled evaluation functions from GP posterior. As the number of observations made with Bayesian optimization on the target model grows, base models accumulate misrankings, while the target model generalizes better. Eventually all the weight will be shifted to the target model, thus, converging to standard Bayesian optimization.

In this study, we aim to leverage such model ensembles built on prior tasks evaluation meta-data to solve the model selection problem. The surrogate ensemble will have a better anytime performance: it will need much fewer examples to give accurate predictions than a single surrogate model trained only on the target task.

5. Methodology

In this section, we describe our novel method for the model selection with surrogate model ensembles and the weighting of models in the ensemble.

5.1 Model Selection with Surrogate Model Ensembles

We describe our approach for model selection with surrogate model ensembles in detail. Suppose that we are given a set of $j-1$ prior optimizations denoted by $t_1, t_2, ..., t_{j-1}$ and the current problem to be solved is t_j. We refer to these optimizations collectively as tasks $t_p \in T$ with $p = [1, j]$. Let $A^k \in \mathcal{A}$ $k = [1, q]$ denote a finite set of classification algorithms. Suppose that each algorithm was applied over the prior tasks and we have evaluations in the form of a collection of tuples (θ, y), where θ is an input configuration (usually a set of scalar and categorical values) and y is the evaluation score, respectively. For example, in a classification problem y may represent accuracies or precision scores. Let data set $\mathcal{D}_p^k = \{(\theta_i^{k,p}, y_i^{k,p})\}_{i=1}^{n_{k,p}}$ hold evaluation measures from each task $t_p \in T \setminus \{t_j\}$ over each algorithm $A^k \in \mathcal{A}$. Typically these would be collected from previous optimization runs and stored in a file or database like OpenML (Vanschoren et al., 2013). The goal is to optimize the current task, i.e. to find $\arg\min_{A^k \in \mathcal{A}, \theta_i \in \Theta_{A^k}} f(A^k, \theta_i)$ through these data sets. To support us in this task, we use Bayesian optimization procedures, building Gaussian processes and exploring them with an acquisition function, as shown in Figure 3.

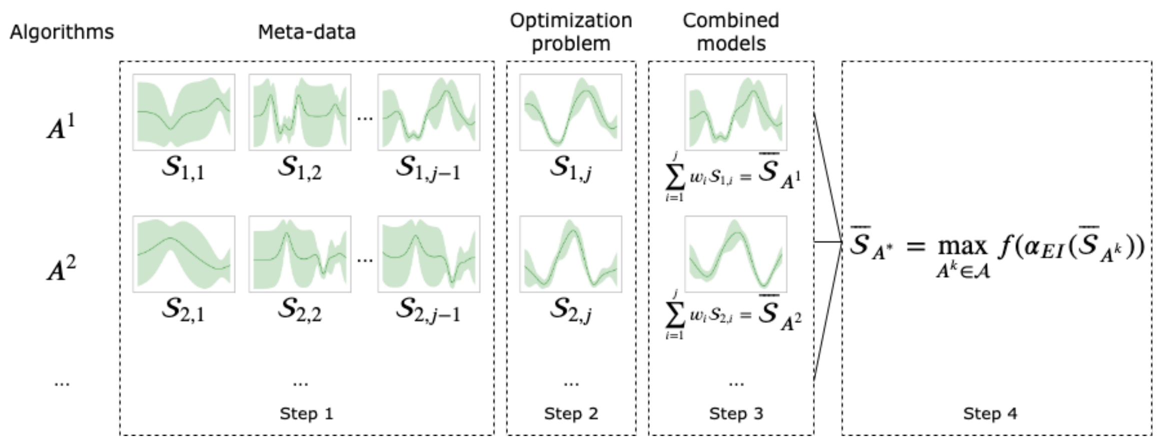

Figure 3: Illustration of proposed model selection with ensemble models. $A^1, A^2, ..., A^k \in \mathcal{A}$ represent the learning algorithms applied to each task $t_1, t_2, ..., t_{j-1}, t_j$ and $\mathcal{S}_{k,1}, \mathcal{S}_{k,2}, ..., \mathcal{S}_{k,j-1}, \mathcal{S}_{k,j}$ are the GP surrogate models fit to task observations for each algorithm. Ensemble model $\overline{\mathcal{S}}_{A^*}$ with the best observed function evaluation indicates the current best algorithm A^*.

First, we fit GPs to all evaluations of each algorithm-task combination from prior optimization problems. In previous work (Feurer et al., 2018), one surrogate model is built over the hyperparameter space of a given algorithm. We build one surrogate model for each task and each algorithm in $A^k \in \mathcal{A}$, under the assumption that that a surrogate model is simpler (only a few hyperparameters instead of many) and more transferable. We call these models base models and we denote them by $S_{k,p}$ for $k = [1, q]$ and $p = [1, j-1]$.

We want to solve the optimization problem for a new task t_j. We compute a set of m observations $m < n$ for each algorithm using the corresponding algorithm-hyperparameter space Θ_{A^k}. We use a random set of hyperparameters and we evaluate their performance, computing a set $\mathcal{D}^k_{p'} = \{(\theta^{k,p'}_i, y^{k,p'}_i)\}^{m_{k,p'}}_{i=1}$ for each algorithm on task t_j for $j = p'$. We then fit target GPs on every $\mathcal{D}^k_{p'}$ meta-data set and we denote them by $S_{k,p'}$.

We combine the base and target models per algorithm according to a weight w (Equation 7). Our combined models are similar to Feurer et al. (2018), although we compute weights based on the relative landmark score and we compute them for multiple algorithms. The weights are updated at each iteration. Our GP weighting is discussed in section 5.2.

From each of the combined $\overline{\mathcal{S}}_{A^k}$ we select the most promising configuration with the EI acquisition function. We use the combined GP's mean and variance $\overline{\mu}(\theta) = \sum_{i=1}^{p} w_i \mu_i(\theta)$ and $\overline{\sigma^2}(\theta) = \sum_{i=1}^{p} w_i^2 \sigma_i^2(\theta)$ with $p = [1, j]$, thus, being able to apply standard EI (Equation 3). After collecting all new configurations θ^k_{new} suggested by the acquisition function, we evaluate them obtaining y^k_{new} and increase the observation history $\mathcal{D}^k_{p'} \cup \{(\theta^k_{new}, y^k_{new})\})$ for each $\overline{\mathcal{S}}_k$. Best algorithms are ranked based on the y^k value. After n iterations, we identify $A^*_{\theta_*} = \arg\min_{(\theta^{k*}, y^{k*}) \in \mathcal{D}^{k*}}$ y^k.

Algorithm 2 Model Selection Framework

```
Require:
    A set of $j-1$ data sets $\mathcal{D}^k_p = \{(\theta^{k,p}_i, y^{k,p}_i)\}^{n_{k,p}}_{i=1}$ from each prior task $t_p \in T$ for each
    algorithm $A^k \in \mathcal{A}$, observation history $\mathcal{D}^k_{p'} = \{(\theta^{k,p'}_i, y^{k,p'}_i)\}^{m_{k,p'}}_{i=1}$ for each algorithm on
    current task $t_j$ with $p' = j$, search spaces $\Theta_{A^k}$ for each algorithm, number of iterations
    $n$, objective function $f$, acquisition function $\alpha_{EI}$
Ensure:
    Optimal algorithm-hyperparameter
 1: To each prior task, fit base models $\mathcal{S}_{k,p}$ on $\mathcal{D}^k_p = \{(\theta^{k,p}_i, y^{k,p}_i)\}^{n_{k,p}}_{i=1}$
 2: Fit target models $\mathcal{S}_{k,p'}$ on observations $\mathcal{D}^k_{p'} = \{(\theta^{k,p'}_i, y^{k,p'}_i)\}^{m_{k,p'}}_{i=1}$
    for $i \leftarrow m, n$ do
 3:     Compute ensemble surrogate models $\overline{\mathcal{S}}_{A^k} = \sum^j_{i=1} w_i \mathcal{S}_{k,i}$ for each algorithm, where
    weights are updated accordingly.
 4:     Select next hyperparameter configuration $\theta^k_{new} \leftarrow \arg\max_{\theta \in \Theta_{A^k}} \alpha_{EI}(\overline{\mathcal{S}}_{A^k})$,
        Evaluate $y^k_{new} \leftarrow f(x^k_{new})$
        Update observations $\mathcal{D}^k_{p'} = \mathcal{D}^k_{p'} \cup \{(\theta^{k,p'}_{new}, y^{k,p'}_{new})\}$
return $A^*_{\theta_*} \leftarrow \arg\min_{(\theta^{k*}, y^{k*}) \in \mathcal{D}^{k*}} \mathrm{y}^k$
```

Algorithm 2 shows how ensemble models are built over algorithms. Inputs of the algorithm are: (1) a set of $j-1$ data sets $\mathcal{D}^k_p = \{(\theta^{k,p}_i, y^{k,p}_i)\}^{n_{k,p}}_{i=1}$ from each prior task $t_p \in T$ for each algorithm $A^k \in \mathcal{A}$, (2) current task t_j observation history $\mathcal{D}^k_{p'} = \{(\theta^{k,p'}_i, y^{k,p'}_i)\}^{m_{k,p'}}_{i=1}$ with $j = p'$, (3) search spaces Θ_{A^k} for each algorithm, (4) number of iterations n, (5) objective function f, (6) acquisition function α_{EI}, and it returns optimal algorithm-hyperparameter for a given machine learning problem.

5.2 Relative Landmark Weighted Gaussian Processes

In this section we further explain the weighting of surrogate models with relative landmarks (RL). Given an algorithm with a fixed set of hyperparameter configurations Θ, a task $t \in T$ and a performance measure M (e.g. accuracy), RL indicates the performance of a configuration relative to another configuration for the given task:

$$RL(\theta_i, \theta_j, t) = M(\theta_i, t) \geq M(\theta_j, t) \tag{10}$$

The decision concerning the $\geq$ operator may be established through a statistical significance test or a simple scalar comparison. Such sets of landmarks can be used to characterize a task and, thus, compute similarities between tasks. In this work, the goal is to measure the performance of GPs.

Since GPs are compared, models that can correctly predict the relative performance of configurations on a new task are better. Consider a pair of hyperparameter configurations (θ_i, θ_j) where $\theta_i, \theta_j \in \Theta$ and a task $t \in T$. Then, for a GP, define the RL function as:

$$\mathrm{RL}(\theta_i, \theta_j, f^t) = 1(f^t(\theta_i) > f^t(\theta_j) \wedge (y^t_i > y^t_j)) \tag{11}$$

where $f^t(\theta_i)$ and $f^t(\theta_j)$ $i \neq j$ is the performance predicted by the surrogate models at each specified point θ_i, θ_j, and y^t_i and y^t_j the actual performance measures $M(\theta_i, t), M(\theta_i, t)$.

Then, Function $1(\textit{condition})$ returns 1 if the surrogate correctly predicts that configuration θ_i evaluates better than θ_j on task t, and 0, otherwise. In practice, we measure how well the target model trained on all previous evaluations on the target task predicts the relative performance of configurations on other tasks. We use leave-one-out models which are stored so that it can be updated rather than retrained from scratch every time. As such, an observation (θ_i, y_i) is left out at a time:

$$\mathrm{RL}(\theta_i, \theta_j, f^{t_{new}}) = 1(f_{-i}^{t_{new}}(\theta_i) > f_{-i}^{t_{new}}(\theta_j) \wedge (y_i^{t_{new}} > y_j^{t_{new}})) \tag{12}$$

Based on this, for each task (including the target task case), we count the number of times $(f^t(\theta_i) > f^t(\theta_j) \wedge (y_i^t > y_j^t)$. With target observations $\mathcal{D} = \{(\theta_k, y_k)\}_{k=0}^{n}$ for which optimization is being executed, we introduce a new function called RLGP to rank each model:

$$\mathrm{RLGP}(f^t, \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{RL}(\theta_i, \theta_j, f^t) \tag{13}$$

Finally, a model is weighted by its ability to predict relative performance of configurations on the new task. Similarly to Feurer et al. (2018), S samples are drawn from the GP posterior at each past observations to obtain the ranks on configurations evaluated so far. By sampling, the overall uncertainty of the models is also taken into account when weighted. Weights for each model i are, thus, computed as:

$$w_i = \frac{1}{S} \sum_{s=1}^{S} \mathrm{RLGP}(f^{t_i}, \mathcal{D}) \tag{14}$$

where ties are either broken randomly or weighted on t_{new} if included in the tie.

We can further refine this method by taking into account prediction accuracy and task similarity. Four variations to the RLGP function are proposed: RLGP-diff, RLGP-fdiff, RLGP-mudiff and Sim. The first three methods attempt to increase the RL score if models can accurately predict the relative performance of configurations, while the last method attempts to weight models by how similar they are to the a new task.

RLGP-diff takes into account how large is the difference between the sampled evaluations functions $f^t(\theta_i)$ and $f^t(\theta_j)$, the method returning this difference if GPs order correctly observations according to the actual value y:

$$\mathrm{RLGP\text{-}diff}(f^t, \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{RL}(\theta_i, \theta_j, t) \cdot (f^t(\theta_i) - f^t(\theta_j)) \tag{15}$$

RLGP-fdiff method computes the difference between the sampled evaluations functions and true values, as such $(f^t(\theta_i) - f^t(\theta_j)) - (y^t(\theta_i) - y^t(\theta_j))$.

$$\mathrm{RLGP\text{-}fdiff}(f^t, \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{RL}(\theta_i, \theta_j, t) \cdot ((f^t(\theta_i) - f^t(\theta_j)) - (y^t(\theta_i) - y^t(\theta_j))) \tag{16}$$

RLGP-mudiff is similar to RLGP-fdiff only that instead of the actual difference, computes the mean difference μ over all samples s drawn from the GP posterior.

$$\text{RLGP-mudiff}(f^t, \mathcal{D}) = \mu \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \text{RL}(\theta_i, \theta_j, t) \tag{17}$$

Sim uses Leite et al. (2012) ATWs similarity $Sim(t, t_{new})$ between a prior task $t \in T$ and the new task t_{new}. Up to now, we assumed all tasks are similar to the target task. Let c the counter of hyperparameter-evaluation pairs between a prior task $t \in T$ and the new task t_{new} for which the estimations match. Then, Sim is defined as:

$$c = \sum_{i=1}^{n}\sum_{j=1}^{n} 1((f^t(\theta_i) > f^t(\theta_j)) \wedge (f_{-i}^{t_{new}}(\theta_i) > f_{-i}^{t_{new}}(\theta_j)))$$

$$Sim(t, t_{new}) = \frac{2c-1}{2n^2-1} \tag{18}$$

$$\text{Sim}(f^t, \mathcal{D}) = Sim(t, t_{new}) \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} \text{RL}(\theta_i, \theta_j, t)$$

6. Empirical Evaluation

This section presents a series of experiments to explore how our methodology performs in practice. We use both synthetic and real data problems to provide a rigorous study of our two main contributions. First, section 6.1 provides a detailed study of RLGP and its variants using simple synthetic functions and a set of SVM benchmarks, comparing them to the other state-of-art methods on a single algorithm. Then, the performance of the proposed automated model selection is assessed on several functions and a large collection of classification algorithms in section 6.2.

Implementation Details In the experiments, GP regression is done using a Matérn 5/2 kernel for smoothness (Shahriari et al., 2016), and the posterior distributions for the kernel hyperparameters is inferred with the NUTS sampler (Hoffman and Gelman, 2014). Bayesian optimization procedures are implemented with the ROBO library (Klein, 2017).

Metrics The reported results are estimated using leave-one-data-set-out cross-validation on the available data, every time using the one left out data set as the optimization task and the rest as meta-data. We compute the average ranks and the distance to global minimum at each iteration (also known as regret or optimization gap) of each compared method, two common evaluation metrics for Bayesian optimization problems. Both strategies are plotted against the number of iterations.

For the average ranks, at each iteration, each optimization strategy is ranked for each task according to their best score. In case of ties, ranks are averaged. The ranks are finally averaged over all data sets to get the final score. Here, better scores are ranked smaller. The second strategy uses the distance to the global minimum at each iteration defined as:

$$DTGM(\mathcal{D}_n, t_j) = \min_{\theta_i \in \mathcal{D}_n = \{(\theta_i, y_i)\}_{i=0}^{n-1}} f^j(\theta_i) - f^j_{min} \tag{19}$$

where $\mathcal{D}_n$ is the observation history after n iterations, t_j is the task being optimized and f^j_{min} is the global minimum. We finally average this score over all optimization tasks.

In order to perform multiple comparisons between our proposals and the rest of the techniques considered we use Friedman-Nemenyi post-hoc statistical significance tests at a significance level of 0.05.

6.1 Ensemble Weighting

Synthetic Tests This section describes the experiment settings to explore how RLGP performs on two different synthetic functions. This experiment in particular evaluates the capability of the proposed ensemble models to find the best hyperparameter configuration since we know the actual optimum. Furthermore, we use this set of tests to establish a baseline for the algorithm selection experiments. For the first synthetic function experiment, the setup used by Feurer et al. (2018) is reproduced.

A modified version of Alpine1 function (Jamil and Yang, 2013) is used for generating similar data sets. The mathematical notation is shown in Equation 20 and its representation in Figure 4 (top-left plot).

$$f(x, \rho) = x \sin(x + \pi + \rho + \frac{x}{10}) \tag{20}$$

Parameter ρ is used as a variation of similarity, with $k = 0$ as data set to optimize hyperparameters for, and $\rho = \frac{k\pi}{12}$ with $k = 1, 2, ..., 5$, data sets used as past runs for meta-learning. This data allows a baseline to be set between evaluated methodologies.

We also investigate how well our methods perform when data sets are not similar. As such, the second synthetic test uses a collection of functions generating random data sets. Six functions from Jamil and Yang (2013) are adapted to create artificial data sets, including Alpine1, Alpine2 and their variations, and Ginuta and Griewank (Jamil and Yang, 2013). The mathematical notation of each of those functions is shown in Equation 21 and their representation in Figure 4 (bottom-left plot). Similarly, parameter $k = 1, 2, ..., 5$ indicates data sets used as past runs for meta-learning, while $k = 0$ is the target data set.

$$l(x, k) = \begin{cases} \frac{(1-x)(\sin(x+5)+0.1)}{10} + 5 + k - 1, & \text{if } k = 0 \\ x \sin(x + \pi + k + \frac{x}{10}), & \text{if } k = 1 \\ \sqrt{x} \sin(x + k - 1) + k - 1, & \text{if } k = 2 \\ \frac{6}{10} + \sin(\frac{16}{15}x - 1) + \sin(\frac{16}{15}x - 1)^2 & \\ + \frac{1}{50} \sin(4(\frac{16}{15}x - 1)) + k - 1, & \text{if } k = 3 \\ \frac{x^2}{4000} - \cos(\frac{x}{3}) + 1 + k - 1, & \text{if } k = 4 \\ \frac{1-x}{10}(\sin(x + 4) + \frac{1}{10}) + k - 1, & \text{otherwise} \end{cases} \tag{21}$$

For both tests GPs are fitted to each meta-learning data set on 20 randomly selected configurations. Then, each optimization run is initialized with three randomly selected configurations, until a total of 20 runs is conducted. The optimization is repeated 120 times, each with a different random selection of configuration initializations.

Figure 4 summarizes the results from the synthetic experiment on the ensemble. Seven methods are evaluated on this setting: standard Bayesian optimization which fits a GP

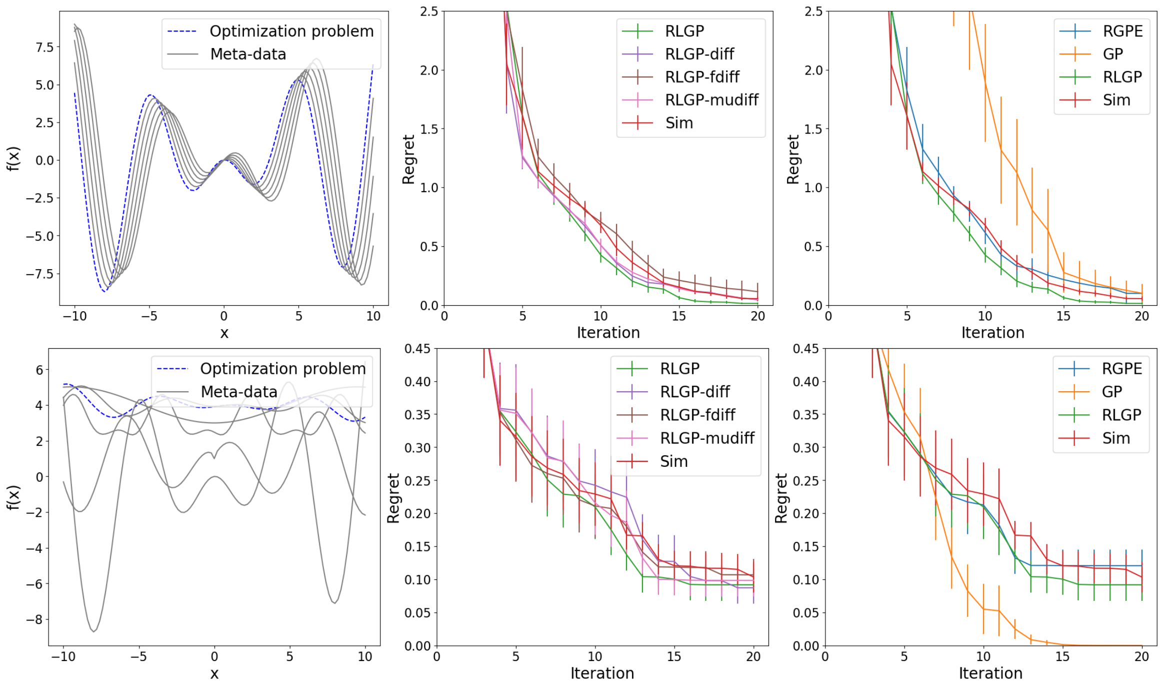

Figure 4: Results for Bayesian optimization ensembles with synthetic functions. The plots show the regret for both Alipne1 function (Equation 20) and random functions (Equation 21) for the proposed variants (middle plots) and the comparing methods (right plots).

model to the target data set only (GP), RGPE (Feurer et al., 2018) and the five proposed RLGP variants. The five variants are plotted separately in the middle plots. We discuss the variants first. For both experiments RLGP performs the best, as opposed to Sim and RLGP-diff.

We proceed to pick RLGP and Sim to compare with the other two methods for both synthetic functions, as shown in Figure 4 (right plots). For both synthetic tests, the warm-starting RLGP variant preforms best, overcoming RGPE, although their difference is not statistically significant ($p > 0.05$). Also, we randomize the initial design in the Bayesian optimization, according to Feurer et al. (2018). If the historically best on average hyperparameters were chosen first, it would potentially provide an extra warm-starting. As expected, when data is not similar and the search space is noisy, the ensembles perform worse in the second synthetic test. In this case it is better to revert to standard Bayesian optimization, randomly picking configurations to be tried instead of using warm-starting.

SVM The second experimental validation for the ensembles comprises 50 data sets on which SVM is trained, used in previous studies (Wistuba et al., 2016, 2018; Feurer et al., 2018). This experiment further consolidates the proposed solutions, since synthetic data impairs credibility as it is artificially created. Here, the number of hyperparameters is

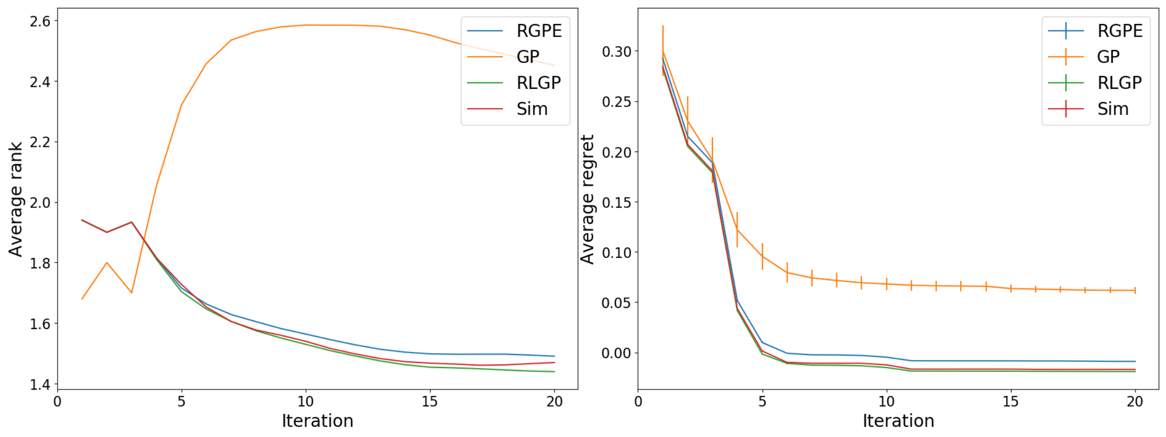

Figure 5: Results for Bayesian optimization ensembles with SVM benchmarks.

six, composed of both scalar and categorical values, with a total of 288 hyperparameter combinations sampled per data set. More details about the meta-data are given in Wistuba et al. (2016).

We reproduce the experiment setup used by Feurer et al. (2018). For each data set, GPs are fitted to each of the other 49 meta-learning data sets on 50 randomly selected configurations, treating the remaining data set as the target optimization problem. Then, each optimization run is initialized with three randomly selected configurations, until a total of 20 iterations is conducted. Bayesian optimization is repeated 20 times for each data set, for a total of 1000 optimization executions, each time with a different random selection of configurations.

Four methods from section 5.2 are benchmarked, namely RGPE, GP, RLGP and Sim. Figure 5 shows the average rank and average regret over 1000 Bayesian optimizations. Overall, RLGP ranks first followed by Sim and RGPE, however, we cannot conclude this since we do not identify significant difference between the methods ($p > 0.05$). Also Sim is not able to learn task similarity over RLGP.

6.2 Model Selection Benchmarks

Synthetic Tests For this experiment, we use the same set of functions from Equation 21 for generating six different sets of data. Each function substitutes an algorithm behaviour for which we produce six similar data sets. As such, parameter k is used as a variation of similarity for each algorithm data set, with $k = 0, 1, 2, ..., 5$. Thus, for each algorithm function there are six similar data sets, with $k = 0$ representing the new target data of a specific algorithm function.

For the model selection, we use three randomly selected configurations to initialize each model. Similarly, we fit GPs to each meta-learning dataset on 20 randomly selected configurations. Then, each optimization run is initialized with three randomly selected configurations, until a total of 20 runs is conducted. The optimization is repeated 120 times, each with a different random selection of configurations. This is repeated for each algorithm

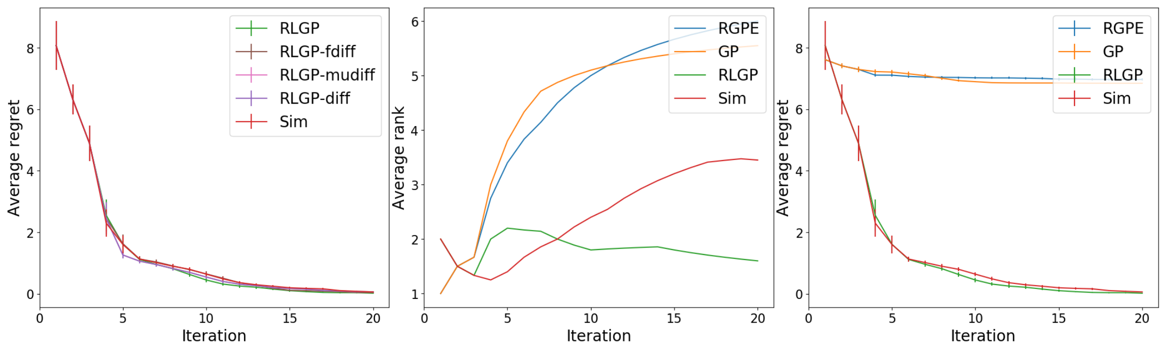

Figure 6: Results for Model Selection framework with synthetic function algorotihms. The plots show the regret for the proposed variants (left plots) and the average ranks (middle plots) and regret for the comparing methods (right plots). Methods are averaged over functions.

function, for a total of 720 optimization runs.

Figure 6 summarizes the results from the synthetic experiment on the model selection framework. We shows the average rank and the average regret of compared methods RGPE, GP, and RLGP and Sim, for which proposed model selection is applied. Once again, we see that the variants cannot learn better models, and we pick RLGP and Sim for further comparisons. With the selection of the algorithm there is a clear difference ($p < 0.001$ as shown in Table 6.2) in performance for RLGP and Sim in comparison to RGPE and GP. Furthermore, Sim has an early drop by the forth iteration, however, RLGP performs slightly better than Sim.

WEKA Our main experimental validation of the automated model selection approach is based on a large set of 553 algorithm-hyperparameter data sets used in Fusi et al. (2018). This data comprises 11 algorithms (as shown in Table 6.2) for which 42000 hyperparameter configurations have been gathered.

Like in section 6.1, we use only 50 randomly selected data sets, treating 49 data sets as the generated evaluation meta-data each time. Base GPs are fitted to a set of 50 randomly selected configurations. We use ten randomly selected configurations to initialize model selection.

The same four methods are benchmarked: RGPE and GP for which we run Bayesian optimization per algorithm, and RLGP and Sim for which we apply our model selection framework. The results of this optimization are shown in Figure 7. Sim and RLGP have the best average rank, which is consistent throughout the optimization process. Additionally, RGPE has an earlier drop in regret by the forth iteration, but overall, the two variants perform similarly according to the Nemenyi tests which identifies highly significant difference ($p < 0.001$) (as shown in Table 6.2). Our solution is significantly better than RGPE and GP. Also, we are able to learn good configurations quicker, with less than ten iterations.

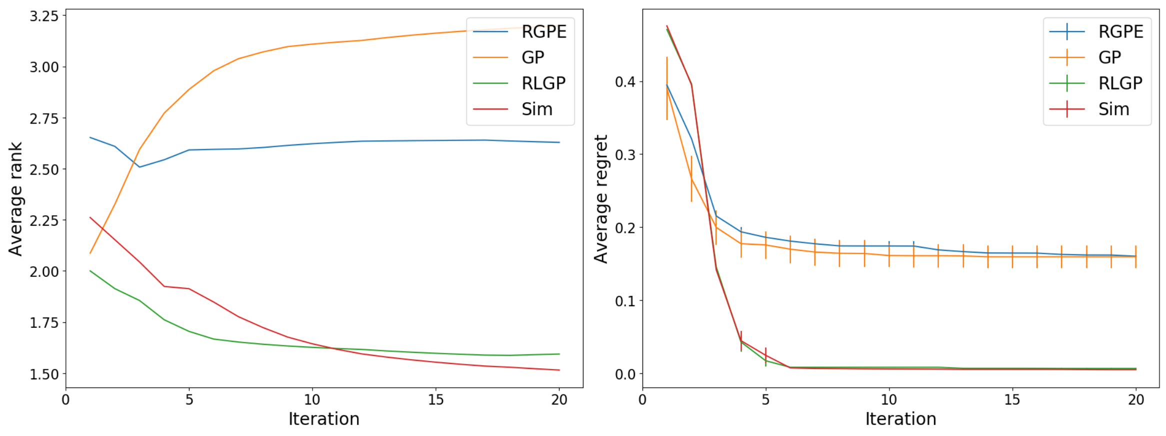

Figure 7: Results for the Model Selection framework with WEKA classifieres. Methods are averaged over all algorithms.

Table 1: WEKA Classifiers. The table shows the number of hyperparameters for each algorithm which consist of both categorical and numerical types.

Classifier	**Hyperparameters**
Bernoulli Naive Bayes	2
Decision Tree	4
Extra Trees	5
Extreme Gradient Boosting	5
Gradient Boosted Decision Trees	7
Gaussian Naive Bayes	2
K Neighbors	3
Linear Discriminant Analysis	4
Multinomial Naive Bayes	2
Quadratic Discriminant Analysis	1
Random Forest	5

Table 2: Statistical significance tests between the compared methods. p_F is the Friedman significance level. We reject the hull hypothesis if p_F is smaller than the significance level (i.e. $p_F < 0.05$).

Experiment	p_F		**RLGP**	**Sim**
Synthetic Alpine1/Random	> 0.05	**RGPE**	-	-
		GP	-	-
SVM	> 0.05	**RGPE**	-	-
		GP	-	-
Synthetic Function Algorithms	< 0.05	**RGPE**	$< 0.001^{***}$	$< 0.001^{***}$
		GP	$< 0.001^{***}$	$< 0.5^{*}$
WEKA Algorithms	< 0.05	**RGPE**	$< 0.001^{***}$	$< 0.001^{***}$
		GP	$< 0.001^{***}$	$< 0.001^{***}$

7. Conclusion

In this work, we have presented a new approach to effectively select a machine learning algorithm as well as the its hyperparameters using meta-learning and Bayesian optimization procedures. Rather than starting from scratch, we can learn across prior tasks and warm-start Bayesian optimization, identifying good configurations with fewer iterations.

Our solution is sample-efficient for expensive algorithms, which otherwise are difficult to tune. Also, compared to active testing, we discretize the configuration space enabling more hyperparameter options to be tried. We achieve that by learning one surrogate model per algorithm, under the assumption that surrogate modes are simpler and more transferable.

We benchmarked our approach against the state-of-the-art with a large number of data sets with different sample sizes. Although generally, our proposed model weighting seems to perform better than the state-of-the-art for warm-starting Bayesian optimization, there is not enough statistical significance. However, using the proposed model selection framework, better scoring algorithms are chosen and tuned for a new optimization task. Our method always identifies optimal algorithm setting, learning algorithm-hyperparameters pairs across tasks, not only hyperparameters. From the results of the Nemenyi test, it can be concluded that the selection model produced significantly better rankings then all the other methods, which justifies the proposal.

Acknowledgments

The author would like to thank the supervisor of this project Joaquin Vanschoren for all the support, time and feedback provided during the whole period of this master thesis.

References

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2755–2763. MIT Press, Cambridge, MA, USA, 2015. URL `http://dl.acm.org/citation.cfm?id=2969442.2969547`.

Matthias Feurer, Benjamin Letham, and Eytan Bakshy. Scalable Meta-Learning for Bayesian Optimization. *arXiv e-prints*, art. arXiv:1802.02219, Feb 2018. URL `https://arxiv.org/abs/1802.02219`.

Johannes Fürnkranz and Johann Petrak. An Evaluation of Landmarking Variants. In *Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001)*, pages 57–68, 2001.

Nicolo Fusi, Rishit Sheth, and Melih Elibol. Probabilistic Matrix Factorization for Automated Machine Learning. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 3352–3361, USA, 2018. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=3327144.3327254`.

Matthew D. Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Momin Jamil and Xin-She Yang. A Literature Survey of Benchmark Functions For Global Optimization Problems. *arXiv e-prints*, art. arXiv:1308.4008, Aug 2013. URL `https://arxiv.org/abs/1308.4008`.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998. ISSN 1573-2916. doi: 10.1023/A:1008306431147. URL `https://doi.org/10.1023/A:1008306431147`.

Aaron Klein. RoBO : A Flexible and Robust Bayesian Optimization. `https://github.com/automl/RoBO`, 2017. URL `https://arxiv.org/abs/1807.01774`. [Online; accessed 2-January-2019].

Alexandre Lacoste, Mario Marchand, Franois Laviolette, and Hugo Larochelle. Agnostic Bayesian Learning of Ensembles. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 611–619, Bejing, China, 22–24 Jun 2014. PMLR. URL `http://proceedings.mlr.press/v32/lacoste14.html`.

Rui Leite, Pavel Brazdil, and Joaquin Vanschoren. Selecting Classification Algorithms with Active Testing. In *In Proceedings of the 8th international conference on Machine Learning and Data Mining in Pattern Recognition, MLDM12*, pages 117–131. Springer-Verlag, 2012.

Élizbar A. Nadaraya. On Estimating Regression. *Theory of Probability & Its Applications*, 9(1):141–142, January 1964. doi: 10.1137/1109020. URL `https://doi.org/10.1137%2F1109020`.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jacob Van der Plas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011. URL `https://arxiv.org/abs/1201.0490`.

Valerio Perrone, Rodolphe Jenatton, Matthias Seeger, and Cedric Archambeau. Multiple Adaptive Bayesian Linear Regression for Scalable Bayesian Optimization with Warm Start. *arXiv e-prints*, art. arXiv:1712.02902, Dec 2017. URL `https://arxiv.org/abs/1712.02902`.

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*, 104:148–175, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proceedings of the 25th International Conference on*

Neural Information Processing Systems - Volume 2, NIPS'12, pages 2951–2959, USA, 2012. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999325.2999464`.

Jost T. Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian Optimization with Robust Bayesian Neural Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4134–4142. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6117-bayesian-optimization-with-robust-bayesian-neural-networks.pdf`.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task Bayesian Optimization. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2004–2012, USA, 2013. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999792.2999836`.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2174-7. doi: 10.1145/2487575.2487629. URL `http://doi.acm.org/10.1145/2487575.2487629`.

Joaquin Vanschoren. Meta-Learning: A Survey. *arXiv e-prints*, art. arXiv:1810.03548, Oct 2018. URL `https://arxiv.org/abs/1810.03548`.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked Science in Machine Learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL `http://doi.acm.org/10.1145/2641190.2641198`.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter Search Space Pruning – A New Component for Sequential Model-Based Hyperparameter Optimization. In Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, João Gama, Alípio Jorge, and Carlos Soares, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 104–119, Cham, 2015. Springer International Publishing. ISBN 978-3-319-23525-7.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Two-Stage Transfer Surrogate Model for Automatic Hyperparameter Optimization. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 199–214, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46128-1.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable Gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1): 43–78, January 2018. ISSN 1573-0565. doi: 10.1007/s10994-017-5684-y. URL `https://doi.org/10.1007/s10994-017-5684-y`.