Meta-Learning for Algorithm and Hyperparameter Optimization with Surrogate Model Ensembles

Georgiana Manolache **Joaquin Vanschoren**

Abstract

We propose a novel meta-learning method for combined algorithm selection and hyperparameter optimization with Bayesian optimization that leverages the surrogate models learned on prior tasks to more efficiently perform model selection for new tasks. It creates an ensemble of both prior and new algorithm-specific surrogate models, able to transfer configurations across similar tasks and efficiently select both algorithms and their hyperparameters for the new task. Empirical results show that this approach quickly identifies near-optimal models, outperforming the current state-of-the-art.

1 Introduction

Automating the process of choosing the best algorithms and hyperparameters for a given machine learning task has been an active research topic over the past years [5]. Some of the most efficient techniques rely on meta-learning. Similarly to how machine learning experts leverage prior knowledge acquired by solving earlier machine learning tasks, meta-learning techniques leverage information gathered through evaluating learning algorithms on earlier tasks [11].

Meta-learning has recently been leveraged in combination with Bayesian optimization techniques for hyperparameter optimization [14, 13, 1]. Bayesian optimization can efficiently search large hyperparameter spaces by constructing a probabilistic (surrogate) model of the performance of learning algorithms, and then using this model together with an acquisition function to decide which hyperparameter configuration to evaluate next [10]. Hence, these surrogate models contain valuable information on how well different algorithms perform on a given task. To do meta-learning, the surrogate models trained on previous tasks can be stored and leveraged when optimizing the hyperparameters for new tasks [13, 1].

Current work in this direction considers hyperparameter optimization for single algorithms exclusively. In this paper, we propose a novel meta-learning technique that generalizes this idea to the more challenging problem of combined algorithm selection and hyperparameter optimization [2]. First, we formulate the problem and discuss related work in Section 2. We introduce our approach in Section 3, and evaluate it across a wide range of machine learning tasks in Section 4. Section 5 concludes.

2 Problem Statement and Related Work

The goal of Combined Algorithm Selection and Hyperparameter optimization (CASH) is to find an optimal algorithm $A^k \in \mathcal{A}$, a set of K algorithms, as well as its optimal hyperparameter configuration $\theta_* \in \Theta_A^k$, where Θ_A^k is the hyperparameter space for algorithm A^k. The optimal solution is the one that minimizes the loss $\mathcal{L}$ (e.g. misclassification rate), measured by training the corresponding model on sample data D^{train} and evaluating it on different sample data D^{valid}:

$$A^*_{\theta_*} = \arg \min_{A^k \in \mathcal{A}, \theta_i \in \Theta_{A^k}} \mathcal{L}(A^k_{\theta_i}(D^{train}), D^{valid}) = \arg \min_{A^k \in \mathcal{A}, \theta_i \in \Theta_{A^k}} f(A^k, \theta_i) \tag{1}$$

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

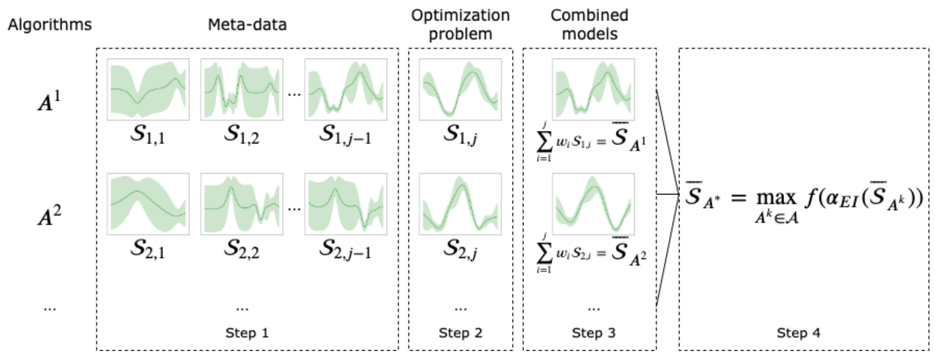

Figure 1: Illustration of proposed methodology. $A^k \in \mathcal{A}$ are the learning algorithms under consideration. $\mathcal{S}_{k,1}, \mathcal{S}_{k,2}, ..., \mathcal{S}_{k,j-1}, \mathcal{S}_{k,j}$ are the GPs obtained by running Bayesian optimization for algorithm A^k on tasks $t_1, t_2, ..., t_{j-1}, t_j$. The ensemble model $\overline{\mathcal{S}}_{A^*}$ with the best observed function evaluation indicates the current best algorithm A^*. The steps correspond to the steps in Algorithm 1.

Bayesian optimization tackles Equation 1 by first evaluating a number of random configurations, and then fitting a probabilistic surrogate model $\mathcal{S}(A^k, \theta_i)$ to predict the loss of unseen configurations. An acquisition function $\alpha(\mathcal{S}(A^k, \theta_i))$, such as Expected Improvement (EI) [7], samples the surrogate model to propose the next configuration to try, trading off exploration and exploitation. This procedure is repeated for a number of iterations [10].

In the meta-learning setting, for a target task t_j, we can leverage meta-data $\mathcal{D}_p = \{(x_i^p, y_i^p)\}_{i=1}^{n_p}$ $p = [1, j-1]$ consisting of n_p performance evaluations made for $j-1$ prior tasks, where x_i^p is a hyperparameter configuration and y_i^p the corresponding evaluation score. There exist many ways to leverage such meta-data [11]. One simple but effective approach is to warm-start Bayesian optimization with the best configurations on *similar* tasks [2] rather than starting from a random initialization. A more recent improvement, ranking-weighted Gaussian process ensembles (RGPE) [1] transfers the surrogate models $\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_{j-1}$ fitted on the previous $j-1$ tasks, hereafter referred to as the *base surrogate models*, and fits a new *target surrogate model* $\mathcal{S}_j$ on the new task. All surrogate models are Gaussian Processes (GPs), and they are combined in a weighted sum $\overline{\mathcal{S}} = \sum_{i=1}^{j} w_i \mathcal{S}_i$. The ensemble maintains the distributional proprieties of a GP and can be used with standard acquisition functions to obtain the next configuration to evaluated on the new task. In every iteration, the corresponding surrogate model are ranked on how well they can predict the relative performance of all configurations already evaluated on the new task, and weighted accordingly. However, this approach has only been evaluated for single-algorithm hyperparameter optimization. Applying it for the combined algorithm and hyperparameter optimization problem is non-trivial since the surrogate models have to learn a more complex and high-dimensional hyperparameter space.

3 Methodology

In this paper, we address this issue by building one GP surrogate model for each combination of a task t_p and algorithm $A^k \in \mathcal{A}$, under the assumption that the surrogate models are simpler (fitting only a few hyperparameters instead of many) and more transferable. This approach also scales better with the number of algorithms since GPs scale cubically in the number of dimensions (hyperparameters). Hence, training k GPs for every task will scale much better than building one GP with the combined hyperparameter space of k algorithms. Moreover, we weight the surrogate models based on the *combined* meta-data from all algorithms, rather than only the meta-data specific to that algorithm, under the assumption that tasks for which *all* algorithms behave similarly to the new task are indeed intrinsically similar to the new task, and the surrogate models fit on them should carry more weight. The method is illustrated in Figure 1 and detailed in Algorithm 1.

Base and target surrogate models. We denote the base surrogate models as $S_{k,p}$ for an algorithm A^k with $k = [1, K]$ and a task t_p with $p = [1, j-1]$. They are GPs trained on all available meta-data $D_p^k = \{(\theta_i^{k,p}, {}_i^{k,p})\}_{i=1}^{n_{k,p}}$. We want to solve the CASH problem for a new task t_j. We compute a set

Algorithm 1 Relative Landmark-weighted Gaussian Processes (RLGP)

```
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 $\mathcal{S}_{k,j}$ on observations $\mathcal{D}^k_j = \{(\theta^{k,j}_i, y^{k,j}_i)\}^{m_{k,j}}_{i=1}$
   for $i \leftarrow m, n$ do
      for $k \leftarrow 1, K$ do
3:       Compute $\overline{\mathcal{S}}_{A^k} = \sum^j_{p=1} w_p \mathcal{S}_{k,p}$ for each $A^k \in \mathcal{A}$, where $w_p \leftarrow \frac{1}{S}\sum^S_{s=1}$ RLGP$(\mathcal{S}_{k,p}, \mathcal{D})$
   based on $S$ samples drawn from the GP posterior
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 $\mathcal{D}^k_j = \mathcal{D}^k_j \cup \{(\theta^{k,j}_{new}, y^{k,j}_{new})\}$
return $A^*_{\theta_*} \leftarrow \arg\min_{(\theta^{k*}, y^{k*}) \in \mathcal{D}^{k*}}$ y$^k$
```

of m initial random hyperparameter evaluations $m < n$ (where n is the total number of iterations) for each algorithm using the corresponding algorithm-hyperparameter space Θ_{A^k}, yielding target meta-data $\mathcal{D}^k_j = \{(\theta^{k,j}_i, y^{k,j}_i)\}^{m_{k,j}}_{i=1}$, and use it to fit the target models $S_{k,p'}$. Per algorithm, we combine the base and target models into a weighted ensemble $\overline{\mathcal{S}}_{A^k} = \sum^j_{p=1} w_p \mathcal{S}_{k,p}$.

Relative landmark-weighting. These per-algorithm ensembles are similar to [1], although we weight the surrogate $\mathcal{S}_{k,p}$ based on the ability of *all* surrogates $\mathcal{S}_{k,p}$ to predict the relative performance of pairs of configurations (θ_i, θ_j) on the new task. Tasks gain more weight if *all* algorithms behave similarly on them. Consider a pair of configurations (θ_i, θ_j) where $\theta_i, \theta_j \in \Theta_{A^k}$. Similarly to [9, 1], we define a relative landmark function that evaluates whether the relative performance of every pair of observed configurations on a target task t can be accurately predicted by the corresponding surrogate models of another task:

$$\text{RLGP}(f^t, \mathcal{D}^k_t) = \sum^n_{i=1}\sum^n_{j=1} \mathbb{1}(f^t(\theta_i) > f^t(\theta_j) \wedge (y^t_i > y^t_j)) \qquad (2)$$

where $f^t(\theta_i)$ and $f^t(\theta_j)$ $i \neq j$ is the performance predicted by the corresponding surrogate models at each specified point θ_i, θ_j, and y^t_i and y^t_j the actual performance measured on task t. Function $\mathbb{1}(condition)$ returns 1 if the surrogate correctly predicts that θ_i evaluates better than θ_j on task t, and 0, otherwise. To include the uncertainty in the GP surrogates, S samples are drawn from the GP posterior at each past observation [1]. Finally, the weights for each task t are computed as:

$$w_t = \frac{1}{S}\sum^S_{s=1} \text{RLGP}(f^t, \mathcal{D}^k_t) \qquad (3)$$

Selecting the next configuration. 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 $\overline{\mu}(\theta) = \sum^p_{i=1} w_i \mu_i(\theta)$ and $\overline{\sigma^2}(\theta) = \sum^p_{i=1} w^2_i \sigma^2_i(\theta)$ with $p = [1, j]$, thus, being able to apply standard EI. 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 $\mathcal{D}^k_t \cup \{(\theta^k_{new}, y^k_{new})\}$ for each $\overline{\mathcal{S}}_k$. The algorithms are ranked based on the y^k value. After n iterations, we identify $A^*_{\theta_*} = \arg\min_{(\theta^{k*}, y^{k*}) \in \mathcal{D}^{k*}} y^k$.

4 Evaluation

Our main empirical validation is based on a large meta-dataset from [3], comprising 553 OpenML [12] datasets and 11 classification algorithms for which 42000 hyperparameter configurations have been evaluated. We use 50 randomly selected datasets and perform leave-one-data-set-out cross-validation, every time using one left out (target) dataset as the new task and the rest as meta-data. Base GPs are fitted to a set of 50 randomly selected configurations. We use three randomly selected configurations to initialize Bayesian optimization and we evaluate for 20 iterations, similarly to [1]. Bayesian optimization is repeated 20 times for each target dataset. In all experiments, GP regression is done using a Matérn 5/2 kernel for smoothness [10], and the posterior distributions for the kernel

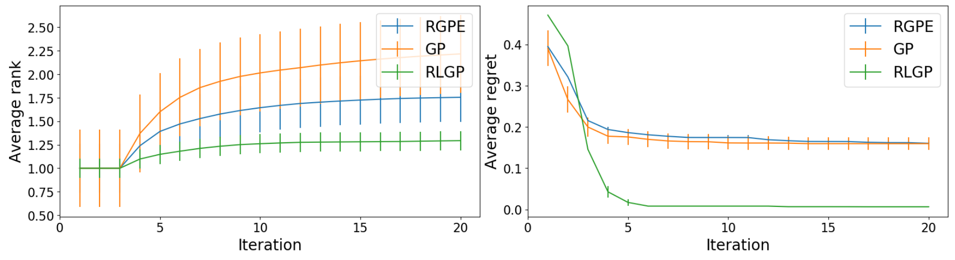

Figure 2: Evolution of average rank and regret using meta-data from 11 WEKA classification algorithms evaluated on 50 OpenML datasets from [3]. Error bars show ± 1 standard deviation.

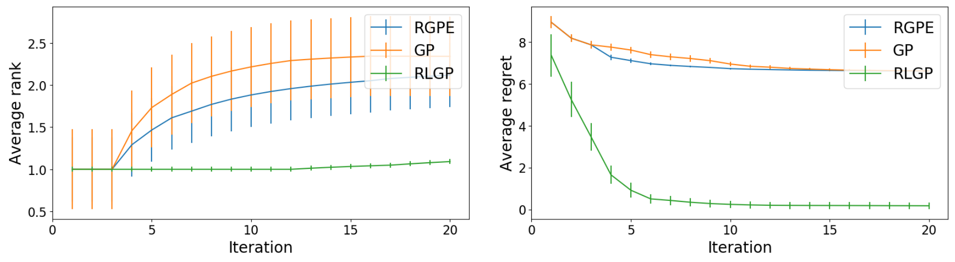

Figure 3: Results on synthetic datasets covering 6 global optimization functions from [6]

hyperparameters is inferred with the NUTS sampler [4]. Bayesian optimization procedures are implemented with the ROBO library [8].

Three methods are benchmarked: RGPE [1] and GP (standard Bayesian optimization) for which we run Bayesian optimization per algorithm, and our Relative Landmark-weighted Gaussian Processes (RLGP). In order to compare them we perform 11 runs, each time predicting the best configuration for a single classification algorithm for RGPE and GP. For RLGP we always predict the best algorithm-configuration from all 11 algorithms. We compute the average rank and regret (the distance to the global minimum at each iteration) of each method over the 11 runs. All strategies are plotted against the number of iterations and lower scores are better. Results are shown in Figure 2. RLGP has the best average rank, which is consistent throughout the optimization process. Our solution is significantly better than RGPE and GP, which seems to indicate that the additional meta-data from other algorithms (used in the weights) provides a significant boost and helps guide the search. Also, we are able to learn good configurations quicker, in fewer than five iterations. We performed a Friedman-Nemenyi post-hoc statistical significance tests and found p-values smaller than 0.001 at a significance level of 0.05. In future work we aim to compare with variants of RGPE operating over combined algorithm hyperparameter spaces, select harder datasets, as well as explore at how many iterations the methods will eventually converge.

Figure 3 shows results obtained over a set of synthetic datasets covering six very different global optimization functions from [6]. We observe that again, RLGP consistently outperforms both RGPE and GP. Further details are available in the Supplementary Materials.

5 Conclusion

We present a novel meta-learning approach that leverages Bayesian optimization to select a machine learning algorithm as well as its hyperparameters for a given task. Compared to the state-of-the-art ranking-weighted Gaussian process ensembles (RGPE) [1], our method enables meta-learning across algorithms and also scales better to many algorithms. Compared to Active testing [9], we do not need to discretize the configuration space, enabling more hyperparameter options to be tried. We achieve this by learning ensembles of algorithm and task-specific surrogate models, under the assumption that such surrogate modes are simpler and more transferable. The method significantly outperforms both RGPE and normal GPs on both real-world and synthetic datasets.

References

[1] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. “Scalable Meta-Learning for Bayesian Optimization”. In: *arXiv e-prints* (Feb. 2018). arXiv: `1802.02219 [stat.ML]`. URL: `https://arxiv.org/abs/1802.02219`.

[2] Matthias Feurer et al. “Efficient and Robust Automated Machine Learning”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 2755–2763. URL: `http://dl.acm.org/citation.cfm?id=2969442.2969547`.

[3] 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. Montr&#233;al, Canada: Curran Associates Inc., 2018, pp. 3352–3361. URL: `http://dl.acm.org/citation.cfm?id=3327144.3327254`.

[4] Matthew D. Hoffman and Andrew Gelman. “The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1593–1623.

[5] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, eds. *Automated Machine Learning: Methods, Systems, Challenges*. In press, available at http://automl.org/book. Springer, 2018.

[6] Momin Jamil and Xin-She Yang. “A Literature Survey of Benchmark Functions For Global Optimization Problems”. In: *arXiv e-prints* (Aug. 2013). arXiv: `1308.4008 [cs.AI]`. URL: `https://arxiv.org/abs/1308.4008`.

[7] Donald R. Jones, Matthias Schonlau, and William J. Welch. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* 13.4 (Dec. 1998), pp. 455–492. ISSN: 1573-2916. DOI: `10.1023/A:1008306431147`. URL: `https://doi.org/10.1023/A:1008306431147`.

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

[9] 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, MLDM’12*. Springer-Verlag, 2012, pp. 117–131.

[10] Bobak Shahriari et al. “Taking the Human Out of the Loop: A Review of Bayesian Optimization”. In: *Proceedings of the IEEE* 104 (2016), pp. 148–175.

[11] Joaquin Vanschoren. “Meta-Learning: A Survey”. In: *arXiv e-prints* (Oct. 2018). arXiv: `1810.03548 [cs.LG]`. URL: `https://arxiv.org/abs/1810.03548`.

[12] Joaquin Vanschoren et al. “OpenML: Networked Science in Machine Learning”. In: *SIGKDD Explorations* 15.2 (2013), pp. 49–60. DOI: `10.1145/2641190.2641198`. URL: `http://doi.acm.org/10.1145/2641190.2641198`.

[13] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. “Scalable Gaussian process-based transfer surrogates for hyperparameter optimization”. In: *Machine Learning* 107.1 (Jan. 2018), pp. 43–78. ISSN: 1573-0565. DOI: `10.1007/s10994-017-5684-y`. URL: `https://doi.org/10.1007/s10994-017-5684-y`.

[14] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. “Two-Stage Transfer Surrogate Model for Automatic Hyperparameter Optimization”. In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Paolo Frasconi et al. Cham: Springer International Publishing, 2016, pp. 199–214. ISBN: 978-3-319-46128-1.

Supplementary Materials

A 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:

$$RL(\theta_i, \theta_j, t) = M(\theta_i, t) \geq M(\theta_j, t) \tag{4}$$

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:

$$\text{RL}(\theta_i, \theta_j, f^t) = 1(f^t(\theta_i) > f^t(\theta_j) \wedge (y_i^t > y_j^t)) \tag{5}$$

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_i^t and y_j^t the actual performance measures $M(\theta_i, t), M(\theta_i, t)$. Then, Function $1(condition)$ 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:

$$\text{RL}(\theta_i, \theta_j, f^{t_{new}}) = 1(f_{-i}^{t_{new}}(\theta_i) > f_{-i}^{t_{new}}(\theta_j) \wedge (y_i^{t_{new}} > y_j^{t_{new}})) \tag{6}$$

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:

$$\text{RLGP}(f^t, \mathcal{D}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \text{RL}(\theta_i, \theta_j, f^t) \tag{7}$$

Finally, a model is weighted by its ability to predict relative performance of configurations on the new task. Similarly to [1], 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:

$$w_i = \frac{1}{S} \sum_{s=1}^{S} \text{RLGP}(f^{t_i}, \mathcal{D}) \tag{8}$$

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. **Sim** is a variant that uses [9] 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:

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

$$Sim(t, t_{new}) = \frac{2c - 1}{2n^2 - 1} \tag{9}$$

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

However, we find that **Sim** does not outperform our base algorithm.

B Extended Evaluation Results

B.1 Synthetic Functions

Six functions from [6] are adapted to create artificial data sets, including Alpine1, Alpine2 and their variations, and Ginuta and Griewank [6]. The mathematical notation of each of those functions is shown in Equation 10 and their representation in Figure 5. Parameter $k = 1, 2, ..., 5$ indicates data sets used as past runs for meta-learning, while $k = 0$ is the target data set.

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

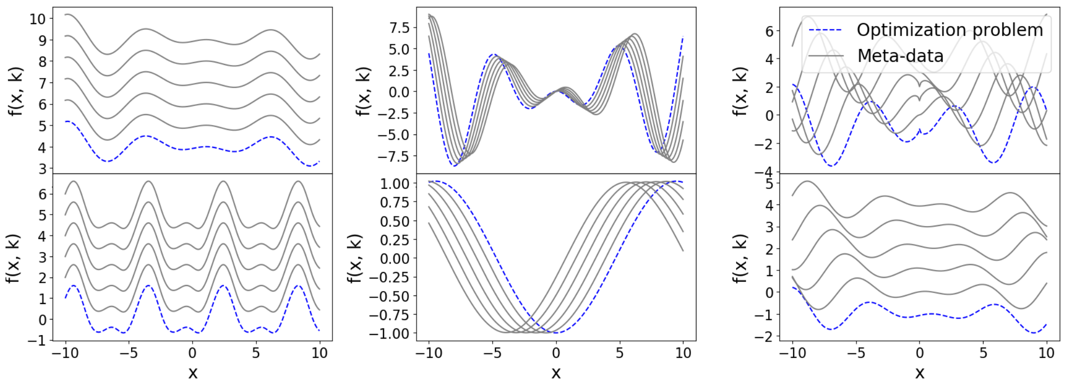

Figure 4: Graphical representation of the synthetic data.

B.2 Real Data

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

C Code

We made code and data available in a GitHub repo: github.com/georgianamanolache/model-selection

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**
Synthetic Function Algorithms	< 0.05	**RGPE**	$< 0.001^{***}$
		GP	$< 0.001^{***}$
WEKA Algorithms	< 0.05	**RGPE**	$< 0.001^{***}$
		GP	$< 0.001^{***}$

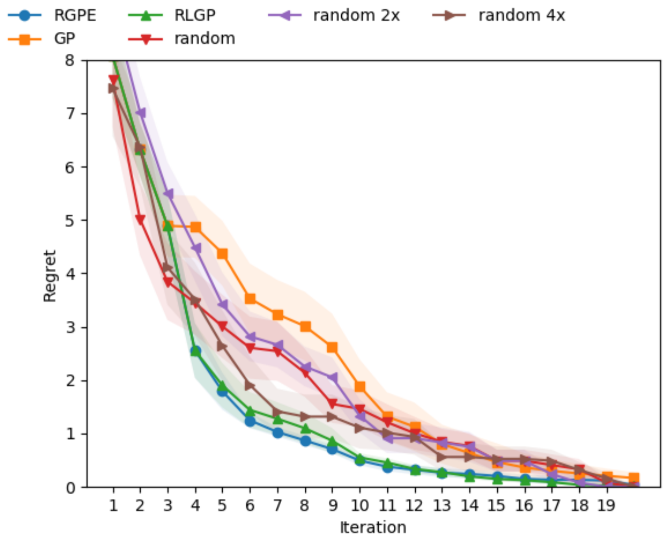

Figure 5: Results on synthetic dataset covering one optimization function from Equation 10.