Hyperparameter Search Space Pruning – A New Component for Sequential Model-Based Hyperparameter Optimization

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab
Universitätsplatz 1, 31141 Hildesheim, Germany
{wistuba,schilling,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. The optimization of hyperparameters is often done manually or exhaustively but recent work has shown that automatic methods can optimize hyperparameters faster and even achieve better final performance. Sequential model-based optimization (SMBO) is the current state of the art framework for automatic hyperparameter optimization. Currently, it consists of three components: a surrogate model, an acquisition function and an initialization technique. We propose to add a fourth component, a way of pruning the hyperparameter search space which is a common way of accelerating the search in many domains but yet has not been applied to hyperparameter optimization. We propose to discard regions of the search space that are unlikely to contain better hyperparameter configurations by transferring knowledge from past experiments on other data sets as well as taking into account the evaluations already done on the current data set.
Pruning as a new component for SMBO is an orthogonal contribution but nevertheless we compare it to surrogate models that learn across data sets and extensively investigate the impact of pruning with and without initialization for various state of the art surrogate models. The experiments are conducted on two newly created meta-data sets which we make publicly available. One of these meta-data sets is created on 59 data sets using 19 different classifiers resulting in a total of about 1.3 million experiments. This is by more than four times larger than all the results collaboratively collected by OpenML.

1 Introduction

Most machine learning algorithms depend on hyperparameters that need to be tuned. In contrast to model parameters, hyperparameters are not estimated during the learning process but have to be set before. Since the hyperparameter tuning often decides whether the performance of an algorithm is state of the art or just moderate, the task of hyperparameter optimization is as important as developing new models [2,7,20,22,25]. Typical hyperparameters are for example the trade-off parameter C of a support vector machine or the regularization constant of a Tikhonov-regularized model. Taking a step further, the chosen model

as well as preprocessing steps can be considered as hyperparameters [25]. Then, hyperparameter optimization not only involves model selection but also model class selection, choice of learning algorithms and preprocessing.

The conventional way of hyperparameter optimization is a combination of manual search with a grid search. This is an exhaustive search in the hyperparameter space which involves multiple training of the model. For high-complex hyperparameter spaces or large data sets this becomes infeasible. Therefore, methods to accelerate the process of hyperparameter optimization are currently an interesting topic for researchers [3,22,25]. Sequential model-based optimization (SMBO) [15] is a black-box optimization process and has proven to be effective in accelerating the hyperparameter optimization process. SMBO is based on a surrogate model that approximates the response function of a data set for given hyperparameters such that sequentially possibly interesting hyperparameter configurations can be evaluated.

Recent work tries to transfer knowledge about the hyperparameter space from past experiments to a new data set [1,24,29]. They motivate this idea by assuming that regions of the hyperparameter space that perform well for few data sets likely contain promising hyperparameter configurations for new data sets.

1.1 Our Contributions

The SMBO framework currently has at most three components. First, the surrogate model that predicts the performance for each possible hyperparameter configuration. Secondly, the acquisition function which uses the surrogate model to propose the next hyperparameter configuration to evaluate. These are the two mandatory components. The third optional component is some initialization technique which usually starts which a hyperparameter configuration that has proven to be good on many data sets [9,11]. We propose to add a fourth component which is orthogonal to all the others. Our idea is to reduce the hyperparameter search space by using knowledge from past experiments to discard regions that are very likely not interesting. This avoids that the acquisition function chooses hyperparameter configurations in these regions because of high uncertainty and therefore avoids unnecessary function evaluations.

Additionally, we created two meta-data sets and make them publicly available. One is a meta-data set created by running a kernel support vector machine on 50 different data sets with 288 different hyperparameter configurations resulting into 14,000 meta-instances. The second is a large scale meta-data set created by using 19 different classifiers provided by Weka [13] on 59 data sets. In total 1,290,389 meta-instances were created such that the number of runs is by more than 4 times larger than the number of runs collaboratively collected by OpenML [26].

2 Related Work

Pruning is a well known technique to accelerate the search in several domains. Thus, for example, various pruning techniques are applied to the minimax algorithm such as the killer heuristic or null move pruning [8]. Branch-and-Bound [18] is a pruning technique that is applied in the domain of operations research for discrete and combinatorial optimization problems and is very common for NP-hard optimization problems [17]. Nevertheless, we are not aware of any published work that is trying to prune the search space in the SMBO framework for hyperparameter optimization.

Since pruning as proposed by us is some way of transferring knowledge from past experiments to a new experiment, other techniques that try exactly the same are the closest related work but as we will see, orthogonal to our contribution. One common and easy way to use experience in the hyperparameter optimization domain is to define an initialization, a sequence of hyperparameter configurations that are chosen first. These are usually those hyperparameter configurations that performed best on average across data sets [9,11]. The second and last method to do so is by using the surrogate model. Instead of learning the surrogate model only on the new data set, the surrogate model is learned across all data sets [1,24,29]. We want to highlight that all these three possibilities are not mutually exclusive and can be combined and thus these ideas are orthogonal to each other.

Leite et al. [19] propose a similar distance function between data sets as we use. But they propose a hyperparameter selection strategy that is limited to the hyperparameter configurations that have been seen on the meta-training data.

Furthermore, there also exist strategies to optimize hyperparameters that are based on optimization techniques from artificial intelligence such as tabu search [4], particle swarm optimization [12] and evolutionary algorithms [10] as well as gradient-based optimization techniques [6] designed for SVMs.

3 Background

3.1 The Formal Setup

A machine learning algorithm $\mathcal{A}_\lambda$ is a mapping $\mathcal{A}_\lambda : \mathcal{D} \rightarrow \mathcal{M}$ where $\mathcal{D}$ is the set of all data sets, $\mathcal{M}$ is the space of all models and $\lambda \in \Lambda$ is the chosen hyperparameter configuration with $\Lambda = \Lambda_1 \times \ldots \times \Lambda_p$ being the p-dimensional hyperparameter space. The learning algorithm estimates a model $M_\lambda \in \mathcal{M}$ that minimizes a regularized loss function $\mathcal{L}$ (e.g. misclassification rate):

$$\mathcal{A}_\lambda \left(D^{(train)} \right) := \arg \min_{M_\lambda \in \mathcal{M}} \mathcal{L} \left(M_\lambda, D^{(train)} \right) + \mathcal{R} \left(M_\lambda \right) \quad . \tag{1}$$

Then, the task of *hyperparameter optimization* is finding the optimal hyperparameter configuration λ^* using a validation set i.e.

$$\lambda^* := \arg \min_{\lambda \in \Lambda} \mathcal{L} \left(\mathcal{A}_\lambda \left(D^{(train)} \right), D^{(valid)} \right) := \arg \min_{\lambda \in \Lambda} f_D \left(\lambda \right) \quad . \tag{2}$$

3.2 Sequential Model-based Optimization

Exhaustive hyperparameter search methods such as grid search are becoming more and more expensive. Data sets are growing, models are getting more complex and have high-dimensional hyperparameter spaces. Sequential model-based optimization (SMBO) [15] is a black-box optimization framework that replaces the time-consuming function f to evaluate with a cheap-to-evaluate surrogate function Ψ that approximates f. With the help of an acquisition function such as expected improvement [15] it sequentially chooses new points such that a balance between exploitation and exploration is found and f is optimized. In our scenario evaluating f is equivalent to learning a model on some training data for a given hyperparameter configuration and estimating the performance of this model on a hold-out data set.

Algorithm 1 outlines the SMBO framework. It starts with an observation history $\mathcal{H}$ that equals the empty set in cases where no knowledge from past experiments is used [2,14,22] or is non-empty in cases where past experiments are used [1,24,29]. First, the optimization process can be initialized. Then, the surrogate model Ψ is fitted to $\mathcal{H}$ where Ψ can be any regression model. Since the acquisition function usually needs to assess prediction uncertainty of the surrogate, common choices are Gaussian processes [1,22,24,29] or ensembles such as random forests [14]. The acquisition function chooses the next candidate to evaluate. A common choice for the acquisition function is expected improvement [15] but further acquisition functions exist such as probability of improvement [15], the conditional entropy of the minimizer [27] or a criterion based on multi-armed bandits [23]. The evaluated candidate is finally added to the set of observations. After T-many SMBO iterations, the best currently found hyperparameter configuration is returned.

Line 6 is our proposed addition to the SMBO framework. Selecting the identity function as *prune* results in the typical SMBO framework. In the next section we propose a more suitable pruning function.

Algorithm 1 Sequential Model-based Optimization

Input: Hyperparameter space Λ, observation history $\mathcal{H}$, number of iterations T, acquisition function a, surrogate model Ψ, initial hyperparameter configurations $\Lambda^{(\text{init})}$.
Output: Best hyperparameter configuration found.

1: **for** $\lambda \in \Lambda^{(\text{init})}$ **do**
2: Evaluate $f(\lambda)$
3: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\lambda, f(\lambda))\}$
4: **for** $t = \left|\Lambda^{(\text{init})}\right| + 1$ to T **do**
5: Fit Ψ to $\mathcal{H}$
6: $\Lambda^{(\text{pruned})} \leftarrow \text{prune}(\Lambda)$
7: $\lambda \leftarrow \arg\max_{\lambda \in \Lambda^{(\text{pruned})}} a(\lambda, \Psi)$
8: Evaluate $f(\lambda)$
9: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\lambda, f(\lambda))\}$
10: **return** $\arg\max_{(\lambda, f(\lambda)) \in \mathcal{H}} f(\lambda)$

4 Pruning the Search Space

The idea of pruning is to consider only a subset of the hyperparameter configuration space Λ to avoid unnecessary function evaluations in regions where we do not expect any improvements. It is obvious that if it is possible to identify regions that are for sure not of interest without evaluating any point in this region highly accelerates the hyperparameter optimization. We propose to predict the potential of regions by transferring knowledge from past experiments. The key idea is that similar data sets to the new data set have similar or even the same regions that are not interesting and therefore not worth investigating.

4.1 Formal Description

We define a region R by its center $\lambda \in \Lambda$ and diameter $\delta \in \mathbb{R}^p,\ \delta > \mathbf{0}$. The potential of this region after t trials on the new data set $D^{(new)}$ is defined by

$$\text{potential}\left(R = (\lambda, \delta), \Lambda_t\right) := \sum_{D' \in \mathcal{N}\left(D^{(test)}\right)} \tilde{f}_{D'}(\lambda) - \max_{\lambda' \in \Lambda_t} \tilde{f}_{D'}(\lambda') \tag{3}$$

where Λ_t is the set of already evaluated hyperparameter configurations on $D^{(new)}$ and $\mathcal{N}\left(D^{(new)}\right)$ is the set of data sets that are closest to the new data set. $\tilde{f}_D$ is the normalized version of the response function f_D of data set D. f_D is scaled to the interval $[0, 1]$ such that each data set has the same influence on the potential. Thus, the potential is the predicted improvement when choosing λ over the hyperparameter configurations already evaluated. Since f_D is not fully observed for $D \in \mathcal{D}$, where $\mathcal{D}$ is the meta-training set, we approximate $\tilde{f}_D$ with a plug-in estimator $\hat{y}_D$. We use a Gaussian process [21] that is trained on all normalized meta-instances of a data set such that we get for each training data set a plug-in estimator

$$\tilde{f}_D(\lambda) \sim \hat{y}_D(\lambda) := \mathcal{GP}\left(m_D(\lambda), k_D(\lambda, \lambda')\right) \tag{4}$$

where we define m_D as the mean function and k_D as the covariance function of $\tilde{f}_D$. As a kernel function we are using the squared exponential kernel

$$k(\lambda, \lambda') := \exp\left(-\frac{\|\lambda - \lambda'\|_2^2}{2\sigma^2}\right) \quad . \tag{5}$$

This allows to estimate $\tilde{f}_D$ for arbitrary hyperparameter configurations. Then, we replace the definition from Equation 3 with

$$\text{potential}\left(R = (\lambda, \delta), \Lambda_t\right) := \sum_{D' \in \mathcal{N}\left(D^{(new)}\right)} \hat{y}_{D'} - \max_{\lambda' \in \Lambda_t} \hat{y}_{D'}(\lambda') \quad . \tag{6}$$

To estimate the nearest neighbors of the new data set $D^{(new)}$ we have to define a distance function between data sets. A common choice for this is the

Euclidean distance with respect to the meta-features [1,29]. Since we experienced better results with a distance function based on rank correlation metrics such as the Kendall tau rank correlation coefficient [16], we are using following distance function

$$\mathrm{KTRC}\left(D_1, D_2, \Lambda_t\right) := \frac{\sum_{\lambda_1,\lambda_2 \in \Lambda_t} \mathbb{I}\left(\hat{y}_{D_1}(\lambda_1) > \hat{y}_{D_1}(\lambda_2) \oplus \hat{y}_{D_2}(\lambda_1) > \hat{y}_{D_2}(\lambda_2)\right)}{(|\Lambda_t|-1)|\Lambda_t|} \tag{7}$$

where $\oplus$ is the symbol for an exclusive or.

Algorithm 2 Prune

Input: Hyperparameter space Λ, observation history $\mathcal{H}$, region radius δ, fraction of the pruned space ν.

Output: Pruned hyperparameter space $\Lambda^{\text{pruned}} \subseteq \Lambda$.

1: Estimate the most similar data sets of the new data set $\mathcal{N}\left(D^{new}\right)$ using Equation 7.
2: Estimate the set Λ' containing the $\nu\,|G|$ hyperparameter configurations $\lambda' \in G \subset \Lambda$ with little potential using Equation 6.
3: $\Lambda^{(pruned)} := \{\lambda \in \Lambda \mid \mathrm{dist}\left(\lambda, \lambda'\right) > \delta,\ \lambda' \in \Lambda'\}$.
4: **return** $\Lambda^{(pruned)} \cup \{\lambda \in \Lambda \mid \mathrm{dist}\left(\lambda, \lambda'\right) \leq \delta, \lambda' \in \Lambda_t\}$

Algorithm 2 summarizes the pruning function. Line 1 estimates the k most similar data sets which we know from past experiments using the KTRC distance function defined in Equation 7. In Line 2 the potential of hyperparameter configurations are estimated using the plug-in estimators (Equation 6) on a fine grid $G \subset \Lambda$. The $\nu\,|G|$ hyperparameter configurations with little potential define regions where no improvement is predicted. Hence, the pruned hyperparameter space is defined as the set of hyperparameter configurations that are not within an δ-region of these low-potential hyperparameter configurations (Line 3). Additionally, the hyperparameter configurations that are within a δ-region of already evaluated hyperparameter configurations are added (Line 4). The intuition here is that since we have already observed an evaluation in this region, the acquisition function will not choose a hyperparameter combination close to these points for exploration but only for exploitation. Hence, no evaluations will be done by the standard SMBO framework without a very likely improvement. For the distance function between hyperparameter configurations we need to consider one that does not take discrete variables into account. Obviously, the loss does not change smoothly when changing a categorical variable that e.g. indicates which algorithm was chosen. Therefore, we define the distance function in Algorithm 2 as

$$\mathrm{dist}\left(\lambda, \lambda'\right) := \begin{cases} \infty & \text{if } \lambda \text{ and } \lambda' \text{ differ in a categorical variable} \\ \|\lambda - \lambda'\| & \text{otherwise} \end{cases}. \tag{8}$$

5 Experimental Evaluation

First, we will introduce the reader to the state of the art tuning strategies which are used to evaluate pruning. Then, the evaluation metrics are defined and the meta-data sets are introduced. Finally, the results are presented.

5.1 Tuning Strategies

We want to give a short introduction to all the tuning strategies we will consider in our experiments. We are considering both strategies that are using no knowledge from previous experiments and those that do.

Random Search This is the only strategy that is not using any surrogate model. Hyperparameter configurations are sampled uniformly at random. This is a common strategy in cases where a grid search is not possible. Bergstra and Bengio [3] have shown that this is very effective for hyperparameters with low effective dimensionality.

Independent Gaussian Process (I-GP) This tuning strategy uses a Gaussian process [22] with squared-exponential kernel as a surrogate model. It only uses knowledge from the current data set and is not using any knowledge from previous experiments.

Independent Random Forest (I-RF) Next to Gaussian processes, random forests are the most widely used surrogate models [14] and hence we are using them in our experiments. Like the independent Gaussian process, the I-RF does not use any knowledge from previous experiments.

Sequential Model-based Algorithm Configuration++ (SMAC++) SMAC [14] is a tuning strategy that is based on a random forest as a surrogate model without background knowledge of previous experiments. SMAC++ is our extension to SMAC. SMAC++ is using the typical SMBO framework but the random forest is also trained on the meta-training data.

Surrogate Collaborative Tuning (SCoT) SCoT [1] uses a Gaussian process with squared-exponential kernel with automatic relevance determination and is trained on hyperparameter observations of previous experiments evaluated on other data sets and the few knowledge achieved on the new data set. An SVMRank is learned on the data set and its predictions are used instead of the hyperparameter performances. Bardenet et al. [1] argue that this overcomes the problem of having data sets with different scales of hyperparameter performances. In the original work it was proposed to use an RBF kernel for SVMRank. For reasons of computational complexity we follow the lead of Yogatama and Mann [29] and use a linear kernel instead.

Gaussian Process with MKL (MKL-GP) Similarly to Bardenet et al. [1], Yogatama and Mann [29] propose to use a Gaussian process as a surrogate model for the SMBO framework. Instead of using SVMRank to deal with the different scales, they are adapting the mean of the Gaussian process, accordingly. Additionally, they are using a specific kernel, a linear combination of an SE-ARD kernel with a kernel modelling the distance between data sets.

Optimal This is an artificial tuning strategy that always evaluates the best hyperparameter configuration and is added to plots for orientation purposes.

Kernel parameters are learned by maximizing the marginal likelihood on the meta-training set [21]. Hyperparameters of the tuning strategies are optimized in a leave-one-out cross-validation on the meta-training set.

The results reported are the average of at least ten repetitions. For the strategies with random initialization (Random, I-GP, I-RF), the mean of 1000 repetitions is reported.

5.2 Evaluation Metrics

In our experiments we are using three different evaluation metrics which we will explain here in detail.

Average Rank The *average rank among different hyperparameter tuning strategies* or for short simply *average rank* is a relative metric between different tuning strategies. The tuning strategies are ranked by the best hyperparameter configuration that they have found so far, ties are solved by granting them the average rank. If we have for example four different tuning strategies that have found hyperparameter configurations that achieve an accuracy of 0.78, 0.77, 0.77 and 0.76, respectively, then the ranking is 1, 2.5, 2.5 and 4.

Normalized Average Loss The disadvantage of the average rank is that it gives no information about by which margin the found hyperparameters of one tuning strategy are better than another and it will vary when strategies are added or are removed. One metric that overcomes this disadvantage is the *normalized average loss*. In our experiments we will consider only classification problems such that $f_D(\lambda)$ is the accuracy on data set D using hyperparameter configuration λ. Since the scale of f_D varies for different D we normalize f_D between 0 and 1 such that every data set has the same impact on the evaluation metric. Thus, the normalized average loss at iteration t is defined as

$$\mathrm{NAL}(\mathcal{D}, \Lambda_t) := \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} 1 - \frac{\max_{\lambda \in \Lambda_t} f_D(\lambda) - \min_{\lambda \in \Lambda} f_D(\lambda)}{\max_{\lambda \in \Lambda} f_D(\lambda) - \min_{\lambda \in \Lambda} f_{D(\lambda)}} \quad . \tag{9}$$

Average Hyperparameter Rank The average hyperparameter rank is another way to overcome the disadvantages of the average rank. Compared to the average rank it is not ranking the tuning strategies but ranking the hyperparameter configurations. Let $r_D(\lambda)$ be the rank of the hyperparameter configuration λ on data set D, then the average hyperparameter rank is defined as

$$\mathrm{AHR}(\mathcal{D}, \Lambda_t) := \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \min_{\lambda \in \Lambda_t} r_D(\lambda) - 1 \ . \tag{10}$$

5.3 Meta-Data Sets

The SVM meta-data set was created by using 50 classification data sets chosen at random. All instances were merged in cases where splits were already given, shuffled and split into 80% train and 20% test. We then used a support vector machine (SVM) [5] to create the meta-instances. We trained the SVM using three different kernels (linear, polynomial and Gaussian) and estimated the labels of the meta-instances by evaluating the trained model on the test split. The hyperparameter space dimension is six, three dimensions for binary features that indicate which kernel was chosen, one for the trade-off parameter C, one for the degree of the polynomial kernel d and the width γ of the Gaussian kernel. If the hyperparameter is not involved, e.g. the degree if we are using the linear kernel, it was set to 0. The test accuracy was precomputed on a grid $C \in \left\{2^{-5}, \ldots, 2^{6}\right\}$, $d \in \{2, \ldots, 10\}$, $\gamma \in \left\{10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20, 50, 10^{2}, 10^{3}\right\}$ resulting into 288 meta-instances per data set. Since meta-features are a vital part for many surrogate models and mandatory for SCoT and MKL-GP, we added the meta-features that were used by [1,29] to our meta-data. First , we extracted the number of training instances n, the number of classes c and the number of predictors m. The final meta-features are c, $\log(m)$ and $\log(n/m)$ scaled to [0.1].

The Weka meta-data set was created using 59 classification data sets which were preprocessed like the classification data sets used for the SVM meta-data set. We used 19 different Weka classifiers [13] and produced 21,871 hyperparameter configurations per data set. The dimension of the hyperparameter space is 102 including the indicator variables for the classifier. Thus, this meta-data set focuses stronger on the model class selection. Overall, this meta-data set contains 1,290,389 instances. In comparison, OpenML [26] has collaboratively collected 344,472 runs.[1]

The meta-data sets are available on our supplementary website together with a visualization of the meta-data as well as more details about how the meta-data sets were created and a detailed list which data sets were used [28].

5.4 Hyperparameter Optimization for SVMs

To show that the proposed plug-in estimators work (Equation 4), we did not use all 288 hyperparameter configurations for training but only 50 per data set. The

[1] Status 2015/03/27 by `http://openml.org`

evaluation is nevertheless done on all 288 of the new data set. We choose G to contain these 288 configurations and fixed $|\mathcal{N}(D^{new})| = 2$, $\nu = 1 - |G|^{-1}$ and δ such that the two closest neighbored hyperparameter configurations of the test region are within δ-distance.

We want to conduct two different experiments. First, we want to compare a surrogate model with pruning to current state of the art tuning strategies. We once again want to stress that pruning in the SMBO framework is an orthogonal contribution such that these results are actually of minor interest. Second, we want to compare different surrogate models with and without pruning or initialization. Pruning is a useful contribution as long as it does not worsen the optimization speed in general and accelerates it in some cases.

Figure 1 shows the results of the comparison of pruning to the current state of the art method. As a surrogate model we decided to choose the Gaussian process that is *not* learned across data sets since it is the most common and simple surrogate model. Surprisingly, the pruning alone with the Gaussian process is able to outperform all the competitor strategies with respect to all three evaluation metrics.

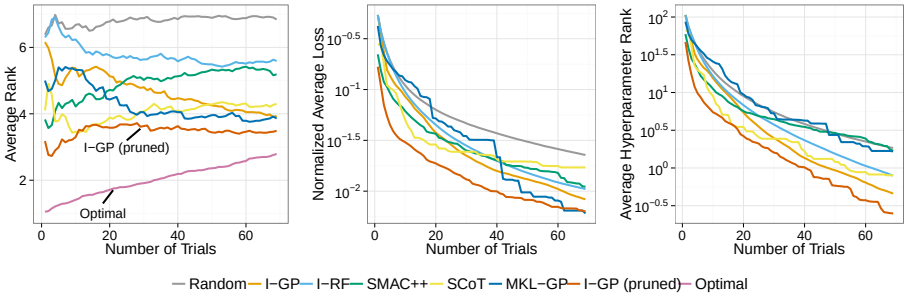

Fig. 1. Pruning is an orthogonal contribution to the SMBO framework. Nevertheless, we compare a pruned independent Gaussian process to many current state of the art tuning strategies without pruning.

Figures 2 to 6 show the results of different surrogate models. We distinguish four different cases: i) only the surrogate model, ii) the surrogate model with pruning, iii) the surrogate model with three steps of initialization and iv) the surrogate model with three steps of initialization and pruning. Figures 2 and 3 show the results for the surrogate models that do not learn across data sets and the remaining three Figures show the results for the surrogate models that learn across data sets. Our expectation before the experiments were that the lift is higher i) for the experiments without initialization and ii) for the experiments with the surrogate models that do not learn across data sets. The reason for this is simple. An initialization is a fixed policy that proposes hyperparameter configurations that has been good on average while pruning discards regions that were

not useful. Thus, pruning will also have an effect of initialization. The difference between initialization and pruning is that initialization proposes a specific hyperparameter while pruning reduces the full hyperparameter space to a set of good hyperparameter configurations and pruning is applied at each iteration and not just for the initial iterations. Additionally, pruning is a way to transfer knowledge between data sets such that those strategies that do not use this knowledge at all benefit more and are prevented from conducting unnecessary exploration queries.

This is exactly what the results of the experiments show. The SMBO experiments with pruning have comparable good starting points like those with initialization. If we compare the results of the independent Gaussian process and random forest for the setting with only initialization with the one with only pruning, we clearly see the unnecessary exploration queries after a good start. The setting with both initialization and pruning does not suffer from this problem and thus is clearly the best strategy. This effect is weaker for the surrogate models that are learned across data sets in Figures 4 and 6. Only for SCoT (Figure 5) pruning does not accelerate the hyperparameter optimization on *this* meta-data set but it also does not worsen it. Table 1 shows the results for all evaluation metrics and surrogate models.

The reader may notice two important things. First, the results in the plot will always converge to the same value across different tuning strategies if you allow only enough trials. Second, even a very small improvement of the performance *just* by choosing a better hyperparameter configurations is already a success especially since this optimization is usually limited in time. This little improvement may result in significantly better results for a new model compared to the competitors or decides whether a research challenge will be won or not.

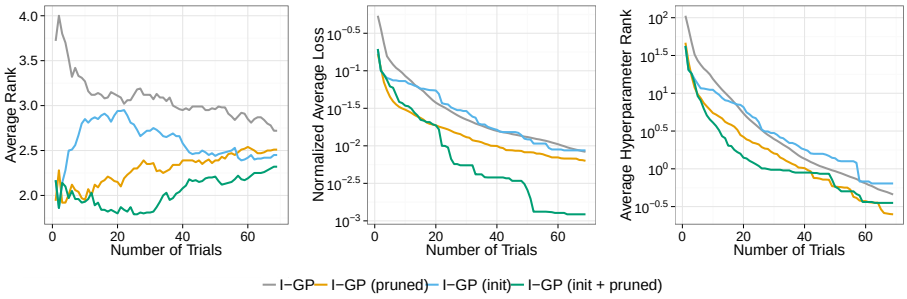

Fig. 2. Average rank, normalized average loss and average hyperparameter rank for I-GP on the SVM meta-data set.

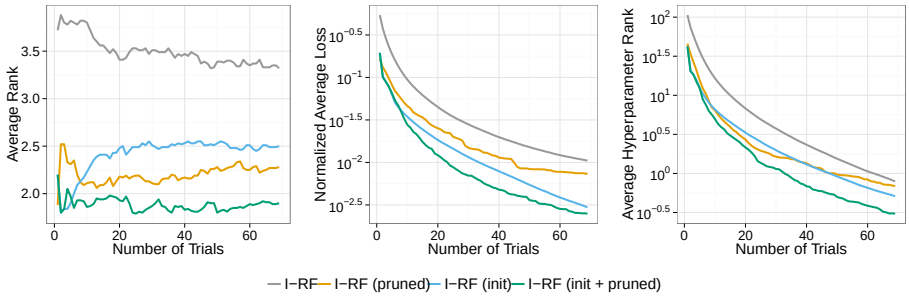

Fig. 3. Average rank, normalized average loss and average hyperparameter rank for I-RF on the SVM meta-data set.

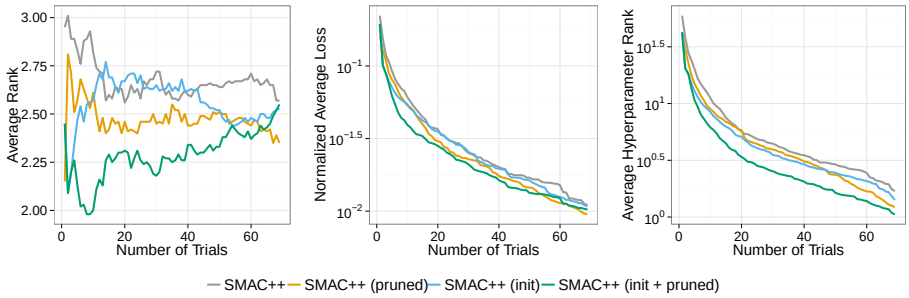

Fig. 4. Average rank, normalized average loss and average hyperparameter rank for SMAC++ on the SVM meta-data set.

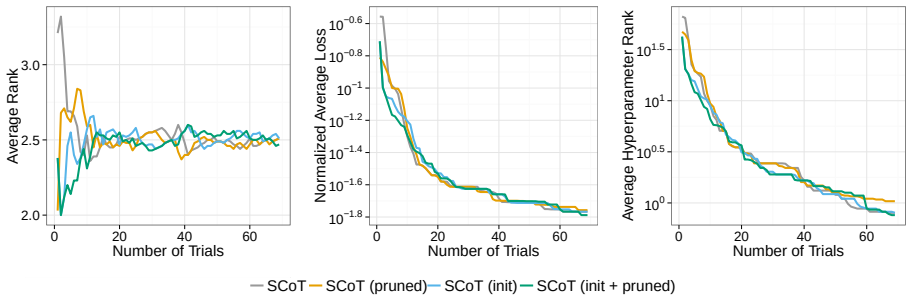

Fig. 5. Average rank, normalized average loss and average hyperparameter rank for SCoT on the SVM meta-data set.

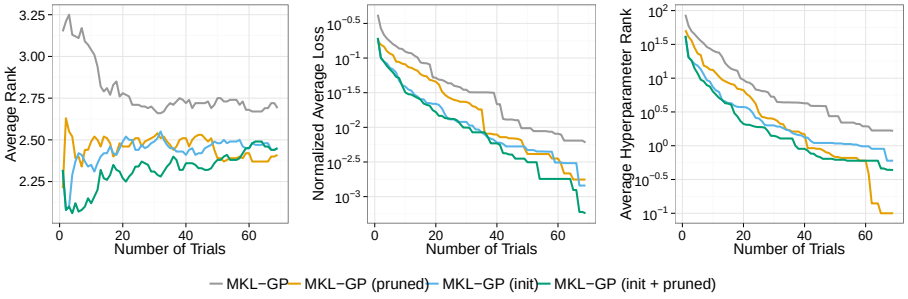

Fig. 6. Average rank, normalized average loss and average hyperparameter rank for MKL-GP on the SVM meta-data set.

Table 1. Average rank, normalized average loss and average hyperparameter rank after 30 trials on the SVM meta-data set. Best results are bold.

I-GP	no pruning/init	pruned	init	init + pruned
Average Rank@30	3.12	2.35	2.72	**1.81**
NAL@30	0.0224	0.0131	0.0291	**0.0055**
AHR@30	3.48	2.60	3.98	**1.97**
I-RF	no pruning/init	pruned	init	init + pruned
Average Rank@30	3.51	2.11	2.51	**1.87**
NAL@30	0.0281	0.0149	0.0116	**0.0070**
AHR@30	4.75	2.64	2.98	**2.14**
SMAC++	no pruning/init	pruned	init	init + pruned
Average Rank@30	2.72	2.45	2.65	**2.18**
NAL@30	0.0251	0.0228	0.0256	**0.0210**
AHR@30	5.42	4.92	4.52	**3.53**
SCoT	no pruning/init	pruned	init	init + pruned
Average Rank@30	2.55	2.55	2.47	**2.43**
NAL@30	0.0244	0.0244	**0.0237**	**0.0237**
AHR@30	3.44	3.44	3.02	**2.90**
MKL-GP	no pruning/init	pruned	init	init + pruned
Average Rank@30	2.68	2.52	2.49	**2.31**
NAL@30	0.0349	0.0232	0.0120	**0.0099**
AHR@30	6.30	3.48	3.00	**2.40**

5.5 Hyperparameter Optimization for Weka

In the last chapter, we have seen little improvement in cases where an initialization is combined with surrogate models that are learning across data sets. We expect pruning to be useful in two scenarios: if i) the dimensionality of the hyperparameter space is very high and ii) the meta-data set is too large such that surrogate models that are learning across data sets are no longer a cost-efficient alternative to evaluating the true function. Since most surrogate models are based on Gaussian processes, a further problem is storing the kernel matrix. In our next meta-data set we are using more than a million meta-instances which result into a kernel matrix of dimensions $10^6 \times 10^6$ which needs 8 TB of memory for storing it.

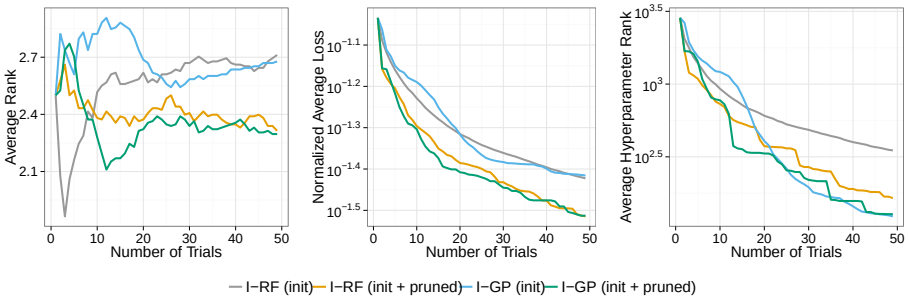

Fig. 7. Average rank, normalized average loss and average hyperparameter rank for I-RF and I-GP on the Weka meta-data set.

For the Weka meta-data set we conducted a similar experiment as for the SVM meta-data set. Due to the size we restricted ourselves to the tuning strategies that do not learn across data sets. Previously, we have seen that a tuning strategy without initialization and pruning is outperformed by a large margin by the same strategy only using pruning. Hence, we show here only the comparison between the strategy i) only using an initialization step and ii) using both initialization and pruning. Figure 7 concludes our experiments. As we have seen on the SVM meta-data set, pruning again indicates that it is a useful addition to the SMBO framework by further accelerating the hyperparameter optimization.

6 Conclusion and Future Work

We propose pruning as an orthogonal contribution the the SMBO framework and show in elaborated experiments on two different data set that it accelerates the hyperparameter optimization in most cases and in the worst case does not worsen it. It can be especially considered for tuning strategies that do not use information from the past for the surrogate model. Additionally, we created a

new meta-data set which is the largest to the best of our knowledge with about four times more experiments than OpenML and make it publicly available.

Acknowledgments. The authors gratefully acknowledge the co-funding of their work by the German Research Foundation (DFG) under grant SCHM 2583/6-1.

References

1. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. pp. 199–207 (2013)
2. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain. pp. 2546–2554 (2011)
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305 (Feb 2012)
4. Cawley, G.: Model selection for support vector machines via adaptive step-size tabu search. In: Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 434-437, Prague, Czech Republic, April 2001. pp. 434–437. Prague, Czech Republic (April 2001)
5. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
6. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S.: Choosing multiple parameters for support vector machines. Machine Learning 46(1-3), 131–159 (2002)
7. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011. pp. 215–223 (2011)
8. David-Tabibi, O., Netanyahu, N.S.: Verified null-move pruning. ICGA Journal 25(3), 153–161 (2002)
9. Feurer, M., Springenberg, J.T., Hutter, F.: Using meta-learning to initialize bayesian optimization of hyperparameters. In: ECAI workshop on Metalearning and Algorithm Selection (MetaSel). pp. 3–10 (2014)
10. Friedrichs, F., Igel, C.: Evolutionary tuning of multiple svm parameters. Neurocomput. 64, 107–117 (Mar 2005)
11. Gomes, T.A.F., Prudêncio, R.B.C., Soares, C., Rossi, A.L.D., Carvalho, A.C.P.L.F.: Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing 75(1), 3–13 (2012)
12. Guo, X.C., Yang, J.H., Wu, C.G., Wang, C.Y., Liang, Y.C.: A novel ls-svms hyper-parameter selection based on particle swarm optimization. Neurocomput. 71(16-18), 3211–3215 (Oct 2008)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explor. Newsl. 11(1), 10–18 (Nov 2009)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proceedings of the 5th International Conference on Learning and Intelligent Optimization. pp. 507–523. LION'05, Springer-Verlag, Berlin, Heidelberg (2011)

15. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. of Global Optimization 13(4), 455–492 (Dec 1998)
16. Kendall, M.G.: A New Measure of Rank Correlation. Biometrika 30(1/2), 81–93 (Jun 1938)
17. Land, A.H., Doig, A.G.: An Automatic Method for Solving Discrete Programming Problems. Econometrica 28, 497–520 (1960)
18. Lawler, E.L., Wood, D.E.: Branch-And-Bound Methods: A Survey. Operations Research 14(4), 699–719 (1966)
19. Leite, R., Brazdil, P., Vanschoren, J.: Selecting classification algorithms with active testing. In: Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition. pp. 117–131. MLDM'12, Springer-Verlag, Berlin, Heidelberg (2012)
20. Pinto, N., Doukhan, D., DiCarlo, J.J., Cox, D.D.: A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS Computational Biology 5(11), e1000579 (2009), PMID: 19956750
21. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
22. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 2960–2968 (2012)
23. Srinivas, N., Krause, A., Seeger, M., Kakade, S.M.: Gaussian process optimization in the bandit setting: No regret and experimental design. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10). pp. 1015–1022. Omnipress (2010)
24. Swersky, K., Snoek, J., Adams, R.P.: Multi-task bayesian optimization. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 2004–2012 (2013)
25. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: 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. pp. 847–855. KDD '13, ACM, New York, NY, USA (2013)
26. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: Networked science in machine learning. SIGKDD Explorations 15(2), 49–60 (2013)
27. Villemonteix, J., Vazquez, E., Walter, E.: An informational approach to the global optimization of expensive-to-evaluate functions. Journal of Global Optimization 44(4), 509–534 (2009)
28. Wistuba, M.: Supplementary website: `http://hylap.org/publications/Hyperparameter-Search-Space-Pruning` (Jun 2015)
29. Yogatama, D., Mann, G.: Efficient transfer learning method for automatic hyperparameter tuning. In: International Conference on Artificial Intelligence and Statistics (AISTATS 2014) (2014)