Learning Data Set Similarities for Hyperparameter Optimization Initializations

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. Current research has introduced new automatic hyperparameter optimization strategies that are able to accelerate this optimization process and outperform manual and grid or random search in terms of time and prediction accuracy. Currently, meta-learning methods that transfer knowledge from previous experiments to a new experiment arouse particular interest among researchers because it allows to improve the hyperparameter optimization.
In this work we further improve the initialization techniques for sequential model-based optimization, the current state of the art hyperparameter optimization framework. Instead of using a static similarity prediction between data sets, we use the few evaluations on the new data sets to create new features. These features allow a better prediction of the data set similarity. Furthermore, we propose a technique that is inspired by active learning. In contrast to the current state of the art, it does not greedily choose the best hyperparameter configuration but considers that a time budget is available. Therefore, the first evaluations on the new data set are used for learning a better prediction function for predicting the similarity between data sets such that we are able to profit from this in future evaluations. We empirically compare the distance function by applying it in the scenario of the initialization of SMBO by meta-learning. Our two proposed approaches are compared against three competitor methods on one meta-data set with respect to the average rank between these methods and show that they are able to outperform them.

1 Introduction

Most machine learning algorithms depend on hyperparameters and their optimization is an important part of machine learning techniques applied in practice. Automatic hyperparameter tuning is catching more and more attention by the machine learning community for two simple but important reasons. Firstly, the omnipresence of hyperparameters affects the whole community such that everyone is affected by the time-consuming task of optimizing hyperparameters either by manually tuning them or by applying a grid search. Secondly, in many cases the final hyperparameter configurations decides whether an algorithm is state of the art or just moderate such that the task of hyperparameter optimization is as important as developing new models [2,4,13,17,20]. Furthermore, hyperparameter optimization has shown to be able to also automatically perform algorithm and preprocessing selection by considering this as a further hyperparameter [20].

Sequential model-based optimization (SMBO) [9] is the current state of the art for hyperparameter optimization and has proven to outperform alternatives such as grid search or random search [17,20,2]. Recent research try to improve the SMBO framework by applying meta-learning on the hyperparameter optimization problem. The key concept of meta-learning is to transfer knowledge gained for an algorithm on past experiments on different data sets to new experiments. Currently, two different, orthogonal ways of transferring this knowledge exist. One possibility is to initialize SMBO by using hyperparameter configurations that have been best on previous experiments [5,6]. Another possibility is to use surrogate models that are able to learn across data sets [1,19,23].

We improve the former strategy by using an adaptive initialization strategy. We predict the similarity between data sets by using meta-features and features that express the knowledge gathered about the new data set so far. Having a more accurate approximation of the similarity between data sets, we are able to provide a better initialization. We provide empirical evidence that the new features provide better initializations in two different experiments.

Furthermore, we propose an initialization strategy that is based on the active learning idea. We try to evaluate hyperparameter configurations that are non-optimal for the short term but promise better results than choosing greedily the hyperparameter configuration that will provide the best result in expectation. To the best of our knowledge, we are the first that propose this idea in context of hyperparameter optimization for SMBO.

2 Related Work

Initializing hyperparameter optimization through meta-learning was proven to be effective [5,7,15,6]. Reif et al. [15] suggests to choose those hyperparameter configurations for a new data set that were best on a similar data set in the context of evolutionary parameter optimization. Here, the similarity was defined through the distance among meta-features, descriptive data set features. Recently, Feurer et al. [5] followed their lead and proposed the same initialization for sequential model-based optimization (SMBO), the current state of the art hyperparameter optimization framework. Later, they extended their work by learning a regression model on the meta-features that predicts the similarity between data sets [6].

Learning a surrogate model, the component in the SMBO framework that tries to predict the performance for a specific hyperparameter configuration on a data set, that is not only learned on knowledge of the new data set but additionally across knowledge from experiments on other data sets [1,19,23,16] is another option to transfer knowledge as well as pruning [22]. This idea is related but orthogonal to our work and can benefit from a good initialization and is no replacement for a good initialization strategy.

We propose to add features based on the performance of an algorithm on a data set for a specific hyperparameter configuration. Pfahringer et al. [12] propose to use landmark features. These features are estimated by evaluating simple algorithms on the data sets of the past and new experiment. In comparison to our features, these features are no by-product of the optimization process but have to be computed and hence need additional time. Even though these are simple algorithms, these features are problem-dependent (classification, regression, ranking, structured prediction all need their own landmark features). Furthermore, "simple" classifiers such as nearest neighbors as proposed by the authors can become very time-consuming for large data sets which are those data sets we are interested in.

Relative landmarks proposed by Leite et al. [10] are not and cannot be used as meta-features. They are used within the hyperparameter optimization strategy that is used instead of SMBO but are similar in that way that they are also given as a by-product and are computed using the relationship between hyperparameter configurations on each data set.

3 Background

In this section the hyperparameter optimization problem is formally defined. For the sake of completeness, also the sequential model-based optimization framework is presented.

3.1 Hyperparameter Optimization Problem Setup

A machine learning algorithm $\mathcal{A}_{\boldsymbol{\lambda}}$ is a mapping $\mathcal{A}_{\boldsymbol{\lambda}} : \mathcal{D} \to \mathcal{M}$ where $\mathcal{D}$ is the set of all data sets, $\mathcal{M}$ is the space of all models and $\boldsymbol{\lambda} \in \Lambda$ 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 $M_{\boldsymbol{\lambda}} \in \mathcal{M}$ that minimizes the objective function that linearly combines the loss function $\mathcal{L}$ and the regularization term $\mathcal{R}$:

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

Then, the task of *hyperparameter optimization* is finding the hyperparameter configuration $\boldsymbol{\lambda}^*$ that minimizes the loss function on the validation data set i.e.

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

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) [9] 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 [9], sequentially new points are chosen such that a balance between exploitation and exploration is met and f is optimized. In our scenario, evaluating f is equivalent to learning a machine learning algorithm on some training data for a given hyperparameter configuration and estimate the models performance 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 [2,8,17] or is non-empty in cases where past experiments are used [1,19,23] or SMBO has been initialized [5,6]. First, the surrogate model Ψ is fitted to $\mathcal{H}$ where Ψ can be any regression model. Since the acquisition function a usually needs some certainty about the prediction, common choices are Gaussian processes [1,17,19,23] or ensembles such as random forests [8]. The acquisition function chooses the next candidate to evaluate. A common choice for the acquisition function is expected improvement [9] but further acquisition functions exist such as probability of improvement [9], the conditional entropy of the minimizer [21] or a multi-armed bandit based criterion [18]. The evaluated candidate is finally added to the set of observations. After T-many SMBO iterations, the best currently found hyperparameter configuration is returned.

Algorithm 1 Sequential Model-based Optimization

```
Input: Hyperparameter space $\Lambda$, observation history $\mathcal{H}$, number of iterations $T$, acquisition function $a$, surrogate model $\Psi$.
Output: Best hyperparameter configuration found.
1: for $t = 1$ to $T$ do
2:     Fit $\Psi$ to $\mathcal{H}$
3:     $\boldsymbol{\lambda}_* \leftarrow \arg\max_{\boldsymbol{\lambda}_* \in \Lambda} a\left(\boldsymbol{\lambda}_*, \Psi\right)$
4:     Evaluate $f\left(\boldsymbol{\lambda}_*\right)$
5:     $\mathcal{H} \leftarrow \mathcal{H} \cup \left\{\left(\boldsymbol{\lambda}_*, f\left(\boldsymbol{\lambda}_*\right)\right)\right\}$
6: return $\arg\min_{\left(\boldsymbol{\lambda}^*, f\left(\boldsymbol{\lambda}^*\right)\right) \in \mathcal{H}} f\left(\boldsymbol{\lambda}^*\right)$
```

4 Adaptive Initialization

Recent initialization techniques for sequential model-based optimization compute a static initialization hyperparameter configuration sequence [5,6]. The advantage of this idea is that during the initialization there is no time overhead for computing the next hyperparameter configuration. The disadvantage is that knowledge gained during the initialization about the new data set is not used for further initialization queries. Hence, we propose to use an adaptive initialization technique. Firstly, we propose to add some additional meta-features generated from this knowledge, which follows the idea of landmark features [12] and relative landmarks [10], respectively. Secondly, we apply the idea of active learning and try to choose the hyperparameter configurations that will allow to learn a precise ranking of data sets with respect to their similarity to the new data set.

4.1 Adaptive Initialization Using Additional Meta-Features

Feurer et al. [5] propose to improve the SMBO framework by an initialization strategy as shown in Algorithm 2. The idea is to use the hyperparameter configurations that have been best on other data sets. Those hyperparameter configurations are ranked with respect to the predicted similarity to the new data set D_{new} for which the best hyperparameter configuration needs to be found. The true distance function $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ between data sets is unknown such that Feurer et al. [5] propose to approximate it by $\hat{d}\left(\mathbf{m}_i, \mathbf{m}_j\right) = \left\|\mathbf{m}_i - \mathbf{m}_j\right\|_p$ where $\mathbf{m}_i$ is the vector of meta-features of data set D_i. In their extended work [6], they propose to use a random forest to learn $\hat{d}$ using training instances of the form $\left(\left(\mathbf{m}_i, \mathbf{m}_j\right)^T, d\left(D_i, D_j\right)\right)$. This initialization does not consider the performance of the hyperparameters configurations already evaluated on the new data set.

We propose to keep Feurer's initialization strategy [6] untouched and only add additional meta-features that capture the information gained on the new data set. Meta-features as defined in Equation 3 are added to the set of meta-features for all hyperparameter configurations $\boldsymbol{\lambda}_k, \boldsymbol{\lambda}_l$ that are evaluated on the new data set. The symbol $\oplus$ denotes an exclusive or. An additional difference is that now after each step the meta-features and the model $\hat{d}$ needs to be updated (before Line 2).

$$m_{D_i,D_j,\boldsymbol{\lambda}_k,\boldsymbol{\lambda}_l} = \mathbb{I}\left(f_{D_i}\left(\boldsymbol{\lambda}_k\right) > f_{D_i}\left(\boldsymbol{\lambda}_l\right) \oplus f_{D_j}\left(\boldsymbol{\lambda}_k\right) > f_{D_j}\left(\boldsymbol{\lambda}_l\right)\right) \tag{3}$$

We make here the assumption that the same set of hyperparameter configurations were evaluated across all training data sets. If this is not the case, this problem can be overcome by approximating the respective value by learning surrogate models for the training data sets as well. Since for these data sets much information is available, the prediction will be reliable enough. For simplicity, we assume that the former is the case.

The target $\mathbf{d}$ can be any similarity measure that reflects the true similarity between data sets. Feurer et al. [6] propose to use the Spearman correlation coefficient while we are using the number of discordant pairs in our experiments, i.e.

$$d\left(D_i, D_j\right) := \frac{\sum_{\boldsymbol{\lambda}_k,\boldsymbol{\lambda}_l \in \Lambda} \mathbb{I}\left(f_{D_i}\left(\boldsymbol{\lambda}_k\right) > f_{D_i}\left(\boldsymbol{\lambda}_l\right) \oplus f_{D_j}\left(\boldsymbol{\lambda}_k\right) > f_{D_j}\left(\boldsymbol{\lambda}_l\right)\right)}{|\Lambda|\left(|\Lambda| - 1\right)} \tag{4}$$

where Λ is the set of hyperparameter configurations observed on the training data sets. This change will have no influence on the prediction quality for the traditional meta-features but is better suited for the proposed landmark features in Equation 3.

Algorithm 2 Sequential Model-based Optimization with Initialization

```
Input: Hyperparameter space $\Lambda$, observation history $\mathcal{H}$, number of iterations $T$, acquisition function $a$, surrogate model $\Psi$, set of data sets $\mathcal{D}$, number of initial hyperparameter configurations $I$, prediction function for distances between data sets $\hat{d}$.
Output: Best hyperparameter configuration found.
1: for $i = 1$ to $I$ do
2:     Predict the distances $\hat{d}(D_j, D_{new})$ for all $D_j \in \mathcal{D}$.
3:     $\boldsymbol{\lambda}_* \leftarrow$ Select best hyperparameter configuration on the i-th closest data set.
4:     Evaluate $f(\boldsymbol{\lambda}_*)$
5:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\lambda}_*, f(\boldsymbol{\lambda}_*))\}$
6: return SMBO($\Lambda, \mathcal{H}, T - I, a, \Psi$)
```

4.2 Adaptive Initialization Using Active Learning

We propose to extend the method from the last section by investing few initialization steps by carefully selecting hyperparameter configurations that will lead to good additional meta-features and provide a better prediction function $\hat{d}$. An additional meta-feature is useful if the resulting regression model $\hat{d}$ predicts the distances of the training data sets to the new data set such that the ordering with respect to the predicted distances reflects the ordering with respect to the true distances. If I is the number of initialization steps and $K < I$ is the number of steps to choose additional meta-features, then the K hyperparameter configurations need to be chosen such that the precision at $I - K$ with respect to the ordering is optimized. The precision at n is defined as

$$\text{prec@n} := \frac{\left|\{\text{n closest data sets to } D_{new} \text{ wrt. } d\} \cap \left\{\text{n closest data sets to } D_{new} \text{ wrt. } \hat{d}\right\}\right|}{n} \quad (5)$$

Algorithm 3 presents the method we used to find the best first K hyperparameter configurations. In a leave-one-out cross-validation over all training data sets $\mathcal{D}$ the pair of hyperparameter configurations $(\boldsymbol{\lambda}_j, \boldsymbol{\lambda}_k)$ is sought that achieves the best precision at $I - K$ on average (Lines 1 to 7). Since testing all different combinations of K different hyperparameter configurations is too expensive, only the best pair is searched. The remaining $K - 2$ hyperparameter configurations are greedily added to the final set of initial hyperparameter configurations Λ_{active} as described in Lines 8 to 15. The hyperparameter configuration is added to Λ_{active} that performs best on average with all hyperparameter configurations chosen so far. After choosing K hyperparameter configurations as described in Algorithm 3, the remaining $I - K$ hyperparameter configurations are chosen as described in Section 4.1. The time needed for computing the first K hyperparameter highly depends on the size of Λ. To speed up the process, we reduced Λ to those hyperparameter configurations that have been best on at least one training data set.

Algorithm 3 Active Initialization

Input: Set of training data sets $\mathcal{D}$, set of observed hyperparameter configurations on $\mathcal{D}$ Λ, number of active learning steps K, number of initial hyperparameter configurations I.

Output: List of hyperparameter configurations that will lead to good predictors for $\hat{d}$.

```
 1: for $i = 1$ to $|\mathcal{D}|$ do
 2:   $\mathcal{D}_{test} \leftarrow \{D_i\}$
 3:   $\mathcal{D}_{train} \leftarrow \mathcal{D} \setminus \{D_i\}$
 4:   for all $\boldsymbol{\lambda}_j, \boldsymbol{\lambda}_k \in \Lambda, j \neq k$ do
 5:     Train $\hat{d}$ on $\mathcal{D}_{train}$ using the additional feature $m_{\cdot,\cdot,\boldsymbol{\lambda}_j,\boldsymbol{\lambda}_k}$ (Eq. 3)
 6:     Estimate the precision at $I - K$ of $\hat{d}$ on $\mathcal{D}_{test}$.
 7: $\Lambda_{active} \leftarrow \{\boldsymbol{\lambda}_j, \boldsymbol{\lambda}_k\}$ with highest average precision.
 8: for $k = 1$ to $K - 2$ do
 9:   for $i = 1 \ldots |\mathcal{D}|$ do
10:     $\mathcal{D}_{test} \leftarrow \{D_i\}$
11:     $\mathcal{D}_{train} \leftarrow \mathcal{D} \setminus \{D_i\}$
12:     for all $\boldsymbol{\lambda} \in \Lambda \setminus \Lambda_{active}$ do
13:       Train $\hat{d}$ on $\mathcal{D}_{train}$ using the additional features (Eq. 3) implied by $\Lambda_{active} \cup \{\boldsymbol{\lambda}\}$.
14:       Estimate the precision at $I - K$ of $\hat{d}$ on $\mathcal{D}_{test}$.
15:   Add $\boldsymbol{\lambda}$ with highest average precision to $\Lambda_{active}$.
16: return $\Lambda_{active}$
```

5 Experimental Evaluation

5.1 Initialization Strategies

Following initialization strategies will be considered in our experiments.

Random Best Initialization (RBI) This initialization is a very simple initialization. I training data sets from the training data sets $\mathcal{D}$ are chosen at random and its best hyperparameter configurations are used for the initialization.

Nearest Best Initialization (NBI) This is the initialization strategy proposed by Reif et al. and Feurer et al. [5,15]. Instead of choosing I training data sets at random, they are chosen with respect to the similarity between the meta-features listed in Table 1. Then, like for RBI, the best hyperparameter configurations on these data sets are chosen for initialization.

Predictive Best Initialization (PBI) Feurer et al. [6] propose to learn the distance between data sets using a regression model based on the meta-features. These predicted distances are used to find the most similar data sets to the new data set and like before, the best hyperparameter configurations on these data sets are chosen for initialization.

Adaptive Predictive Best Initialization (aPBI) This is our extension to PBI presented in Section 4.1 that adapts to the new data set during initialization by including the features defined in Equation 3.

Active Adaptive Predictive Best Initialization (aaPBI) Active Adaptive Predictive Best Initialization is described in Section 4.2 and extends aPBI by using the first K steps to choose hyperparameter configurations that will result in promising meta-features. After the first K iterations, it behaves equivalent to aPBI.

Table 1. List of all meta-features used.

Number of Classes	Class Probability Max
Number of Instances	Class Probability Mean
Log Number of Instances	Class Probability Standard Deviation
Number of Features	Kurtosis Min
Log Number of Features	Kurtosis Max
Data Set Dimensionality	Kurtosis Mean
Log Data Set Dimensionality	Kurtosis Standard Deviation
Inverse Data Set Dimensionality	Skewness Min
Log Inverse Data Set Dimensionality	Skewness Max
Class Cross Entropy	Skewness Mean
Class Probability Min	Skewness Standard Deviation

5.2 Meta-Features

Meta-features are supposed to be discriminative for a data set and can be estimated without evaluating f. Many surrogate models [1,19,23] and initialization strategies [5,15] use them to predict the similarity between data sets. For the experiments, the meta-features listed in Table 1 are used. For an in-depth explanation we refer the reader to [1,11].

5.3 Meta-Data Set

For creating the two meta-data sets, 50 classification data sets from the UCI repository are chosen at random. All instances are merged in cases there were already train/test splits, shuffled and split into 80% train and 20% test. A support vector machine (SVM) [3] was used to create the meta-data set. The SVM was trained using three different kernels (linear, polynomial and Gaussian) and the labels of the meta-instances were estimated by evaluating the trained model on the test split. The total hyperparameter dimension is six, three dimensions for indicator variables that indicates which kernel was chosen, one for the trade-off parameter C, one for the width γ of the Gaussian kernel and one for the degree of the polynomial kernel d. If the hyperparameter is not involved, e.g. the degree if the Gaussian kernel was used, it is set to 0. The test accuracy is precomputed on a grid $C \in \{2^{-5}, \ldots, 2^6\}$, $\gamma \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20, 50, 10^2, 10^3\}$ and $d \in \{2, \ldots, 10\}$ resulting into 288 meta-instances per data set. The meta-data sets is extended by the meta-features listed in Table 1.

5.4 Experiments

Two different experiments are conducted. First, state of the art initialization strategies are compared with respect to the average rank after I initial hyperparameter configurations. Second, the long term effect on the hyperparameter optimization is compared. Even though the initial hyperparameter configuration lead to good results after I results, the ultimate aim is to have good results at the end of the hyperparameter optimization after T iterations.

We evaluated all methods in a leave-one-out cross-validation per data set. All data sets but one are used for training and the data set not used for training is the new data set. The results reported are the average over 100 repetitions. Due to the randomness, we used 1,000 repetitions whenever RBI was used.

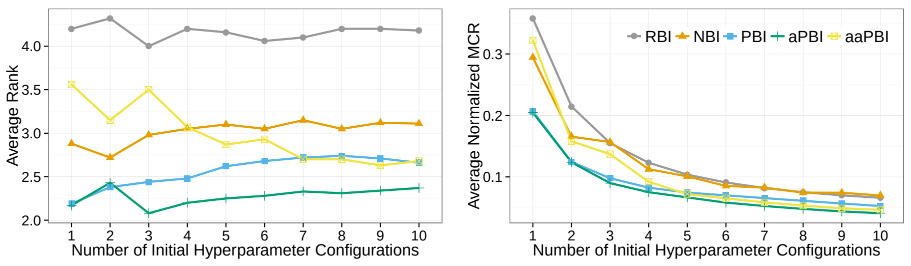

Fig. 1. Five different tuning strategies are compared. Our strategy aPBI is able to outperform the state of the art. Our second strategy aaPBI is executed under the assumption that it has 10 initial steps. The first steps are used for active learning and hence the bad results in the beginning are not surprising.

Comparison to Other Initialization Strategies For all our experiments 10 initialization steps are used, $I = 10$. In Figure 1 the different initialization strategies are compared with respect to the average rank in the left plot. Our initialization strategy aPBI benefits from the new, adaptive features and is able to outperform the other initialization strategies. Our second strategy aaPBI uses three active learning steps, $K = 3$. This explains the behavior in the beginning. After these initial steps it is able to catch up and finally surpass PBI but it is not as good as aPBI. Furthermore, a distance function learned on the meta-features (PBI) provides better results than a fixed distance function (NBI) which confirms the results by Feurer et al. [6]. All initialization strategies are able to outperform RBI which confirms that the meta-features contain information that concludes information about the similarity between data sets. The right plot shows the average misclassification (MCR) rate where the MCR is scaled to 0 and 1 for each data set.

Comparison with Respect to the Long Term Effect To have a look at the long term effect of the initialization strategies, we compare the different initialization strategies in the SMBO framework using two common surrogate models. One is a Gaussian process with a squared exponential kernel with automatic relevance determination [17]. The kernel parameters are estimated by maximizing the marginal likelihood on the meta-training set [14]. The second is a random forest [8]. Again, aPBI provides strictly better results than PBI for both surrogate models and outperforms any other initialization strategy for the Gaussian process. Our alternative strategy aaPBI performs mediocre for the Gaussian process but good for the random forest.

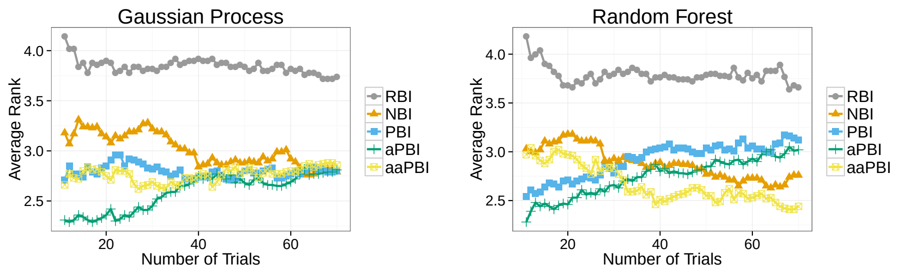

Fig. 2. The effect of the initialization strategies on the further optimization process in the SMBO framework using a Gaussian process (left) and a random forest (right) as a surrogate model is investigated. Our initialization strategy aPBI seems to be strictly better than PBI while aaPBI performs especially well for the random forest in the later phase.

6 Conclusion and Future Work

Predicting the similarity between data sets is an important topic for hyperparameter optimization since it allows to successfully transfer knowledge from past experiments to a new experiment. We have presented an easy way of achieving an adaptive initialization strategy by adding a new kind of landmark features. We have shown for two popular surrogate models that these new features improve over the same strategy without these features. Finally, we introduced a new idea that in contrast to the current methods considers that there is a limit of evaluations. It tries to exploit this knowledge by applying a guided exploitation at first that will lead to worse decisions for the short term but will deliver better results when the end of the initialization is reached. Unfortunately, the results for this method are not fully convincing but we believe that it can be a good idea to choose hyperparameter configurations in a smarter way but always assuming that the next hyperparameter configuration chosen is the last one.

In this work we were able to provide a more accurate prediction of the similarity between data sets and used this knowledge to improve the initialization. Since not only initialization strategies but also surrogate models rely on an exact similarity prediction, we plan to investigate the impact on these models. For example Yogatama and Mann [23] use a kernel that measures the distance between data sets by using the p-norm between meta-features of data sets only. A better distance function may help to improve the prediction and will help to improve the SMBO beyond the initialization.

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

References

1. Bardenet, R., Brendel, M., Kégl, B., Sebag, M.: Collaborative hyperparameter tuning. In: Dasgupta, S., Mcallester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning (ICML-13). vol. 28, pp. 199–207. JMLR Workshop and Conference Proceedings (May 2013)
2. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 24, pp. 2546–2554. Curran Associates, Inc. (2011)
3. 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`
4. Coates, A., Lee, H., Ng, A.: An analysis of single-layer networks in unsupervised feature learning. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, vol. 15, pp. 215–223. JMLR W&CP (2011)
5. 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)
6. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 1128–1135 (2015)
7. Gomes, T.A., Prudêncio, R.B., Soares, C., Rossi, A.L., Carvalho, A.: Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing 75(1), 3 – 13 (2012), brazilian Symposium on Neural Networks (SBRN 2010) International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010)
8. 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)

9. 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)
10. Leite, R., Brazdil, P., Vanschoren, J.: Selecting classification algorithms with active testing. In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition, Lecture Notes in Computer Science, vol. 7376, pp. 117–131. Springer Berlin Heidelberg (2012)
11. Michie, D., Spiegelhalter, D.J., Taylor, C.C., Campbell, J. (eds.): Machine Learning, Neural and Statistical Classification. Ellis Horwood, Upper Saddle River, NJ, USA (1994)
12. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: In Proceedings of the Seventeenth International Conference on Machine Learning. pp. 743–750. Morgan Kaufmann (2000)
13. 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
14. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press (2005)
15. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine Learning 87(3), 357–380 (2012)
16. Schilling, N., Wistuba, M., Drumond, L., Schmidt-Thieme, L.: Hyperparameter Optimization with Factorized Multilayer Perceptrons. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015 (2015)
17. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 25, pp. 2951–2959. Curran Associates, Inc. (2012)
18. 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)
19. Swersky, K., Snoek, J., Adams, R.P.: Multi-task bayesian optimization. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 26, pp. 2004–2012. Curran Associates, Inc. (2013)
20. 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)
21. 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)
22. Wistuba, M., Schilling, N., Schmidt-Thieme, L.: Hyperparameter Search Space Pruning - A New Component for Sequential Model-Based Hyperparameter Optimization. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015 (2015)
23. Yogatama, D., Mann, G.: Efficient transfer learning method for automatic hyperparameter tuning. In: International Conference on Artificial Intelligence and Statistics (AISTATS 2014) (2014)