Learning Hyperparameter Optimization Initializations

Martin Wistuba
Information Systems and
Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
wistuba@ismll.uni-hildesheim.de

Nicolas Schilling
Information Systems and
Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
schilling@ismll.uni-hildesheim.de

Lars Schmidt-Thieme
Information Systems and
Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

***Abstract*—Hyperparameter optimization is often done manually or by using a grid search. However, recent research has shown that automatic optimization techniques are able to accelerate this optimization process and find hyperparameter configurations that lead to better models. Currently, transferring knowledge from previous experiments to a new experiment is of particular interest because it has been shown that it allows to further improve the hyperparameter optimization.**

We propose to transfer knowledge by means of an initialization strategy for hyperparameter optimization. In contrast to the current state of the art initialization strategies, our strategy is neither limited to hyperparameter configurations that have been evaluated on previous experiments nor does it need meta-features. The initial hyperparameter configurations are derived by optimizing for a meta-loss formally defined in this paper. This loss depends on the hyperparameter response function of the data sets that were investigated in past experiments. Since this function is unknown and only few observations are given, the meta-loss is not differentiable. We propose to approximate the response function by a differentiable plug-in estimator. Then, we are able to learn the initial hyperparameter configuration sequence by applying gradient-based optimization techniques.

Extensive experiments are conducted on two meta-data sets. Our initialization strategy is compared to the state of the art for initialization strategies and further methods that are able to transfer knowledge between data sets. We give empirical evidence that our work provides an improvement over the state of the art.

I. Introduction

Tuning hyperparameters is an omnipresent problem for practitioners and researchers in the data mining domain. The performance of an algorithm highly depends on an adequate hyperparameter configuration choice which requires the necessary expertise in the respective domain. In contrast to model parameters, hyperparameters are often tuned manually in combination with a grid or random search [1]. Recent research proposes automatic hyperparameter optimization techniques that can find good hyperparameters in less time than usual optimization methods and are even able to find better hyperparameters configurations than human experts [2], [3], [4], [5]. Taking a step further, the chosen model as well as preprocessing steps can be considered as hyperparameters [6]. Then, hyperparameter optimization includes also model and preprocessing selection. Hence, automatic hyperparameter optimization has become an interesting topic for researchers. Sequential model-based optimization (SMBO) [7], originally a framework for black-box optimization, has been successfully applied for hyperparameter optimization [5] and is the current state of the art. SMBO is based on a surrogate model that approximates the response function of a data set. The response function maps a hyperparameter configuration and a data set to the evaluation measure on a hold-out data set. Then, sequentially, possibly interesting hyperparameter configurations are evaluated and the newly acquired knowledge can be used for further hyperparameter configuration acquisitions. Recent approaches for improving SMBO try to transfer knowledge from previous tuning processes to the current one. This is either done using an initialization for SMBO [8], a specific surrogate model that is learning across experiments [9], [10], [11] or search space pruning [12].

Recent work tries to transfer knowledge about the hyperparameter space from past experiments to a new data set. This is either done by using surrogate models that consider past experiments [9], [13], [10] or by using the information on past experiments to initialize the new search [8], [14], [15]. The motivation behind this approach is that subsets of the hyperparameter space that are good on few data sets are likely good candidates for others. While the first approach is limited to the application in the SMBO framework, the second can be used for any hyperparameter optimization strategies.

A. Our Contributions

We propose to transfer knowledge of hyperparameter configurations from past experiments to new data sets using an initialization strategy that can be used for, but is not limited to, the SMBO framework. We mathematically formalize the problem of hyperparameter optimization and derive a hyperparameter optimization loss. Since this meta-loss is yet not differentiable, we propose to use a differentiable plug-in estimator. Given this estimator, initial hyperparameter configurations can be learned by gradient-based optimization techniques that minimize the meta-loss. In contrast to existing initialization strategies, our strategy is neither limited to hyperparameter configurations that have been evaluated on previous data sets nor does it depend on meta-features. Additionally, we can show that a direct optimization for the right loss leads to better initializations and ultimately to an accelerated hyperparameter optimization.

II. Related Work

Initializing hyperparameter optimization through meta-learning has been proven to be effective [8], [14], [15]. Reif et al. [15] suggest 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. [8] followed their lead and proposed the same initialization for sequential model-based optimization (SMBO), the current state of the art hyperparameter optimization framework. In comparison, we propose to learn the optimal initial hyperparameter configurations. Our method is not limited to hyperparameter configurations that have been tried in past experiments such as the state of the art and does not depend on meta-features.

Recently, a further way of transferring knowledge from past experiments to a new experiment was proposed that is limited to the application in the SMBO framework. The surrogate model, the component in the SMBO framework that tries to predict the performance for a specific hyperparameter configuration on a data set, is not learned on knowledge of the new data set only but additionally on knowledge from experiments on other data sets [9], [13], [10], [11]. This work is related but orthogonal to our work. It can benefit from a good initialization and is no replacement for a good initialization strategy.

III. Background

In this section the hyperparameter optimization problem is formally defined. For the sake of completeness, the sequential model-based optimization framework and the concept of Gaussian processes are presented.

A. Hyperparameter Optimization Problem Setup

A machine learning algorithm $\mathcal{A}_{\boldsymbol{\lambda}}$ is a mapping $\mathcal{A}_{\boldsymbol{\lambda}} : \mathcal{D} \rightarrow \mathcal{M}$ where $\mathcal{D}$ is the set of all data sets, $\mathcal{M}$ is the space of all models and $\boldsymbol{\lambda} \in \Lambda$ 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 a regularized loss function $\mathcal{L}$ (e.g. misclassification rate):

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

Then, the task of *hyperparameter optimization* is to find the optimal hyperparameter configuration $\boldsymbol{\lambda}^*$ using a validation set i.e.

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

For demonstration purposes, in the remaining sections we consider the problem of tuning the hyperparameters of a classifier. Thus, f will be the misclassification rate. This is no limitation but shall help the reader to understand the concepts at a concrete example.

B. Sequential Model-based Optimization

Naïve hyperparameter optimization techniques such as grid search or random search [1] are becoming more and more expensive. This has several reasons. For one thing data sets are getting larger, for another thing models are getting more complex and have high-dimensional hyperparameter spaces (e.g. Deep Belief Networks [16] and Convolutional Neural Networks [17]). Sequential model-based optimization (SMBO) [7] was proposed as a black-box optimization framework. The idea is to replace the expensive-to-evaluate function f to minimize with a cheap-to-evaluate surrogate function Ψ that approximates f. An acquisition function (e.g. expected improvement [7]) is used to tackle the exploitation-exploration dilemma. Candidate points of the search space Λ are evaluated and sequentially chosen 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 model's performance on a hold-out data set.

Algorithm 1 outlines the SMBO framework. The observation history $\mathcal{H}$ is either the empty set in cases where no knowledge from past experiments is used [2], [18], [5] or contains information gathered in past experiments on other data sets [9], [13], [10]. The SMBO process can be initialized, usually by applying meta-knowledge [8]. In each iteration, the surrogate model Ψ is fitted to the instances in the observation history $\mathcal{H}$. In theory, Ψ can be any regression model but since the acquisition function a depends on the prediction and the uncertainty about the prediction, common choices for the surrogate model are Gaussian processes [9], [5], [13], [10] or ensembles such as random forests or neural networks [18], [11]. The candidate point that maximizes the acquisition function is chosen as the next candidate to evaluate. A common choice for the acquisition function is the expected improvement [7]. Further acquisition functions are the probability of improvement [7], the conditional entropy of the minimizer [19] or a multi-armed bandit based criterion [20]. 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 $\Psi$, initial hyperparameter configurations $\Lambda_I = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_I\}$.
Output: Best hyperparameter configuration found.
1: for $i = 1$ to $I$ do
2:     Evaluate $f(\boldsymbol{\lambda}_i)$ (See Eq. (2))
3:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\lambda}_i, f(\boldsymbol{\lambda}_i))\}$
4: for $t = I + 1$ to $T$ do
5:     Fit $\Psi$ to $\mathcal{H}$
6:     $\boldsymbol{\lambda}_* \leftarrow \arg\max_{\boldsymbol{\lambda}_* \in \Lambda} a(\boldsymbol{\lambda}_*, \Psi)$
7:     Evaluate $f(\boldsymbol{\lambda}_*)$
8:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\boldsymbol{\lambda}_*, f(\boldsymbol{\lambda}_*))\}$
9: return $\arg\min_{(\boldsymbol{\lambda}^*, f(\boldsymbol{\lambda}^*)) \in \mathcal{H}} f(\boldsymbol{\lambda}^*)$
```

C. Gaussian Processes

Gaussian processes [21] are a conventional prediction model for state of the art hyperparameter selection strategies [9], [5], [13], [10]. In this work, they are not only used to be the surrogate model for the new, unknown data set but also for the training data sets. Thus, for the sake of completeness, we recapture that definition.

Given a training data set $D = \{(\boldsymbol{\lambda}_i, f_i)\,,\ i = 1 : N\}$ where $f_i := f(\boldsymbol{\lambda}_i)$ and $\boldsymbol{\lambda}_i \in \mathbb{R}^P$. We want to predict $\mathbf{f}_*$ of a testing data set $\boldsymbol{\Lambda}_*$ of size $N_* \times P$. Then, $\mathbf{f}$ and $\mathbf{f}_*$ are jointly Gaussian distributed

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_y & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix}\right) \tag{3}$$

where $\mathbf{K}_y := \mathbf{K} + \sigma_y^2 \mathbf{I}_N$, $\mathbf{K} := \kappa(\boldsymbol{\Lambda}, \boldsymbol{\Lambda})$, σ_y is a noise hyperparameter, $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ the identity matrix and $\kappa : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$ a kernel function. Furthermore, $\mathbf{K}_* := \kappa(\boldsymbol{\Lambda}, \boldsymbol{\Lambda}_*) \in \mathbb{R}^{N \times N_*}$ and $\mathbf{K}_{**} := \kappa(\boldsymbol{\Lambda}_*, \boldsymbol{\Lambda}_*) \in \mathbb{R}^{N_* \times N_*}$. The predicted probability distribution conditioned on previous observations $\mathbf{f}$ of a single instance $\boldsymbol{\lambda}_*$ simplifies to

$$p(f_* | \boldsymbol{\lambda}_*, \boldsymbol{\Lambda}, \mathbf{f}) := \mathcal{N}\left(f_* | \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{f}, k_{**} - \mathbf{k}_*^T \mathbf{K}_y^{-1} \mathbf{k}_*\right). \tag{4}$$

The predicted mean is referred to as $\hat{f}$.

1) Efficient Computations: The most expensive step in training a Gaussian process is the inversion of the kernel matrix. In the SMBO framework an update of the Gaussian process is needed after each iteration and a complete retraining is inefficient. A common trick to decrease the running time for the update step from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$ for a symmetric kernel function is to decompose the kernel matrix using the Cholesky decomposition $\mathbf{K}_y = \mathbf{L}\mathbf{L}^T$. Then, the predicted probability distribution for a single instance $\boldsymbol{\lambda}_*$ is

$$p(f_* | \boldsymbol{\lambda}_*, \boldsymbol{\Lambda}, \mathbf{f}) := \mathcal{N}\left(f_* | \mathbf{k}_*^T \boldsymbol{\alpha}, k_{**} - \mathbf{v}^T \mathbf{v}\right) \tag{5}$$

with $\boldsymbol{\alpha} = \mathbf{L}^T \setminus (\mathbf{L} \setminus \mathbf{f})$ and $\mathbf{v} = \mathbf{L} \setminus \mathbf{k}_*$ where $\setminus$ is the operator for solving an equation system. If a new instance $(\boldsymbol{\lambda}_*, f_*)$ is added, the triangular matrix $\boldsymbol{L}$ is updated via

$$\boldsymbol{L}_{new} := \begin{pmatrix} \boldsymbol{L} & \mathbf{0} \\ \mathbf{l}^T & l_* \end{pmatrix} \tag{6}$$

where $\mathbf{l} = \mathbf{L} \setminus \mathbf{k}_*$ and $l_* = \sqrt{\kappa(\boldsymbol{\lambda}_*, \boldsymbol{\lambda}_*) - \|\mathbf{l}\|^2 + \sigma_y^2}$. Now $\boldsymbol{\alpha}$ can be recomputed as described above.

IV. Learning Initializations

In this paper, the task of initializing hyperparameter optimization strategies is generalized and a novel approach to choose initial hyperparameter configurations is proposed. Instead of choosing from hyperparameter configurations that have been best on previous data sets [8], [14], we directly learn hyperparameter configurations and thus are not limited to hyperparameter configurations that have been evaluated on previous data sets. The idea is to initialize the initial sequence of hyperparameter configurations with promising configurations and further improve them by minimizing the proposed hyperparameter loss. Since this loss is differentiable, the initial sequence of hyperparameter configurations can be learned with gradient-based optimization techniques such as stochastic gradient descent. This loss is derived by formally expressing a useful evaluation measure for hyperparameter optimization in general.

A. Meta-Notation

In the following, the prefix *meta* is used to distinguish between the different learning problems. The traditional learning problem is to learn some parameters $\boldsymbol{\theta}$ on a given data set containing instances with predictors. For the hyperparameter optimization problem one can create *meta-data sets* containing *meta-instances* with *meta-predictors*. A meta-data set contains meta-instances $((\boldsymbol{\lambda}_i, \mathbf{m}_D), f_D(\boldsymbol{\lambda}_i))$ where $f_D(\boldsymbol{\lambda}_i)$ is the target and $(\boldsymbol{\lambda}_i, \mathbf{m}_D)$ are the meta-predictors with data set-dependent meta-features $\mathbf{m}_D$. A meta-data set can contain meta-instances from more than one data set.

In our setting, the hyperparameter response function $f_D : \Lambda \to \mathbb{R}$ returns the misclassification rate after training a specific classifier on a specific data set D using the hyperparameter configuration $\boldsymbol{\lambda} \in \Lambda$. The task is to find the best I initial hyperparameter configurations such that the hyperparameter optimization, that is limited to T evaluations, will find the best hyperparameter configuration for an unknown, new data set D_{new}. To achieve this, meta-instances for other data sets $D \in \mathcal{D}$ are given. We denote the final initial hyperparameter configurations by Λ_I. Let Λ_T be the sequence of the T evaluated hyperparameter configurations on $f_{D_{new}}$.

B. Evaluation Measure for Hyperparameter Optimization

First, a general evaluation measure for hyperparameter optimization is proposed. Then, we discuss a loss function for initialization techniques for hyperparameter optimization. Verbally, a set of T hyperparameter configurations $\Lambda_T^* \subseteq \Lambda$ is sought that minimizes the difference between the global minimum of f_D and the best chosen hyperparameter configuration $\boldsymbol{\lambda}$, i.e.

$$\Lambda_T^* = \underset{\Lambda_T \subset \Lambda}{\arg\min} \min_{\boldsymbol{\lambda} \in \Lambda_T} f_D(\boldsymbol{\lambda}) - f_D^{min},\ |\Lambda_T| = T. \tag{7}$$

It is important to notice that only the best hyperparameter configuration is of importance. The final evaluation measure for a given data set D and a set of hyperparameter configurations Λ_T is the distance to the global minimum (DTM) (Equation (8)).

$$\mathrm{DTM}(\Lambda_T, D) := \min_{\boldsymbol{\lambda} \in \Lambda_T} f_D(\boldsymbol{\lambda}) - f_D^{min} \tag{8}$$

To compute the DTM for more than one data set, it is not sufficient to just average the DTM of each data set. The misclassification rate has different scales on different data sets which leads to unequal consideration of the errors done on each data set. The scaled average DTM defined in Equation (9) is a simple solution to this problem. By scaling the misclassification rate per data set to $[0, 1]$, the evaluation measure will be 0 for the optimal solution and 1 for the worst solution. Additionally, every data set equally contributes to the measure.

$$\mathrm{ADTM}(\Lambda_T, \mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \min_{\boldsymbol{\lambda} \in \Lambda_T} \frac{f_D(\boldsymbol{\lambda}) - f_D^{min}}{f_D^{max} - f_D^{min}} \tag{9}$$

This kind of scaling was already proposed by Yogatama and Mann [10] to overcome the problem of different scales on different data sets. We want to highlight that the true scaling can only be done precisely if the true maximum and minimum is known. This information is used only by hindsight to evaluate the performance in the experiments.

For notational purposes, whenever f is used in the following, the approximated scaled version of it is intended. This approximation is computed by using the largest and smallest value that has been seen as an approximation for the maximum and minimum, respectively.

C. A Loss for Initializing the Hyperparameter Optimization

In the last section, the evaluation measure for hyperparameter optimization in general is formalized. A good initialization should support a faster convergence of the hyperparameter search such that the evaluation measure is minimal after T steps. Although, this cannot necessarily be achieved by minimizing the same measure within the I initialization iterations, we use this loss as a proxy and optimize the initial sequence of hyperparameter configurations for the same evaluation measure. Thus, the meta-loss in Equation (10) is minimized.

$$\mathcal{L}\left(\Lambda_I, \mathcal{D}\right) = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \min_{\boldsymbol{\lambda} \in \Lambda_I} f_D\left(\boldsymbol{\lambda}\right) \tag{10}$$

D. Differentiable Meta-Loss

The goal in this work is to learn a set of I initial hyperparameter configurations that are not limited to the candidates evaluated on previous data sets. This sequence is learned by minimizing the meta-loss defined in Equation (10).

Obviously, this loss is not differentiable. The minimum function is a non-differentiable function and the function f_D is only partially observed. The minimum function can be approximated by the differentiable softmin function $\sigma : \mathbb{R}^m \rightarrow \mathbb{R}^m$ as defined in Equation (11). We choose $\beta = -100$ as proposed in [22] such that $\sum_{i=1}^{m} \mathbf{x}_i \sigma\left(\mathbf{x}\right)_i$ is very close to the true minimum $\min\left\{\mathbf{x}_1, \ldots, \mathbf{x}_m\right\}$.

$$\sigma\left(\left(\mathbf{x}_1, \ldots, \mathbf{x}_m\right)^T\right)_i := \frac{e^{\beta \mathbf{x}_i}}{\sum_{j=1}^{m} e^{\beta \mathbf{x}_j}} \tag{11}$$

The function f_D is approximated by a differentiable plug-in estimator or surrogate model $\hat{f}_D$. This can be any differentiable regression model that is able to learn from few observations on D and generalize to unseen meta-instances. Thus, the final, differentiable meta-loss is given in Equation (12).

$$\mathcal{L}\left(\Lambda_I, \mathcal{D}\right) = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \sum_{i=1}^{I} \sigma_{D,i} \hat{f}_D\left(\boldsymbol{\lambda}_i\right) \tag{12}$$

where for notational convenience, the following notation is used in the remainder of the paper,

$$\sigma_{D,i} \quad = \quad \frac{e^{\beta \hat{f}_D\left(\boldsymbol{\lambda}_i\right)}}{\sum_{j=1}^{I} e^{\beta \hat{f}_D\left(\boldsymbol{\lambda}_j\right)}}. \tag{13}$$

E. Gradients for the Meta-Loss

In order to apply a gradient-based learning algorithm that minimizes the meta-loss the gradients for this loss with respect to the initial hyperparameter configurations need to be estimated.

$$\begin{aligned} \frac{\partial}{\partial \lambda_{l,j}} \mathcal{L}\left(\Lambda_I, \mathcal{D}\right) &= \frac{\partial}{\partial \lambda_{l,j}} \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \sum_{i=1}^{I} \sigma_{D,i} \hat{f}_D\left(\boldsymbol{\lambda}_i\right) \\ &= \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \sigma_{D,l} \left(\frac{\partial}{\partial \lambda_{l,j}} \hat{f}_D\left(\boldsymbol{\lambda}_l\right)\right) \\ &\qquad \cdot \left(\beta\left(1 - \sigma_{D,l}\right) \hat{f}_D\left(\boldsymbol{\lambda}_l\right) + 1\right) \end{aligned} \tag{14}$$

Hyperparameter response function are highly non-linear functions. Hence, our plug-in estimator needs to be able to model this property. Gaussian processes have proven to be a good surrogate model for the meta-testing data, hence we decided to also use it as a surrogate model for each training data set D. Hence, using the notation from Section III-C, the derivative for this specific prediction model is

$$\begin{aligned} \frac{\partial}{\partial \lambda_{l,j}} \mathcal{L}\left(\Lambda_I, \mathcal{D}\right) &= \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \sigma_{D,l} \left(\boldsymbol{\alpha}_D^T \frac{\partial}{\partial \lambda_{l,j}} \mathbf{k}_{D,*}\right) \\ &\qquad \cdot \left(\beta\left(1 - \sigma_{D,l}\right) \mathbf{k}_{D,*}^T \boldsymbol{\alpha}_D + 1\right) \end{aligned} \tag{15}$$

F. Learning Algorithm

After deriving the gradients, the final learning algorithm is presented in Algorithm 2. First, a surrogate model for each training data set is learned. Then, some initial values for $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_I$ are chosen. Instead of using a more sophisticated initialization step such as k-means, a relatively simple strategy is used. A subset containing I of all $|\mathcal{D}|$ training data sets are chosen at random and the best hyperparameter configurations for these data sets are used as initial values. Afterwards, the initial solution is iteratively improved by updating the hyperparameter configurations into the negative direction of the gradient weighted by the learning rate η. For our experiments we identified $\eta = 10^{-3}$ and $E = 10^3$ epochs as useful values for our problem.

Algorithm 2 Learning Hyperparameter Optimization Initialization

```
Input: Meta-training set $\mathcal{D}$, number of initial hyperparameter
    configurations $I$, number of epochs $E$, meta-learn rate $\eta$.
Output: Optimal set of hyperparameter configurations for
    initialization.
1: for $D \in \mathcal{D}$ do
2:     Train $\hat{f}_D(\boldsymbol{\lambda}_*) = \mathbf{k}_{D,*}^T \boldsymbol{\alpha}_D$ on observed meta-instances
       $(\boldsymbol{\lambda}_i, f_D(\boldsymbol{\lambda}_i))$.
3: Initialize $\Lambda_I = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_I\}$ with the best hyperparameter
   configurations of $I$ data sets $D \in \mathcal{D}$ chosen at random.
4: for $e = 1$ to $E$ do
5:     Precompute $\hat{f}_D(\boldsymbol{\lambda}_i)$ and $\boldsymbol{\sigma}_D$ for all $i = 1 \ldots I$ and
       $D \in \mathcal{D}$.
6:     for $i = 1$ to $I$ do
7:         for $j = 1$ to $P$ do
8:             $\lambda_{i,j} \leftarrow \lambda_{i,j} - \eta \frac{\partial}{\partial \lambda_{i,j}} \mathcal{L}(\Lambda_I, \mathcal{D})$
9: return $\Lambda_I$
```

G. Adaptive Data Set Weights

So far, the influence of each training data set for finding the optimal initial hyperparameter configurations is equal.

Naturally, it is expected that one data set may contain more information than others, hence, we propose to weight the influence of each data set with a cost function that depends on one training data set D and the new data set D_{new} for which the optimal hyperparameter configuration is sought.

$$\mathcal{L}\left(\Lambda_I, \mathcal{D}\right) = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} c\left(D, D_{new}\right) \sum_{i=1}^{I} \sigma_{D,i} \hat{f}_D\left(\boldsymbol{\lambda}_i\right) \tag{16}$$

The cost function c should be 1 if D is similar to D_{new} and close to 0 if they are completely different. This cost function may depend on meta-features [8], [15] but we propose to iteratively reupdate and recompute the initial hyperparameter configurations depending on the outcome of already performed evaluations $f_{D_{new}}\left(\boldsymbol{\lambda}_1\right), \ldots, f_{D_{new}}\left(\boldsymbol{\lambda}_t\right)$, $t < I$. The assumption is that the I initial hyperparameter configurations are not evaluated in parallel but sequentially. Then, at time step $t+1$, it is sufficient to minimize the loss in Equation (16) by keeping the already evaluated hyperparameter configurations $\Lambda_t = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_t\}$ fixed and find $\boldsymbol{\lambda}_{t+1}$ that minimizes the loss. The weighting of the data sets is updated using the approximated Kendall Tau Rank Correlation Coefficient [23] defined by

$$\begin{aligned} c\left(D, D_{new}, \Lambda_t\right) &:= \frac{\sum_{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \in \Lambda_t} s\left(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, D, D_{new}\right)}{\left|\Lambda_t\right| \left(\left|\Lambda_t\right| - 1\right)} \\ s\left(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, D, D_{new}\right) &:= \chi\left(\hat{f}_D\left(\boldsymbol{\lambda}_1\right) > \hat{f}_D\left(\boldsymbol{\lambda}_2\right)\right. \\ &\qquad \left. \oplus f_{D_{new}}\left(\boldsymbol{\lambda}_1\right) > f_{D_{new}}\left(\boldsymbol{\lambda}_2\right)\right) \end{aligned} \tag{17}$$

where the exclusive-or operator is denoted by $\oplus$ and $\chi(\cdot)$ is 1 if the logical expression is true and 0 otherwise. For the new data set D_{new}, the true values for all hyperparameter configurations in Λ_t are given. This is not the case for the training data set D and thus this value is approximated using $\hat{f}_D$.

H. Comparison to State of the Art

The state of the art strategy for initializing hyperparameter optimization [8], [15] can be understood as a special case of our general loss function in Equation (16). They optimize for the same loss but use a different cost function. While we are using a constant cost function in Section IV-F or an adaptive function in Section IV-G, they propose to consider only the data sets that are nearest regarding a distance function δ over the meta-features of the data set. Formally, they use

$$c\left(D, D_{new}\right) := \begin{cases} 1 & \text{if } D \text{ is among the } I \text{ nearest} \\ & \text{data sets regarding } \delta \text{ to } D_{new} \\ 0 & \text{otherwise} \end{cases} . \tag{18}$$

To optimize the loss function with respect to this cost function, no difficult optimization technique is needed. Simply taking the best hyperparameter configurations of the I nearest data sets to D_{new} is sufficient.

This choice of initial hyperparameter configurations is limited to the hyperparameter configurations that have been investigated on the training data sets. We will show that, if this constraint is not given, better initial hyperparameter configurations can be found.

V. Experimental Evaluation

A. Tuning Strategies

We want to give a short introduction to all tuning strategies considered in the experiments. Both, strategies that do not use knowledge from previous experiments and those that use it, are considered. All strategies are based on the SMBO framework and use the expected improvement as the acquisition function. The only difference is the surrogate model and whether knowledge from other data sets is transferred or not.

Independent Gaussian Process (GP): This tuning strategy uses a Gaussian process with squared-exponential kernel with automatic relevance determination (SE-ARD) 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 (RF): Next to Gaussian processes, random forests are the most widely used surrogate model [18] and are used in the experiments. Like the independent Gaussian process, the RF does not use any knowledge from previous experiments.

Surrogate Collaborative Tuning (SCoT): SCoT [9] uses a Gaussian process with SE-ARD kernel and is trained on previous and the current experiment. Instead of using the original labels, an SVMRank is learned on the data set and its predictions are used as the labels instead. Bardenet et al. [9] argue that this overcomes the problem of having data sets with different scales of labels. 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 [10] and use a linear kernel instead.

Gaussian Process with MKL (MKLGP): Similarly to Bardenet et al. [9], Yogatama and Mann [10] 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 and a kernel modeling the distance between data sets.

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 in Figures 1 to 6 show the averages of ten repetitions of a leave-one-out cross-validation on the data sets. Thus, each of the 50 data sets is once the new data sets while all others are part of the meta-training set.

B. Initialization Strategies

Following initialization strategies will be considered in our experiments.

a) No Initialization (No Init): No initialization is used. This is equivalent to a random initialization for all surrogate models that do not transfer knowledge from previous experiments. Thus, these results were repeated 1,000 times instead of only 10 times and averaged.

TABLE I. 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

b) Random Best Initialization (RBI): This initialization is a very simple initialization. I training data sets from the set of all training data sets $\mathcal{D}$ are chosen uniformly at random. Then, the best hyperparameter configurations on these data sets are used for the initialization. Hence, this corresponds to the initialization used in Algorithm 2.

Nearest Best Initialization (NBI): This is the initialization strategy proposed by Reif et al. and Feurer et al. [8], [15]. Instead of choosing I training data sets uniformly at random, they are chosen with respect to the similarity between the meta-features listed in Table I. Then, like for RBI, the best hyperparameter configurations on these data sets are chosen for initialization.

Learning Initializations (LI): Learning Initializations is the strategy introduced in Section IV and summarized in Algorithm 2. Its advantage is that it is directly optimized for the loss and the selected hyperparameter configurations are not limited to hyperparameter configurations that were previously observed in past experiments.

Adaptive Learning Initializations (aLI): Adaptive Learning Initializations is presented in Section IV-G. It is an extension to Learning Initializations (LI) that tries to incorporate the knowledge about the new data set that is sequentially gathered by re-weighting the influence of each training data set.

C. Meta-Features

Meta-features are supposed to be discriminative for a data set and can be estimated without evaluating f. Many surrogate models [9], [13], [10] and initialization strategies [8], [15] use them to predict the similarity between data sets. For the experiments, the meta-features listed in Table I are used. For an in-depth explanation we refer the reader to [9], [24].

D. Evaluation Metrics

As proposed before, the scaled average distance to the minimum is used as the evaluation measure.

$$\mathrm{ADTM}\left(\Lambda_T, \mathcal{D}\right) := \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \min_{\boldsymbol{\lambda} \in \Lambda_T} \frac{f_D\left(\boldsymbol{\lambda}\right) - f_D^{min}}{f_D^{max} - f_D^{min}} \tag{19}$$

For computational reasons, evaluations of f on the data sets are precomputed by doing an extensive grid search. Then, the global minimum and maximum is approximated by using the smallest and largest value that was found during the grid search, respectively.

The initial hyperparameter configurations found by Learning Initialization will not be on this grid. To avoid the time-consuming evaluation for the exact hyperparameter configurations, the hyperparameter configurations were chosen that are closest to the one proposed by Learning Initializations.

E. Meta-Data Sets

For creating the two meta-data sets, 50 classification data sets are used. In case there were already train/test splits, all instances were merged, shuffled and split into 80% train and 20% test. Then, two different classifiers were used to create the meta-instances for two meta-data sets: a support vector machine (SVM) [25] and AdaBoost [26].

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 misclassification rate was precomputed on a grid $C \in \left\{2^{-5}, \ldots, 2^{6}\right\}$, $d \in \{2, \ldots, 10\}$ and $\gamma \in \left\{10^{-4}, 10^{-3}, 0.01, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20, 50, 10^2, 10^3\right\}$ resulting in 288 meta-instances per data set.

The AdaBoost meta data set was created as proposed by Bardenet et al. [9]. Decision products are chosen as weak learners such that the number of hyperparameters is two. The number of iterations i and the number of product terms p. The misclassification rate was precomputed on a grid with values $i \in \{2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$ and $p \in \{2, 3, 4, 5, 7, 10, 15, 20, 30\}$ resulting in 108 meta-instances per data set.

Both meta-data sets are extended by the meta-features listed in Table I. To show that being able to choose hyperparameter configurations that have not been seen on the meta-training set yields an improvement, a coarse grid was used for the meta-training data set and the full, fine grid for the meta-testing data set.

F. Experiments

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

1) Comparison to other Initialization Strategies: The ADTM on the two meta-data set for $I = 1 \ldots 10$ for different initialization strategies is shown in Figure 1. This experiment analyzes i) the performance of initialization strategies versus surrogate models that are transferring knowledge from previous experiments (SCoT and MKLGP), ii) the legitimacy of the use of meta-features as a measure for similarity between data sets and iii) compares our proposed initialization strategies

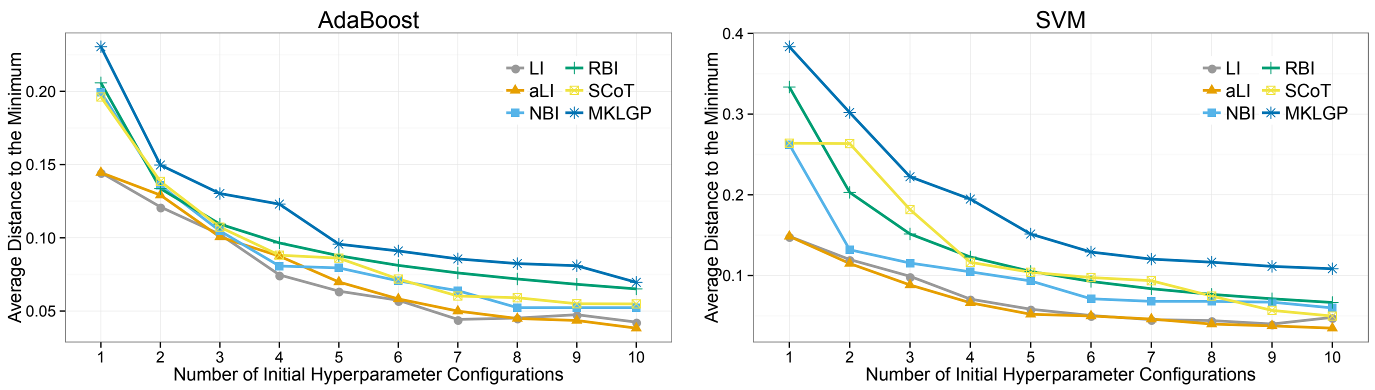

Fig. 1. Development of the ADTM for increasing number of initial hyperparameter configurations on both meta-data sets. Our proposed strategies LI and aLI are outperforming the state of the art initialization strategies (RBI/NBI) and state of the art surrogate models that transfer knowledge from previous experiments (SCoT and MKLGP).

to the current state of the art. One may argue, that in the case that a surrogate model is used to transfer knowledge from previous experiments, no further initialization strategy is needed. The experiments do not indicate that this is true. While we acknowledge that at least SCoT provides a moderate initialization sequence, it is still not able to beat the best initialization strategy. A direct comparison between RBI and NBI indicates that meta-features can be used to estimate the similarity between data sets. The use of meta-features in NBI leads to an improved initialization compared to RBI. Finally, our proposed initialization strategy LI provides a very good initialization for $I = 1$ and is consistently the best initialization. Its variation aLI does not seem to provide better results.

2) Comparison with Respect to the Long Term Effect: The aim of an initialization strategy is to accelerate the hyperparameter search and convergence to the best or at least a good hyperparameter configuration. Thus, not the performance at the end of the initialization is essential but the further convergence. Therefore, a further experiment was conducted. The SMBO framework was initialized with five hyperparameter configurations using the respective initialization strategy and then was continued using the SMBO procedure. All initialization strategies are also compared to the case where no initialization was used. The experiments are conducted for all surrogate models presented in Section V-A.

We expect that transferring knowledge from previous experiments has more impact if this is not done by the surrogate model. This can be seen in Figure 2. Comparing Figure 1 with Figure 2 on the SVM meta-data set, one can see that all initialization strategies provide better results for the first initial value than the GP without initialization does after six SMBO iterations. The results in Figure 2 and 3 are of special interest because the effect of the knowledge transfer is not distorted by a surrogate model that also transfers knowledge. Our proposed strategy LI outperforms any other initialization strategy on both meta-data sets and for both surrogate models. Again, aLI does not provide better results than LI. It is interesting to notice that RBI provides similar results to NBI.

The results in Figure 4 and 5 demonstrate the impact of an initialization strategy on a surrogate model that transfers already knowledge from previous experiments. Obviously, the gap between no initialization and some specific initialization shrinks. SCoT without initialization achieves similar results as SCoT with RBI or NBI but SCoT with LI consistently achieves the best results. Account should be taken of the fact that the result lines will compulsory get closer with growing number of SMBO iterations and will meet after just enough iterations. For the MKLGP the results are similar, but here NBI is able to outperform LI on the SVM meta-data set.

So far we argued carefully that, even though surrogate models that are transferring knowledge across data sets exist, there is still need for an initialization strategy. Moreover, one can question: Are these surrogate models necessary if a good initialization strategy is used? Surrogate models that learn across data sets have three big disadvantages:

1) The run-time for updating the surrogate model after each SMBO iteration is by a factor of $\mathcal{O}\left(|\mathcal{D}|^2\right)$ higher than a surrogate model that is only trained on the current data set (assuming the surrogate model is based on a GP which is the case for SCoT and MKLGP).
2) A specific surrogate model is needed and no out of the box machine learning model such as GP or RF can be used. Additionally, hand-crafted, problem-dependent meta-features need to be used.
3) These surrogate models are specific for the SMBO framework and cannot be used for other optimization frameworks such as the initialization strategy. Furthermore, an initialization strategy can be learned by a single researcher and then shared with other researchers and practitioners while the specific surrogate model always also includes sharing the meta-data.

These disadvantages may be tolerable if there is an improvement in terms of hyperparameter optimization acceleration. Figure 6 compares both surrogate models that do not use meta-features and do not transfer knowledge (GP and RF) to those that do (SCoT and MKLGP). All four strategies are initialized with five hyperparameter configurations by LI since this provided better results for all strategies. As the reader can

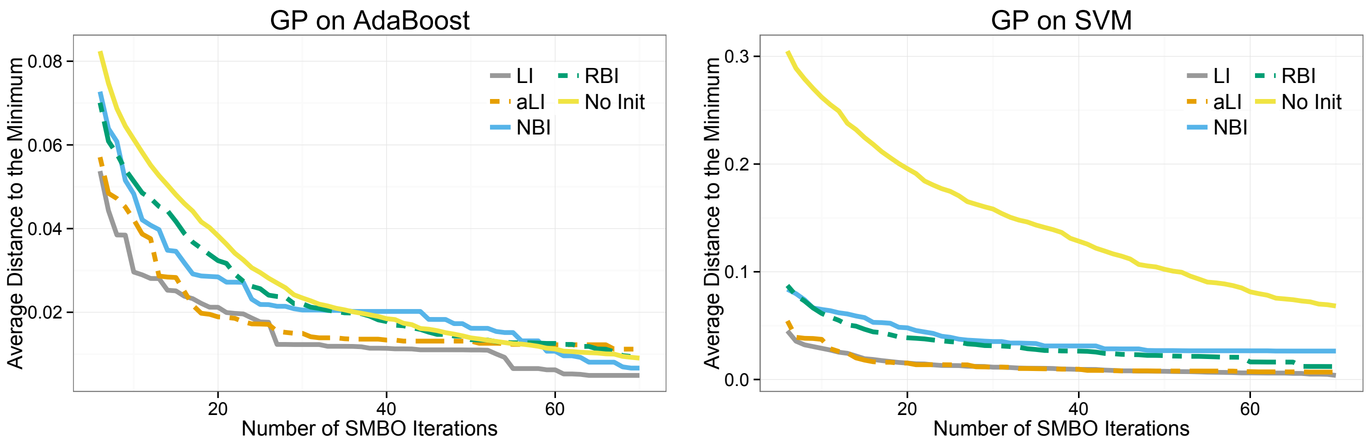

Fig. 2. Impact of an initialization with five hyperparameter configurations on the long term optimization for GP (a surrogate model that does not use information from previous experiments on other data sets). Our proposed strategies LI and aLI are outperforming alternative initialization strategies on both meta-data sets.

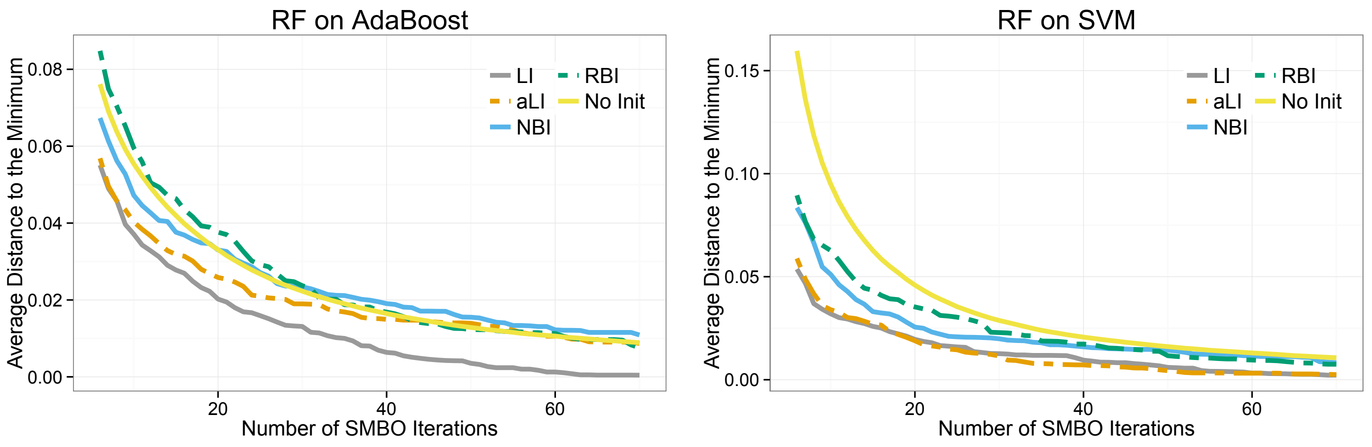

Fig. 3. Impact of an initialization with five hyperparameter configurations on the long term optimization for RF (a surrogate model that does not use information from previous experiments on other data sets). Our proposed initialization strategy LI is outperforming the state of the art on both meta-data sets.

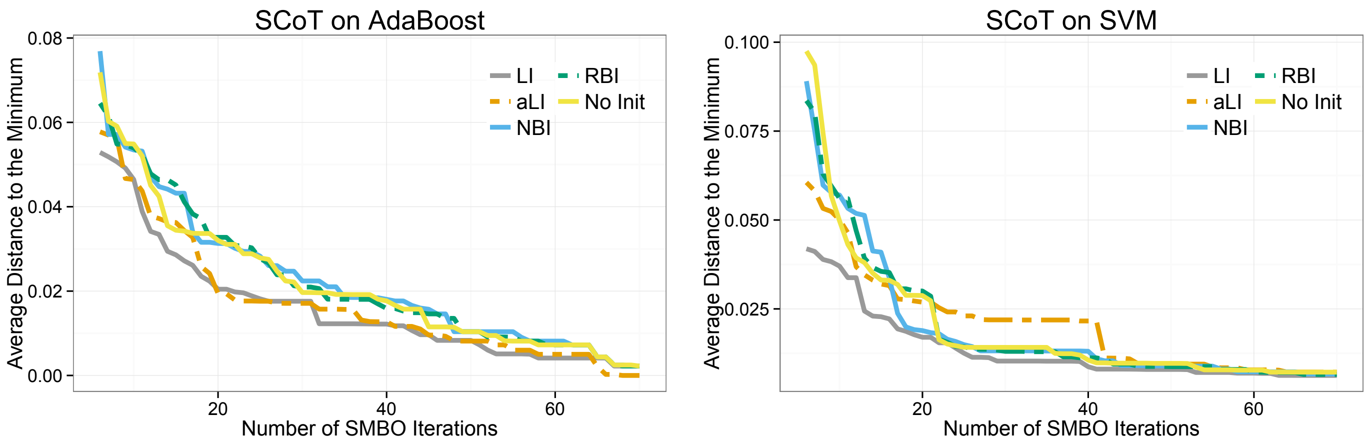

Fig. 4. Impact of an initialization with five hyperparameter configurations on the long term optimization for SCoT (a surrogate model that transfers knowledge from previous experiments to the new data set). Our proposed initialization strategy LI is outperforming the state of the art on both meta-data sets.

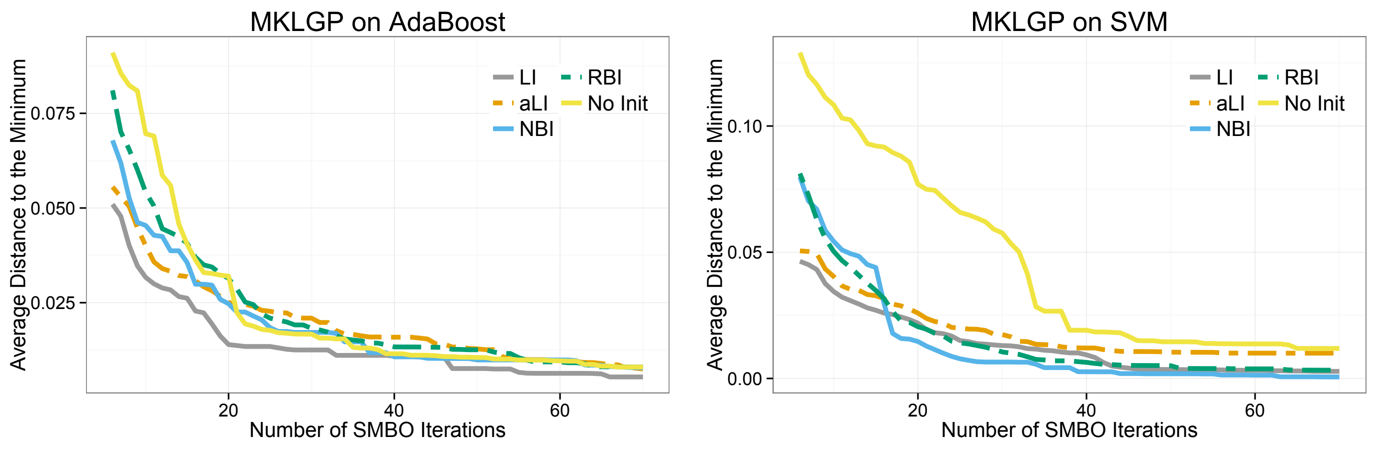

Fig. 5. Impact of an initialization with five hyperparameter configurations on the long term optimization for MKLGP (a surrogate model that transfers knowledge from previous experiments to the new data set). Our proposed initialization strategy LI is outperforming the state of the art on AdaBoost meta-data sets. We acknowledge that NBI provides better results for the SVM meta-data set.

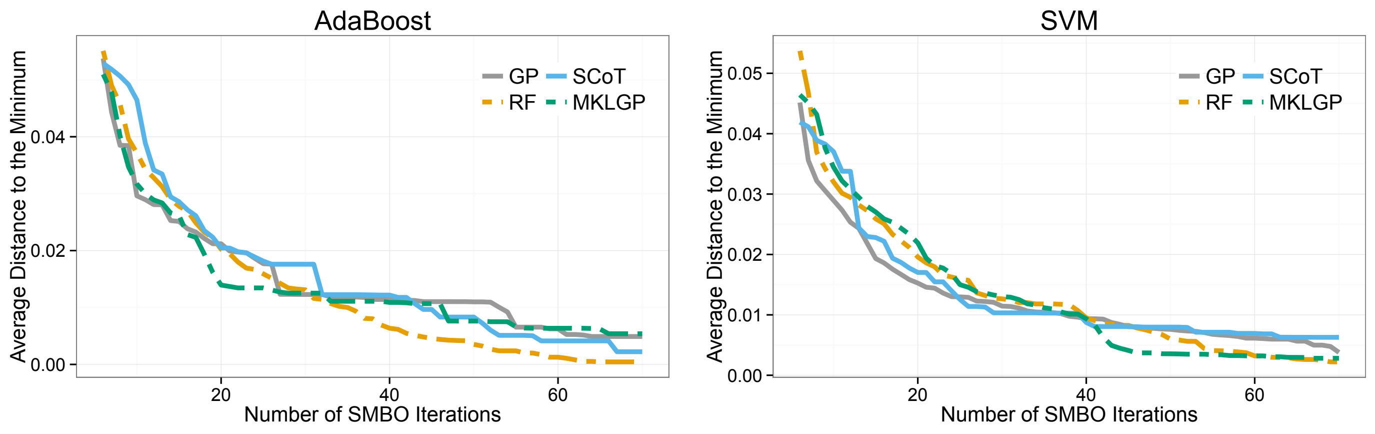

Fig. 6. Comparison of the performance development of four different surrogate models that are all initialized with five LI hyperparameter configurations. RF is a surrogate model that does not transfer knowledge between data sets but yet performs best only due to the LI initialization strategy. Hence, current state of the art surrogate models that transfer knowledge between data sets do not seem to achieve better results if one uses a initialization strategy instead.

see, the surrogate models that learn across data sets do not provide better results.

G. *Experimental Conclusion*

Concluding this section, our proposed initialization strategy LI is able to provide better initial hyperparameter configurations than the state of the art. Furthermore, it demonstrates better results with respect to the long term effect. Initialization strategies for optimization strategies that do not transfer knowledge from other experiments gain most from the initialization. But also for surrogate models that already transfer the knowledge an improvement is recognizable. Furthermore, the current state of the art surrogate models do not seem to be necessary if a good initialization strategy is chosen. The proposed adaptive re-weighting of the data set weights (aLI) has some potential but the re-weighting as proposed by us does not show better results than a constant weighting.

VI. Conclusion

A meta-loss for hyperparameter optimization was derived that depends on the hyperparameter response function of previously seen data sets. Since the response function is unknown and only few observations are given, the meta-loss is not differentiable. By approximating the response function with a differentiable plug-in estimator, the meta-loss becomes differentiable. That in turn enabled us to learn initial hyperparameter configurations that minimize the meta-loss. These learned initial hyperparameter configurations are not limited to configurations that have been seen before.

Our initialization learning algorithm was compared to state of the art initialization strategies and provided better initial values and furthermore has better long term effects on the hyperparameter optimization. Finally, a generalized meta-loss was presented that allows to (dynamically) weight the influence of each data set and we have shown that the current state of the art initialization strategies are optimized for a special case of this general meta-loss.

Acknowledgement

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] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[2] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2546–2554.

[3] A. Coates, H. Lee, and A. Ng, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. JMLR Workshop and Conference Proceedings, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. JMLR W&CP, 2011, pp. 215–223.

[4] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox, “A high-throughput screening approach to discovering good forms of biologically inspired visual representation,” *PLoS Computational Biology*, vol. 5, no. 11, p. e1000579, 2009, PMID: 19956750.

[5] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959.

[6] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Auto-weka: Combined selection and hyperparameter optimization of classification algorithms,” in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’13. New York, NY, USA: ACM, 2013, pp. 847–855.

[7] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *J. of Global Optimization*, vol. 13, no. 4, pp. 455–492, Dec. 1998.

[8] M. Feurer, J. T. Springenberg, and F. Hutter, “Using meta-learning to initialize bayesian optimization of hyperparameters,” in *ECAI workshop on Metalearning and Algorithm Selection (MetaSel)*, 2014, pp. 3–10.

[9] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, “Collaborative hyperparameter tuning,” in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, S. Dasgupta and D. Mcallester, Eds., vol. 28, no. 2. JMLR Workshop and Conference Proceedings, May 2013, pp. 199–207.

[10] D. Yogatama and G. Mann, “Efficient transfer learning method for automatic hyperparameter tuning,” in *International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, 2014.

[11] N. Schilling, M. Wistuba, L. Drumond, and L. Schmidt-Thieme, “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. Proceedings, Part II*, 2015.

[12] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, “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. Proceedings, Part II*, 2015.

[13] K. Swersky, J. Snoek, and R. P. Adams, “Multi-task bayesian optimization,” in *Advances in Neural Information Processing Systems 26*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2004–2012.

[14] T. A. Gomes, R. B. Prudêncio, C. Soares, A. L. Rossi, and A. Carvalho, “Combining meta-learning and search techniques to select parameters for support vector machines,” *Neurocomputing*, vol. 75, no. 1, pp. 3 – 13, 2012, brazilian Symposium on Neural Networks (SBRN 2010) International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010).

[15] M. Reif, F. Shafait, and A. Dengel, “Meta-learning for evolutionary parameter optimization of classifiers,” *Machine Learning*, vol. 87, no. 3, pp. 357–380, 2012.

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Proceedings of the IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.

[18] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, ser. LION’05. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 507–523.

[19] J. Villemonteix, E. Vazquez, and E. Walter, “An informational approach to the global optimization of expensive-to-evaluate functions,” *Journal of Global Optimization*, vol. 44, no. 4, pp. 509–534, 2009.

[20] N. Srinivas, A. Krause, M. Seeger, and S. M. Kakade, “Gaussian process optimization in the bandit setting: No regret and experimental design,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, J. Frnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 1015–1022.

[21] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[22] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, “Learning time-series shapelets,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’14. New York, NY, USA: ACM, 2014, pp. 392–401.

[23] M. G. Kendall, “A New Measure of Rank Correlation,” *Biometrika*, vol. 30, no. 1/2, pp. 81–93, Jun. 1938.

[24] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds., *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, 1994.

[25] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[26] D. Benbouzid, R. Busa-Fekete, N. Casagrande, F.-D. Collin, and B. Kégl, “Multiboost: A multi-purpose boosting package,” *J. Mach. Learn. Res.*, vol. 13, pp. 549–553, Mar. 2012.