Mach Learn (2018) 107:43–78
https://doi.org/10.1007/s10994-017-5684-y

Scalable Gaussian process-based transfer surrogates for hyperparameter optimization

Martin Wistuba[1] · **Nicolas Schilling**[1] · **Lars Schmidt-Thieme**[1]

Received: 7 November 2016 / Accepted: 4 October 2017 / Published online: 22 December 2017

Abstract Algorithm selection as well as hyperparameter optimization are tedious task that have to be dealt with when applying machine learning to real-world problems. Sequential model-based optimization (SMBO), based on so-called "surrogate models", has been employed to allow for faster and more direct hyperparameter optimization. A surrogate model is a machine learning regression model which is trained on the meta-level instances in order to predict the performance of an algorithm on a specific data set given the hyperparameter settings and data set descriptors. Gaussian processes, for example, make good surrogate models as they provide probability distributions over labels. Recent work on SMBO also includes meta-data, i.e. observed hyperparameter performances on other data sets, into the process of hyperparameter optimization. This can, for example, be accomplished by learning transfer surrogate models on all available instances of meta-knowledge; however, the increasing amount of meta-information can make Gaussian processes infeasible, as they require the inversion of a large covariance matrix which grows with the number of instances. Consequently, instead of learning a joint surrogate model on all of the meta-data, we propose to learn individual surrogate models on the observations of each data set and then combine all surrogates to a joint one using ensembling techniques. The final surrogate is a weighted sum of all data set specific surrogates plus an additional surrogate that is solely learned on the target observations. Within our framework, any surrogate model can be used and explore Gaussian processes in this scenario. We present two different strategies for finding the weights

Nicolas Schilling and Martin Wistuba have contributed equally to this work.

Editors: Pavel Brazdil and Christophe Giraud-Carrier.

✉ Martin Wistuba
wistuba@ismll.uni-hildesheim.de

Nicolas Schilling
schilling@ismll.uni-hildesheim.de

Lars Schmidt-Thieme
schmidt-thieme@ismll.uni-hildesheim.de

[1] Information Systems and Machine Learning Lab, Universitätsplatz 1, Hildesheim, Germany

used in the ensemble: the first is based on a probabilistic product of experts approach, and the second is based on kernel regression. Additionally, we extend the framework to directly estimate the acquisition function in the same setting, using a novel technique which we name the "transfer acquisition function". In an empirical evaluation including comparisons to the current state-of-the-art on two publicly available meta-data sets, we are able to demonstrate that our proposed approach does not only scale to large meta-data, but also finds the stronger prediction models.

Keywords Hyperparameter optimization · Gaussian processes · Sequential model-based optimization · Meta-learning

1 Introduction

Hyperparameter optimization and algorithm selection are ubiquitous tasks in machine learning contexts that usually have to be conducted for every individual research task and real-world application. Choosing the correct model and hyperparameter configuration usually improves very poor predictions to state-of-the-art performance.

Hyperparameter optimization tries to find the hyperparameter configuration that minimizes a certain black box function $y(x)$, which is commonly a cross-validation loss of a model learned on some training data using the hyperparameter configuration x. Despite its omnipresence, hyperparameter optimization is usually a difficult task, as the optimization cannot be carried out by minimizing a loss function with nice mathematical properties such as differentiability or convexity. Consider for example the number of hidden layers and hidden neurons for a simple feed-forward neural network. When learning the neural network, both of these hyperparameters have to be set, as the final prediction performance depends heavily on the correct setting of the model complexity.If the model is too complex (i.e. many layers and neurons), the model will very likely overfit the training data or get stuck in a local minimum. However, if the model complexity is not high enough, it might underfit the training data, and miss information vital to the optimization procedure. Thus, the correct setting of these hyperparameters is vital for any serious application of machine learning; however, as mentioned above, the difficulty with this task is that we have no loss function which we can optimize to learn the specific best choices for these hyperparameters.

The majority of efforts to solve this problem are based on the sequential model-based optimization (SMBO) framework, which has its roots in the area of black-box optimization. SMBO is an iterative approach which trains a *surrogate model* Ψ on the observed meta-level instances of y. Then, it can be used in order to predict the performance of an algorithm on a specific data set given the hyperparameter settings and data set descriptors. We use this method to find promising hyperparameter configurations, evaluate y for these configurations, and finally retrain Ψ. The overall process is repeated T many times, and in the end, we take the best hyperparameter configuration found so far. In comparison to exhaustive search methods, SMBO tries to adaptively steer the optimization into promising regions in the hyperparameter space.

More recently, SMBO has been used in conjunction with meta-learning in order to create a "meta-learning system". According to Lemke et al. (2015), a meta-learning system must fulfill two properties:

1. A meta-learning system must include a learning subsystem which adapts with experience.
2. Experience is gained by exploiting meta-knowledge extracted

(a) in a previous learning episode on a single data set, and/or
(b) from different domains or problems.

Our contributions to SMBO lead to a system that fulfills all of these requirements. Our system adapts with experience by updating the surrogate model which represents the meta-knowledge. Furthermore, we exploit meta-knowledge extracted on the new data set and from previous problems.

Throughout this paper, the term meta-data refers to observations of the performances of different sets of hyperparameter configurations evaluated on a various, different data sets. The inclusion of such meta-data in SMBO-based hyperparameter optimization can be accomplished in different ways, with the most commonly used approach being to pretrain the surrogate model on these observations. Such surrogate models, which are pretrained on meta-data, are called "transfer surrogate models" because they are capable of using the meta-data to infer from previously seen data sets to new ones.

Another approach learns a particular initialization, i.e. a set of hyperparameter configurations, which is most likely to work well on the data. This approach has been shown to produce better results; this seems plausible due to the nature of the problem.

Usually, researchers gain more experience in choosing well-performing hyperparameter configurations for their models by running these models on a variety of data sets and hyperparameters. Consequently, if a new data set arrives, the hyperparameter optimization guided by an expert will comprise this knowledge into choosing which initial configurations to test. This intuition makes it clear that incorporating hyperparameter performance on other data sets into the surrogate model used within SMBO aids in speeding up and steering the hyperparameter optimization towards regions where we can suspect to find good hyperparameter configurations.

A number of publications show that using meta-data is beneficial, including but not limited to Bardenet et al. (2013), Yogatama and Mann (2014), Swersky et al. (2013), Schilling et al. (2015), Wistuba et al. (2015, 2016), Feurer et al. (2015).

In this paper, we integrate two pieces of work, Schilling et al. (2016), Wistuba et al. (2016), which both learn individual Gaussian processes on subsets of the meta-data. Each Gaussian process is learned on all the observed performances of a *single* data set, i.e. the hyperparameter configuration and its corresponding performance on this specific data set. Finally, all processes are then combined into a single surrogate model. In this way, we can achieve scalability to large amounts of meta-data because the training effort of the surrogate model is no longer cubic in the number of data sets used to create the meta-data. We compare both papers and extend the ideas therein by learning a transfer acquisition function using an ensemble of Gaussian processes. The resulting approach shows better empirical performance than the state-of-the art for hyperparameter optimization. We present this result via a set of thoroughly conducted experiments which maintain the scalability properties of a simple product of experts model. We then also show that the acquisition function is an elegant way of dealing with different performance scales for different data sets and a decaying use of meta-data.

Our contributions in this work are:

- The unification of our previous work (Schilling et al. 2016; Wistuba et al. 2016) as the scalable Gaussian process transfer surrogate framework.
- Identification of typical problems faced when using meta-data in surrogate models and thus,
- Proposal of using meta-learning in the acquisition function instead of the surrogate model to overcome these issues by using the transfer acquisition function framework.

– Extensive empirical evaluations for comparing all approaches including previous and new methods with additional discussion.

This work is structured as follows: in the next section we review the related work that has been done on SMBO and its combination with meta-data. In Sect. 3 we formally define the problem of hyperparameter optimization. We then present the detailed definition of SMBO and Gaussian processes (often used as an important component of SMBO) in Sects. 4 and 5. In Sect. 6, we present SGPT, our scalable transfer surrogate framework which unifies our previous work (Schilling et al. 2016; Wistuba et al. 2016). Section 7 then contains our extension this work by using the meta-data within the acquisition function of SMBO via our proposed "transfer acquisition function" (TAF) We evaluate our proposed method and compare it to current state of the art methods in Sect. 8, and then conclude the paper in Sect. 9.

2 Related work

The algorithm selection problem is an important problem in many domains and was first introduced in the 1970s (Rice 1976). Application domains include hard combinatorial problems such as SAT (Xu et al. 2008) and TSP (Kanda et al. 2012), software design (Cavazos and O'Boyle 2006), numerical optimization (Kamel et al. 1993), optimization (Nareyek 2004) and many more. In our work, we limit ourselves to the domain of machine learning although there are generalizations that subsume hyperparameter optimization in the broad category of algorithm configuration (Eggensperger et al. 2018). Thus, we investigate both the problem of finding the right algorithm as well as finding suitable hyperparameters for this algorithm.

Existing work can be grouped with respect to different properties. One way to group previous work is by methodological approach, where we have on the one hand approaches that search the hyperparameter space exhaustively and on the other hand methods that use black-box optimization techniques such as sequential model-based optimization (Jones et al. 1998). Other methods under this grouping make use of search algorithms from artificial intelligence such as genetic algorithms.

One can also distinguish between approaches based on those which use meta-data and those that do not. Meta-learning permits us to transfer past experiences with particular algorithms and hyperparameter configurations from one data set to another. Plenty of work has been done in that area and can be found in some recently published books and surveys: Brazdil et al. (2009), Vilalta and Drissi (2002), Lemke et al. (2015).

In the next section we discuss further related work as classified by methodology.

2.1 Exhaustive search methods

The most widely used method to optimize hyperparameters in machine learning is the "grid search". For a grid search, we choose a finite subset of hyperparameter configurations and evaluate them all in a brute force manner, ideally within a parallel computing environment. In some cases, grid search is manually steered by choosing a coarse grid at first to find regions where hyperparameter performance is generally good, with such regions being investigated more closely using a fine-grained grid. This mixture of grid search and manual search techniques appears in many publications: for instance Hinton (2010), Larochelle et al. (2007). The downsides of grid search are rather obvious: if the dimension of the hyperparameter space is

large and no prior knowledge about hyperparameter performance is given, grid search requires many evaluations to deliver good results, often at the expense of many useless computations.

Bergstra and Bengio (2012) introduce Random Search, which essentially replaces the fixed set of points by sampling points from some probability distribution. This has mainly two advantages, at first, one may enter prior beliefs over the hyperparameter space by defining the probability distributions to draw from. Secondly, random search works better in scenarios of low effective dimensionality, which is the case if hyperparameter performance almost stays constant in one dimension of the hyperparameter space and changes drastically in another dimension.

2.2 Model specific methods

Many methods for hyperparameter optimization exist which have been designed for a specific algorithm. These methods are usually based on genetic algorithms (Friedrichs and Igel 2005a; de Souza et al. 2006), although some are deterministic (Keerthi et al. 2007). Beyond this, there are many other methods for specific scenarios, including but not limited to methods for general regression and time-series models (McQuarrie and Tsai 1998), for regression when the sample size is small (Chapelle et al. 2002), for Bayesian topical trend analysis (Masada et al. 2009) and for log-linear models (Foo et al. 2007). Moreover, Schneider et al. deal with hyperparameter learning in probabilistic prototype-based models (Schneider et al. 2010), Seeger employs hyperparameter learning for large scale hierarchical kernel methods (Seeger 2006), and Kapoor et al. are concerned with optimizing hyperparameters for graph-based semi-supervised classification models (Kapoor et al. 2005).

The major limitation of all of these methods is that they are specifically tailored to one particular model and only work well in certain scenarios. This is a drawback that more widely applicable SMBO-based methods can alleviate.

2.3 Sequential model-based optimization

In order to overcome the issues of exhaustive search methods or model specific methods, black box optimization has been used in the context of sequential model-based optimization (SMBO) (Jones et al. 1998). SMBO learns a surrogate model on the observed hyperparameter performance, which is then queried to provide predictions for unobserved hyperparameter configurations. The predicted performance and the uncertainty of the surrogate model is then used within the expected improvement acquisition function to choose which of the many unobserved hyperparameter configurations to test next. The main strand of research along these lines has been committed to finding surrogate models, for example a Gaussian process (Rasmussen et al. 2005) which provided the so-called Spearmint method (Snoek et al. 2012). Other surrogate models, such as the random forests proposed in SMAC (Hutter et al. 2011), have also been investigated. While this earlier work focuses on optimizing hyperparameters only, Auto-WEKA (Thornton et al. 2013) has shown that algorithm-selection can be considered in a similar fashion to hyperparameters, and so the existing work is capable of choosing both algorithms and hyperparameters in combination.

Additionally, research on including meta-data, i.e. observations of hyperparameter performance on other data sets, has been gaining a lot of attention. Bardenet et al. (2013) use SVM^{RANK} as a surrogate and thus consider the hyperparameter selection as a ranking task rather than a regression task. In this way, they are able to overcome the issues of different data sets having different performance levels. To estimate uncertainties, they train a Gaussian process on the output of the SVM^{RANK}, in order to compute expected improvement.

Moreover, Gaussian processes with a meta-kernel have been proposed (Swersky et al. 2013; Yogatama and Mann 2014). Finally, neural networks have been used as surrogate models in combination with a factorization machine (Rendle 2010) in the input layer (Schilling et al. 2015).

2.4 Learning curve predictions

The idea of using meta-data in SMBO is to find better performing prediction models within a smaller fraction of time. Another idea applicable to models that are learned in an iterative fashion is to predict the learning curve, i.e. the performance of the resulting model, after a number of epochs. For example, Domhan et al. (2015) predict the performance of the hyperparameter configuration based on the partially observed learning curve after a few iterations. If the final performance is likely to be worse than the current best configuration, then the process is stopped and the configuration discarded, and the optimization continues with another, different configuration. Swersky et al. (2014) propose a similar approach which never discards a configuration, but instead learns the models for various hyperparameter configurations at the same time and switch from one learning process to another if it turns out to be more promising.

There are other hyperparameter approaches not related to SMBO following the same idea. There are, for example, some population-based approaches, such as Successive Halving (Jamieson and Talwalkar 2016) and Hyperband (Li et al. 2016), which choose a set of hyperparameter configurations at random and incrementally train the learning algorithms in parallel. From time to time, the weakest configurations are discarded. The Racing Algorithm by Maron and Moore (1997) follows a similar principle but is focused on lazy learners where the expensive part is testing rather than training.

2.5 Meta-initializations

There are several strategies to find a set of initial configurations for hyperparameter optimization methods. Reif et al. (2012) propose to initialize a hyperparameter search based on genetic algorithms with the best hyperparameters on other data sets, where the similarity of data sets is defined through meta-features. Feurer et al. (2014) propose the same idea for SMBO which was later extended (Feurer et al. 2015; Wistuba et al. 2015). The drawback of these approaches is that they do not consider whether the initial hyperparameter configurations are very close to each other and therefore may waste computation time by choosing too similar hyperparameters initially. Thus, one of our previous works proposes to learn a set of initial hyperparameter configurations by optimizing a meta-loss that maximizes the overall improvement on the meta-data (Wistuba et al. 2015).

2.6 Meta-features

Meta-features are descriptive characteristics of a data set and thus an essential component of all traditional meta-learning methods that are learning across problems. In this work, we use pairwise comparisons of the performance of two hyperparameter configurations on one data set compared to another. This is a very special instance of landmarkers (Pfahringer et al. 2000). Landmark features are created by applying very fast machine learning algorithms (e.g. decision stumps, linear regression) to the data, with the performance is added as a meta-feature. In contrast, our approach uses only the performance of algorithms and hyperparameter configurations which we have evaluated during our optimization process,

and thus, no additional time was spent for estimating these landmarkers. This idea has been already employed by some others (Leite et al. 2012; Sun and Pfahringer 2013; Wistuba et al. 2015). In contrast to their work, we propose a way of using these meta-features also in cases with continuous hyperparameters, since for continous hyperparameters, it is very unlikely that we have seen the same hyperparameter configurations for all data sets. The approach of pairwise comparisons proposed by the literature works only if we either only want to find the best algorithm and ignore the hyperparameters (Sun and Pfahringer 2013) or discretize the hyperparameters (Leite et al. 2012; Wistuba et al. 2015). We overcome this problem by predicting the performance of a hyperparameter configuration if it is not part of our meta-data set.

2.7 Other approaches

Furthermore, there also exist strategies to optimize hyperparameters that are based on optimization techniques from artificial intelligence such as tabu search (Cawley 2001), particle swarm optimization (Guo et al. 2008) and evolutionary algorithms (Friedrichs and Igel 2005b). Since none of these strategies use information from previous experiments, meta-data can be added analogously to the SMBO counterpart using in initialization (Gomes et al. 2012; Reif et al. 2012). Another interesting recent proposition is the use of bandit optimization techniques for automatic machine learning (Hoffman et al. 2014).

3 Problem definition

In this section we will formally define the problem of hyperparameter optimization and introduce the notation that will be used in the remainder of the paper. We will follow the notation that was introduced by Bergstra and Bengio (2012), but extend it to account for a more general problem of hyperparameter optimization by also including model choice and other tasks.

Let $\mathcal{D}$ denote the space of all data sets and let $\mathcal{M}$ denote the space of all models. Thus, $\mathcal{D}$ consists of all possible data sets, where instances might have a vector representation, but can also be images, time-series, or similar representations.

We then let $\mathcal{M}$ define the space of all machine learning models. This includes all parametric models, besides also trees and other models with parameters and a specific structure such as neural networks. The configuration space $\mathcal{X}$ encodes the choice of algorithms and hyperparameters. Then, let us define a *general* algorithm $\mathcal{A}$ as a mapping

$$\mathcal{A} : \mathcal{X} \times \mathcal{D} \longrightarrow \mathcal{M} \tag{1}$$

that takes as input a choice of hyperparameter $x \in \mathcal{X}$ and a data set $D \in \mathcal{D}$ to then deliver a model $M \in \mathcal{M}$ learned on the training partition D^{train} of data set D. In this formulation, the model choice as well as the hyperparameter setting are both combined in the choice of x. Additionally, preprocessing tasks, the choice of optimization technique, and other such settings can be treated as hyperparameters. While allowing such treatment may arguably not be the best option, we follow the lead of Thornton et al. (2013). We assume that $\mathcal{X}$ has a fixed dimensionality p for notational purposes, although with new models and learning algorithms being researched everyday, the dimensionality of $\mathcal{X}$ is constantly growing. Given a concrete setting of x, $\mathcal{A}$ searches through the model space $\mathcal{M}$ to find a model that minimizes the empirical loss $\mathcal{L}$ on the training partition of data D, D^{train}, considering a regularization $\mathcal{R}$ to

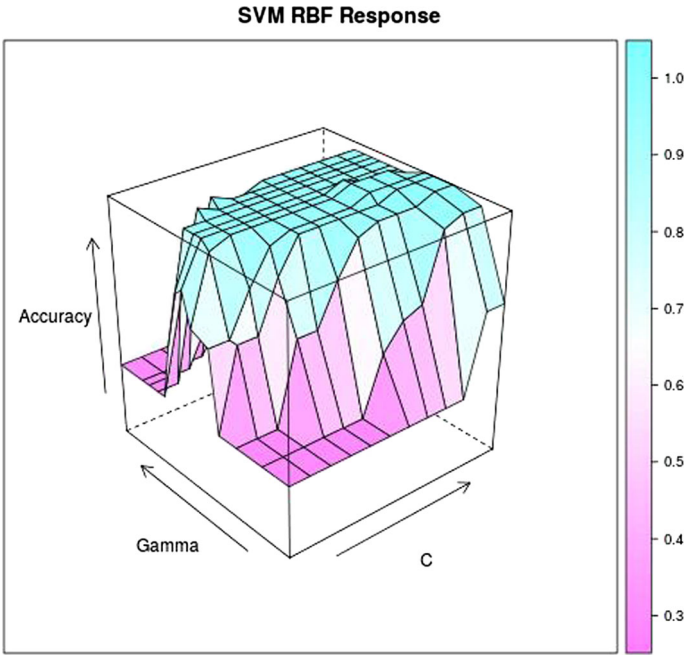

Fig. 1 Response surface of an RBF-SVM on the famous Iris data set. Hyperparameters are the cost of slack (C), and the kernel width γ

avoid overfitting:

$$\mathcal{A}(x, D) = \arg\min_{M \in \mathcal{M}} \mathcal{L}\left(M, D^{\text{train}}\right) + \mathcal{R}(M). \tag{2}$$

Now we can define the task of *hyperparameter optimization* as finding the configuration $x^\star$, that yields a model learned on the training partition which minimizes the loss on the validation partition D^{valid} of the data set:

$$x^\star = \arg\min_{x \in \mathcal{X}} \mathcal{L}\left(\mathcal{A}\left(x, D^{\text{train}}\right), D^{\text{valid}}\right) = \arg\min_{x \in \mathcal{X}} y(x, D). \tag{3}$$

Note that, to shorten the notation a little bit, we introduce the function y instead of the more cumbersome expression in the middle. In the literature, the contours of y are also called *response surface* (Czogiel et al. 2006), a term which we will also use throughout the paper. Figure 1 shows a response surface of an RBF-SVM on the well-known Iris data set.

As y is an unknown black-box function, it cannot be minimized using standard optimization techniques. Usually, y is optimized by doing a grid search, which exhaustively searches through $\mathcal{X}$ for an optimum. Grid search is conducted by defining a finite subset $G \subset \mathcal{X}$ which is usually the Cartesian product of a few points in each dimension of $\mathcal{X}$, $\mathcal{X}_i$,

$$G = \prod_{i=1}^{p} G_i \qquad G_i \subset \mathcal{X}_i \quad |G_i| < \infty \tag{4}$$

and then evaluating y for all of these points. If learning the model takes a lot of time, this exhaustive approach can be a very time-consuming process that results in a lot of useless computations, as usually grid search is not conducted in an adaptive way, where observations of y are taken into account to design the grid. In the recent years, as hyperparameter

optimization has become more and more an issue in machine learning, researchers have used black-box optimization techniques described next.

4 Sequential model-based optimization

Sequential model-based optimization (SMBO) is a technique that iteratively fits a so called *surrogate model*, henceforth denoted by Ψ, on the observed values of y such that $\Psi \approx y$. For brevity, we will denote the set

$$\mathcal{H} = \{(x_1, y(x_1, D)), ..., (x_t, y(x_t, D))\} \tag{5}$$

of t many observed values as *observation history* for data set D. Having estimated a surrogate model, it is queried for not yet observed hyperparameter configurations. Its output will be evaluated by an acquisition function a, which chooses the next hyperparameter optimization to test, which naturally depends on the surrogate prediction as well as $y^{\min}$, which is the best value found so far. This process is repeated until either a certain number of trials has been conducted or the cross-validation performance has achieved an adequate level. An overview of the whole process can be seen in Algorithm 1.

Algorithm 1 Sequential Model-based Optimization

Require: Hyperparameter space $\mathcal{X}$, observation history $\mathcal{H}$, number of trials T, acquisition function a, surrogate model Ψ.
Ensure: Best hyperparameter configuration found.
$y^{\min} \leftarrow \infty$
for $t = 1$ to T **do**
  Fit Ψ to $\mathcal{H}$
  $x \leftarrow \arg\max_{x \in \mathcal{X}} a\left(\Psi(x), y^{\min}\right)$
  Evaluate $y(x)$
  $\mathcal{H} \leftarrow \mathcal{H} \cup \{(x, y(x))\}$
  if $y(x) < y^{\min}$ **then**
    $x^{\min}, y^{\min} \leftarrow x, y(x)$
return $x^{\min}$

Many different surrogate models have been proposed in the recent years, for instance Gaussian processes (Bardenet et al. 2013; Snoek et al. 2012; Swersky et al. 2013; Yogatama and Mann 2014) in different variations. Additionally, random forests (Hutter et al. 2011) and neural networks (Schilling et al. 2015) have been employed. What all these surrogate models have in common is that they are relatively easy and fast to evaluate, at least in comparison to evaluating y, while still being able to learn complex functions. Using linear models to estimate a response surface as in Fig. 1 apparently leads to poor results. Additionally, a growing observation history enables the surrogate model to better approximate the true response surface of y. One key ingredient of surrogate models in the SMBO framework is that, besides predicting the validation performance more accurately, they also have to give an estimation about their uncertainty, i.e. predict a probability distribution instead of a single value. In our work, the surrogate model Ψ will predict for each configuration x a posterior mean $\mu(\Psi(x))$ and a standard deviation $\sigma(\Psi(x))$.

If the surrogate is fitted to the current observation history, we can query it for new hyperparameter configurations to then decide which hyperparameter to choose next; however, this decision is based on an acquisition function a, which helps in balancing both *exploration*

and *exploitation* throughout the hyperparameter optimization. Exploration, on the one hand, is the process of exploring the hyperparameter space, i.e. going into regions where we have next to no observations of y. Naturally, this is what we want to do at the start of the SMBO procedure. Exploitation, on the other hand, is conducted if we have tested enough configurations and believe the surrogate model in its predictions. At this stage, we expect to only test new configurations in the vicinity of the currently best one. It is clear, however, that the search will result in a bad local minimum if no exploration and only exploitation is done. It is easiest to understand how exploitation and exploration are achieved through the acquisition function by having a look at the GP-LCB acquisition function (Srinivas et al. 2010):

$$a_{\text{GP-LCB}}(x) = -\mu(\Psi(x)) + \beta_t \sigma(\Psi(x)). \tag{6}$$

We fix a trade-off β_t between the predicted value and the uncertainty. For higher β_t, we prefer exploration over exploitation because we give x with higher uncertainty a higher weight and vice versa. For a fixed β_t, good candidates x are those with very small predicted posterior mean values or those with a high uncertainty. If the score is dominated by the posterior mean, we are dealing with an exploitation scenario, if it is dominated by the uncertainty, we are dealing with exploration.

While GP-LCB is the best acquisition function to explain how exploration and exploitation can be achieved, we use the expected improvement in our experiments, which achieves a similar effect, but works slightly differently. At the end of this section, we show an example of how a balanced tradeoff between exploration and exploitation is achieved by expected improvement.

As mentioned earlier, a very common choice of acquisition function is the expected improvement (EI) which was firstly used in Jones et al. (1998). The improvement of a new hyperparameter configuration x can be defined as

$$I(x) = \max\left\{y^{\min} - \hat{Y}(x), 0\right\} \tag{7}$$

where $\hat{Y}(x)$ is a random variable that covers our current belief over the performance of x, i.e. $\hat{Y}(x)$ is actually the prediction of our surrogate model Ψ. Then, the expected improvement is simply the expected value of the improvement function given our observation history $\mathcal{H}$:

$$\mathrm{E}[I(x)] = \mathrm{E}\left[\max\left\{y^{\min} - \hat{Y}(x), 0\right\} \,\middle|\, \mathcal{H}\right]. \tag{8}$$

If $\hat{Y}(x)$ follows a Gaussian distribution with mean and variance representing the mean and variance of the surrogate model,

$$\hat{Y}(x) \sim \mathcal{N}\left(\mu(\Psi(x)), \sigma^2(\Psi(x))\right), \tag{9}$$

the expected improvement can be computed analytically. In order to do so, let us first define Z as the best performance $y^{\min}$ standardized by our currently estimated distribution

$$Z = \frac{y^{\min} - \mu(\Psi(x))}{\sigma(\Psi(x))}. \tag{10}$$

Then, $\mathrm{E}[I(x)]$ can be computed as follows

$$\mathrm{E}[I(x)] = \begin{cases} \sigma(\Psi(x))(Z \cdot \Phi(Z) + \phi(Z)) & \text{if } \sigma^2(\Psi(x)) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the Gaussian density and the cumulative distribution function of a standard Gaussian distribution, respectively.

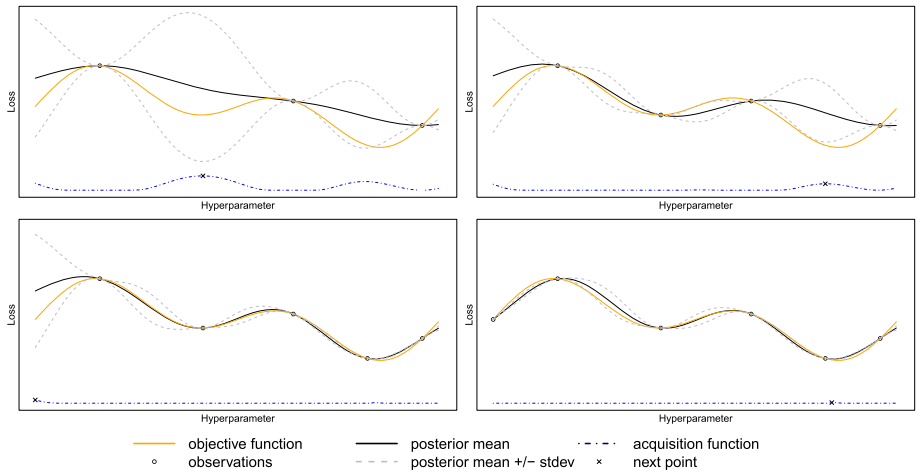

Fig. 2 Overview of SMBO using four trials. In this simple example, we assume that the dimensionality of the hyperparameter configuration space $\mathcal{X}$ is one. The plots show the mapping from the hyperparameter configuration (x-axis) to the corresponding loss (y-axis). We start with three observations. Sequentially, different hyperparameters configurations are evaluated, more knowledge about the function is gathered, and we slowly get closer to the global minimum

An overview of how SMBO works can be seen in Fig. 2, where a Gaussian process (black solid line with uncertainty indicated by dashed gray lines) is initially learned on three data points to approximate the ground truth (yellow solid line). The blue line at the bottom indicates the score of the acquisition function, the cross indicates the maximum of the acquisition function, which is the argument that will be evaluated next by SMBO. In the first three steps, SMBO is doing exploration. The maximum of the acquisition function is with arguments where we have a very high uncertainty about our prediction. In the last step, uncertainty is low and the maximum of the acquisition function is strongly determined by the posterior mea, and thus we are doing exploitation.

4.1 SMBO using meta-information

Using meta-information, i.e. information of hyperparameter performance on another data set, during the process of hyperparameter optimization has attracted a lot of interest within the last years. The motivation is very natural: the more experiments we run on diverse data sets, the better feeling we get for hyperparameters and how they affect the final validation performance. Thus, an experienced researcher usually starts the search for good hyperparameters in a subspace of $\mathcal{X}$ where improvements are likely. In order to use meta-information in SMBO, we now denote the observation history as

$$\mathcal{H} = \left\{\left(x_1, y(x_1, D_1)\right), ..., \left(x_{K_1}, y(x_{K_1}, D_1\right), ..., \left(x_{K_M}, y(x_{K_M}, D_M)\right)\right\}. \tag{12}$$

Note that the hyperparameter configurations evaluated on diverse data sets have to include identical settings.

The majority of work has focused on coming up with surrogate models that can effectively integrate the meta-knowledge, for example by being trained on the observation history prior to starting SMBO on a new data set. In order to do so, we have to add so called *meta-features*, that describe the characteristics of a data set, as otherwise the surrogate model would be

unable to differentiate between instances if the same hyperparameter configuration has been used. Compared to the work in traditional meta-learning, only a few (three or four) meta-features have been used for surrogate models (Bardenet et al. 2013; Yogatama and Mann 2014; Schilling et al. 2015). We extended the number of meta-features in our previous work (Schilling et al. 2016; Wistuba et al. 2016) and will use the same meta-features in this work for all methods. We have listed all those meta-features in Table 1. To simplify the notation, we will from now on assume that by writing D as a variable of y, the meta-features of D are automatically included.

Several surrogate models are explicitly designed for handling meta-data (Bardenet et al. 2013; Yogatama and Mann 2014; Schilling et al. 2015). Given the importance of these approaches for this work, we explain how they work and how meta-data is used in detail in the upcoming subsections.

4.1.1 Surrogate collaborative tuning (SCoT)

Bardenet et al. (2013) were the first to propose a surrogate model in conjunction with meta-data, showing how to learn a single surrogate model over observations from many data sets. Since the same algorithm applied to different data sets leads to loss values that can differ significantly in scale, they recommend tackling this problem using a ranking model instead of a regression model. Finally, they propose to use SVM^{RANK} with an RBF kernel to learn a ranking of hyperparameter configurations per data set. The ranking itself does not provide uncertainty estimations which are needed for the acquisition function, and thus, Bardenet et al. finally fit a Gaussian process to the ranking in order to provide this.

4.1.2 Gaussian process with multi-kernel learning (MKL-GP)

Yogatama and Mann (2014) propose to learn a Gaussian process using meta-data. To overcome the problem of different scales on different data sets, they propose to standardize the loss per data set by subtracting the mean and scaling it to unit variance. Furthermore, they employ a linear combination of a squared exponential kernel with automatic relevance determination (SE-ARD) for observations on the same data set and a nearest neighbor kernel for modeling similarities between data sets. They define the kernel as

$$\begin{aligned} k_{\text{MKL}}\left(\left(x_i, D_k\right), \left(x_j, D_l\right)\right) = &\alpha \delta\left(D_k = D_l\right) k_{\text{SE-ARD}}\left(x_i, x_j\right) \\ &+ (1-\alpha)\, \delta\left(D_l \in \mathcal{N}\left(D_k\right)\right) k_{\text{NN}}\left(x_i, x_j\right) \end{aligned} \tag{13}$$

where the SE-ARD kernel is defined as

$$k_{\text{SE-ARD}}\left(x_i, x_j\right) = \exp\left(-\frac{1}{2}\sum_{k=1}^{p}\frac{\left(x_{i,k} - x_{j,k}\right)^2}{\sigma_k^2}\right). \tag{14}$$

The δ functions returns one if its predicate is true and zero otherwise. The data set similarity kernel is set to

$$k_{\text{NN}}\left(x_i, x_j\right) = 1 - \frac{1}{B}\left\|x_i - x_j\right\|, \tag{15}$$

where B must be chosen such that k_{NN} is always non-negative and $\mathcal{N}(D)$ denotes the set of most similar data sets with respect to a distance function. The distance between two data sets is defined as the Euclidean distance between their meta-features, and is used to determine the neighboring data sets. $x_i \in \mathcal{X}$ is the vectorial representation of the configuration. The tuple (x_i, D_k) indicates that x_i has been evaluated on data set D_k. The similarity between

Table 1 The list of meta-features used in our experiments for all methods

Meta-features	
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

two tuples (x_i, D_k) and $\left(x_j, D_l\right)$ is the weighted sum of the SE-ARD and the NN kernel. The SE-ARD kernel is only considered if x_i and x_j are evaluated on the same data set. The NN kernel is only considered if the settings are evaluated on very similar data sets.

4.1.3 Factorized multilayer perceptron (FMLP)

Schilling et al. (2015) proposed to use a modified multilayer perceptron as a surrogate model. Meta-instances are extended by meta-features and data set indicators. Data set indicators are nothing else but $M + 1$ additional binary predictors, one for each data set. The indicator is one if the meta-instance belongs to the corresponding data set, and zero otherwise. The modified multilayer perceptron uses a special activation function in the first layer. Instead of using a linear signal function, the authors propose

$$\text{logistic}\left(w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} v_i^T v_j x_i x_j\right) \tag{16}$$

where *logistic* is the logistic function,

$$\text{logistic}\,(s) = \left(1 + e^{-s}\right)^{-1}, \tag{17}$$

and $V \in \mathbb{R}^{p \times k}$ are latent variables. This model is based on factorization machines (Rendle 2010), which are very commonly employed prediction models for recommender systems. The underlying idea is to learn a latent representation for each data set to model similarities between data sets.

5 Gaussian processes

As mentioned earlier in this paper, our main focus is on learning Gaussian product ensembles over different parts of the meta-data. However, we first want to remind the reader of the definition and learning of Gaussian processes, and to discuss their advantages and disadvantages.

Given a training data set consisting of inputs (for us the hyperparameters) $X = (x_1, ..., x_n)$ and their associated outputs $y = (y(x_1), ..., y(x_n))$, a Gaussian process assumes that the labels follow a multivariate Gaussian

$$y \sim \mathcal{N}(\mu(X), k(X, X)) \tag{18}$$

where $\mu(X)$ is a mean function that will be set to zero without any loss of generality (Rasmussen et al. 2005), and $k(X, X)$ is a positive semidefinite covariance matrix expressed through a kernel function k that computes the similarity of each pair of instances. A very commonly used kernel function is the squared exponential kernel which is defined as

$$k(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma_l^2}\right) + \delta(x = x')\sigma_y^2, \tag{19}$$

where σ_l is the kernel width and σ_y a small noise constant that is applied on the diagonal to ensure numerical stability. Finally, δ is the indicator function that returns one if its input is true and zero otherwise.

In order to make predictions with a Gaussian process, assume we have training data (X, y) and a new test instance $x_\star$ where the output is unknown. Then, we are interested in the conditional distribution of $y_\star$ given $x_\star$ and the training data. From the definition of Gaussian processes we know that the old and new targets are jointly Gaussian

$$\begin{pmatrix} y \\ y_\star \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K & k_\star \\ k_\star^\top & k_{\star\star} \end{pmatrix}\right), \tag{20}$$

where $K := k(X, X)$, $k_\star := k(X, x_\star)$ and $k_{\star\star} := k(x_\star, x_\star)$ for brevity. Then, the conditional distribution of our test labels is a multivariate Gaussian

$$p(y_\star \mid x_\star, X, y, \theta) = \mathcal{N}\left(\mu(x_\star), \sigma^2(x_\star)\right) \tag{21}$$

with kernel hyperparameters θ and mean and variance (Rasmussen et al. 2005)

$$\mathrm{E}[y_\star] = \mu(x_\star) = k_\star K^{-1} y \tag{22}$$

$$\mathrm{Var}[y_\star] = \sigma^2(x_\star) = k_{\star\star} - k_\star^\top K^{-1} k_\star. \tag{23}$$

Note that the computational expense lies in inverting the kernel matrix K of the training data which has dimensionality $n \times n$. However, once we have estimated the inverse, we can easily predict means and variances for every input that we are interested in.

Another nice aspect of Gaussian processes is that they are *hyperparameter free*, as the kernel hyperparameters θ can effectively be learned by maximizing their marginal likelihood (Rasmussen et al. 2005) which is given by

$$\log p(y \mid X, \theta) = -\frac{1}{2} y^\top K^{-1} y - \frac{1}{2} \log|K|. \tag{24}$$

The above term can be maximized using standard optimization techniques such as gradient ascent. As we will optimize θ on the meta-data prior to starting the hyperparameter optimization for the new data set, we drop them in order to make the notation more compact.

From the nature of SMBO, our observation history is incremented by one additional instance with each SMBO trial that we undertake. For the kernel matrix, this means only one additional column and row, consisting of the kernel function evaluated for the new point and all the old points. However, after adding the new observation, we have to invert K again.

To speed up the overall inversion process, we can use the Cholesky decomposition on K, where K is decomposed as a product of triangular matrices:

$$K = LL^\top. \tag{25}$$

Then, the predicted Gaussian for $y_\star$ resolves to

$$p(y_\star \mid x_\star, X, y) = \mathcal{N}(y_\star \mid k_\star \alpha\ ,\ k_{\star\star} - l^\top l) \tag{26}$$

with $\alpha = \text{solve}\left(L^\top, \text{solve}\,(L, y)\right)$ and $l = \text{solve}\,(L, k_\star)$ for solve being the operation of solving a system of equations. As soon as a new instance has to be included into the Gaussian process, we can simply update the matrix L. This is done by setting the new L_{new} as

$$L_{\text{new}} = \begin{pmatrix} L & 0 \\ l^\top & l_* \end{pmatrix} \tag{27}$$

and setting

$$l_* = \sqrt{k_{\star\star} - \|l\|_2^2 + \sigma_y^2}. \tag{28}$$

This way we effectively reduce the computation from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ where n is the number of observations in the current observation history, due to L being a lower triangular matrix. However, if we are aiming to include a vast amount of meta-data, learning a Gaussian process will become an issue because the run time is still quadratic. We address this issue in the next section.

6 Scalable Gaussian process transfer surrogate framework

In the previous section we discussed the learning of Gaussian processes, where the most computationally expensive step lies in the inversion of the kernel matrix K, which is of size n if we are facing n many training instances. Given the scenario that we are in possession of large scale meta-data, learning a Gaussian process becomes infeasible, as inverting K can only be done in $\mathcal{O}(n^3)$ computations; however, Gaussian processes are a natural choice as surrogate models for SMBO, as they naturally predict uncertainties and are basically hyperparameter free.

Beyond the computational challenges, learning a Gaussian process on all training instances makes the strong assumption that each training instance and data set are equally important. This issue is usually addressed by adding meta-features which leads to an indirect representation of similarity between data sets and their influence. We want to propose a framework that tackles both issues by making Gaussian processes scalable and making the influence of each data set within the meta-data explicit.

Therefore, in order to still learn Gaussian processes, we propose to subdivide the meta-data into M many individual parts and learn a single Gaussian process independently on each of the parts, including a single, additional Gaussian process for all the new observations that we will see during the SMBO trials. Formally, we divide our meta-data

$$X = (X^{(1)}, ..., X^{(M)}) \qquad y = (y^{(1)}, ..., y^{(M)}), \tag{29}$$

in a way where all $X^{(i)}$ are pairwise disjoint. However, instead of taking an arbitrary subdivision of our meta-data, we simply divide it by the data sets we have already observed. This means, for each data set D_i, we create a subset $X^{(i)},\ y^{(i)}$ which contains all meta-instances

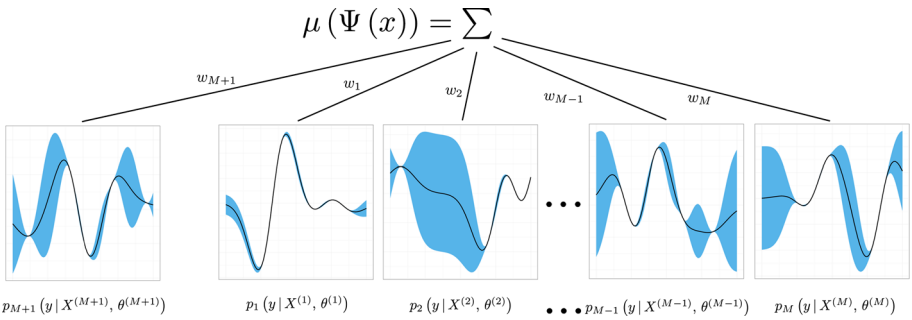

Fig. 3 The proposed framework for our scalable transfer surrogate based on Gaussian processes. A Gaussian process is learned per data set and they are finally combined in a weighted sum

of data set D_i. As a result, we have M Gaussian processes learned, one for each data set, such that for every $i = 1, ...M$

$$p_i\left(y_\star \mid x_\star, X^{(i)}, y^{(i)}\right) = \mathcal{N}\left(\mu_i(x_\star), \sigma_i^2(x_\star)\right). \tag{30}$$

As mentioned earlier, we also learn a Gaussian process for the new observations, which will be updated after every SMBO trial. We will simply use the index $M + 1$ for the target Gaussian process.

Algorithm 2 Scalable Gaussian Process Transfer Surrogate Framework

function TRAIN(X, y)
  Split meta-data by data set:
    $X = (X^{(1)}, ..., X^{(M)})$ $y = (y^{(1)}, ..., y^{(M)})$ ▷ Equation 29
  for $i = 1$ to M **do**
    Estimate $p_i\left(y \mid X^{(i)}, \theta^{(i)}\right) = \mathcal{N}\left(\mu_i(x), \sigma_i^2(x)\right)$. ▷ Equation 21

function PREDICT(x, $p_{M+1}\left(y \mid X^{(M+1)}, \theta^{(M+1)}\right)$)
  ▷ Function called in Algorithm 1 ($\Psi(x)$) to estimate mean and standard deviation.
  $\mu = \frac{\sum_{i=1}^{M+1} w_i \mu_i(x_\star)}{\sum_{i=1}^{M+1} w_i}$ ▷ Equation 31
  $\sigma = \sqrt{\left(\sum_{i=1}^{M+1} v_i \sigma_i^{-2}(x_\star)\right)^{-1}}$ ▷ Equation 32
  return (μ, σ)

We derive our *scalable Gaussian process transfer surrogate framework* (SGPT) by combining all $M + 1$ Gaussian processes into a weighted, normalized sum as sketched in Fig. 3. We define the following mean and precision

$$\mu(x_\star) = \frac{\sum_{i=1}^{M+1} w_i \mu_i(x_\star)}{\sum_{i=1}^{M+1} w_i} \tag{31}$$

$$\sigma^{-2}(x_\star) = \sum_{i=1}^{M+1} v_i \sigma_i^{-2}(x_\star). \tag{32}$$

The final framework is summarized in Algorithm 2. It consists of two different parts. The first involves the training of the individual processes, and the second one combines the

processes for prediction. As mentioned before, training involves dividing the meta-instances in M subsets, one subset for each data set on which we observed evaluations. Thus, every Gaussian process becomes the expert of the respective data set. The prediction uses these experts plus one additional expert that is estimated on the observed performances on the new data set. Based on Eqs. 31 and 32, the mean and uncertainty is estimated. In the following subsections, we will discuss how to derive possible options for choosing w and v which we introduced in Eqs. 31 and 32. Each version is a possible surrogate model Ψ that can be used in SMBO (see Algorithm 1).

6.1 Product of experts

In this section we want to formally derive values for the parameters w and v for the scalable Gaussian process transfer surrogate framework (Algorithm 2). Following our previous work (Schilling et al. 2016), when applying the independence assumption, we can write the joint likelihood in Eq. 21 as a product of individual likelihoods

$$p(y_\star \mid x_\star, X, y) = \prod_{i=1}^{M+1} p_i \left(y_\star \mid x_\star, X^{(i)}, y^{(i)} \right), \tag{33}$$

which is also called a product of experts model and has been introduced by Hinton (1999). Additionally, weighting coefficients β_i have been proposed to use in the product of experts model to derive the *generalized* product of experts

$$p(y_\star \mid x_\star, X, y) = \prod_{i=1}^{M+1} p_i^{\beta_i} \left(y_\star \mid x_\star, X^{(i)}, y^{(i)} \right), \tag{34}$$

where the initial formulation is obtained by setting all $\beta_i = 1$ (Hinton 1999). Usually, the coefficients β_i in the generalized product of experts are chosen to sum up to one.

Computing the product of all these Gaussian densities, we obtain a Gaussian distribution with the following mean and precision:

$$\mu(x_\star) = \sigma^2(x_\star) \sum_{i=1}^{M+1} \beta_i \sigma_i^{-2}(x_\star) \mu_i(x_\star) \tag{35}$$

$$\sigma^{-2}(x_\star) = \sum_{i=1}^{M+1} \beta_i \sigma_i^{-2}(x_\star). \tag{36}$$

Substituting the precision into the formula for the mean, the predicted mean resolves to

$$\mu(x_\star) = \frac{\sum_{i=1}^{M+1} \beta_i \sigma_i^{-2}(x_\star) \mu_i(x_\star)}{\sum_{i=1}^{M+1} \beta_i \sigma_i^{-2}(x_\star)}, \tag{37}$$

which is a sum of means, weighted by the product of β_i and the individual precisions. For our experiments, we set

$$\beta_i = \frac{1}{M+1} \quad \forall i = 1, \ldots, M+1, \tag{38}$$

which does not influence the predicted mean as the terms cancel out; however, this effectively increases the uncertainty which the general model of experts usually tends to underestimate (Deisenroth and Ng 2015). To sum up, generalized products of experts are an instance of

scalable transfer surrogates when setting

$$w_i = \beta_i \sigma_i^{-2}(x_\star) \tag{39}$$

$$v_i = \beta_i \tag{40}$$

as weight parameters in Algorithm 2.

6.2 Kernel regression

In the previous section, we have derived parameters w and v for Algorithm 2 under the assumption that every data set has equal importance for the task of finding optimal hyperparameter configurations for our new data set D_{M+1}; however, in order to find good hyperparameter configurations on a new data set D_{M+1}, it is intuitive to give more weight to the influence of data sets that have a similar hyperparameter response surface. Hence, setting w_i to larger values for these experts to increase their influence makes a lot of sense. Assuming that we know the similarity $k\left(\chi_i, \chi_j\right)$ between two data sets D_i and D_j, where χ_i and χ_j are the data set descriptors of D_i and D_j, respectively, we proposed to set the value of w_i to the similarity between the data set D_i and the new data set D_{M+1} (Wistuba et al. 2016):

$$w_i = k\left(\chi_i, \chi_{M+1}\right). \tag{41}$$

The concrete kernel that we apply is the Epanechnikov quadratic kernel (Nadaraya 1964)

$$k_\rho\left(\chi_i, \chi_j\right) = \gamma\left(\frac{\left\|\chi_i - \chi_j\right\|_2}{\rho}\right), \tag{42}$$

where the γ function is given by

$$\gamma(t) = \begin{cases} \frac{3}{4}\left(1 - t^2\right) & \text{if } t \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

and $\rho > 0$ is the bandwidth. Setting w like this, our scalable Gaussian process transfer surrogate framework is now equivalent to kernel regression with the Nadaraya Watson kernel-weighted average for the mean prediction. Furthermore, we propose to rely on the uncertainty of the surrogate model for the new data set only:

$$v_i = \begin{cases} 1 & i = M+1 \\ 0 & \text{otherwise.} \end{cases} \tag{44}$$

We would like to use the true similarity between the new data set and all other data sets, but since this is not available, we will evaluate two different common techniques to approximate it. One is based on meta-features, i.e. simple, statistical or information theoretic properties that are extracted from the data set which describe the data set (Bardenet et al. 2013; Reif et al. 2012; Smith-Miles 2009). We use the meta-features listed in Table 1. For a more detailed explanation of meta-features, we refer to Michie et al. (1994).

Using these meta-features, however, has one drawback: the meta-features are constant, which means that the knowledge of the target data set enters the model only via the target Gaussian process which is updated after every trial. Therefore, we propose an alternative using a pairwise hyperparameter performance comparison (Leite et al. 2012; Wistuba et al. 2015). The idea is to select pairs $\left(x_i, x_j\right)$ of evaluated hyperparameter configurations on the new data set D_{M+1} and count how often D_{M+1} and another data set D_k agree on the ranking of

these configurations. After evaluating t many hyperparameter configurations during SMBO on the new data set, we estimate the data set descriptors for each data set D_k as

$$(\chi_k)_{j+(i-1)t} = \begin{cases} \frac{1}{t(t-1)} & \text{if } \mu_k(x_i) > \mu_k(x_j) \\ 0 & \text{otherwise} \end{cases}. \tag{45}$$

While the value of $y(\cdot, D_{M+1})$ is known for these t hyperparameter configurations, this is not necessarily true for the data sets $D_1, \ldots, D_M$. Hence, we use the prediction of each individual expert instead.

Computing the Euclidean distance of two meta-feature vectors then yields the number of discordant pairs normalized by dividing by the number of all pairs. This is basically a distance function based on the Kendall rank correlation coefficient (Kendall 1938). In this way, during the SMBO process the coefficients are adapted after each iteration, where the data sets that agree on more hyperparameter pairs with the target data set are weighted higher. This has been shown to improve the performance drastically.

7 Transfer acquisition function framework

The transfer surrogate models of the previous section provide very good results as we will see in the experiments section. Nevertheless, these models face two important problems: first, each data set has different scales of evaluation scores which are reconstructed by each of the experts, and thus, the weighted average will likely not meet the scale of the new data set. One way to overcome this is to normalize the performance meta-data by data set and also create an approximated normalization of the new data set. This is more a work-around than a solution because the problem remains to some degree and the approximated normalization for the new data set is inaccurate in the very beginning.

The second problem stems from the fact that transfer surrogate models assume that the meta-data is equally important throughout the whole optimization process. It seems natural, however, that with more knowledge about the new data set the influence of meta-data could diminish. In fact, we have seen that it is important to reduce the influence over time as soon as all available knowledge is consumed (Wistuba et al. 2016). Thus, we propose to transfer the meta-data within the acquisition function instead of the surrogate model which means that we will use a surrogate model that directly reconstructs the response surface of the new data set, and does not consider any meta-data. We reconstruct this response surface by using a Gaussian process that is trained on all available observations of the response surface for the new data set. We propose a *transfer acquisition function framework* (TAF) that is very similar to the aforementioned transfer surrogate framework presented in Algorithm 2. We define a novel acquisition function that incorporates meta-data as

$$a(x) = \frac{w_{M+1}\mathrm{E}\left[I_{M+1}(x)\right] + \sum_{i=1}^{M} w_i I_i(x)}{\sum_{i=1}^{M+1} w_i} \tag{46}$$

with

$$I_i(x) = \max\left\{y_{D_i}^{\min} - \hat{Y}_i(x), 0\right\}, \tag{47}$$

where $\hat{Y}_i(x) \sim \mathcal{N}\left(\mu_i(x), \sigma_i^2(x)\right)$ and $y_{D_i}^{\min}$ is the best value achieved so far on D_i. Thus, the acquisition function becomes the weighted average of the expected improvement (see Eq. 8) on the new data plus the predicted improvement on all other data sets from previous

experiments. While the framework looks very similar to the one proposed in Eqs. 31 and 32, the implications are entirely different. As we previously tried to learn a mapping between data characteristics and algorithm and hyperparameter behaviour from the meta-data, we now try to learn a mapping between the data characteristics and the improvement. Instead of predicting the performance for each hyperparameter configuration and hence minimizing a regression loss by employing meta-data, each hyperparameter configuration is scored by two components: first, the expected improvement on the new data set is taken into account, which leads to a high uncertainty especially for the early trials; secondly, the predicted improvement on the new data set is taken into account, which leads to a high uncertainty especially for the early trials; secondly, the predicted improvement that the hyperparameter configuration has achieved on other data sets is considered, where both components complement each other. In the early phase of the optimization, when the expected improvement provided by the surrogate for the new data set is still very unreliable, the improvement prediction from the meta-data favors hyperparameter configurations that have good average performance. Over time, more information about the new data set is collected and the expected improvement prediction becomes reliable. At this point, most of the meta-data has been consumed, i.e. many hyperparameter configurations in regions that have been good in previous experiments have been tried and thus, the improvement is comparably small. Thus, the meta-data starts to play a minor role in the acquisition function.

The weights for the transfer acquisition function framework can be set analogously to the propositions in Sects. 6.1 and 6.2.

8 Experiments and results

8.1 Meta-data set creation

We have created in total two meta-data sets, both for the purpose of learning classification models.

The first meta-data set contains the predictive hyperparameter performances for running an SVM on 50 different data sets all taken from the UCI repository. We selected the data sets at random and decided to use an SVM as a classifier since it is one of the most popular classification tools. We learn a linear SVM, an SVM with RBF kernel, and lastly an SVM with a polynomial kernel. The hyperparameters are then the choice of kernel, the cost of the slack variables C and—depending on which kernel we choose—the kernel width γ for RBF and the degree d for the polynomial kernel. We chose C from $C \in \left\{2^{-5}, \ldots, 2^{6}\right\}$, γ was searched in $\gamma \in \left\{10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20, 50, 10^{2}, 10^{3}\right\}$ and the polynomial degree was optimized over $d \in \{2, \ldots, 10\}$. This are 288 different configurations per data set in total. If a data set was already split, we merged all splits and created a new 80% training and 20% validation split. For running the SVM we used the implementation by Tsochantaridis et al. (2004). The creation of this meta-data set took 160 CPU hours.

The other meta-data set is extended so that it contains both hyperparameter performance on different data sets and performance of a set of different algorithms. In order to accomplish this, we used WEKA (Holmes et al. 1994) to run 19 different classifiers on 59 data sets for a total of 21,871 hyperparameter configurations to evaluate per data set. An overview of the employed classifiers can be seen in Table 2. In total, this sums up to roughly 1.3 million experiments. The overall computation of this meta-data set took about 900 CPU hours.

The meta-target for both meta-data sets is the classification error. We cannot assume that all approaches would work for other error metrics such as the logarithmic loss. We did not

Table 2 Overview of all classifiers used within the WEKA meta-data set

REP tree	Random tree	Random forest
LMT	J48	Decision stump
ZERO_R	PART	ONE_R
RIPPER	Decision table	KSTAR
IBK	SMO	Simple logistic
MLP	Logistic regression	Naive bayes
Bayes net		

conduct experiences for other loss metrics but we do not expect different results since the problem remains the same: finding the global minimum of the function y.

In a leave-one-data-set-out cross-validation, we choose one data set as the target data set. No meta-instance of this data set is available at the beginning of the search and needs to be acquired by SMBO. To show that the optimization strategies can deal with yet unobserved hyperparameter configurations, only a subset of all meta-instances is used for training purposes. The evaluation on meta-test is done using all meta-instances of the target data set. Thus, all methods also need to evaluate hyperparameter configurations they have never seen in their meta-data.

8.2 Competing optimization strategies

We compare our proposed optimization strategies to a large set of state-of-the-art optimization strategies which will be described in detail.

Random Search This is a relatively simple baseline that chooses hyperparameter configurations at random. Nevertheless, Bergstra and Bengio (2012) have shown that this strategy can outperform grid search, especially for algorithms with hyperparameters that have a low effective dimensionality.

Independent Gaussian Process (I-GP) The use of Gaussian processes as a surrogate model goes back to the paper of Jones et al. (1998) from 1998. It was first proposed to be applied to the problem of hyperparameter optimization for machine learning by Snoek et al. (2012) under the name Spearmint. In our experiment we used a squared-exponential kernel with automatic relevance detection (SE-ARD) and also for all other optimization methods that are based on a Gaussian process. This surrogate model does not use any knowledge from previous experiments.

Independent Random Forest (I-RF) This surrogate is very similar to I-GP but uses a random forest instead of a Gaussian process. It was proposed by Hutter et al. (2011) under the name SMAC and is applied in auto-sklearn (Feurer et al. 2015) and Auto-WEKA (Thornton et al. 2013).

Initialization for I-GP and I-RF (I-GP (init) and I-RF (init)) Because I-GP and I-RF do not consider any meta-data, we evaluate both surrogate models also with a meta-initialization (Wistuba et al. 2015).

Surrogate Collaborative Tuning (SCoT) SCoT (Bardenet et al. 2013) is the first transfer surrogate model proposed. Furthermore, it also tries to rank the hyperparameter configurations instead of reconstructing the hyperparameter surface. It uses SVM^{RANK} to predict a ranking. Since the ranker does not provide any uncertainties, a Gaussian process is fitted on the output of the ranker. The authors originally proposed to use the RBF-kernel for SVM^{RANK} but due

to computational complexity we follow the lead of Yogatama and Mann (2014) and use the linear kernel instead.
Gaussian Process with Multi-Kernel Learning (MKL-GP) Yogatama and Mann (2014) proposed another transfer surrogate model. This transfer surrogate model learns a Gaussian process with a specific kernel combination on all instances. The kernel is a linear combination of the SE-ARD kernel and a kernel modelling the similarity between data sets based on a set of meta-features. To tackle the problem of different scales of hyperparameter response surfaces for different data sets, they propose to normalize the target.
Factorized Multilayer Perceptron (FMLP) FMLP (Schilling et al. 2015) is another transfer surrogate model that uses a specific neural network to learn the similarity between data sets implicitly in a latent representation.
Scalable Gaussian Process Transfer Surrogate (SGPT) This is the framework we propose in this work. We distinguish different instances of it depending on how we choose the weights. SGPT-PoE is the version that chooses the weights according to Sect. 6.1 and hence is based on product of experts. Then we included also the kernel regression method introduced in Sect. 6.2 with the meta-feature data set descriptors (SGPT-M) and with the pairwise hyperparameter performance descriptors (SGPT-R).
Transfer Acquisition Function (TAF) In Sect. 7 we proposed an acquisition function that makes use of meta-data. We combine it with the surrogate model as used by I-GP and distinguish different versions depending on which weights are chosen as for SGPT.

The kernel parameters used in the Gaussian process are learned by maximizing the marginal likelihood on the meta-training set (Eq. 24). All hyperparameters of the tuning strategies are optimized in a leave-one-data-set-out cross-validation on the meta-training set.

The reported results were estimated using a leave-one-data-set-out cross-validation and are the average of ten repetitions. For strategies with random initialization (Random, I-GP, I-RF), we report the average of over thousand repetitions due to higher variance. For those strategies that use meta-features, we use those meta-features that are described in Table 1.

8.3 Evaluation metrics

We compare all optimization strategies with respect to three different evaluation measures.

Average rank

At each time step t all optimization strategies are ranked for each data set D_i according to their best score achieved, the better the score, the smaller the rank. In case of ties, the average rank is used. Finally, the ranks are averaged over all data sets yielding the average rank.

Average distance to the global minimum

For an optimization strategy the average distance to the global minimum (ADTM) at time step t is defined as follows. The distance of the best score achieved for each data set D to the best score on this data set is computed. The final score is the average over all data sets $D \in \mathcal{D}$. To account for different scales on different data sets, the hyperparameter performance is scaled to [0, 1]. Formally,

$$\text{ADTM} = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \min_{x \in \mathcal{X}_t} \frac{y_D(x) - y_D^{\min}}{y_D^{\max} - y_D^{\min}} \tag{48}$$

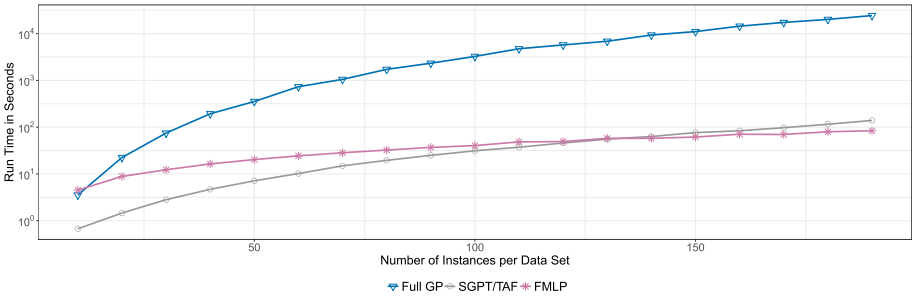

Fig. 4 SGPT (and hence also TAF) is clearly outperforming the state-of-the-art that is training a single Gaussian process on the full meta-data with respect to scalability. FMLP, which is based on a neural network, has a linear training time and provides faster results for larger data sets

where $y_D^{\min}$ and $y_D^{\max}$ are the best and worst performance on the precomputed grid, respectively. $\mathcal{X}_t$ is the set of all hyperparameter configurations tried until time step t. For this evaluation metric it holds that lower values are better and zero is the potential optimum.

Fraction of unsolved data sets

This metric estimates the proportion of data sets in which the optimum according to the precomputed grid has not been found. The evaluation metric ranges between 1 if no optimum has been found to 0 if every optimum is found.

8.4 Scalability experiment

As discussed in Sect. 5, Gaussian processes are computationally expensive. Training time is cubic in the number of training instances and still quadratic when updating it. Our proposed surrogate model SGPT makes use of Gaussian processes in a scalable way. Given d data sets, where on each of these n observations of hyperparameters performance have been made. A typical way of using meta-data for SMBO (Bardenet et al. 2013; Yogatama and Mann 2014) is to train Gaussian process on all instances which has an asymptotic training time of $\mathcal{O}\left(d^3n^3\right)$. We propose to learn for each data set an independent Gaussian process which reduces the training time to $\mathcal{O}\left(dn^3\right)$ which is no longer cubic in the number of data sets. Still, the complexity of both methods is cubic in the number of instances per data set. In an empirical evaluation we show that our method is nevertheless feasible while the state-of-the-art exceeds an acceptable run time.

We have created an artificial meta-data set with $d = 50$ data sets and 5 hyperparameters. The number of instances per data set n varies from 10 to 190. We estimated the run time for a Gaussian process on the full data and SGPT for different n. The results are visualized in Fig. 4. At a point where the Full GP needs almost 7 hours of training, SGPT needs only about 2 minutes. One can even consider to further improve the scalability by learning multiple Gaussian processes per data set. To achieve this, the subsets $X^{(i)}$, $y^{(i)}$ defined in Eq. 29 have to be divided further. One could for example learn an individual Gaussian process for each of the three SVM kernels and then apply the method of Sect. 6.1.

As discussed earlier, the cubic run time make Gaussian processes unattractive for large meta-data sets. Hence, our main goal was to achieve run times for Gaussian processes that are competitive to other models such as neural networks as used by FMLP. Figure 4 shows that our approach needs time very similar to FMLP, for fewer instances, it is even faster.

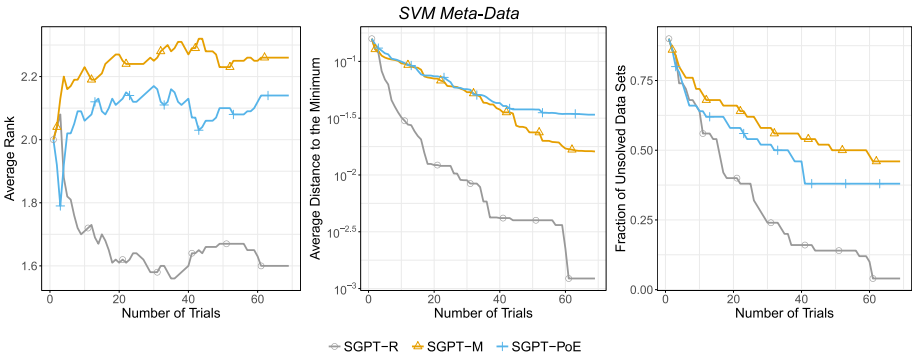

Fig. 5 SGPT-R is outperforming the other two approaches due to the decaying influence of the meta-data for the task of hyperparameter optimization

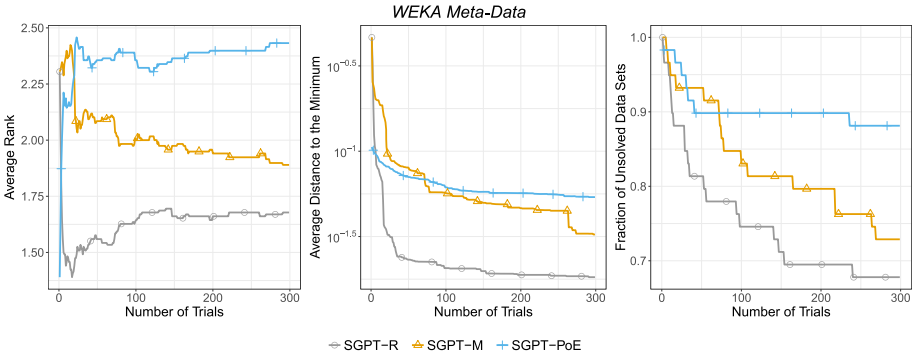

Fig. 6 The adaptive weights also allows SGPT-R to outperform its variants for the task of combined algorithm selection and hyperparameter optimization

8.5 Predictive performance in SMBO

We were able to provide theoretical and empirical evidence that our method is scaling better than the current state-of-the-art methods. Now our aim is to provide empirical evidence that our proposed methods are also competitive for the task of hyperparameter optimization as well as combined algorithm selection and hyperparameter optimization in terms of prediction.

First, we compare our proposed variations of SGPT and explain the results in Sect. 8.5.1. Then, in Sect. 8.5.2, we compare SGPT to six state-of-the-art methods. Finally, we analyze our proposed acquisition function and empirically compare it to its SGPT counterpart in Sect. 8.5.3. In Sect. 8.5.5, we compare the best methods in a final comparison. We use Sect. 8.5.5 to conduct a significance analysis of our results and conclude the experimental section with a comparison considering the training time in Sect. 8.5.6.

8.5.1 Evaluating the scalable gaussian process transfer surrogate framework

In Sect. 6, we derived or proposed three different ways of using meta-data in SGPT, depending on how we set the weights. Before comparing to the state-of-the-art methods, our aim is to focus on these three variations. We evaluated all methods for the task of hyperparameter optimization as well as for the task of combined algorithm selection and hyperparameter

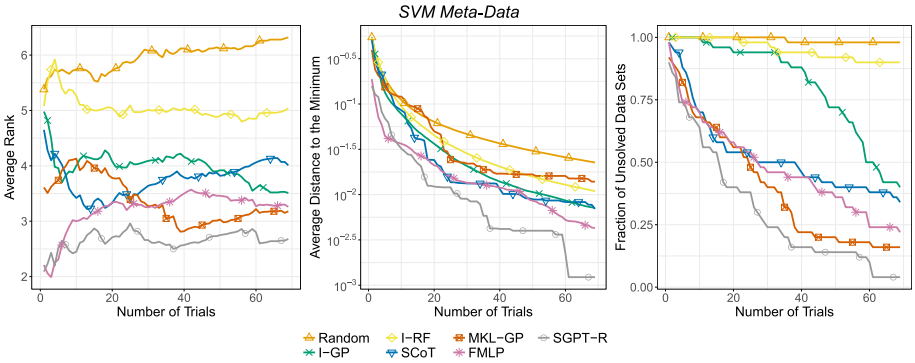

Fig. 7 Our proposed approach SGPT-R is outperforming all competitor methods with respect to all three metrics. For all metrics, the lower the better

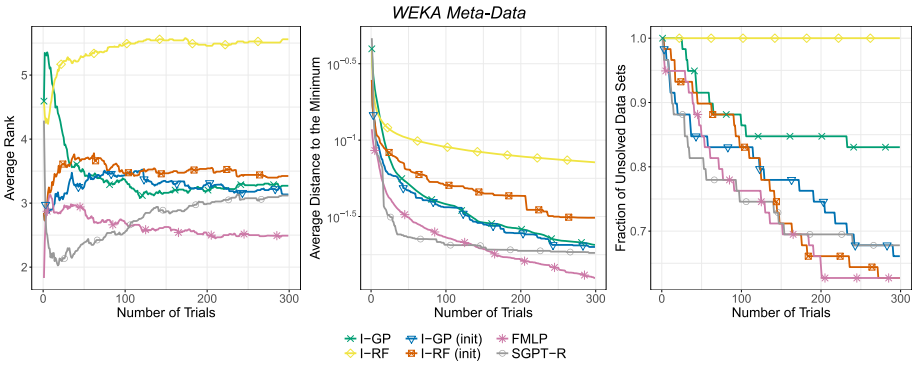

Fig. 8 SGPT-R is outperforming the competitor methods for the task of combined algorithm selection and hyperparameter optimization only in the first iterations. Then, FMLP provides the best results. On this larger meta-data set we were not able to compare to the methods based on a GP trained on the whole meta-data (i.e. SCoT and MKL-GP)

optimization. The results are presented in Figs. 5 and 6. SGPT-R is outperforming the other two alternatives. Our explanation for this is the use of adaptive weights. Firstly, they enable the method to quickly identify data sets that behave similarly and additionally, the influence of meta-data decays over time. Hence, as soon as enough knowledge is gathered from the meta-data, the system is able to rely stronger on the predictions of the surrogate for the new data set. This insight was one of our key motivations for proposing TAF. In the following, we will not use SGPT-M and SGPT-PoE in further comparisons to avoid overcrowded figures.

8.5.2 Comparison to other competitor methods

We saw in the last section that SGPT-R is the best variation of our SGPT framework. We compare it on the task of hyperparameter optimization as well as the task of combined algorithm selection and hyperparameter optimization. As before, we use all three evaluation measures in the comparisons. The results for the task of hyperparameter optimization are presented in Fig. 7. SGPT-R outperforms all competitor methods with respect to all three evaluation metrics.

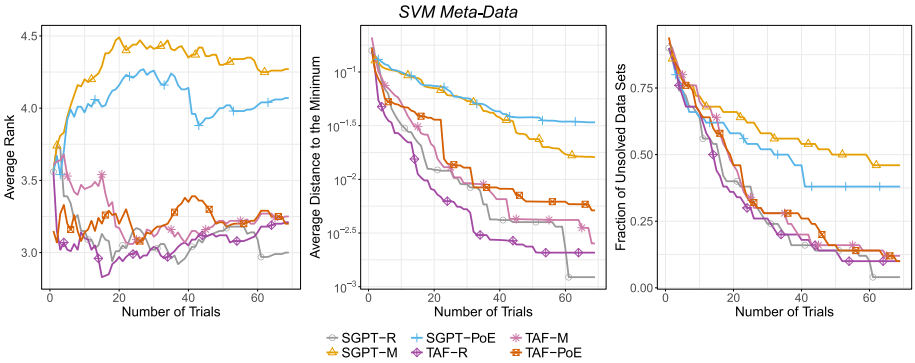

Fig. 9 TAF and SGPT deliver similar competitive results on this meta-data set

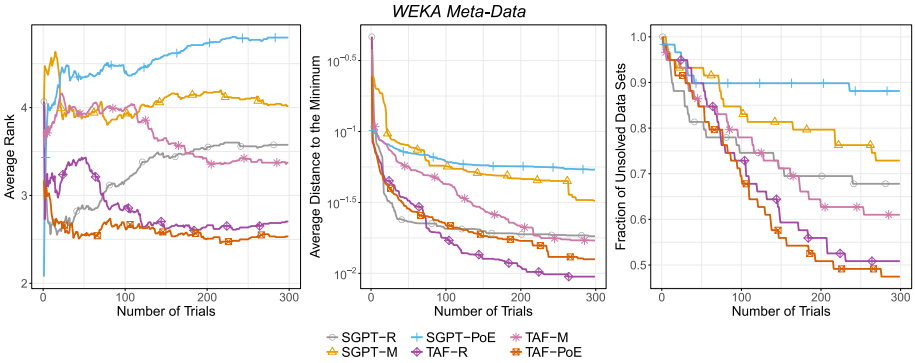

Fig. 10 TAF provides a clear improvement over SGPT thanks to its adaptive use of meta-data and better way of dealing with different data set scales

The results for the task of combined algorithm selection and hyperparameter optimization are presented in Fig. 8. For this task, we do not compare our methods to SCoT and MKL-GP because they are based on a Gaussian process which is trained on the full meta-data set. Since the meta-data set is too large, we were not able to conduct these experiments. Instead, we compare our methods to a Gaussian process and a random forest that use a meta-initialization.

SGPT-R provides very good results for the first trials but then it is not able to further progress and find better hyperparameter configurations. The likely reason is that the WEKA meta-data set is a more challenging optimization problem than the SVM meta-data set. Hence, more trials are needed in comparison to the SVM meta-data, which consequently increases the number of discordant pairs such that the distance of the new data set to the others is increasing too quickly. Then, following the definition of Eqs. 42 and 45, meta-data will not be considered any more and our method performs like a Gaussian process without meta-data. Thus, the strongest competitor FMLP is able to outperform it after some time. Also the other methods are getting close to the performance for SGPT.

8.5.3 Evaluating the transfer acquisition framework (TAF)

Our motivation for introducing TAF was to get rid of the problem of differently scaled data sets and the question how to adaptively employ meta-data. It is a direct extension of SGPT

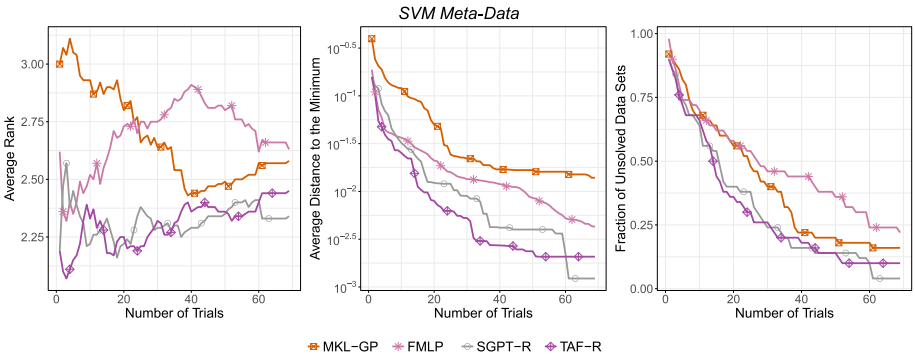

Fig. 11 SGPT-R and TAF-R provide similar performances by outperforming the other competitors

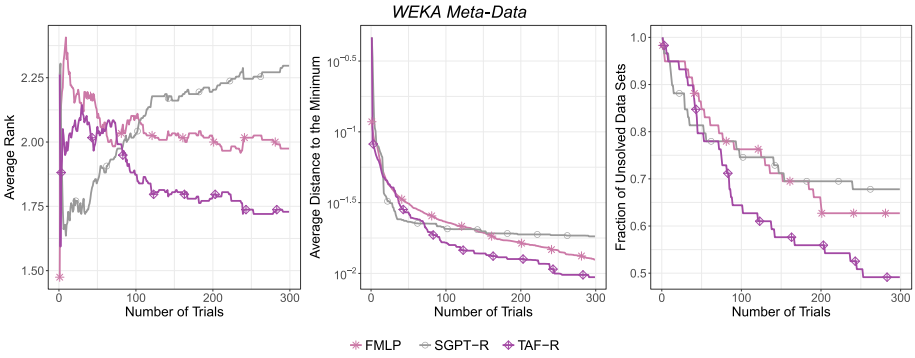

Fig. 12 On this data set TAF-R is able show that it is more robust than SGPT-R

that uses meta-data in the acquisition function instead of by the surrogate model. To provide empirical evidence that we overcame the problems faced by many transfer surrogate models, we compare TAF first to the different versions of SGPT. As a reminder, the postfix "-PoE" is used for the variant that chooses the weights according to Sect. 6.1 and is based on product of experts. The postfixes "-M" and "-R" indicate the variants based on kernel regression with the meta-feature data set descriptors pairwise hyperparameter performance descriptors, respectively (Sect. 6.2). We have conducted our experiments on our two meta-data sets and obtained the results summarized in Figs. 9 and 10. TAF-R obtained best results. We see a strong improvement of TAF-M and TAF-PoE over SGPT-M and SGPT-PoE, respectively. As mentioned before, especially SGPT-M uses static weights that do not account for the progress of the optimization process. Hence, the impact of the meta-data remains constant which is unfavorable. The weights of TAF-M are the same and thus also remain constant but the improvement on the meta-data shrinks over time as better x have been found. Hence, we observe an adaptive use of meta-data that leads to the large improvement over SGPT-PoE and SGPT-M. Since SGPT-R has already a mechanism that decays the influence of the meta-data, SGPT-R and the TAF approaches provide similar good results on the SVM meta-data set in Fig. 9 which likely results from the better way of tackling the problem of different scales.

The improvement of TAF-R over SGPT-R becomes significant on the WEKA meta-data set as presented in Fig. 10. We saw in the previous experiments that SGPT-R is able to have a good start on this problem but then somehow gets stuck and has problems to further improve

the solution. TAF-R is able to overcome this issue. Having a similar good start, it is able to continue its search and finds the optimum for more than 50% of the data sets within 300 trials.

8.5.4 Overall comparison

We conclude our experiments by comparing the best methods from the TAF and SGPT frameworks with the strongest state-of-the-art competitors. Figure 11 presents the results for finding optimal hyperparameter configurations for a kernel SVM. Both, TAF-R and SGPT-R are outperforming the competitor methods with respect to all three metrics and are approximately comparable. For the WEKA data set the story is different. In Fig. 12 it becomes clear that SGPT-R has a good start, but then fails to further improve the solution. FMLP is able to outperform it after some time and also other simpler competitor methods are getting comparable performances. Thanks to the better way of adaptive meta-data usage and handling of different data set scales, TAF-R is once again the best method. While it is a little bit worse than SGPT-R at the beginning, TAF-R quickly outperforms SGPT-R. Additionally, TAF-R is always stronger than the runner-up method FMLP. Thus, we consider our motivation confirmed in proposing TAF. Even though it looks very similar to SGPT, TAF-R is more robust due to the aforementioned reasons. We are able to prove theoretically and show empirically that our approach is faster than the Gaussian process that has been learned on the full meta-data. In our experiment we can show that the run time is approximately as fast as the fastest transfer surrogate models. Hence, our proposed approach is not only effective, it is also efficient.

8.5.5 Significance analysis

Friedman Test. To analyze the significance of our results, we apply a Friedman test (1937, 1940) with the corresponding post-hoc tests. This was proposed by Demšar (2006) who analyzed multiple statistical tests and as a conclusion proposed this method for comparisons of multiple methods over multiple data sets. A very important aspect is that Demšar proposed this test for classifiers and not for optimization methods, as there are vital differences between these problems. Good methods might find a good solution, or even the best in a shorter time period than other worse methods; however, if you provide these methods with more time, they inevitably will catch up. It is clear that with enough trials there will be no significant difference between a random strategy and any other strategy. Since we are not aware of any better solution to analyze the statistical significance of our results, we follow Demšar's recommendation for all methods nonetheless.

For our analysis we added one further method that we call *Best*. This is an oracle method that chooses for any data set the best performing hyperparameter configuration and algorithm with the first trial.

We estimated the critical value for both meta-data sets and rejected the null hypothesis that the measured average ranks are not significantly different from the mean rank for $\alpha = 0.05$. Since no method singled out, we use the Nemenyi test (1962) for pairwise comparisons. We computed the critical values for $p = 0.05$. We conducted the significance tests for four different scenarios: after the first and 30th trial on the SVM meta-data set and after the 30th and 200th trial on the WEKA meta-data set. The results are visualized in Figs. 13 and 14. Groups of methods that are not significantly different are connected.

On the SVM meta-data set the critical difference is 1.7 for $p = 0.05$. The post-hoc tests detect significant differences between TAF-R and SGPT-R to all methods but FMLP

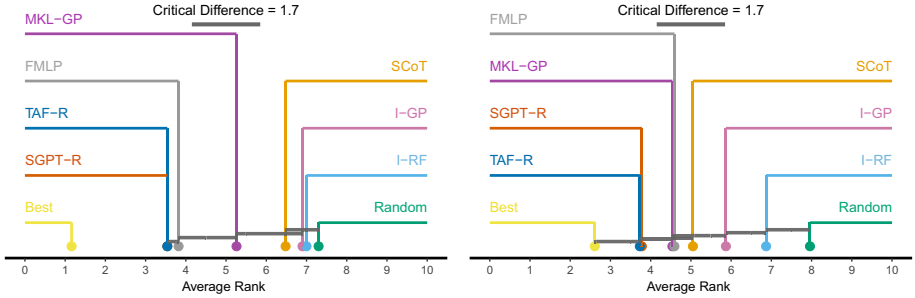

Fig. 13 Comparison of all optimizers against each other with the two-tailed Nemenyi test on the SVM meta-data set. Groups that are not significantly different (at $p = 0.05$) are connected. The left plot shows the results after the first trial, the right plot after the 30th trial. After 30 trials there is no significant difference between our proposed methods TAF and SGPT and the potentially best method

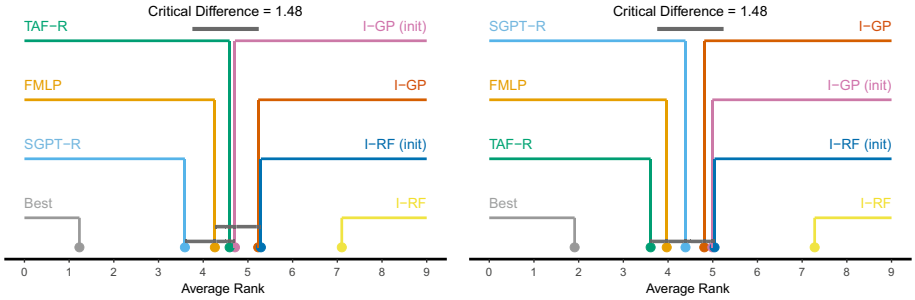

Fig. 14 Comparison of all optimizers against each other with the two-tailed Nemenyi test on the WEKA meta-data set. Groups that are not significantly different (at $p = 0.05$) are connected. The left plot shows the results after 30 trials, the right plot after 200 trials

after the first trial and significant differences to I-GP, I-RF and Random after 30 trials. As discussed earlier, the optimal value is fixed such that more trials mean that more methods achieve good solutions. Nevertheless, even after 30 trials any method that is using meta-data still outperforms any other method that is not. Finally, after 30 trials there is no significant difference between our proposed methods and the potentially best method Best.

On the WEKA meta-data set the critical difference is 1.48 for $p = 0.05$. The post-hoc tests detect that I-RF is significantly worse than any other method. After 30 trials, SGPT-R provides significantly better results than I-GP, I-RF and I-RF with initialization. The experimental data is not sufficient to reach any conclusion regarding FMLP, TAF-R and I-GP with initialization. After 200 trials, no statistically significant statement can be made about all methods but I-RF and Best.

Bayesian Hierarchical Test In addition to the Friedman test, we conduct a second significance test, namely the recently proposed Bayesian hierarchical test (Corani et al. 2016). As proposed by Corani et al., we assume that two classifiers are practically equivalent if their mean difference of accuracy lies within the interval $(-0.01, 0.01)$. The test produces three probabilities, the first two being whether one or the other classifier is significantly better than the other. Additionally it provides a probability whether both classifiers are practically equivalent as their difference of accuracies lies within the region of practical equivalence (rope). Figures 15 and 16 present the posterior probabilities of the hierarchical model for

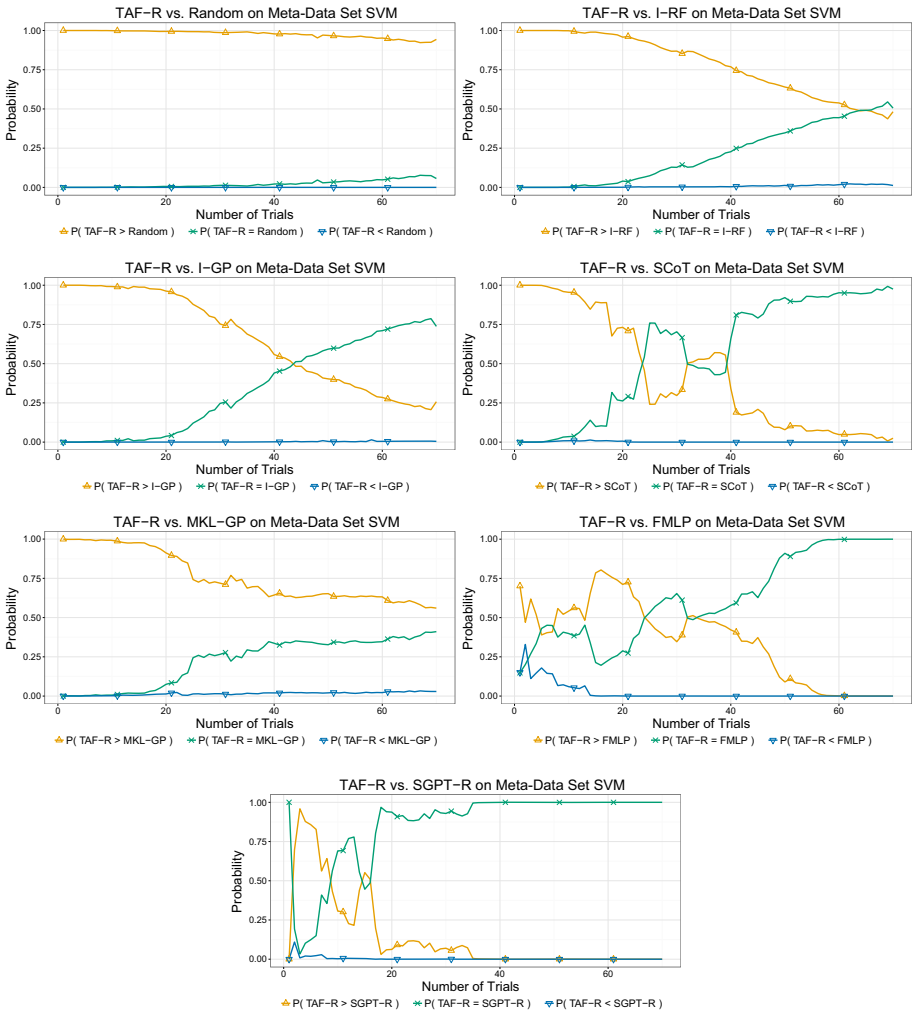

Fig. 15 Comparison of TAF-R against all other optimizers using the Bayesian hierarchical test on the SVM meta-data set. Probabilities above 95% are significant. With increasing number of trials the probability for equality of both methods increases due to a limited number of considered configurations

each trial. For a significance level of $\alpha = 0.05$, the results are significant when the posterior probability is above 95%. However, the advantage of the Bayesian hierarchical test is that posterior probabilities can be meaningfully interpreted even when they do not exceed the 95% threshold (Corani et al. 2016), simply by computing the odds ratios of the different outcomes. We see that TAF-R provides, sometimes significant, better results than Random, I-GP, I-RF, SCoT and MKL-GP on both meta-data sets. In comparison to FMLP and SGPT-R the results are not significant. However, the test indicates that these methods are either equal or TAF-R is better. There is little indication that TAF-R performs worse, as the respective probability only increases when compared to SGPT-R on the WEKA meta-data set in some of the early trials.

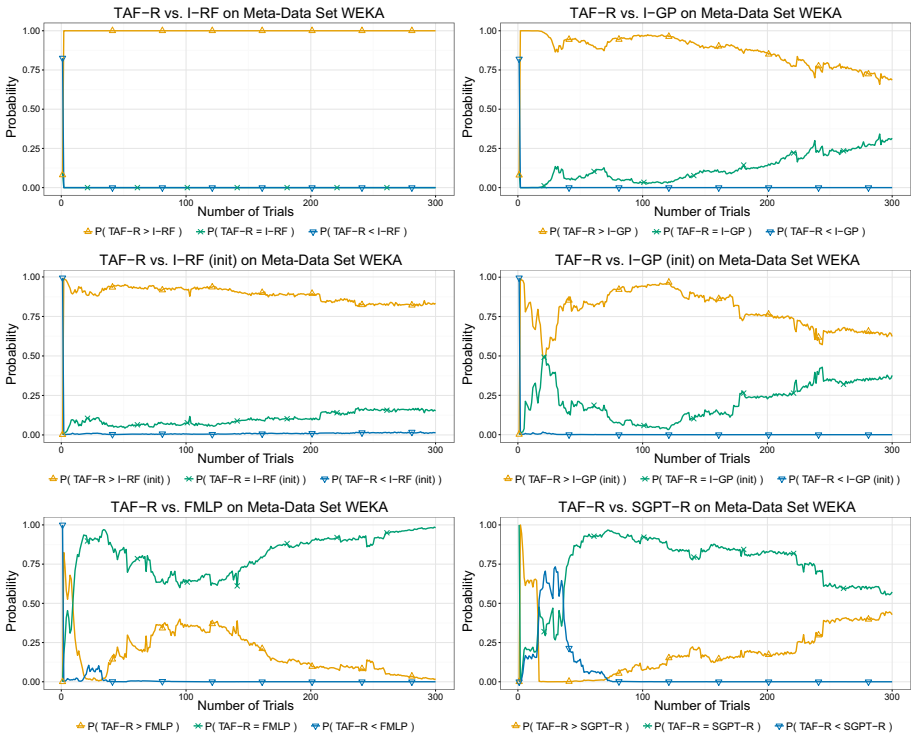

Fig. 16 Comparison of TAF-R against all other optimizers using the Bayesian hierarchical test on the WEKA meta-data set. Probabilities above 95% are significant. With increasing number of trials the probability for equality of both methods increases due to a limited number of considered configurations

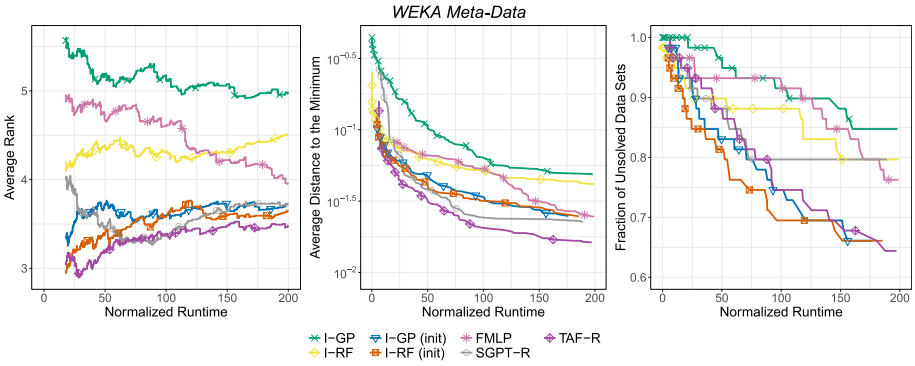

Fig. 17 Comparison of TAF-R to its competitor method considering the run time. TAF-R still performs best. In contrast to the previous experiments, FMLP is worse and RF better

8.5.6 Performance with respect to run time

In the previous experiments we assumed that the training time is always the same no matter which configuration is chosen. The main reason for this is that none of the methods considers the run time needed for evaluating a configuration when choosing which configuration to test next. All of them try to find the global minimum in as few trials as possible. However,

in practice it is of course very important to consider the run time as well. Since for the Weka meta-data set not only the hyperparameters are changing but also the model, we conducted a run time experiment for this particular meta-data set. The results are reported in Fig. 17. In our experiments we used various data sets which differ in the number of predictors and instances. To ensure that each data set has equal influence on the average results, we normalized the run time. The run time for each data set is normalized by dividing it by the average training time needed for a model to be trained on this data set. Hence, the normalized run time in Fig. 17 at 1 shows the performance of all methods after investing the amount of time that is needed to train a single model on average. We report ADTM and fraction of unsolved data sets for each method starting at the time when at least one configuration for all data sets has been evaluated. We report the average rank starting at the time when every method has evaluated at least one configuration for all data sets.

Our proposed method TAF-R still yields the best results. The biggest differences to the previous results are the changes of FMLP and I-RF. I-RF improved a lot. It evaluates far more configurations per time unit than any other method. This leads to worse results in the previous evaluation protocol but pays off under this protocol. For FMLP it is exactly the other way round. It mainly focuses on time-consuming configurations. While it needs only few configurations to find good models, it needs more time than most competitor methods.

9 Conclusions

In this work, we proposed a new transfer surrogate model framework which is able to scale to large meta-data sets. Such considerations will become a necessity in the near future as with more experiments conducted every day, the meta-data that can be employed also grows. To ensure scalability, our transfer surrogate model is built on Gaussian processes which have been learned individually on the observed performances on a single dataset only, to then be combined into a joint surrogate model on the basis of product of experts as well as kernel regression models. Additionally, as Gaussian processes are basically hyperparameter free, we have created a strong but scalable surrogate model, that also does not require a second stage hyperparameter optimization, as opposed to other surrogates in the related work. Overall, we derived different instances of our framework and evaluated them with respect to optimization performance and scalability on two meta-data sets containing both hyperparameter optimization as well as model choice. We show empirically that our method is able to outperform existing methods with respect to both measures by choosing well-performing hyperparameter configurations while maintaining a small computational overhead.

Nevertheless, the transfer surrogate model still suffers from the problem caused by different performance scales on different data sets, which introduces a bias when selecting the next hyperparameter configuration to test. As a result of this issue, we proposed to decay the influence of meta-data as the hyperparameter optimization progresses, we have found empirically that this is strategy proves advantageous. Unfortunately, this decay is relying on heuristics, which was our motivation to use the meta-data within the acquisition function instead of the surrogate model to achieve a more principled effect. Similarly to the transfer surrogate framework, we derived a transfer acquisition framework that keeps the assets of our previously proposed surrogate model but overcomes its disadvantages and directly uses the improvement of the meta-data. In a conclusive evaluation, we were able to confirm our intuition empirically on two meta-data sets, as the transfer acquisition framework shows even higher performance than the transfer surrogate model on the same meta-data sets. Consequently, we

recommend using the transfer acquisition framework for hyperparameter optimization, as it is fast and powerful in delivering well-performing hyperparameter configurations.

Possible future work involves the consideration of learning curves. As mentioned in the related work section, there exist many approaches which use the convergence behavior of the learning algorithm to predict the final performance. This prediction itself will help improving our method. Additionally, learning curve forecasting methods will benefit from observations on other data sets. Furthermore, we want to investigate the case where one is not solely interested in reducing the loss of a predictor but also other cost such as training, prediction or hardware constraints (Abdulrahman et al. 2018) which is relevant for edge devices. We saw in our experiments that trying to minimize the classification error only regardless of the training time needed can lead to non-optimal results. Therefore, we are considering to minimize a loss function which is a combination of classification error and training time as suggested by Abdulrahman et al. (2018). Finally, we want to conduct experiments to see whether our method is able to be migrated to deep learning methods.

Acknowledgements We would like to thank Prof. Pavel Brazdil and the anonymous reviewers for their valuable feedback that helped us improving this work. We are grateful for Bradley Baker's help in perfecting our paper. We acknowledge the co-funding of our work by the German Research Foundation (DFG) under Grant SCHM 2583/6-1.

References

Abdulrahman, S. M., Brazdil, P., van Rijn, J. N., & Vanschoren, J. (2018). Speeding up algorithm selection using average ranking and active testing by introducing runtime. In P. Brazdil & C.Giraud-Carrier (Eds.), *Special issue on metalearning and algorithm selection*. *Machine Learning Journal*, *107*, 1

Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013) Collaborative hyperparameter tuning. In *Proceedings of the 30th international conference on machine learning* (pp. 199–207). ICML 2013, Atlanta, GA, USA, 16–21 June 2013.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*, 281–305.

Brazdil, P., Giraud-Carrier, C. G., Soares, C., & Vilalta, R. (2009). Metalearning—Applications to data mining. *Cognitive technologies*. Springer. https://doi.org/10.1007/978-3-540-73263-1.

Cavazos, J., & O'Boyle, M. F. P. (2006). Method-specific dynamic compilation using logistic regression. In *Proceedings of the 21th annual ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications* (pp. 229–240). OOPSLA 2006, October 22–26, 2006, Portland, Oregon, USA.

Cawley, G. C. (2001). 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*.

Chapelle, O., Vapnik, V., & Bengio, Y. (2002). Model selection for small sample regression. *Machine Learning*, *48*(1–3), 9–23.

Corani, G., Benavoli, A., Demsar, J., Mangili, F., & Zaffalon, M. (2016). Statistical comparison of classifiers through bayesian hierarchical modelling. CoRR abs/1609.08905. http://arxiv.org/abs/1609.08905.

Czogiel, I., Luebke, K., & Weihs, C. (2006). Response surface methodology for optimizing hyper parameters. Tech. rep. https://eldorado.tu-dortmund.de/bitstream/2003/22205/1/tr09-06.pdf.

de Souza, B. F., de Carvalho, A., Calvo, R., & Ishii, R. P. (2006). Multiclass SVM model selection using particle swarm optimization. In *Sixth international conference on hybrid intelligent systems, 2006* (pp. 31–31). HIS'06 , IEEE.

Deisenroth, M. P., & Ng, J. W. (2015). Distributed gaussian processes. *International Conference on Machine Learning (ICML)*, *2*, 5.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30. URL http://www.jmlr.org/papers/v7/demsar06a.html.

Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence* (pp. 3460–3468). IJCAI 2015, Buenos Aires, Argentina, July 25–31, 2015.

Eggensperger, K., Lindauer, M., Hoos, H. H., Hutter, F., & Leyton-Brown, K. (2018). Efficient benchmarking of algorithm configuration procedures via model-based surrogates. In P. Brazdil & C. Giraud-Carrier (Eds.), *Special issue on metalearning and algorithm selection*. *Machine Learning Journal*, *107*, 1.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in neural information processing systems, Vol. 28: Annual conference on neural information processing systems 2015*, December 7–12, 2015, Montreal, Quebec, Canada (pp. 2962–2970). http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.

Feurer, M., Springenberg, J. T., & Hutter, F. (2014). Using meta-learning to initialize bayesian optimization of hyperparameters. In *ECAI workshop on metalearning and algorithm selection (MetaSel)* (pp. 3–10).

Feurer, M., Springenberg, J. T., & Hutter, F. (2015). 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).

Foo, C.s., Do, C. B., & Ng, A. (2007). Efficient multiple hyperparameter learning for log-linear models. In *Advances in neural information processing systems* (pp. 377–384).

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701. https://doi.org/10.1080/01621459.1937.10503522.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, *11*(1), 86–92.

Friedrichs, F., & Igel, C. (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, *64*, 107–117.

Friedrichs, F., & Igel, C. (2005). Evolutionary tuning of multiple svm parameters. *Neurocomputing*, *64*, 107–117. https://doi.org/10.1016/j.neucom.2004.11.022.

Gomes, T. A. F., Prudêncio, R. B. C., Soares, C., Rossi, A. L. D., & Carvalho, A. C. P. L. F. (2012). Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing*, *75*(1), 3–13. https://doi.org/10.1016/j.neucom.2011.07.005.

Guo, X. C., Yang, J. H., Wu, C. G., Wang, C. Y., & Liang, Y. C. (2008). A novel ls-svms hyper-parameter selection based on particle swarm optimization. *Neurocomputing*, *71*(16–18), 3211–3215. https://doi.org/10.1016/j.neucom.2008.04.027.

Hinton, G. E. (1999). Products of experts. In *Artificial neural networks, 1999. ICANN 99. Ninth international conference on (Conf. Publ. No. 470)* (Vol. 1, pp. 1–6). IET.

Hinton, G. (2010). A practical guide to training restricted Boltzmann machines. *Momentum*, *9*(1), 926.

Hoffman, M. D., Shahriari, B., & de Freitas, N. (2014). On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *Proceedings of the seventeenth international conference on artificial intelligence and statistics* (pp. 365–374). AISTATS 2014, Reykjavik, Iceland, April 22–25, 2014.

Holmes, G., Donkin, A., & Witten, I. H. (1994). Weka: A machine learning workbench. In *Intelligent information systems, 1994. Proceedings of the 1994 second Australian and New Zealand conference on* (pp. 357–361). IEEE.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th international conference on learning and intelligent optimization*, LION'05 (pp. 507–523). Berlin, Heidelberg: Springer.

Jamieson, K. G., & Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th international conference on artificial intelligence and statistics* (pp. 240–248). AISTATS 2016, Cadiz, Spain, May 9–11, 2016. http://jmlr.org/proceedings/papers/v51/jamieson16.html

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492. https://doi.org/10.1023/A:1008306431147.

Kamel, M. S., Enright, W. H., & Ma, K. S. (1993). ODEXPERT: An expert system to select numerical solvers for initial value ODE systems. *ACM Transactions on Mathematical Software*, *19*(1), 44–62.

Kanda, J., Soares, C., Hruschka, E. R., & de Carvalho, A. C. P. L. F. (2012). A meta-learning approach to select meta-heuristics for the traveling salesman problem using mlp-based label ranking. In *Neural information processing—19th international conference* (pp. 488–495). ICONIP 2012, Doha, Qatar, November 12–15, 2012, Proceedings, Part III.

Kapoor, A., Ahn, H., Qi, Y., & Picard, R. W. (2005). Hyperparameter and kernel learning for graph based semi-supervised classification. In *Advances in Neural Information Processing Systems* (pp. 627–634).

Keerthi, S., Sindhwani, V., & Chapelle, O. (2007). An efficient method for gradient-based adaptation of hyperparameters in SVM models. *Twenty-first annual conference on neural information processing systems*. Vancouver, Canada

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93. https://doi.org/10.2307/2332226.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on machine learning* (pp. 473–480). ACM.

Leite, R., Brazdil, P., & Vanschoren, J. (2012). Selecting classification algorithms with active testing. In *Machine learning and data mining in pattern recognition—8th international conference* (pp. 117–131). MLDM 2012, Berlin, Germany, July 13–20, 2012. Proceedings.

Lemke, C., Budka, M., & Gabrys, B. (2015). Metalearning: A survey of trends and technologies. *Artificial Intelligence Review*, *44*(1), 117–130. https://doi.org/10.1007/s10462-013-9406-y.

Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2016). Efficient hyperparameter optimization and infinitely many armed bandits. CoRR **abs/1603.06560**. http://arxiv.org/abs/1603.06560.

Maron, O., & Moore, A. W. (1997). The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, *11*(1–5), 193–225. https://doi.org/10.1023/A:1006556606079.

Masada, T., Fukagawa, D., Takasu, A., Hamada, T., Shibata, Y., & Oguri, K. (2009). Dynamic hyperparameter optimization for bayesian topical trend analysis. In *Proceedings of the 18th ACM conference on information and knowledge management* (pp. 1831–1834). ACM.

McQuarrie, A. D., & Tsai, C. L. (1998). *Regression and time series model selection*. Singapore: World Scientific.

Michie, D., Spiegelhalter, D. J., Taylor, C. C., & Campbell, J. (Eds.). (1994). *Machine learning, neural and statistical classification*. Upper Saddle River, NJ: Ellis Horwood.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, *9*(1), 141–142. https://doi.org/10.1137/1109020.

Nareyek, A. (2004). *Choosing search heuristics by non-stationary reinforcement learning* (pp. 523–544). Boston, MA: Springer.

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, *18*, 263. International Biometric Soc 1441 I ST, NW, Suite 700, Washington, DC 20005-2210.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In *Proceedings of the seventeenth international conference on machine learning* (pp. 743–750). Morgan Kaufmann.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian processes for machine learning (Adaptive computation and machine learning)*. Cambridge, MA: The MIT Press.

Reif, M., Shafait, F., & Dengel, A. (2012). Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning*, *87*(3), 357–380. https://doi.org/10.1007/s10994-012-5286-7.

Rendle, S. (2010). Factorization machines. In *Data mining (ICDM), 2010 IEEE 10th international conference on* (pp. 995–1000). IEEE.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, *15*, 65–118. https://doi.org/10.1016/S0065-2458(08)60520-3.

Schilling, N., Wistuba, M., & Schmidt-Thieme, L. (2016). Scalable hyperparameter optimization with products of gaussian process experts. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 33–48). Springer.

Schilling, N., Wistuba, M., Drumond, L., & Schmidt-Thieme, L. (2015). 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.

Schneider, P., Biehl, M., & Hammer, B. (2010). Hyperparameter learning in probabilistic prototype-based models. *Neurocomputing*, *73*(7), 1117–1124.

Seeger, M. (2006). Cross-validation optimization for large scale hierarchical classification kernel methods. In *Advances in neural information processing systems* (pp. 1233–1240).

Smith-Miles, K. A. (2009). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, *41*(1), 6:1–6:25. https://doi.org/10.1145/1456650.1456656.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems 25: 26th Annual conference on neural information processing systems 2012* (pp. 2960–2968). Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, USA.

Srinivas, N., Krause, A., Kakade, S., & Seeger, M. W. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 1015–1022), June 21–24, 2010, Haifa, Israel.

Sun, Q., & Pfahringer, B. (2013). Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, *93*(1), 141–161.

Swersky, K., Snoek, J., & Adams, R. P. (2013). Multi-task bayesian optimization. In *Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013* (pp. 2004–2012). Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, USA.

Swersky, K., Snoek, J., & Adams, R. P. (2014). Freeze-thaw bayesian optimization. *Computing Research Repository*. arXiv:1406.3896.

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*, KDD '13 (pp. 847–855). ACM, New York, NY, USA. https://doi.org/10.1145/2487575.2487629.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning* (p. 104). ACM.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77–95. https://doi.org/10.1023/A:1019956318069.

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015). Learning data set similarities for hyperparameter optimization initializations. In *Proceedings of the 2015 international workshop on meta-learning and algorithm selection* (pp. 15–26), Porto, Portugal, September 7th, 2015.

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015). Learning hyperparameter optimization initializations. In *International conference on data science and advanced analytics*, DSAA 2015, Paris, France, October 19–21, 2015.

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2015). Sequential model-free hyperparameter tuning. In *2015 IEEE international conference on data mining* (pp. 1033–1038). ICDM 2015, Atlantic City, NJ, USA, November 14–17, 2015. https://doi.org/10.1109/ICDM.2015.20

Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2016). Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 199–214). Springer.

Xu, L., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2008). SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research (JAIR)*, *32*, 565–606.

Yogatama, D., & Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *International conference on artificial intelligence and statistics (AISTATS 2014)*.