2015 IEEE International Conference on Data Mining

Sequential Model-free Hyperparameter Tuning

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

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

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

***Abstract*—Hyperparameter tuning is often done manually but current research has proven that automatic tuning yields effective hyperparameter configurations even faster and does not require any expertise. To further improve the search, recent publications propose transferring knowledge from previous experiments to new experiments. We adapt the sequential model-based optimization by replacing its surrogate model and acquisition function with one policy that is optimized for the task of hyperparameter tuning. This policy generalizes over previous experiments but neither uses a model nor uses meta-features, nevertheless, outperforms the state of the art. We show that a static ranking of hyperparameter combinations yields competitive results and substantially outperforms a random hyperparameter search. Thus, it is a fast and easy alternative to complex hyperparameter tuning strategies and allows practitioners to tune their hyperparameters by simply using a look-up table. We made look-up tables for two classifiers publicly available: SVM and AdaBoost. Furthermore, we propose a similarity measure for data sets that yields more comprehensible results than those using meta-features. We show how this similarity measure can be applied to surrogate models in the SMBO framework and empirically show that this change leads to better hyperparameter configurations in less trials.**

I. Introduction

Hyperparameters need to be tuned for most machine learning algorithms and their choice is crucial. The hyperparameter tuning often decides whether the performance of an algorithm is just moderate or state of the art, hence, the task of hyperparameter tuning is as important as developing new algorithms [1], [2], [3], [4], [5]. Hyperparameter performance depends on the data set, i.e. hyperparameter configurations that are great on one data set yield poor performance for another.

Recently, there is an increased interest in the machine learning community to simplify the hyperparameter tuning process [1], [4], [5]. Sequential model-based optimization (SMBO) is one of the current approaches to solve this black box optimization problem. A surrogate model that tries to learn the hyperparameter performance is combined with a heuristic to sequentially try different hyperparameter configurations.

Recent work tries to speed up SMBO by transferring knowledge about the performance of hyperparameter configurations on other data sets [6], [7]. Even though the scale of the evaluation metric between data sets can vary considerably, the goal is to identify regions of hyperparameter configurations that have good performance across data sets. Bardenet et al. and Yogatama et al. [6], [7] use a surrogate model that minimizes the squared error on the meta-training data set and choose promising hyperparameter configurations using a heuristic acquisition function (e.g. Expected Improvement [8]) on the predicted response function to find the balance between exploration and exploitation. Since the surrogate model needs to be relearned in each SMBO iteration, this might be time-consuming for large meta-data sets.

A. Our Contributions

There are two problems that are approached in this work: i) the choice of the hyperparameter configuration depends on a model optimized for the squared error combined with a simple heuristic and ii) the similarity between data sets depends on meta-features. Item i) is a problem because it does not solve the problem of finding the best hyperparameter configuration directly and item ii) is a problem because meta-features do not guarantee that they are descriptive for the data set and are problem-dependent. We want to overcome these problems by proposing a model- and meta-feature-free hyperparameter tuning strategy that is optimized for a hyperparameter tuning loss. Its benefit is shown empirically in comparison to state of the art model-based tuning strategies that optimize its models for regression losses combined with heuristics. Since a hyperparameter tuning loss was never introduced before, we propose a quality measure for hyperparameter tuning. This is firstly used to optimize our method for and secondly allows better comparison of tuning strategies across papers compared to strategies such as average rank among tuning strategies. Because we do not want to rely on meta-features, a distance measure between data sets that does not depend on meta-features is proposed. We empirically show that this new distance measure is able to improve existing hyperparameter optimization strategies by applying and evaluating it on existing model-based tuning strategies and compare it to the currently used meta-feature-based distance functions.

II. Related Work

The task of hyperparameter tuning is important and often decides whether an algorithm has mediocre or state of the art performance on a task. In the last few years much research was done on the field of automatic hyperparameter tuning. Bergstra and Bengio [1] have shown that for algorithms with low effective hyperparameter dimensionality a random hyperparameter search can outperform a grid search by using just a small fraction of trials. Recent state of the art tuning strategies are using the sequential model-based optimization (SMBO) [8]. Strategies such as SMAC [9], TPE [10] and Spearmint [4] sequentially learn the hyperparameter response function to find a promising next hyperparameter combination.

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.20

Current research tries to use knowledge from previous experiments on other data sets to apply this to future, unknown data sets. This is either done by an initialization using meta-knowledge [11] or by learning a surrogate model directly on the meta-data of the current and past experiments [6], [7].

Furthermore, there also exist strategies to optimize hyperparameters that are based on optimization techniques from artificial intelligence such as tabu search [12], particle swarm optimization [13] and evolutionary algorithms [14]. Since none of these strategies use information from previous experiments, meta-knowledge can be added analogously to the SMBO counterpart using an initialization [15], [16]. Active Testing [17] proposes to sequentially try the algorithm configuration that is likely better than the currently best configuration.

III. Background

A. The Formal Setup

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

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

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

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

To distinguish between the task of hyperparameter optimization from model parameter estimation the prefix *meta* is used.

B. A New Evaluation Metric

It is very common to compare different tuning strategies either by an average rank between the methods over different data sets or by the performance of the found hyperparameters on the data sets directly [6], [7]. Both comparisons make absolutely sense but come with some disadvantages. First of all, for both evaluation strategies you can either compare the plots depending on the number of tries done so far or you fix the number of tries to T and report the average rank after T tries. Additionally, the average rank between tuning strategies changes if you add or remove strategies. Hence, the values cannot be reused in further publications. Furthermore, if one tuning strategy has a better average rank than another it does not necessarily mean that it is also the better strategy in general. Lets assume there are three data sets where the best hyperparameter configuration needs to be found. Tuning strategy A finds the best hyperparameter configuration for two of them and the worst for the last one. Strategy B finds the second best hyperparameter configuration for all three data sets. By average rank, strategy A has a rank of 1.3 compared to strategy B with rank 1.7. But strategy B provides stable predictions and is not much worse than strategy A in cases where A is the better strategy and A was very good on most of the data sets but really bad on one of them.

We want to overcome both disadvantages. The evaluation measure should be a number that rewards fast convergence against the best hyperparameter configuration and is not a measure that is relative to the performance of other methods. Thus, it can be easily compared to new methods if they are applied on the same meta-data set. At this point, one could think about a ranking measure such as normalized discounted cumulative gain (NDCG). But we are not interested in finding the perfect ranking of hyperparameter configurations (this means first trying the best hyperparameter configuration, then the second best and so on) but on finding a decent hyperparameter configuration as soon as possible. Additionally, after finding the best hyperparameter configuration, the choice of further hyperparameter configurations should not affect the metric.

We propose the cumulative average normalized error (CANE) as a new metric which we define as

$$\mathrm{CANE}\left(\mathcal{D}, \Lambda_T\right) := \sum_{t=1}^{T} \mathrm{ANE}\left(\mathcal{D}, \Lambda_t\right) \tag{3}$$

where the average normalized error is defined as

$$\mathrm{ANE}\left(\mathcal{D}, \Lambda_T\right) := \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \frac{\min_{\lambda \in \Lambda_T} f_D\left(\lambda\right) - f_D^{min}}{f_D^{max} - f_D^{min}} \tag{4}$$

where $\mathcal{D}$ is the set of data sets, $\Lambda_T = (\lambda_1, \ldots, \lambda_T)$ is the ordered sequence of evaluated hyperparameter configurations at time T, Λ_t the ordered subsequence of Λ_T until time $t < T$, $f_D\left(\lambda\right)$ is the error of the hyperparameter configuration λ on the validation partition of data set D and f_D^{max} and f_D^{min} are the maximum and minimum value of f_D using $\lambda \in \Lambda$, respectively. This metric can only be applied if the hyperparameter tuning is limited to a finite subset of all possible hyperparameter configurations. In real applications, the number of hyperparameter configurations is often infinite but for the evaluation of hyperparameter tuning strategies it is common that the meta-data is generated by applying a grid search using a finite set of hyperparameter configurations that is used on all data sets. In the remainder of this paper, Λ denotes this finite subset of all feasible hyperparameter configurations.

CANE is lower bounded by 0 and attains this value only if in the first try on every data set the best hyperparameter configuration was chosen. The function value $f_D\left(\lambda\right)$ is scaled between 0 and 1 per data set in Equation 3 to overcome different scales between data sets.

IV. Sequential Model-free Optimization

This section will introduce one of the core contributions of this work. With A-SMFO we propose a fast and parallelizable hyperparameter tuning strategy that can be applied easily. This will be extended to NN-SMFO which uses information about the similarity between past seen data sets. We propose a new distance function to measure this similarity.

A. Average SMFO

Our idea is to combine the surrogate model with the acquisition function. Additionally, we want a method that is independent of any meta-features. Given a new, unknown data set, the best you can do is to take that hyperparameter configuration that has proven to be the best on past experiments. This intuition is extended to arbitrary many tries. Formally, at time step t the hyperparameter configuration λ is chosen that improves the performance on the meta-training set the most. Let Λ_t be the set of evaluated hyperparameter configurations at step t. Then, we define the performance of a hyperparameter configuration λ_t at t as

$$\text{perf}\left(\lambda_t, \Lambda_{t-1}, \mathcal{D}\right) = \sum_{D \in \mathcal{D}} \min_{\lambda' \in \Lambda_{t-1} \cup \{\lambda_t\}} \left\{r_D\left(\lambda', \Lambda\right)\right\} \quad (5)$$

where $r_D\left(\lambda, \Lambda\right)$ is the rank of λ on data set D over all hyperparameter configuration in Λ. Again, Λ is the finite set of feasible hyperparameter configurations as defined above that were evaluated on $D \in \mathcal{D}$ in previous experiments. The ranking for a new data set is obviously unknown. Then, at each time step t, $f\left(\arg\min_{\lambda \in \Lambda} \text{perf}\left(\lambda, \Lambda_{t-1}, \mathcal{D}\right)\right)$ is evaluated.

Algorithm 1 CANE Optimal Sequence

Input: Set of feasible hyperparameter configurations Λ, set of data sets $\mathcal{D}$, number of maximal tries T.
Output: Sequence of hyperparameter configurations to evaluate.

1: $\Lambda_0 \leftarrow ()$
2: **for** $t = 1$ to T **do**
3: $\quad \lambda_t \leftarrow \arg\min_{\lambda \in \Lambda} \text{perf}\left(\lambda, \Lambda_{t-1}, \mathcal{D}\right)$
4: $\quad \Lambda_t \leftarrow \left(\Lambda_{t-1}, \lambda_t\right)$
5: $\quad$ **if** $\frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \min_{\lambda \in \Lambda_t} r_D\left(\lambda, \Lambda\right) = 1$ **then**
6: $\quad\quad$ **return** Λ_t
7: **return** Λ_T

Algorithm 1 will find the best hyperparameter configurations for the meta-training set by sequentially selecting the hyperparameter configurations that minimize Equation 5 given Λ. The stopping criterion in Line 5 is fulfilled as soon as Λ_t contains all hyperparameter configurations that were best on the data sets $D \in \mathcal{D}$. The resulting sequence Λ_T is optimal for the meta-training set but not necessarily for the meta-testing set. Actually, it is likely that more trials are needed on meta-test than on meta-train resulting in a sequence returned by Algorithm 1 that is too short. To overcome this problem, the final tuning strategy is given in Algorithm 2. The idea is to optimize the CANE on meta-train in the first iteration. In the next iterations, CANE is optimized by not considering previously chosen hyperparameter configurations in the evaluation and in the pool of feasible candidates Λ. This is equivalent to optimizing for rank t in iteration t instead of optimizing for rank 1.

B. Nearest Neighbor SMFO

Average SMFO acquires hyperparameter configurations only dependent on the meta-training set $\mathcal{D}_{train}$. This means, there is a fixed sequence of hyperparameter configurations to evaluate and hence this method can be parallelized and implemented easily. Prediction can be done in constant time, memory consumption is linear in the number of feasible

Algorithm 2 Average SMFO

Input: Set of feasible hyperparameter configurations Λ, set of data sets $\mathcal{D}$, number of maximal tries T.
Output: Sequence of hyperparameter configurations to evaluate.

1: $\hat{\Lambda} \leftarrow ()$
2: **while** $T > 0$ **do**
3: $\quad \Lambda' \leftarrow$ Algorithm1 $\left(\Lambda \setminus \hat{\Lambda}, T, \mathcal{D}\right)$
4: $\quad T \leftarrow T - |\Lambda'|$
5: $\quad \hat{\Lambda} \leftarrow \left(\hat{\Lambda}, \Lambda'\right)$
6: **return** $\hat{\Lambda}$

hyperparameter configurations and this static sequence of hyperparameter configurations can be shared among researchers and practitioners. Nevertheless, Average SMFO has one big disadvantage. It does not consider the valuable evaluations on the current data set D_{test}. Therefore, Average SMFO is extended to Nearest Neighbor SMFO (NN-SMFO) to overcome this problem. NN-SMFO is not considering every data set when predicting the best hyperparameter configuration but the k most similar. The distance function between D_{test} and any other data set $D \in \mathcal{D}$ is inspired by the Kendall tau rank correlation [18] coefficient. Assuming that the ranking of hyperparameter configurations contains solely concordant or discordant pairs and ignoring the constants, the resulting KTRC distance function is

$$\begin{aligned} \text{KTRC}\left(D_1, D_2, \Lambda_t\right) &:= \frac{\sum_{\lambda_1 \in \Lambda_t} \sum_{\lambda_2 \in \Lambda_t} s\left(\lambda_1, D_1, \lambda_2, D_2\right)}{\left(|\Lambda_t| - 1\right)|\Lambda_t|} \\ \Lambda_t &:= \Lambda_t^{(D_1)} \cap \Lambda_t^{(D_2)} \end{aligned} \quad (6)$$

$$\begin{aligned} s\left(\lambda_1, D_1, \lambda_2, D_2\right) \quad := \quad & \mathbb{I}\left(f_{D_1}\left(\lambda_1\right) > f_{D_1}\left(\lambda_2\right)\right. \\ & \left.\oplus f_{D_2}\left(\lambda_1\right) > f_{D_2}\left(\lambda_2\right)\right) \end{aligned}$$

where $\oplus$ is the symbol for an exclusive or and $\Lambda_t^{(D)}$ is the set of hyperparameter configurations $\lambda \in \Lambda$ where $f_D\left(\lambda\right)$ is already evaluated for data set $D \in \mathcal{D}$, f_D being the response function of data set D.

To apply these changes, the data sets in $\mathcal{D}$ before Line 3 in Algorithm 1 needs to be reduced to the k most similar data sets to D_{test}.

V. On a Distance Measure Between Data Sets

The current state of the art models the similarity between data sets either by learning it implicitly using meta-features [6], [9] or modelling them explicitly using the Euclidean distance on the meta-features [16], [7]. What is meant with similar data sets is that they behave similarly with respect to the hyperparameter configurations. This means, that two similar data sets have a similar ranking of hyperparameter configurations. Measuring this ranking using a rank correlation metric such as proposed in Equation 6 is actually a natural choice. Otherwise, the similarity cannot be estimated if no evaluations are observed. We claim that few evaluations are enough to approximate the true rank correlation such that this distance measure is nevertheless more expressive than alternatives such as distance functions based on meta-features.

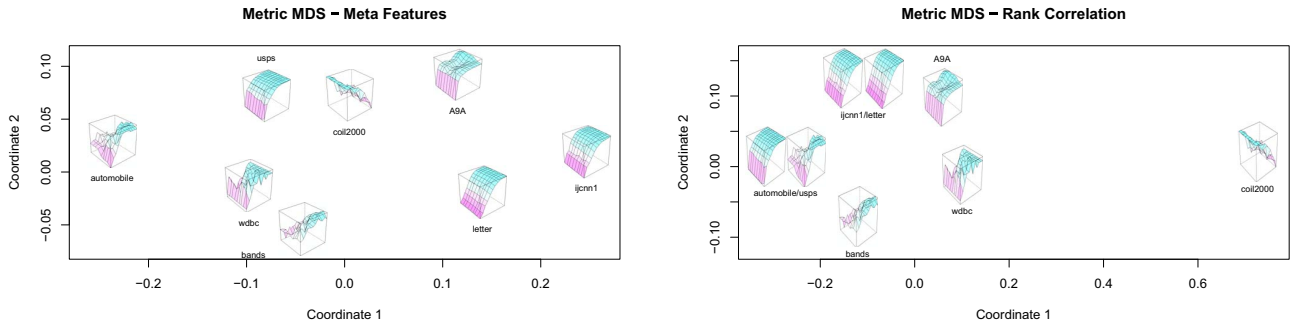

Fig. 1. Metric multidimensional scaling of a distance metric using Euclidean distance on the meta-features (left) and Equation 6 using only the first four hyperparameter configurations recommended by Average SMFO (right). The shown response surfaces are that from a AdaBoost classifier, the meta-features used are described in Section VI.

Figure 1 supports this claim. In the left plot, it shows the multidimensional scaling (MDS) of the distance matrix using the Euclidean distance on the meta-features. The data set names indicate the position in the space, the near-by little plots the (hidden) response function. The response function in this case maps two hyperparameters (x and z-axis) to the f value, in this case the classification accuracy. One can see that in some cases this similarity measure works, like in the cases of *letter* and *ijcnn1* but it can also fail. For instance, *usps* and *coil2000* are very close to each other but the hyperparameter configurations are almost ranked contrary.

In the right plot the KTRC distance function defined in Equation 6 is used. The distance is computed using only four hyperparameter configurations, i.e. the best four recommended by Average SMFO. The message is that only little information about the ranking can already provide a good clustering. Clearly, *coil2000* is identified as completely different to any other data set and the reader may see for herself that the distances are more intuitive just from visual inspection.

VI. Experimental Evaluation

The tuning strategies proposed in Section IV are arguably simple and still competitive. We are comparing them to state of the art hyperparameter tuning strategies (competitors are published on top conferences, e.g. NIPS 2012, ICML 2013, AISTATS 2014). Empirically, we show that they can outperform those strategies and have the capability to scale to big meta-data.

A. Experimental Setup

To compare different hyperparameter tuning strategies, we used 25 classification data sets randomly chosen from the UCI repository to create two meta-data sets. We merged existing splits, shuffled all instances and created new splits where 80% was used as train and 20% as test, respectively.

One meta-data set was created as proposed by [6] using AdaBoost with decision products as weak learners [19]. This results into two hyperparameters, the number of iterations I and the number of product terms M. The target measure is the classification error. Validation errors are precomputed on a grid with values $I \in \{2, 5, 10, 20, 50, 100, 200, 500, 10^3, 2 \cdot 10^3, 5 \cdot 10^3, 10^4\}$ and $M \in \{2, 3, 4, 5, 7, 10, 15, 20, 30\}$ resulting into 108 meta-instances per data set.

The second meta-data set was created using a support vector machine [20]. In this case the hyperparameters are the chosen kernel (linear, polynomial or Gaussian), the trade-off between margin and training error C and kernel specific hyperparameters such as the degree of the polynomial kernel d and the width γ of the Gaussian kernel. Again, the validation errors are precomputed on a grid with values $C \in \{2^{-5}, \ldots, 2^6\}$, $d \in \{2, \ldots, 10\}$ and $\gamma \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5, 1, 2, 5, 10, 20, 50, 10^2, 10^3\}$ resulting into 288 meta-instances per data set.

Meta-features are a vital part in most of the competitor strategies. Therefore, we added the meta-features to our meta-data that were also used by [6], [7]. First, we are extracting the number of training instances n, the number of classes c and the number of predictors p. The final meta-features are c, $\log(p)$ and $\log(n/p)$ scaled to $[0, 1]$.

The meta-data sets are available on our supplementary website [21]. It provides also the source code, further experiments (e.g. runtime), plots and information.

B. Tuning Strategies

We are comparing our proposed methods with five different competitor strategies. One is random search (Random) [1], the only fully parallelizable strategy besides A-SMFO. Then we compare to different variations of the SMBO framework. I-GP is using a Gaussian process with squared-exponential kernel as a surrogate model and does not consider any meta-knowledge [10]. Surrogate Collaborative Tuning (SCoT) [6] and Gaussian process with multi kernel learning (MKL-GP) [7] are using surrogate models that consider meta-knowledge. Furthermore, we compare to SMAC++, a variation of SMAC [9] that also considers meta-knowledge during the optimization process. To empirically evaluate the influence of the distance function between data sets, we propose Rank Correlation-based Gaussian Process (RC-GP). This is a variation of MKL-GP but instead of using the Euclidean distance of meta-features to estimate the similarity between data sets, we use the distance function defined in Equation 6. Optimal is an artificial

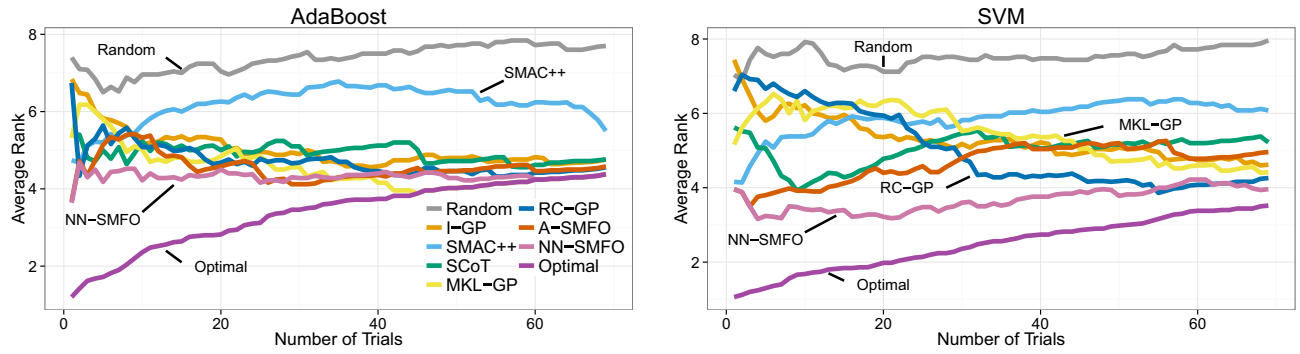

Fig. 2. Development of the average rank among different hyperparameter tuning strategies with increasing number of trials (best viewed in color or online [21]).

tuning strategy that always evaluates the best hyperparameter configuration and is added to some plots for orientation

C. Results

Before interpreting the results, the hypotheses made before the experiments are recalled: 1) The strategies proposed in Section IV are better because they are optimized directly for CANE. 2) The KTRC distance measure proposed in Equation 6 provides better information about the distance between data sets than a distance function based on the meta-features.

To confirm the first hypothesis, consider Figure 2. It shows the development of the average rank among different hyperparameter tuning strategies.[1] On the AdaBoost meta-data set all tuning strategies but Random and SMAC++ are close together. Nevertheless, NN-SMFO is one of the best. Especially in the beginning, a larger gap to the other approaches is recognizable. Starting at iteration 40, MKL-GP seems to become very good but this is misleading as the reader will see in Figure 3. The improvement here is less than one may assume, improving the classification error on average at the third decimal place. The good performance of NN-SMFO becomes substantial on the SVM meta-data set. Here it takes about 50 trials until any other tuning strategy gets even close to the performance of NN-SMFO. A-SMFO performs worse with respect to the average rank than NN-SMFO but still performs well if you consider that it is a static sequence and does not use any information about the data set that is currently being investigated. We want to remark that performance of the tuning strategies will always converge against the same value since at some point they have tried all feasible hyperparameter configurations.

Having a look at the development of the average normalized error in Figure 3 gives the reader the impression how fast the tuning strategies actually converge against the best hyperparameter configuration on average. One can see that both, A-SMFO and NN-SMFO, are converging considerably faster than the other strategies. Again, this marked difference is more substantial on the SVM meta-data set. Our assumption is that the SVM meta-data set is more difficult because it contains more than twice that many feasible hyperparameter configurations and the SVM has more hyperparameters to tune.

[1]To facilitate the readability, all plots can be found with an arbitrary subset of tuning strategies on the supplementary website [21].

Finally, also the results of the third metric presented in Table I lead to the same conclusions.

To validate the second hypothesis, we will have a closer look at the tuning strategies MKL-GP (yellow) and RC-GP (dark blue) because the only difference between those two is the distance function. Again, we first compare them with respect to the average rank. On the AdaBoost meta-data set, the difference is little showing a small advantage for MKL-GP. On the SVM data set the difference is more substantial. Here, RC-GP can show that the KTRC distance function is supporting.

Comparing RC-GP and MKL-GP with respect to the average rank does not lead to an unambiguous conclusion. Things look entirely different when having a look at the results with respect to the average normalized error. The average convergence of RC-GP to the best hyperparameter configuration is considerably faster than the convergence of MKL-GP. Also, for the last evaluation metric CANE, one can see that RC-GP provides better results than MKL-GP.

VII. Conclusion

We introduced three new hyperparameter tuning strategies and demonstrated that hyperparameter tuning is also possible without surrogate models and showed empirically that it is currently even better. A-SMFO has very nice properties. It is parallelizable which is only possible for grid search and random search while all other tuning strategies are not trivially parallelizable. Additionally, it only depends on the training meta-data and not on the currently evaluated data set. Therefore, we made the best predicted ranking for the models SVM and AdaBoost accessible to the community [21]. This offers practitioners a static hyperparameter tuning strategy that has proven to be very competitive compared to the state of the art and easy to apply for future experiments. Another nice property is that it makes results reproducible. Reproducing random search is not exactly possible because often only the distribution over the hyperparameters is made public and not which hyperparameter configurations are finally chosen.

NN-SMFO is an improvement over A-SMFO and can be used to further improve the results. It was compared to important state of the art competitor strategies and has empirically shown to be effective.

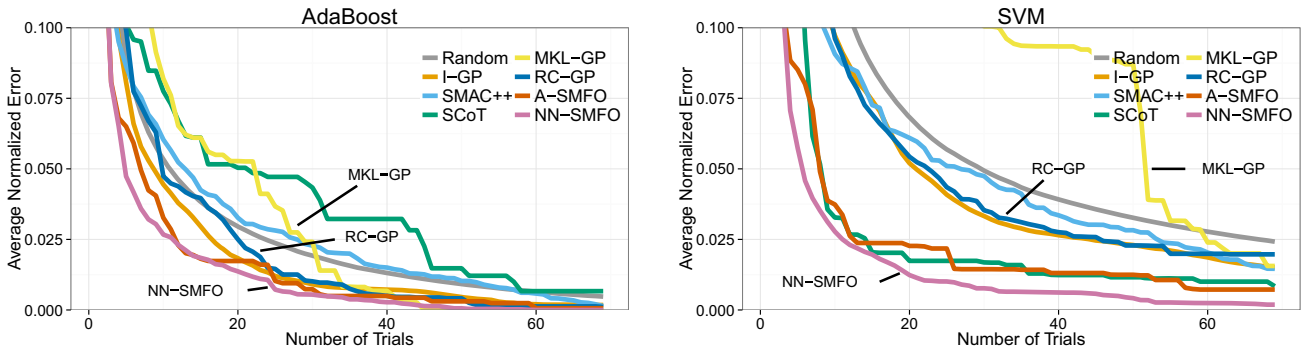

Fig. 3. Development of the average normalized error with increasing number of trials (best viewed in color or online [21]).

TABLE I. SUMMARY OF THE CANE OF THE TUNING STRATEGIES ON BOTH META-DATA SETS. NUMBER IN BRACKETS INDICATE THE RANKING ACROSS THE STRATEGIES. THE STRATEGIES INTRODUCED IN THIS PAPER ARE BOLD.

	Random	I-GP	SMAC	SCoT	MKL-GP	**RC-GP**	**A-SMFO**	**NN-SMFO**
AdaBoost	0.089 (6)	0.065 (3)	0.085 (5)	0.120 (8)	0.119 (7)	**0.071 (4)**	**0.047 (2)**	**0.040 (1)**
SVM	0.280 (7)	0.254 (5)	0.172 (4)	0.115 (3)	0.361 (8)	**0.259 (6)**	**0.082 (2)**	**0.053 (1)**

ACKNOWLEDGMENT

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

[12] G. Cawley, "Model selection for support vector machines via adaptive step-size tabu search," in *Artificial Neural Nets and Genetic Algorithms*, V. Kůrkov, R. Neruda, M. Krn, and N. Steele, Eds. Springer Vienna, 2001, pp. 434–437.

[13] X. C. Guo, J. H. Yang, C. G. Wu, C. Y. Wang, and Y. C. Liang, "A novel ls-svms hyper-parameter selection based on particle swarm optimization," *Neurocomput.*, vol. 71, no. 16-18, pp. 3211–3215, Oct. 2008.

[14] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters," *Neurocomput.*, vol. 64, pp. 107–117, Mar. 2005.

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

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

[17] R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in *Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition*, ser. MLDM'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 117–131.

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

[19] B. Kégl and R. Busa-Fekete, "Boosting products of base classifiers," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 497–504.

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

[21] M. Wistuba. (2015, Aug) Supplementary website: http://www.hylap.org/publications/Sequential-Model-free-Hyperparameter-Tuning.