2015 IEEE 27th International Conference on Tools with Artificial Intelligence

Joint Model Choice and Hyperparameter Optimization with Factorized Multilayer Perceptrons

Nicolas Schilling, Martin Wistuba, Lucas Drumond and Lars Schmidt-Thieme
University of Hildesheim
Information Systems and Machine Learning Lab
Hildesheim, Germany
Email: {schilling, wistuba, ldrumond, schmidt-thieme}@ismll.uni-hildesheim.de

***Abstract*—Recent work has demonstrated that hyperparameter optimization within the sequential model-based optimization (SMBO) framework is generally possible. This approach replaces the expensive-to-evaluate function that maps hyperparameters to the performance of a learned model on validation data by a surrogate model which is much cheaper to evaluate. The current state of the art in hyperparameter optimization learns these surrogate models across a variety of solved data sets where a grid search has already been employed. In this way, surrogate models are learned across data sets, and thus able to generalize better. However, meta features that describe characteristics of a data set are usually needed in order for the surrogate model to differentiate between same hyperparameter configurations on different data sets. Another research area that is closely related focuses on model choice, i.e. picking the right model for a given task, which is also a problem that many practitioners face in machine learning.**

In this paper, we aim to solve both of these problems with a unified surrogate model that learns across different data sets, different classifiers and their respective hyperparameters. We employ factorized multilayer perceptrons, a surrogate model that consists of a multilayer perceptron architecture, but offers the prediction of a factorization machine in the first layer. In this way, data sets, models and hyperparameters are being represented in a joint lower dimensional latent feature space. Experiments on a publicly available meta data set containing 59 individual data sets and 19 prediction models demonstrate the efficiency of our approach.

***Keywords*-Hyperparameter Optimization; Model Choice; Sequential Model-Based Optimization**

I. Introduction

One of the largest obstacles that keeps practitioners from applying machine learning is to choose the right prediction model for the task to be solved. Each model has its advantages and disadvantages, and only experienced practitioners know which model is best suited for which task. Thus, the problem of model choice is apparent everywhere in applied machine learning.

On top of that, these prediction models contain hyperparameters which have to be set before learning, as they cannot be optimized using standard optimization methods such as gradient descent. This is due to several reasons, for example for hyperparameters being discrete and therefore not learnable with continuous optimization methods. However, also the objective function to be minimized needs to be expressed explicitly with respect to the hyperparameters, which in many cases is simply not possible. There is a huge variety of different hyperparameters but what they all share is the fact that there is no general approach to optimize them which causes many researchers to perform hyperparameter optimization on a pre-defined grid of fixed hyperparameter values. Obviously, this leads to many runs of the machine learning algorithm and therefore is computationally infeasible for models that have a high dimensional hyperparameter space.

Usually, experienced researchers know which model is suited best for which task and also which regions in hyperparameter space offer promising validation performance. This knowledge is usually obtained from applying the same model to a variety of different data sets and assessing the final validation performance. In this paper, we seek to directly learn this kind of knowledge across data sets by associating latent features to hyperparameters, models and data sets. We extend previous work [1] on factorized multilayer perceptrons to also take into account model choice and present results in a standard sequential model-based optimization setting as well as give insight into the models that have been chosen frequently.

II. Related Work

To enable a wide audience to employ machine learning in their applications, research on hyperparameter optimization as well as model choice has been conducted in the last years. We review the related work on both hyperparameter optimization and model choice.

The most widely used method for hyperparameter optimization is probably a grid search, where a discrete subset of the hyperparameter space is created and validation performance of hyperparameters is assessed on this grid only. This is apparently not efficient, as a lot of useless results are produced and discarded in the end, an effort that can usually only be conducted on compute clusters which are not accessible for everyone. A more sophisticated variant of

1082-3409/15 $31.00 © 2015 IEEE
DOI 10.1109/ICTAI.2015.24

grid search is the random search [2], which samples hyperparameters according to pre-defined probability distributions and is supposed to work well in cases of low effective dimensionality when the validation performance is sensitive in only some dimensions of the hyperparameter space.

Another class of hyperparameter optimization methods bases on strong assumptions such as the specific choice of a model. Two instances, that work for the specific model choice of an SVM are for instance [3] and [4]. In the case of graph based semi-supervised learning, hyperparameters can be optimized using [5]. In cases of regression with a very small sample size, [6] may be applied. There are many approaches that are based on strong assumptions such as a specific model choice, however, they all have the common disadvantage of not being applicable in general.

A third class of hyperparameter optimization methods is based on artificial intelligence algorithms. For example, [7] optimizes SVM Kernel hyperparameters, [8] is based on evolutionary algorithms whereas [9] employs particle swarm optimization.

Finally, the most recent research on hyperparameter optimization employs the Sequential Model-Based Optimization (SMBO) framework [10]. SMBO methods learn a surrogate model on already observed hyperparameter performances to ultimately predict a new hyperparameter configuration to test. This choice is based on the predicted performance as well as the uncertainty of the surrogate model for that very hyperparameter configuration. Spearmint [11] employs Gaussian processes as surrogate model, whereas SMAC uses a random forest [12]. Several proposals have been made to incorporate meta knowledge from previously solved data sets, for example [13] learns an initialization strategy based on meta features, [14] employs active testing on observed hyperparameter performance. The work of [15] proposes to learn a ranking SVM on hyperparameter performances whose predictions are then used in a Gaussian process, whereas [16] uses a Gaussian process with a Kernel function that also models the distance of different tasks to each other. All Gaussian process approaches suffer the mutual disadvantage that for large meta data sets they are not applicable anymore as the kernel matrix grows too big to store. Another common downside of many of these models is that they rely on meta features to prevent confusion of the surrogate model with same hyperparameter configurations across different data sets. Finally, our previous work [1] learns a multilayer perceptron with an additional factorization term in the first layer to directly learn latent representations for hyperparameters and tasks, which make meta features obsolete.

The most prominent work on model choice and automated machine learning is probably Auto-WEKA [17], which uses a random forest based on [12] and introduces a meta data set containing the performance of a plethora of prediction models on 21 data sets. Moreover, multi armed bandits are employed [18] in the field of model choice. A very close work that also proposes to learn latent features for algorithm choice is [19], where the latent representation is learned using Singular Value Decomposition. However, their work is more fictional as they do not consider hyperparameters and therefore learn the latent representation only after all algorithms have finished, which then of course is not needed anymore.

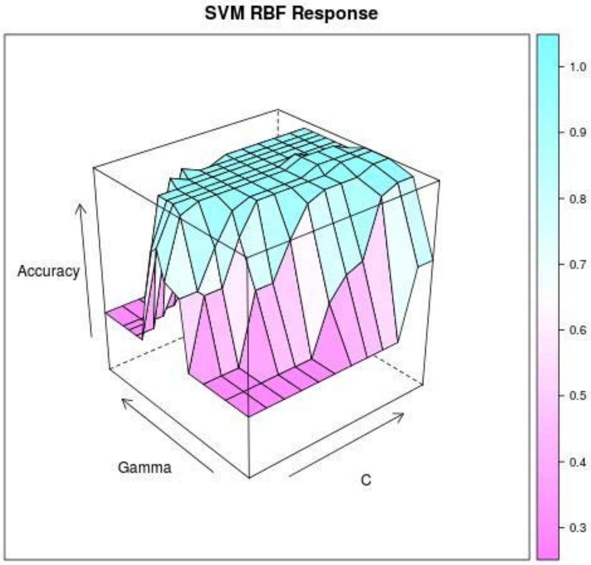

Figure 1. Response surface of an RBF Support Vector Machine on the Iris data set. Validation Accuracy is plotted against two hyperparameters, the choice of tradeoff parameter C and the kernel width γ (Best viewed in color)

We seek to treat both problems, hyperparameter optimization and model choice as the same problem. In this way, we can map prediction models as well as hyperparameters and tasks into one joint latent feature space from which interactions can be estimated and used in a cross data set, cross model surrogate.

III. Background

In this section, we will first give a brief definition of hyperparameter optimization and model choice. Afterwards, we review the sequential model-based optimization (SMBO) framework and present our choice of surrogate model.

A. Hyperparameter Optimization

Let us denote by $\mathcal{D}$ the set of all data sets. Then, following the notation by [2], for a general but fixed model class $\mathcal{M}$, a machine learning algorithm $\mathcal{A}$, parametrized by some hyperparameters $\lambda \in \Lambda$ is defined as mapping

$$\mathcal{A}_\lambda : \mathcal{D} \longrightarrow \mathcal{M}_\lambda \ , \tag{1}$$

that maps training data D^{train} to a learned model M_λ under consideration of the hyperparameters λ. Note that the model class $\mathcal{M}$ does not necessarily have to depend on λ, however, in many cases λ also defines the model's complexity. In most cases, Λ is the direct product of individual hyperparameter spaces, i.e. $\Lambda = \Lambda_1 \times \ldots \times \Lambda_p$. Usually, $\mathcal{A}_\lambda$ searches through the model space $\mathcal{M}_\lambda$ and picks the model M_λ with lowest objective function

$$\mathcal{A}_\lambda(D) = \underset{M_\lambda \in \mathcal{M}_\lambda}{\arg\min}\, \mathcal{L}(M_\lambda, D) + \mathcal{R}(M_\lambda) \ , \tag{2}$$

where $\mathcal{L}(M_\lambda, D)$ is a loss functional and $\mathcal{R}(M_\lambda)$ is a regularization term. The *hyperparameter optimization problem* is then defined as finding the optimal hyperparameter λ^*, where the resulting model after learning minimizes the loss $\mathcal{L}$ on a given validation set.

$$\lambda^* = \underset{\lambda \in \Lambda}{\arg\min}\, \mathcal{L}(\mathcal{A}_\lambda(D^{\text{train}}), D^{\text{val}}) \tag{3}$$

In Figure 1, validation accuracy is plotted for an RBF-SVM for the hyperparameters tradeoff C and kernel width γ on the well-known Iris data set from the UCI repository[1]. We denote the function that maps hyperparameters to a learned model and then evaluates the learned model on validation data by the function f, i.e. in the above equation we replace

$$f(D, \lambda) = \mathcal{L}(\mathcal{A}_\lambda(D^{\text{train}}), D^{\text{val}}) \ . \tag{4}$$

The observations of f are called hyperparameter response or validation performance throughout this paper.

For simplicity, the reader may consider a linear regression model learned by gradient descent for the notation presented above. Then, $\mathcal{M} = \mathbb{R}^k$ for a linear regression on $k-1$ features plus an additional bias term and λ can be denoted as $\lambda = (\mu, \nu)$ consisting of regularization constant $\mu > 0$ and learning rate $\nu > 0$. Consequently, $\mathcal{L}$ is the Root Mean Squared Error (RMSE) and f would map a data set and hyperparameters to the RMSE error of a learned model on the validation part of the data set.

B. Model Choice

Using the notation defined above, we can write down the problem of model choice in the same manner. Given a set of n algorithms with fixed model class $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ to learn, the problem of *model choice* can be stated as finding the algorithm $\mathcal{A}^*$ where the resulting model has minimal error on validation data.

$$\mathcal{A}^* = \underset{\mathcal{A} \in \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}}{\arg\min}\, \mathcal{L}(\mathcal{A}(D^{\text{train}}), D^{\text{val}}) \tag{5}$$

As both problems are very similar, we seek to denote them as one joint problem. This can be done by letting λ also describe the choice of model, then we return to the hyperparameter optimization but would care specifically for only one dimension of λ.

C. Sequential Model-Based Optimization

As f involves calling the whole learning algorithm and assessing the resulting validation performance, direct evaluation of f is usually very expensive, therefore black-box optimization may be readily applied. Suppose we have observed f on a discrete subset $G \subset \Lambda$ of the hyperparameter space, and let us denote the set of already observed tuples by $\mathcal{H}$. Then, instead of querying f at certain points, SMBO learns a surrogate model $\Psi \approx f$ to predict the response of f for unknown hyperparameter configurations, and use its predictions to choose a next hyperparameter configuration to test. Usually, predictions of Ψ are then fed into an acquisition function that seeks for an optimal trade-off between exploitation and exploration. On the one hand, exploitation means that only hyperparameter configurations in well-known regions of f are chosen, which might lead to finding only local optima in hyperparameter space. Exploration, on the other hand, means that also regions of Λ are chosen where the surrogate Ψ is really uncertain and therefore improvement might be large. As acquisition function, the Expected Improvement (EI) [10] is the most widely used, however, there exist other acquisition functions such as probability of improvement [10], conditional entropy of the minimizer [20] or multi armed bandit based criteria [21]. Afterwards, the new hyperparameter configuration is evaluated and the observation is added to the current observation history $\mathcal{H}$. In the end, after T many iterations the best hyperparameter configuration found is returned. The resulting algorithm is sketched in Algorithm 1. Note that we use now x instead of λ, as usually the feature vector is augmented by additional features describing meta entities such as the current data set and model choice.

Algorithm 1 Sequential Model-based Optimization

```
Input: Hyperparameter space $\Lambda$, observation history $\mathcal{H}$,
    target data set $D^{\text{new}}$, number of iterations $T$, acquisition
    function $a$, surrogate model $\Psi$.
Output: Best hyperparameter configuration $x^{\text{best}}$ for $D^{\text{new}}$
for $t = 1$ to $T$ do
    Fit $\Psi$ to $\mathcal{H}$
    $x^{\text{new}} = \underset{x}{\arg\max}\ a(x, \Psi(x))$
    Evaluate $f(x^{\text{new}}, D^{\text{new}})$
    if $f(x^{\text{new}}, D^{\text{new}}) > f(x^{\text{best}}, D^{\text{new}})$ then
        $x^{\text{best}} = x^{\text{new}}$
    $\mathcal{H} = \mathcal{H} \cup (x^{\text{new}}, f(x^{\text{new}}, D^{\text{new}}))$
return $x^{\text{best}}$
```

D. Involving Meta Knowledge

As was already mentioned above, the SMBO procedure is not limited to the current data set on which hyperparameter optimization is being conducted. Before starting SMBO, we may already have observed hyperparameter performance

[1] https://archive.ics.uci.edu/ml/datasets/Iris

for learning the same model on a different data set. This is possibly valuable information for a surrogate model, as usually hyperparameter performance does not change too drastically for different data sets. However, for a fixed choice of model the hyperparameter space is also fixed, and therefore the observation history across data sets would contain a lot of instances with exactly the same features but a different label, which possibly only confuses the surrogate model.

To overcome this problem, meta features that are supposed to describe the relevant characteristics of a data set have been derived [22] and been used for example in collaborative hyperparameter tuning [15] and for a faster hyperparameter optimization through a smarter initialization [13]. These features can be very simple statistics such as number of classes in a classification problem, number of instances, number of features and so on. However, meta features can also be more computationally demanding, such as the prediction accuracy of a simple linear model, usually they are then called landmark features [23]. One problem of meta features is that up to now it is not entirely clear which meta features are helpful and which are not, the hope is usually that some help and the surrogate model performs a feature selection automatically. Moreover, meta features as defined above are dependent on the task, for example the distribution of classes has no meaning in case of a regression task and therefore cannot be applied in general.

If we consider model choice, we might not face this problem as models usually have different hyperparameters and therefore features change depending on the model choice. In general, however, this is not true and therefore we have to think of ways of making observations also distinctive for different models. Following the road of meta features, we could now start to think of meta features that describe models, such as if the model is linear or nonlinear and so on. However, in this case we have to define new meta features for every model, if we continue and want to generalize not only over data sets and model choice but for instance preprocessing as well, we have to start over again and define these features for the new entity we want to generalize over. Clearly, this does not seem to be the optimal strategy.

Another way to overcome the problem of same but differently labeled instances is to add binary categorical features that encode directly to which data set or model choice the observation can be mapped. Thus, for a set $\{D_1, \ldots, D_n\}$ of observed data sets and a set of $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ prediction models, we augment the input of f by a variable x which then also carries information about the data set and the prediction model involved

$$x = (\lambda, \delta, \mu) \ . \tag{6}$$

The variables δ and μ are defined as

$$\delta(D_j) = (\delta_1, ..., \delta_n) \quad \delta_i = \mathbb{1}(i = j) \ , \tag{7}$$

and in a very analogous way

$$\mu(\mathcal{M}_j) = (\mu_1, ..., \mu_n) \quad \mu_i = \mathbb{1}(i = j) \ , \tag{8}$$

where $\mathbb{1}$ attains the value 1 if its predicate is true and 0 otherwise. This is probably the simplest but also most uninformative meta feature that we can think of. In case of a standard regression model as surrogate we would only be able to learn a bias for the as well as for model choice, respectively. However, we argue that by associating latent features to these variables and optimizing for them, we are able to directly learn latent characteristics of the entity we want to generalize over. In this way, we are also able to model interactions between all entities.

E. Factorized Multilayer Perceptrons

In [1], we have proposed to use factorized multilayer perceptrons (FMLP) for hyperparameter optimization. They basically are a feedforward multilayer perceptron but instead of linear signals forward the prediction of a factorization machine [24] in the input layer. Let us denote by $w^l_{ik} \in \mathbb{R}$ the weight that maps the i-th input to the k-th neuron in layer l, and by $v^l_{ik} \in \mathbb{R}^K$ we define the K-dimensional vector of latent features that maps the i-th input to the k-th neuron in layer l. Then, we write

$$s^l_k = w^l_{0k} + \sum_{i=1}^{n} w^l_{ik} x^{l-1}_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v^l_{ik}, v^l_{jk} \rangle x^l_i x^l_j \ , \tag{9}$$

as signal function for $l = 1$ and for subsequent layers, we use the regular perceptrons

$$s^l_k = w^l_{0k} + \sum_{i=1}^{n} w^l_{ik} x^{l-1}_i \ . \tag{10}$$

The input for the next layer is then computed by applying a sigmoid function to the signals

$$x^l_k = \sigma(s^l_k) \ , \tag{11}$$

where we use the logistic function $\sigma(t) = (1 + \exp(-t))^{-1}$. As the categorical indicator variables are only given in the input layer we restrict the model to factorize regression weights only in the first layer. In this way, we combine the approximation quality of a complex nonlinear model such as multilayer perceptrons with the latent feature representation that a factorization is capable to model. As our previous work shows [1], using only factorization machines does not show strong performance as the prediction model is simply not complex enough to model arbitrary hyperparameter response surfaces. We learn FMLPs using the famous backpropagation algorithm [25] for the Root Mean Squared Error, which requires one forward and one backward pass through the network to update all model parameters.

The resulting optimization procedure is given in algorithm 2. Note that we dropped the dependency on the layer for the update of latent features and use the upper index to

Algorithm 2 SGD-Backpropagation for Factorized Multilayer Perceptron

```
Input: Data Set $D$, Loss function $\mathcal{L}$, step length $\eta > 0$.
Initialize $v$ and $w$
repeat
    Draw $(x, y) \in D$
    Predict $\hat{y}(x)$
    Compute $\delta_k^l = \frac{\partial \mathcal{L}(\hat{y}(x),y)}{\partial s_k^l} \cdot \frac{\mathrm{d}\sigma}{\mathrm{d}s_k^l}$ for all layers $l$ and neurons $k$
    Update $w_{i,k}^l = w_{i,k}^l - \eta \delta_k^l x_i$ for all layers $l$ and neurons $k$
    For the input layer $l = 1$:
    Precompute $\mu_j^k = \sum_{i=1}^n v_{i,j}^{(k)} x_i$
    Update $v_{i,j}^{(k)} = v_{i,j}^{(k)} - \eta \delta_k^l (x_i \mu_j^k - v_{i,j}^{(k)} x_i^2)$
until Convergence
return $(w, v)$
```

denote individual elements of a latent feature vector v_{ik}. In our implementation, we also use update parameters with respect to a momentum term, this has been dropped to avoid unnecessary clutter. We use a Gaussian initialization centered around 0 for latent features v and the Nguyen-Widrow initialization [26] for the weights w, as it empirically results in faster convergence of the whole network.

IV. Experiments

In this section, we first present the meta data set, then discuss experiment setups, competing surrogate models and in the end present the results.

A. Meta Data Set Creation

For assessing empirical evidence, we created a large meta data set that encompasses 59 classification data sets. If splits were already existing, we merged the given data and created one split where 80% have been used for training and the remaining 20% have been used to obtain test performance. From now on, let us denote the set of observed data sets by $\mathcal{D}$. As the task is classification, $f(\lambda, D)$ models the classification accuracy of a learned classifier with given hyperparameters λ on the validation partition of D. Moreover, we applied 19 different prediction models for the classification task by using WEKA [27]. An overview of employed prediction models is given in Figure 3. For a more detailed overview over data sets and hyperparameter grids that have been chosen, we refer the reader to our supplementary web page [28], where also the full meta data set and program code (Java) can be downloaded for reproduction purposes. For a fixed data set, in total we have conducted $21,871$ experiments, which for 59 experiments leads to a total number of approximately 1.29 million observed instances.

B. Competing Methods

We give a brief overview over all competing surrogate models.

1) Independent Gaussian Process (I-GP): This tuning strategy learns a Gaussian Process with squared exponential automatic relevance determination (SE-ARD) kernel [11], independent from observations on all the other data sets. Uncertainty is assessed by using the predictive posterior distribution.

2) Independent Random Forest (I-RF): This tuning strategy learns a Random Forest as proposed in [12] and which is also used by AUTO WEKA [17]. As well as the I-GP, it is learned independently from observations on other data sets. It consists of 100 individual decision trees, uncertainty is obtained by computing the sample variance.

3) Random Forest (RF): This tuning strategy learns a Random Forest with additional data set and model indicator variables to allow generalization over data sets and model choices. Again, 100 individual trees are learned and uncertainty is assessed using the sample variance.

4) Factorized Multilayer Perceptron (FMLP): This tuning strategy is the one we propose in [1] and in section and show how it can be augmented to learn across models in section III-E. It embeds hyperparameters, models and data set characteristics into a joint latent feature space. Similar to random forests, we compute an averaging ensemble of 100 models and use the sample mean and variance as prediction and uncertainty.

C. Average Normalized Accuracy

We start by normalizing accuracies f with respect to the best and worst configuration found so far for a fixed data set D:

$$\tilde{f}(\lambda, D) = \frac{f(\lambda) - f_{\min}(D)}{f_{\max}(D) - f_{\min}(D)} \tag{12}$$

By normalizing the individual accuracies per data set we obtain normalized accuracies $\tilde{f}$, which range from 0 to 1 but more importantly are comparable between different data sets. Average normalized accuracy sums up the best normalized accuracies that were found on every data set and in the end averages them.

$$\text{avgAcc} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \max \left\{ \tilde{f}(\lambda, D_i) \right\} \tag{13}$$

This evaluation metric gives an impression of how fast the SMBO approach finds the maximum accuracy. For average normalized accuracy, the higher the better.

D. Experiment Setup

We performed a leave one out experiment for each of the 59 data sets being the target data set D^{new} once. However, to account for a more realistic setting we subsampled the original grid of $21,871$ hyperparameter configurations by

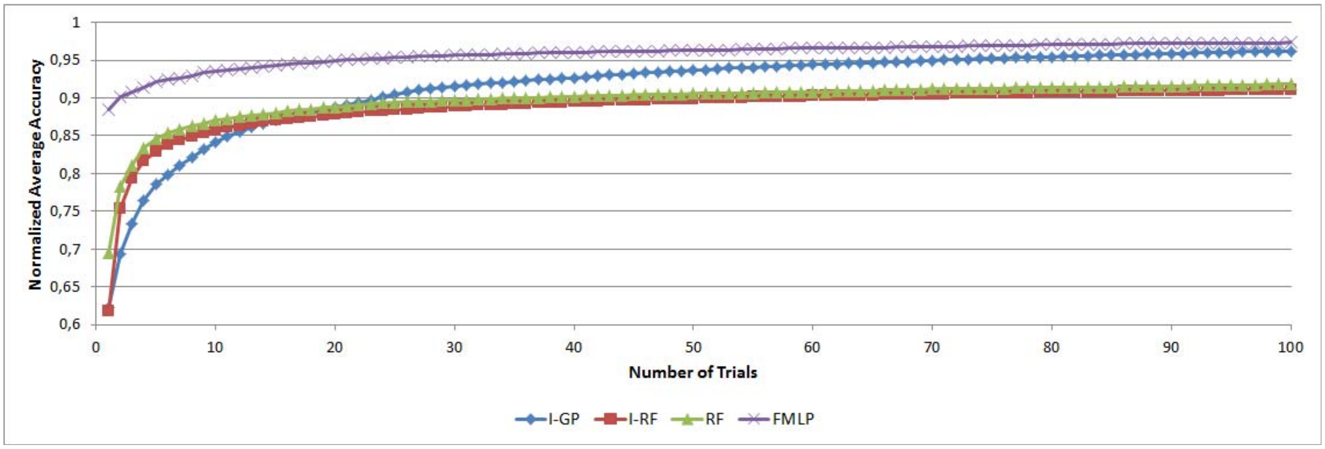

Figure 2. Average Normalized Accuracy is shown versus the number of trials already conducted. Optimal is omited as it is has a constant accuracy (Best viewed in color).

taking only every third hyperparameter of the training data into account. For a model with p many hyperparameters, this reduces the total grid by a factor of $(1/3)^p$, a total of $1,873$ hyperparameter configurations remained for each training data set. For the test data set, however, we used the full grid. We argue that this scenario is much more realistic as for the test data there are infinitely many possible hyperparameter configurations, an amount which we will never have for training purposes.

As acquisition function, the expected improvement has been used, as it currently is the most widely used acquisition function in SMBO based hyperparameter optimization. For the data set independent surrogates we performed 1000 repetitions of the experiment as usually the start of the SMBO procedure shows some random effects. For all other surrogates, we computed 10 repetitions, as they use meta knowledge across data sets their initial variance was much lower. In total, we allowed each SMBO run to query $T = 100$ different hyperparameter configurations, which is relatively tiny compared to the grid size, but if we manage to find well performing hyperparameter configurations, this would constitute a great achievement.

The implementation of Gaussian processes with SE-ARD Kernel was done by ourselves in Java as well as the implementation of FMLPs. For the random forests, we used the implementation by MLTK[2]. GP hyperparameters are automatically tuned by optimizing the marginal likelihood. Random forest hyperparameters have been optimized using leave one out cross validation, whereas the hyperparameters of the FMLP have been chosen fixed according to previous experience with the model. We use one hidden layer with 5 neurons, the dimension of the latent feature vectors is $K = 8$, and for optimization we use a step size $\mu = 0.01$ and the same value as momentum term.

V. Results

Figure 2 shows the development of the average normalized accuracy versus trials of the SMBO algorithm for all competing methods and a total of 100 trials. In the beginning, all methods more or less suffer from the cold start issue, as no observations of the new data set have been made so far and therefore no such information could be used in learning the models. As can clearly be seen, the independent models have quite a bad start with rather poor accuracy achieved. This, however, makes a lot of sense considering that they do not employ any meta knowledge at all. It can also be observed that the random forest which uses meta knowledge has a much better start than its independent counterpart, although this advantage fades quite early after 20 to 30 trials. Interestingly, both random forests surrogate cannot maintain their advantage compared to an independent Gaussian process, which overtakes them after roughly 15 trials.

Clearly, the FMLP proposes better hyperparameter configuration at a much earlier point in time than all competing methods. It already starts out at an average normalized accuracy of almost 0.9, thus reaching already 90% of the maximum accuracy on average per data set in the first trial. We believe that this is an amazing result, compared to the other surrogates where the best one (RF) starts at almost 0.7 average normalized accuracy and the independent surrogates starting even worse at a value a little above 0.6.

Moreover, after having conducted 100 trials, which amounts for 0.005% of the hyperparameter grid, we are able to achieve an average normalized accuracy of almost 0.975 using FMLPs. We argue that this advantage comes largely from learning latent representations for data sets, hyperparameters and models. The cold-start issue remains also for an FMLP, but having learned latent features for all other entities seems to mitigate this issue to a certain degree.

[2] http://www.cs.cornell.edu/~yinlou/projects/mltk/

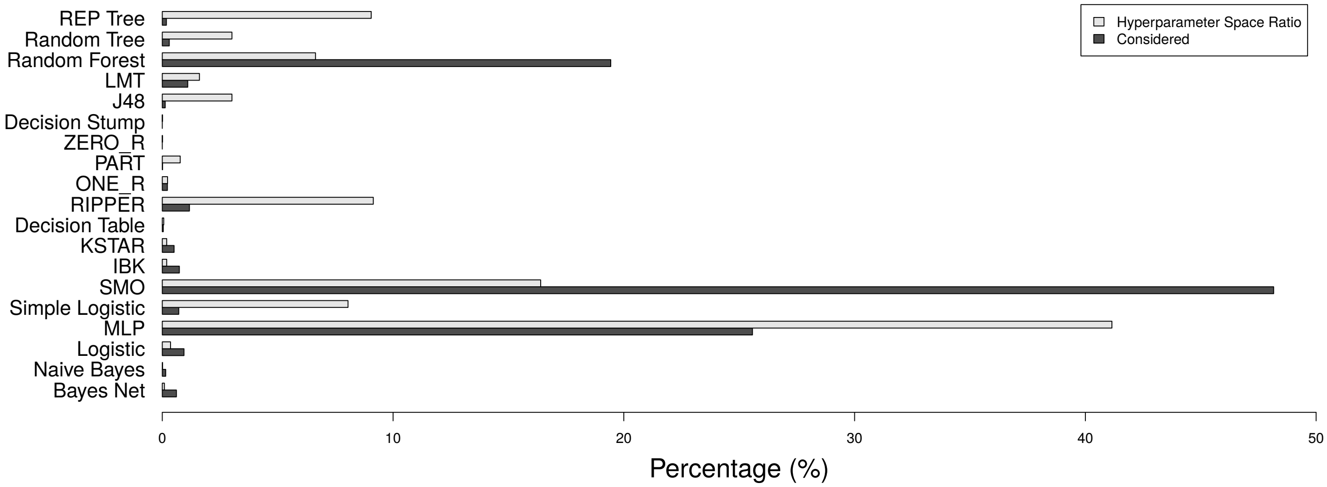

Figure 3. List of employed models. The dark bars indicate the average frequency of chosing a model, whereas the light bars indicate the portion of a model to the overall hyperparameter space.

Of course, with more observations also the FMLP learns characteristics of the new data set and therefore improves with time.

Figure 3 shows a list of employed models and sheds light on the model choice that an SMBO with FMLPs performs. The bars for a model indicate the frequency of how often this model has been chosen as an average over all experiments (dark), as well as the relative size of the hyperparameter space of a given model with respect to the whole hyperparameter space (light). In this way, the white bars can be understood as probability of choosing a model with a surrogate model that picks a random hyperparameter configuration.

As we can see, the FMLP surrogate highly favors the more complex models such as multilayer perceptrons, random forests and Support Vector Machines. Consequently, it chooses other models less often, for example Decision Stumps are never considered, which is likely due to their simplicity and their bad performance on other data sets. For many other models it seems that a few observations are already enough to make a decision whether it makes sense to explore the model's hyperparameter space even further. This knowledge across data sets and models is exactly the knowledge that experienced practitioners have and that we have sought to learn by applying FMLPs.

VI. Conclusion

In this paper, we have shown that a joint optimization of both hyperparameters and model choice can very effectively be done by using factorized multilayer perceptrons as surrogate models in the sequential model-based optimization framework. We conducted experiments on a large meta optimization data set of 59 individual tasks, solved by 19 different prediction models forming in total a grid of roughly 1.2 million instances. The results show that our proposed method outperforms competitor methods by a decent margin, and that a joint learning across data sets and prediction models is generally possible without the need of any kind of meta features.

For future work, one may envision also learning across different tasks, for example across classification and regression, and possibly ranking. A factorization machine [24], for instance, can solve all three tasks and employs the same hyperparameters in every scenario. Thus, a generalization across prediction tasks should also be possible, again, without defining extra meta features for tasks. We believe that our research helps non-experienced practitioners to use the vast area of machine learning for solving their application problems, and brings the area one step closer to fully automatized machine learning.

Acknowledgment

The authors gratefully acknowledge the co-funding of their work by the German Research Foundation (DFG) in the project Hyperparameter Learning Across Problems (HyLAP) under grant SCHM 2583/6-1.

References

[1] N. Schilling, M. Wistuba, L. Drumond, and L. Schmidt-Thieme, "Hyperparameter optimization with factorized multilayer perceptrons." in *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2015.

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

[3] S. S. Keerthi, V. Sindhwani, and O. Chapelle, "An efficient method for gradient-based adaptation of hyperparameters in svm models," in *Advances in neural information processing systems*, 2006, pp. 673–680.

[4] M. M. Adankon and M. Cheriet, "Model selection for the LS-SVM. Application to handwriting recognition," *Pattern Recognition*, vol. 42, no. 12, pp. 3264–3270, 2009.

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

[6] O. Chapelle, V. Vapnik, and Y. Bengio, "Model selection for small sample regression," *Machine Learning*, vol. 48, no. 1-3, pp. 9–23, 2002.

[7] P. Koch, B. Bischl, O. Flasch, T. Bartz-Beielstein, C. Weihs, and W. Konen, "Tuning and evolution of support vector kernels," *Evolutionary Intelligence*, vol. 5, no. 3, pp. 153–170, 2012.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] M. Feurer, J. T. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[14] R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in *Machine Learning and Data Mining in Pattern Recognition*. Springer, 2012, pp. 117–131.

[15] R. Bardenet, M. Brendel, B. Kegl, 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.

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

[17] 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.

[18] M. W. Hoffman, B. Shahriari, and N. de Freitas, "On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning."

[19] Y. Malitsky and B. O'Sullivan, "Latent features for algorithm selection," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

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

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

[22] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, "Machine learning, neural and statistical classification," 1994.

[23] B. Pfahringer, H. Bensusan, and C. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms," in *In Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 743–750.

[24] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 995–1000.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, 1988.

[26] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*. IEEE, 1990, pp. 21–26.

[27] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[28] N. Schilling. Supplementary website: http://hylap.org/joint-model-choice-and-hyperparameter-opt-with-fmlp.