Mach Learn (2018) 107:1–14
https://doi.org/10.1007/s10994-017-5692-y

EDITORIAL

Metalearning and Algorithm Selection: progress, state of the art and introduction to the 2018 Special Issue

Pavel Brazdil[1] · **Christophe Giraud-Carrier**[2]

Received: 27 September 2017 / Accepted: 8 December 2017 / Published online: 29 December 2017

Abstract This article serves as an introduction to the Special Issue on Metalearning and Algorithm Selection. The introduction is divided into two parts. In the the first section, we give an overview of how the field of metalearning has evolved in the last 1–2 decades and mention how some of the papers in this special issue fit in. In the second section, we discuss the contents of this special issue. We divide the papers into thematic subgroups, provide information about each subgroup, as well as about the individual papers. Our main aim is to highlight how the papers selected for this special issue contribute to the field of metalearning.

Keywords Metalearning · Algorithm selection and configuration · Hyperparameter optimization · Automated ensemble construction

1 Advances in metalearning

The field of metalearning, especially with respect to algorithm selection and configuration, has been an active area of research since the seminal work of Rice (1976). The field gained strength in the 1990's thanks to collaborative European projects, beginning with STATLOG (Michie et al. 1994), and followed by METAL (Giraud-Carrier 2005) and e-LICO (2012), among others. By the first decade of this century, the field was considered quite mature. It is interesting to analyze the text in the Introduction to the first Special Issue on Metalearning of *Machine Learning Journal* (Giraud-Carrier et al. 2004):

> The application of Machine Learning (ML) and Data Mining (DM) tools to classification and regression tasks has expanded outside the boundaries of research into the

✉ Pavel Brazdil
pbrazdil@inesctec.pt

Christophe Giraud-Carrier
cgc@cs.byu.edu

1 LIAAD Inesc Tec, University of Porto, Porto, Portugal

2 Department of Computer Science, Brigham Young University, Provo, UT, USA

> realm of applied research, industry, commerce, and government. Two key aspects play an important role in the successful application of these tools. One is the selection of a suitable predictive model (or combination of models) where expertise is seldom available a priori; users of commercial ML and DM tools must either resort to trial-and-error or expert advice. Clearly, neither solution is completely satisfactory for the end user who wishes to access the technology more directly and cost-effectively. The effectiveness of this process can be enhanced by metalearning. Metalearning assistants can provide automatic and systematic user guidance on model selection and method combination.

From the current perspective, we would still agree with most of this. However, on closer analysis we have identified a certain shift of perspective when trying to respond to the following four questions:

1. How can the metalearning process be characterized?
2. What constitutes meta-knowledge?
3. Should algorithm selection be extended to automatic design of solutions (workflows)?
4. What kind of application domains exist for this methodology?

We believe that the current answers to these issues are somewhat different from those in 2004, as discussed in what follows.

1.1 How can the metalearning process be characterized?

In 2004 metalearning was seen as a one-step process and the common view was as follows. The metalearning system would use the gathered meta-knowledge to obtain a (meta-)model which would be applied to a new problem. However, this perspective has changed. There is more stress on the process of search for the right algorithm or solution in general. While meta-knowledge can be useful in this process, many researchers stress the importance of the way the search is conducted. The aim is to obtain good solutions early in the style of *anytime algorithms*.

Various solutions have been proposed. In one, the process was speeded up by using a simplified model, referred to sometimes as *surrogate model*, to determine which test to conduct first. The articles by Eggensperger et al. and by Wistuba et al. in this Special Issue follow this line of research. Another solution takes a somewhat different approach. It schedules the fast tests before the slower ones. The meta-knowledge gathered in past tests is then used to estimate how fast and how precise the algorithms are. One such system is discussed in the article by Abdulrahman et al.

1.2 What constitutes meta-knowledge?

Surprisingly, we note that even the view of what meta-knowledge is seems to have evolved. The earlier view was that meta-knowledge includes performance data on previously seen datasets and some characteristics of these datasets. Although the definition of metalearning in Brazdil et al. (2009), which states that

> Metalearning is the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes.

is general enough, one figure in that publication (Fig. 1.1, p. 5) suggests that meta-knowledge is gathered prior to its application to some new (target) dataset of interest. However, the shift

towards search on the new dataset has led to a somewhat different view. As tests proceed on the new dataset, the tests already carried out can be viewed as also constituting meta-data. The following definition of metalearning by Lemke et al. (2015), to which we fully subscribe, makes this explicit:

> A metalearning system must include a learning subsystem, which adapts with experience. Experience is gained by exploiting meta-knowledge extracted: (a) in a previous learning episode on a single dataset and/or (b) from different domains or problems.

Hence, a new interesting topic in metalearning is how to exploit the new meta-data capturing performance results to adapt the search. This line is taken up in some systems described in this Special Issue.

1.3 Should algorithm selection be extended to automatic design of solutions (workflows)?

The introduction to the original Special Issue on metalearning states that:

> Advances in the field of metalearning hinge around one specific question: how can we acquire and exploit knowledge about learning systems (i.e., meta-knowledge) to understand and improve their performance? To answer this question we need to explain what is meant by a learning system. For our purposes, a learning system can be either (a) a single learning algorithm; or (b) a set of different learning algorithms, all employed on the same task.

Regarding (a), the authors further explain that:

> ... it is important that learning algorithms are able to profit from their repetitive use over similar tasks. Ideally, the models should continuously adapt to new needs. This is usually done by re-learning. Meta-knowledge can capture the cumulative expertise gained on different tasks. The process of re-learning either maintains the learning algorithm unchanged (as in incremental learning), or else enables modifications. Meta-knowledge can be used to control these modifications, which can be either rather simple, such as opting for a particular parameter setting or a particular strategy for parameter optimization; or more complex, such as is the case with architectures that evolve through experience.

Many researchers in the last decade have worked on the problem of how to determine the potentially best parameter setting of a given algorithm (or a set of algorithms) for the target dataset. The articles by Eggensperger et al. and by Wistuba et al. in this Special Issue propose new solutions along these lines, while the article by Kordik et al. proposes the use of genetic algorithms for architectures that evolve through experience.

Regarding (b), the authors state that:

> Given a set of learning algorithms, we can ask the following: (i) which algorithm is best suited for a given task or application domain? or (ii) which is the preferred ordering of the given algorithms? or (iii) what is the form of the composite classifier to be employed in the new task?

These questions are of course still valid. Some researchers have focused on trying to find a satisfactory answer to (ii), as this provides one simple way to answer (i). The article by Abdulrahman et al. in this Special Issue discusses one such strategy.

Although we would not disagree with the perspective presented in 2004, it does miss certain important aspects. The simple categorization into:

> (a) a single learning algorithm; or (b) a set of different learning algorithms, all employed on the same task

fails to cover the breadth of current work. Many systems today are concerned both with algorithm configuration (including hyper-parameter optimization) and selection of algorithms. Again, the articles by Eggensperger et al., by Wistuba et al., and by Kordik et al. in this Special Issue tackle this more complex problem.

The original notion of algorithm has also been extended to a sequence of operations, often referred to as a *workflow* or sometimes also as *pipeline of operations*. Many articles have been published on this since the seminal work of Bernstein and Provost (2001) and Bernstein et al. (2005). The problem of how to design workflows automatically was also investigated by various researchers within e-LICO (2012). Various methods have been incorporated in a number of systems, such as RapidMiner (Hofmann and Klinkenberg 2013; Kietz et al. 2012), WEKA (Witten et al. 2011), and software library Scikit-learn (Pedregosa et al. 2011), giving rise, for example, to Auto-WEKA (Thornton et al. 2013; Kotthoff et al. 2017) and AUTO- SKLEARN (Feurer et al. 2015). A recent DARPA-led initiative, known as Data-driven Discovery of Models, is also focused on the automation of the complete data mining process (i.e., workflows). Results are not yet available. While we cannot present an exhaustive review of all current approaches and systems, our aim has been to select articles describing some of the latest advances in metalearning, that could be used in their re-design or in the design of new systems that respond better to current user requirements.

1.4 What kind of application domains exist for this methodology?

The work in the area of metalearning and algorithm selection carried out in the 1990's was focused predominantly on machine learning. As time went on, researchers began looking at other areas, which lead to improvements in the methodology. An insightful overview by Smith-Miles (2008) discusses applications to sorting, forecasting, constraint satisfaction, and optimization, and the extension of these ideas to, for instance, bioinformatics.

The papers in this Special Issue also cover the application of metalearning to various areas. The articles by Abdulrahman et al., by Muñoz et al., by van Rijn et al., and by Kordik et al. focus on classification. The article by Lorena et al. addresses regression. The article by Malone et al. is concerned with Bayesian approaches. Finally, the articles by Eggensperger et al. and by Wistuba et al. focus on optimization.

2 Organization of the special issue and main contributions

The call for papers resulted in 19 submissions in May 2016. Each submitted paper was reviewed by at least 3 reviewers, while some of the papers have undergone several cycles of revision and reviewing. This process resulted in 9 accepted papers for this Special Issue. We have organized the accepted papers into four thematic subgroups, as follows.

- Configuration of algorithms for hyperparameter optimization (2 papers).
- Extending/improving metalearning and algorithm selection methods (2 papers).
- Ensemble construction (2 papers).

– Applying metalearning and algorithm selection methods to new application domains (3 papers).

To provide an overview and assist interested readers in their selection of further reading, we summarize and highlight the main contributions of each paper.

2.1 Configuration of algorithms for hyperparameter optimization

Eggensperger, Lindauer, Hoos, Hutter and Leyton-Brown in their paper entitled *Efficient Benchmarking of Algorithm Configuration Procedures via Model-Based Surrogates* discuss various approaches whose aim is to identify a potentially best set of hyperparameters for a varied set of tasks. This task is often referred to as *hyperparameter optimization (HPO)* or *algorithm configuration (AC)*. The authors exploit the notion of simple models referred to as surrogate models that enable to predict effectively a promising configuration to be tested next. The use of surrogate models is of course not new and builds on previous work. Hutter et al. (2014) and Eggensperger et al. (2015) have considered several common regression algorithms for predicting algorithm performance. The best performers were *random forests (RFs)* (Breiman 2001) and *Gaussian processes (GPs)* (Rasmussen and Williams 2006). In this paper, the authors have opted for quantile regression forest (QRF). They optimize the hyperparameters using sequential model-based configuration (SMAC) (Hutter et al. 2009). This process exploits *empirical performance models (EPMs)* (Leyton-Brown et al. 2009; Hutter et al. 2014), which are regression models that characterize a given algorithm performance across problem instances and/or parameter settings.

Conducting experiments with parameter configurations is in general a tedious task. Many AC procedures include so called *adaptive capping* technique based on an earlier idea of *racing*. This technique enables to terminate prematurely a configuration if it is provably worse than another one.

Previous approaches had, however, one limitation, as typically the method would exploit only the information gathered on the target dataset. Typically, these approaches would not consider potentially useful knowledge that could already be available from previous experiments on other datasets. In other words, previous approaches used just the meta-knowledge gathered on the target dataset. One of the contributions of this paper is that the authors incorporate this knowledge into the model. As in other work in the area of metalearning the authors used certain datasets as training data to train their meta-model. This involved testing various hyperparameter configurations on the training datasets. This method has the advantage that it permits to focus on the high-performance regions of the parameter configuration space, in a way somewhat similar to *irace* (Lópes-Ibáñez et al. 2011). The authors generate new training data to give preference to these high-performance regions of the parameter configuration space.

The authors have conducted experiments with various AC systems including SMAC, ParamILS (Hutter et al. 2009), ROAR (Hutter et al. 2011) and *irace* (Lópes-Ibáñez et al. 2011). The last system was used on some datasets only. Eleven datasets were used in total covering combinatorial problems (mixed-integer programming (MIP), propositional satisfiability (SAT), AI planning, and answer set programming (ASP)) and HPO tasks. Each dataset can provide various dataset instances. The authors had two aims. The first one was to evaluate the precision of predictions of time on a target dataset in LOCO mode (the target dataset was not a part of the training data). SMAC, QRF and ROAR QRF were the best performers. The second aim was to evaluate the effect of incorporating surrogate models in different systems. The authors show that this is overall quite advantageous and can lead to significant speed-ups of 1–3 orders of magnitude.

Wistuba, Schilling and Schmidt-Thieme in their article *Scalable Gaussian Process based Transfer Surrogates for Hyperparameter Optimization* also employ surrogate models based on sequential model-based optimization (SMBO). This work is concerned both with hyperparameter optimization and algorithm selection. In this paper the authors have opted for Gaussian processes, unlike the previous paper that exploited quantile random forest (QRF). As the authors point out, recent work on SMBO includes also meta-data, that is performance of different hyperparameter configurations on other datasets (e.g., Bardenet et al. 2013). However, the authors point out that Gaussian processes do not scale-up well with growing meta-data. This is because they involve inversion of a kernel matrix, which limits their applicability. The authors propose to learn individual surrogate models on different datasets and an additional surrogate model learned on the target dataset. Different surrogate models are then combined into a joint model using an ensembling technique. The final surrogate is a weighted sum of the individual surrogate models. The authors call this approach SGPT and have defined three different variants. In this overview we mention just one, SGTP-R, that uses pairwise hyperparameter performance descriptors. This variant obtained better experimental results than the other two variants.

The basic proposal outlined above is then developed further. The authors note that the basic proposal suffers from two shortcomings. One stems from the fact that the performance on different datasets may cover different ranges of values. The second one is that in the basic proposal the weights of different components do not change, despite the fact that as tests proceed on the target dataset, the meta-data on this dataset is richer. These observations led the authors to propose a new variant of the surrogate framework, referred to as *transfer acquisition function (TAF)*. This function is defined as the weighted average of the expected improvement on the new data and the predicted improvement on all other data sets from previous experiments. Each hyperparameter configuration is characterized by the two components that complement each other. The first one, the expected improvement on the new data set is rather unreliable in the early trials. Hence, the suggestions provided by the meta-data are followed. This favors hyperparameter configurations that have been good on average on different datasets. As time proceeds and as more information about the new data set has been collected, the expected improvement prediction becomes more reliable, and consequently, the meta-data starts to play a minor role in the process. Similarly, as with SGPT, the authors have defined three different variants. Here, we focus on just TAF-R, which, similarly as SGPT-R, uses pairwise descriptors. It obtained better experimental results than the other two variants.

An empirical evaluation was carried out on two problems. The first meta-dataset was created by running SVM with different hyperparameter configurations on 50 UCI data sets. The authors have compared their proposed system to 6 other state-of-the-art systems on this dataset. They have shown that after 30 trials there is no significant difference between their proposed methods TAF and SGPT and the oracle method that chooses for any data set the best performing hyperparameter configuration after the first trial. The next best system is MKL-GP, Gaussian Process with Multi-Kernel Learning (Yogatama and Mann 2014).

The second problem involved running 19 different Weka classifiers on 59 data sets with 21,871 hyperparameter configurations. SGPT was the winner after 30 trials, but two other systems were not significantly worse. The situation changed after 200 trials. TAF-R was the winner, but several other systems (mainly GP-based) were not significantly different. The system I-RF (Independent Random Forest) that uses SMAC (Hutter et al. 2011), which is used in AUTO- SKLEARN (Feurer et al. 2015) and Auto-WEKA (Thornton et al. 2013), was slightly inferior to the top equivalence group.

2.2 Extending/improving metalearning and algorithm selection methods

The paper *Speeding up Algorithm Selection using Average Ranking and Active Testing by Introducing Runtime* by Abdulrahman, Brazdil, van Rijn and Vanschoren discuss two methods oriented towards selection of algorithms, whose aim is to exploit the meta-data in past experiments and identify the potentially best algorithm. The issue is then how to use the meta-data to schedule tests on the target dataset. The first method (AR) is a rather simple one. It calculates an average ranking for all algorithms over all datasets. As the aim is to identify well-performing algorithms early, the upgraded method gives preference to those algorithms that achieve relatively high accuracies, but are also fast to train. The novelty here lies in the use of A3R measure that combines accuracy and run time, instead of just accuracy. So, different algorithms are simply ranked based on this composite measure. The rankings obtained on different datasets are combined in the usual manner.

It is of course necessary to establish the correct balance between accuracy and runtime. Giving too much weight to runtime promotes rather poor algorithms and hence time could be wasted testing them. Giving too little weight to runtime has the opposite effect. It promotes tests on algorithms, which exhibit good performance in many situations, but may occasionally fail. Unfortunately, such algorithms tend to be rather slow and so testing them early may actually delay the whole process of identifying good algorithms. The authors present a relatively simple solution by setting the value of parameter P in the A3R measure. The best setting was determined experimentally and the corresponding optimized version is referred to as AR*. The authors have shown that the optimized version can identify the potentially best algorithm much faster than the AR version that uses just accuracy. The speed-up is quite substantial, between 2 or 3 orders of magnitude.

The authors have also investigated the problem of how the performance of this method is affected by incomplete meta-data. Upgrading AR to this setting requires a method that is capable of aggregating a set of incomplete rankings. This is resolved by attributing different weights to different rankings, depending on their length. The proposed aggregation method is quite fast and still achieves quite good results. The authors demonstrate that the AR* method is relatively robust to omissions in the meta-data. This finding could be explored in future studies, as it shows that it is not necessary to carry out exhaustive testing to collect the performance meta-data.

The second method discussed in this paper, active testing (AT), is based on the work of Leite et al. (2012). It uses the meta-data from past experiments to initialize the search for the potentially best algorithm. The search itself uses both the past meta-data and the meta-data obtained on the target dataset obtained in previous tests. The approach uses a concept of the current best algorithm and the aim is to identify the best competitor. This is done using the estimate of performance gain, ΔPf. This concept is somewhat related to the concept of *expected improvement (EI)*, used in some approaches to algorithm selection/configuration that exploit surrogate models, as in the work of Witsuba et al. described here.

The authors also show how their AT method can be upgraded to incorporate the measure that combines accuracy and runtime. Similarly as with AR, it is necessary to establish the right balance between the contributions of accuracy and runtime. The performance of the upgraded version AT* exceeds by large the original version AT.

The experimental study of both AT* and AR* methods showed that they led to comparable results, although AR* was marginally better. This is surprising, as AR* is much simpler than AT*. Thus it represents quite useful, but tough baseline that could be used by others. We expect that if more candidate algorithms (or their variants) were used, the AT* method could

beat AR*. It is also conceivable that the search method used within AT* could be improved and hence compete better with AR*.

The paper *Instance Spaces for Machine Learning Classification* by Muñoz, Villanova, Baatar and Smith-Miles address the issue of performance evaluation of machine learning classifiers. The authors examine the relationship between the data sets used in the evaluation—referred to as *test instances*—and the validity of the conclusions drawn. Ideally, the test instances should reveal the strengths and weaknesses of different algorithms. The authors argue that if this is not the case, new instances should be sought to provide the necessary insights.

A comprehensive methodology has been developed to enable to assess the quality of a given set of test instances. These included 235 dataset instances in total, 210 of which were from UCI, while the remaining ones were from other repositories. The methodology proposed relies on a good characterizations of all instances. The authors have considered 509 features for this task. The aim was to select a small subset that would characterize well the hardness of the classification task. The authors have proposed a novel method for this purpose. Each problem instance is used to generate a new version which is either more or less challenging in terms of a specific classification challenge. The authors identify 12 challenges. One, for instance, is non-linear separability. The original and altered datasets are compared in terms of their values for all candidate features. A statistically significant difference in values suggests that the applied alteration results in a change of the feature value and hence that the feature is relevant to measuring the degree of the challenge presented by an instance.

This way the authors have identified just 10 features. These are: maximum normalized entropy of the attributes, normalized entropy of class attribute, mean mutual information of attributes and class, error rate of the decision node, standard deviation of the weighted distance, maximum feature efficiency, collective feature efficiency, training error of linear classifier, fraction of points on the class boundary, nonlinearity of nearest neighbor classifier. These features represent an interesting finding in itself that can be exploited by the ML community.

The ten most important features are used in further analysis presented in this paper. The aim of the authors is to project the 10-dimensional space into a 2-dimensional space, enabling thus to visualize classification datasets as points in a two-dimensional space. The visualization reveals pockets of hard and easy instances and area in the 2D space, referred to as footprint, where the particular algorithm is expected to do well. Quantitative metrics, such as the area of the footprint, provide objective measures of the relative power and robustness of an algorithm across the given range of test instances.

The results presented in this paper demonstrate the lack of diversity of the given test instances, as most algorithms had similar footprints, suggesting that either the algorithms are all essentially rather similar, or that the instances are not revealing the strengths and weaknesses of each algorithm as much as is desired, or that the features may not be discriminant enough.

Regarding the last point, the authors have proposed a method to generate new test instances, aiming to enrich the diversity of datasets. The proposed method requires defining the target vector of features, f_T, as input. The method then tunes a Gaussian Mixture Model (GMM) with the objective to decrease the difference (in terms of MSE) between f_T and f_S, representing the feature vector of a sample from the GMM. As the authors have shown, this process can lead to datasets covering better the 2D space.

The authors have outlined a method for generating new data instances, which can be potentially useful. Future work should show that the richer meta-data is useful and indeed facilitates the process of selecting algorithm for a new dataset instance.

2.3 Ensemble construction

The work of van Rijn, Holmes, Pfahringer and Vanschoren, described in the paper *The Online Performance Estimation Framework: Heterogeneous Ensemble Learning for Data Streams*, is a small departure from the traditional definition of metalearning, but relevant in the sense that it dynamically exploits the complementary strengths of various classification learning algorithms to create ensembles of increased performance over changing data streams. The basic idea is to track the performance of various learning algorithms over recent observations and to combine their predictions accordingly. Specifically, the authors propose the notion of online performance evaluation where the performance of a classifier is given by the value of an appropriate loss function averaged over a fixed-size window sliding across the data stream as new training examples become available. When the average is unweighted, the approach is known as a *windowed (WD) estimation*; when the relative recency of observations is considered so that higher importance is given to more recent training examples and the effect decreases through time, the approach, controlled by a parameter alpha, is known as *fading factor estimation (FF)*.

Beginning with a set of 25 data stream classification learning algorithms, the authors, acknowledging the fact that there may exist both clear distinctions among these algorithms, but also some significant similarities, proceed to cluster them using classifier output difference (COD) as the distance function. The resulting hierarchical COD-based clustering, together with an adequate threshold value, allows the authors to reduce the total number of candidate classifiers from 25 to 7 by selecting a representative in each of the corresponding clusters. Using their proposed Best Last (BLAST) ensemble framework to implement their approach (both WD and FF), they then report on experimental results with 60 data stream classification tasks. For comparison, they use (1) the classifier that performs best on average (here, Hoeffding Option Tree), to highlight the gain obtained by ensembling several classifiers; (2) a majority vote ensemble of the same 7 base classifiers, to highlight the value of online performance evaluation over a simple vote across classifiers; and (3) a number of state-of-the-art techniques for data stream classification (here, online bagging, leveraging bagging and accuracy weighted ensemble, each with 128 ensemble members), to validate their approach. The results show that, on average, the proposed approach performs competitively with the best state-of-the-art approaches at a significantly lower computational cost. Given the variations across data streams, future work could consider a further analysis, possibly using metalearning, of what types of data streams would cause each approach to work best. Interestingly, neither the value of the fading parameter nor the number of active classifiers considered by BLAST, have significant impacts on performance.

In their paper *Discovering Predictive Ensembles for Transfer Learning and Meta-learning*, Kordik, Cerny and Fryda describe a method that employs the concept of evolutionary algorithms to optimize the topology and parameters of hierarchical ensembles. So, for instance, a given bagging ensemble can include individual parametrized algorithms, such as a decision tree (DT) with the *splitmin* parameter set to 5, a boosting ensemble, 5NN and another DT with another parameter setting. Hierarchical ensembles have attracted attention recently, as they can achieve very good performance. As the authors have pointed out, gradient boosting and multi-level stacking of neural networks were parts of the winning solution in the Netflix competition (Töscher and Jahrer 2009; Bennett and Lanning 2007). However, hierarchical ensembles are often tailored to one particular problem (data set), where they exhibit excellent performance, but tend to fail with different data. The prevailing approach to constructing these ensembles is manual trial-and-error combined with extensive hyperparameter optimization.

This motivated the authors to develop an automatic method that employs the concept of evolutionary algorithms.

The basic building blocks in this process are so-called *templates*, which can be evolved by genetic programming. The process starts with a set of simple solutions, which are evolved into more complex ones. This process is controlled both by fitness and by a kind of grammar that controls how a particular node can be expanded. So, for instance, the node *metaClass* can be expanded into *Bagging(classify^n)* or *Boosting(classify^n)*, etc., where *n* represents the number of elements. Similarly, the node *baseClassification* can be expanded into *NeuralNetClassifier* or *DecTree*, etc. As this process can be rather slow when the dataset is large, the authors develop their solutions on small data samples and apply these to full data.

The process of evolving templates can either start from scratch, or else reuse metalearning templates (in meta-database) developed earlier to start the process. This strategy can be compared to the strategy described by Feurer et al. (2015) that uses meta-data to initialize the surrogate-based search for the best parameter settings.

The authors found out that template-base methodology is prone to overfitting the data in spite of very low number of their parameters, especially when a meta-database is employed. Meta-database can be however very useful in speeding up the convergence when enough data is available for independent model validation.

The generalization accuracy of classifiers generated exceed the accuracy obtained by the most popular classifiers, including various types of ensembles. The authors demonstrate how one particular evolved template (simple ensemble of fast sigmoidal regression models) outperforms state-of-the-art approaches on a rather large Airline data set.

2.4 Applying metalearning and algorithm selection methods to new application domains

While work on metalearning has mostly focused on classification algorithm selection, Lorena, Maciel, Miranda, Costa and Prudêncio, in their paper *Data Complexity Meta-features for Regression Problems*, investigate its use in the context of regression. They begin by arguing, quite correctly, that meta-features designed for classification tasks are not appropriate for regression tasks, and thus introduce a set of twelve novel, regression-specific meta-features. As in the case of classification, these meta-features attempt to capture certain salient aspects, such as complexity of regression tasks that are likely to influence the choice and performance of regression algorithms. In particular, four meta-features compute information about the correlation between the values of the input variables with the values of the target output, where high correlation would be indicative of simpler tasks; two meta-features are intended to determine whether the underlying data can be modeled by a linear function, where linear fits would also clearly be indicative of simpler tasks; three meta-features attempt to characterize the smoothness and shape of the function that may best fit the data, where smoother functions are indicative of tasks with simpler relationships between inputs and outputs and hence simpler tasks; and three meta-features focused on the geometry of the underlying data, where simpler topology and higher density are indicative of simpler tasks.

Having proposed these regression-specific meta-features, the authors embark on a number of validation experiments involving both synthetic and real-world data sets. The synthetic data sets consist of data generated from polynomial of various degrees and sine functions of various frequencies and phases. The first experiment simply reports on the distributions of values of the various meta-features with respect to the different types of synthetic data sets, and demonstrates that in noiseless scenarios the meta-feature values vary nicely according to the complexity of the tasks. The second experiment is a straightforward application of

metalearning for algorithm selection in the context of regression, where the data sets are characterized by the proposed twelve meta-features. Instead of a regression algorithm the target is the label of the underlying function used to generate the synthetic data. A number of meta-classifiers are evaluated on the data using 10-fold cross-validation, and the results show that they all do very well at mapping noise-free data sets to their correct function types. The third major experiment focuses on a single regression algorithm, namely support vector regression (SVR) with a RBF kernel, and uses metalearning to predict the best parameter setting for a data set. In this experiment, the authors use 39 real-world data sets characterized by the twelve proposed meta-features and labeled by their optimal SVR parameter setting (i.e., kernel coefficient gamma, penalty C, and margin of tolerance epsilon) obtained through an extensive empirical analysis of 3,192 possible parameter configurations. Using a simple 1-nearest neighbor metalearner, evaluated with leave-one-out cross-validation, they show that the proposed meta-features lead to a more accurate model than previously proposed meta-features, and that the model predictions are not statistically different from the optimal ones (Wilcoxon test, 95%) for 30 of the 39 tested data sets. The final major experiment applies metalearning to the prediction of algorithm performance, i.e., an instance of meta-regression. The authors select fourteen well-known regression algorithms to create fourteen meta-datasets, each consisting of the 39 real-world data sets characterized by the twelve meta-features and labeled with algorithm performance (measured by NMSE). Using SVR as a meta-regressor, their results show improved NMSE over baseline for all fourteen regression algorithms.

In their paper *Empirical Hardness of Finding Optimal Bayesian Network Structures: Algorithm Selection and Runtime Prediction* Malone, Kangas, Järvisalo, Koivisto and Myllymäki apply metalearning to the problem of Bayesian network structure learning (BNSL) by the three best performing families of solvers, namely A*, integer linear programming (ILP), and constraint programming (CP). As in the work of Lorena et al. for regression, the authors first propose a set of relevant meta-features intended to characterize significant differences among BNSL problem instances. They start with two well-known basic meta-features, namely the number of variables and the mean number of candidate parent sets per variable, which they extend in three different ways. First, they add six meta-features that compute statistical information about the numbers and sizes of candidate parent sets. Second, they consider two relaxed versions of the original BNSL problem, and extract fifteen pieces of structure information from each of the corresponding dependency graphs as meta-features, such as *number of root nodes*, *average node in-degree*, *number of non-trivial strongly connected components*, etc. Finally, they use the *probing method*, where a solver is run for a short, fixed amount of time (here, 5 seconds), and again structure information is extracted from the resulting dependency graph; probing is performed with a greedy approach, A*, ILP and CP, to produce 12 meta-features each for a total of 48 additional meta-features.

The authors then go on to use metalearning to assess the value of the proposed meta-features in algorithm selection and in performance prediction. They draw from 39 real data sets, 19 sampled data sets, and 477 synthetic data sets to create a total of 1179 BNSL problem instances by varying parameters and removing simplistic instances. Each instance is run against four ILP solvers, three A* solvers, and one CP solver, and labelled by the run time of each solver, making it possible to design an algorithm selection task by considering the algorithm with the smallest run time as the winner, as well as a performance prediction task for each of the solvers. For the metalearner, the authors use random forest together with a single pre-processing step, optimized by AUTO- SKLEARN (Feurer et al. 2015). Nested 10-fold cross-validation is used for evaluation, where the inner loop optimizes hyper-parameters and the outer loop computes performance measures. Following a rigorous set of experiments,

the authors show that while the two basic meta-features are sufficient to build an algorithm selection model that accurately predicts which algorithm will perform best on a given data set, information is lacking to build models capable of predicting the solver's run times. In this latter case, the more sophisticated meta-features lead to significant improvements. Not surprisingly, the most helpful meta-features tend to vary from solver to solver, validating both the use of a wider range of task- and domain-specific meta-features, and the value of metalearning in improving our understanding and use of BNSL algorithms.

Similarly, Olier, Sadawi, Bickerton, Vanschoren, Grosan, Soldatova and King, in their paper *Meta-QSAR: A Large-scale Application of Meta-learning to Drug Design and Discovery* present another domain-specific application of metalearning. They combine their interest and expertise in drug design and discovery with the targeted use of metalearning to select the best performing activity prediction algorithm given a specific instance of quantitative structure activity relationships (QSARs). Although the domain is the same for all QSAR problems, individual instances differ in significant ways suggesting that different machine learning methods should be used on different instances, thus providing motivation for a metalearning approach.

In their extensive set of experiments, the authors consider 2764 proteins from the ChEMBL database, each with three possible representations: one consisting of the complete available set of 1447 molecular descriptors; one consisting of a subset of 43 basic of these descriptors; and finally one known as the fingerprint representation. Hence, there is a total of 8292 data sets. As the set of algorithms to choose from, the authors consider 18 regression algorithms. Their results clearly show that different algorithms and different protein representations lead to different performances. For each protein, meta-features a set of 2394 meta-features are extracted. To build an algorithm selection model, the authors consider all 2764 proteins, characterized by all of the meta-features, and labeled by the best QSAR strategy, consisting of one of the 18 regression algorithms together with one of the three protein representation. The metalearner is the random forest algorithm with 500 trees. The results, which include choosing one in 52, as well as choosing one in the top 16, 11, 6, 3, and 2, demonstrate that metalearning for algorithm selection in the context of QSAR problems predicts regression algorithm/protein representation pairs whose performance is above baseline. All data sets, source code, and experiments have been made available within OpenML, ensuring reproducibility and providing a great source of data for future experiments.

3 Conclusion

Our goal with this Special Issue has been to report on some of the progress made in metalearning over the last decade. While much has been done, much remains to be done still, and there are a number of interesting avenues of research, including the following.

– *Pipeline/workflow design* Most of the work so far has remained rather focused on predicting one (or a ranking of) applicable algorithm for a given dataset. It is well-known that most real-world data mining applications often rely on data preprocessing operations, including data transformation (e.g., missing data imputation), feature selection, and sampling, together with classification and/or regression algorithms to build a final model. Hence, we must turn our attention to the design of complete pipelines of algorithms to solve problems. Some systems, such as Auto-WEKA (Kotthoff et al. 2017), AUTO- SKLEARN (Feurer et al. 2015), and TPOT (Olson and Moore 2016), go some way towards this goal, as they are capable of returning certain types of pipelines. Several

relevant workshops have been organized at various ML conferences, including ICML 2017, ECML/PKDD 2017, and ICDM 2017 (e.g., see Brazdil et al. (2017)). The continued upsurge of activity in *Automatic Machine Learning*, including DARPA's recently launched program on Data-driven Discovery of Models,[1] is likely to fuel valuable research in this area for many years to come.

– *Complex data handling* Most metalearning systems rely on the ability to extract meta-features from datasets, so that the metalearner induces a mapping from meta-feature space to performance space. To date, datasets have been assumed to be in tabular form, with instances represented by vectors of values over a set of attributes. With increased interest in images, audio, video, and other data formats, we need to revisit the problem of focussing on the appropriate part of the data space and extracting meta-features from these data sources.
– *Novel approaches* Although much of metalearning has consisted of a kind of bootstrapping exercise where base-level learning techniques are now used at the meta-level, it may be beneficial to consider approaches from related fields, such as matrix factorization in recommender systems or multi-armed bandit systems, to address the metalevel learning problem.

We look forward to what the future may bring in these and other areas. In the meantime, we wish to express our sincere thanks to the authors who contributed high-quality work to this Special Issue, to the reviewers who worked tirelessly to provide constructive feedback to authors, to Machine Learning's Editor-in-Chief for allowing us to produce this Special Issue and for offering valuable comments throughout, and to the editorial and publishing staff at Springer for bringing this Special Issue to production.

References

Bardenet, R., Brendel, M., Kégl, B., & Sebag, M. (2013). Collaborative hyperparameter tuning. In *Proceedings of the thirtieth international conference on machine learning*, pp. 199–2017.

Bennett, J., & Lanning, S. (2007). The netflix prize. In *Proceedings of KDD Cup and Workshop*.

Bernstein, A., & Provost, F. (2001). An intelligent assistant for the knowledge discovery process. In *Proceedings of the IJCAI-01 workshop on wrappers for performance enhancement in KDD*.

Bernstein, A., Provost, F., & Hill, S. (2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 503–518.

Brazdil, P., Giraud-Carrier, C., Soares, C., & Vilalta, R. (2009). *Metalearning: Applications to Data Mining*. Berlin: Springer.

Brazdil, P., Vanschoren, J., Hutter, F., & Hoos, H. (2017). AutoML. In *Proceedings of the international workshop on automatic selection, configuration and composition of machine learning algorithms* (Vol. 1998). CEUR.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

e-LICO. (2012). An e-laboratory for interdisciplinary collaborative research in data mining and data-intensive science. http://www.e-lico.eu/.

Eggensperger, K., Hutter, F., Hoos, H., & Leyton-Brown, K. (2015). Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the twenty-ninth AAAI conference on artificial intelligence*, pp. 1114–1120.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28, pp. 2962–2970). Red Hook: Curran Associates, Inc.

Giraud-Carrier, C. (2005). The data mining advisor: Meta-learning at the service of practitioners. In *Proceedings of the fourth international conference on machine learning and applications*, pp. 113–119.

[1] See https://www.darpa.mil/program/data-driven-discovery-of-models.

Giraud-Carrier, C., Brazdil, P., & Vilalta, R. (2004). Introduction to the special issue on meta-learning. *Machine Learning*, *54*(3), 187–193.

Hofmann, M., & Klinkenberg, R. (2013). *RapidMiner: Data mining use cases and business analytics applications*. London: Chapman & Hall/CRC.

Hutter, F., Hoos, H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: An automatic configuration framework. *Journal of Artificial Intelligence Research*, *36*, 267–306.

Hutter, F., Hoos, H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Proceedings of the fifth international conference on learning and intelligent optimization (LION'11)*, pp. 507–523.

Hutter, F., Xu, L., Hoos, H., & Leyton-Brown, K. (2014). Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, *206*, 79–111.

Kietz, J. U., Serban, F., Bernstein, A., & Fischer, S. (2012). Designing KDD-workflows via HTN-planning for intelligent discovery assistance. In *Proceedings of the ECAI-12 workshop on planning to learn*.

Kotthoff, L., Thornton, C., Hoos, H., Hutter, F., & Leyton-Brown, K. (2017). Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, *18*(1), 826–830.

Leite, R., Brazdil, P., & Vanschoren, J. (2012). Selecting classification algorithms with active testing. In *Proceedings of the eighth international conference on machine learning and data mining in pattern recognition (LNCS 7376)*, pp. 117–131.

Lemke, C., Budka, M., & Gabrys, B. (2015). Metalearning: A survey of trends and technologies. *Artificial Intelligence Review*, *44*(1), 117–130.

Leyton-Brown, K., Nudelman, E., & Shoham, Y. (2009). Empirical hardness models: Methodology and a case study on combinatorial auctions. *Journal of the ACM*, *56*(4), 22:1–22:52.

Lópes-Ibáñez, M., Dubois-Lacoste, J., Stüzle, T., & Birattari, M. (2011). *The irace package, iterated race for automatic algorithm configuration*. Tech. rep.: IRIDIA, Université Libre de Bruxelles.

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

Olson, R., & Moore, J. (2016). TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Proceedings of the ICML 2016 AutoML Workshop (JMLR Workshop and Conference Proceedings 64)*, pp. 66–74.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge: The MIT Press.

Rice, J. (1976). The algorithm selection problem. *Advances in Computers*, *15*, 65–118.

Smith-Miles, K. (2008). Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, *41*(1), 6:1–6:25.

Thornton, C., Hutter, F., Hoos, 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*, pp. 847–855.

Töscher, A., & Jahrer, M. (2009). *The bigChaos solution to the netflix grand prize*. Tech. rep., Commendo Research & Consulting.

Witten, I., Eibe, F., & Hall, M. (2011). *Data mining: Practical machine learning tools and techniques* (3rd ed.). San Francisco, CA: Morgan Kaufmann.

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