Meta-learning as a scheme-based search with complexity control

Abstract. Recent years revealed growing need for efficient meta-learning. For much longer time it is known that there is no single adaptive algorithm, eligible to provide satisfactory (i.e. close to optimal) solutions for every kind of problem, however computing power facilitates practical applications of more and more sophisticated learning strategies and more and more thorough search in the space of models. Because testing all possible models is not (and never will be) feasible, we need intelligent tools to combine human expert knowledge, the knowledge extracted by means of computational intelligence and different search strategies to disclose the nature of a problem and provide attractive models. We present some techniques, we have successfully used in our meta-learning approaches, describe the crucial ideas of our general architecture for meta-learning, and show some applications.

1 Introduction

The need for and feasibility of successful meta-learning are growing. For know, searching for as accurate models as possible was usually within the domain of humans, but we are never as precise, thorough and systematic as computers can be, so there is no reason why machines could not perform better than us in such tasks.

The progress of computational capabilities, already now, facilitates successful artificial approaches to meta-learning. Obviously, searching for optimal models is NP-hard, so the progress in computing hardware does not facilitate a complete search through the space of possible models, so we need (and will always need) intelligent systems for this purpose.

There may be many different views of meta-learning and many different algorithms putting stress on different aspects of the field. Till now, the term "meta-learning" was used in a number of meanings. For example, some articles use this name when talking about building rankings of methods on the basis of their predicted eligibility for solving particular tasks [1, 2]. The rankings were constructed according to some similarity of the problem being solved to other known problems. The similarity was measured as a distance in some space of the datasets, where each dataset was described by a number of values corresponding to the types of values contained within the data, some other statistical coefficients [3], results obtained with some simple learning algorithms [4, 5] (this technique was given a name of *landmarking*),some features of decision trees built for the data [6], etc. In some other approaches instead of measuring similarity between datasets, decision trees were used to decide which algorithm should perform better [7]

Another meaning was given to meta-learning in numerous papers devoted to ensemble models, since building complex structured model can also be seen as a meta-level

task. The complex models include different kinds of committees (including ones considering member-model competence when making decisions [8, 9]), stacking models etc. [10–12]

All the above-mentioned methods build complex models or prepare models rankings to act as method selection advisers, which is not satisfactory for us. We understand meta-learning as automation of the process of finding most accurate models for given task (which eventually should replace human interaction). It is not enough to compare the datasets to provide a reliable advise on which method can be more useful to solve the problem, especially because different methods usually require different data preprocessing to obtain optimal results and data transformations may move datasets into completely different point in the space of datasets. Thus we emphasize the need for an intelligent search for the (sub)optimal solution which can not be based only on some statistical information about the form of the data, but must integrate meta-level information of different kinds and sources including human expert knowledge and the knowledge gained by means of computational intelligence (CI).

2 Meta-learning

From our point of view, meta-learning is the process of learning how to learn, to obtain as good solution to defined problem as possible. It corresponds exactly to what humans are trying to do when mining given data: in order to find very good models we try many different methods, their combinations etc., observe and analyze the subsequent results and use our general knowledge about building CI models, to control the search in such a way that only the sensible combinations are tested and those maximizing our suspicions of attractiveness. So meta-learning is a very complex process incorporating the search for (sub)optimal solutions, using meta-knowledge to conduct the search and (simultaneously) gaining new meta-knowledge.

The meta-knowledge which indicates attractive search directions may have different sources:

- human experts can formulate rules to restrict the search to what they regard as sensible, and what they know about the specificity and requirements of particular methods,
- statistics calculated for different data analyzed in the past, about which methods were successful in which circumstances, correlations between the different methods results (usefulness), usability of different data transformation techniques for different learning methods and different data etc. can also help us restrict the search to the most promising areas,
- performing tests of different methods is always a source of valuable information about the characteristics of the data being currently analyzed, which sometimes may be in contradiction with the statistics mentioned in the previous item—such context-dependent information must be of higher priority and must be continuously updated.

It is important to provide a uniform representation of the meta-knowledge, regardless its source, so that for example the expert knowledge may be extended, adjusted

according to performed tests, etc. It must be capable of expressing rules of miscellaneous types, concerning different levels of abstraction.

Exact representation of the meta-knowledge satisfying these conditions is itself a subject for a broad discussion, so we do not go into more details here, where we pay more attention to the algorithmic part of our meta-learning approach.

2.1 Goals of meta-learning

Meta-learning aims at finding optimal (regarding some criteria) models. It is not restricted to maximizing classification accuracy, minimizing regression error or any limited set of tasks. It is important to have the possibility of providing the criterion without the need of any changes within the engine of the data mining system. Fortunately it is not very difficult in search-based approaches - although particular methods optimize different measures, they are usually quite compatible (models optimal from different points of view, are close to each other in the space of models), so that the meta-pursuit can be successful when searching around the optimal and close to optimal models. Out system provides such openness and has already been used to obtain different goals. Here we discuss only classification tasks, but still some maximized (classical) accuracy and some balanced accuracy.

In classification problems, the goal is usually to maximize the classification accuracy (or balanced accuracy), but its estimation can never be devoid of error. Thus we are often interested in high stability of our validation results (i.e. the minimization of their variance). To achieve this, we prefer maximization of

$$\mu - \alpha\sigma, \tag{1}$$

where μ is an estimation of the expected value of accuracy, σ is the standard deviation of the accuracy within the validation tests, and α is a parameter (usually equal to 1 in our approaches). Such measure is sound with the ideas of testing statistical hypotheses and its optimization may be seen as the maximization of the threshold, below which we will not fall with given probability (equal to 0.5 in the case of $\alpha = 0$, and greater for larger values of α).

Another important feature of adaptive methods is time complexity. Obviously the methods which are fast should be tested before more time consuming ones. Moreover simple models are preferable to complex ones, when their quality does not differ significantly. In this context, a comfortable measure of model complexity is the *Levin's complexity*, which is defined as the sum of model (description) length and logarithm of the time of its adaptive method execution:

$$L + \log(T). \tag{2}$$

Control of this complexity allows us to stop long-lasting processes and those, that will certainly end up with unacceptably complex models, without waiting till the end of their adaptive processes (thus saving computation time).

When striving to meta-learning goals we must not forget about justification of the validation methods we use. Incorrect validation usually leads to overoptimistic results, and provides no real confirmation of generalization achievements of the model. Thus, it

is very important to validate not just the final model (e.g. classifier or approximator), but the whole sequence of operations performed from raw data to the decider. No supervised part of the sequence is allowed to be put outside of the validation process and treated as an element of the data preprocessing stage. The split of data analysis processes into data preprocessing and final learning is very common in the literature, but it is often not justifiable.

2.2 Meta-level exploration

In our approach, the pursuit to the optimum model is a heuristic search, which can be seen as divided into a number of stages, although only at the beginning the borders between stages are sharp—further stages are rather fuzzy, because in fact, the whole search is a single loop in which we test different models starting with the fastest and simplest ones and proceeding to more and more complex and time-consuming methods. Such order is natural because we do not want to test complicated models when simple models provide satisfactory solutions or run time-consuming processes when fast ones perfectly do the job.

Thus, to control model complexity we introduced the mechanisms of *abstraction levels* and *computational complexity level*. The abstraction levels allow to organize the search by imposing constraints on each stage of the search, for example:

- at the beginning only simple classifiers will be used,
- then classifiers after simple data transformations,
- then sequences consisting of a normalization followed by feature selection and then by a classifier etc.

The meta learner can define model structures which are to be exploited at subsequent *levels of abstraction*. The structures are defined by means of meta-schemes described in section 2.4. This facilitates penetrating different areas of the model space with different density and avoiding insensible combinations (like multiple standardization). We always try to define the scenarios we regard as most sensible, however we also accept some less convincing constructions to leave some chance for a surprising invention.

In parallel, there are another constraints on Levin complexity of methods to be tried. This is to avoid using time-consuming or building very complex models too early. As soon as the first model is built (regardless its optimality) we may put restrictions on learning new models (both running time and model complexity). The methods, for which we can predict (or at least show lower bounds of) model complexity and running time, may be put into proper queue to wait for their time. The methods, for which the prediction is not possible, are run and properly controlled. Thanks to using Levin complexity, we may calculate the thresholds of acceptable values of method run time and model complexity. For estimating model complexity we use a criterion resembling Minimum Description Length, which reflects the numbers and types of values describing the model. Passing to next stage of the meta-search is in fact including additional level of abstraction in the set of accepted meta-schemes.

Although definition of the abstraction levels is up to meta-learning methods, there are some tests which in our opinion should definitely be performed (and their order also seems natural). These are:

- tests of simple methods of different types (e.g. classifiers of different nature,
- combinations of different data transformation algorithms (normalizations, feature selecion, vector selection etc.) and methods specialized in solving problems of the type (classifiers, approximators etc.),
- multiple data transformations, both sequential (like standardization followed by feature selection) and parallel (like committees of feature selection models),
- ensemble methods including committees respecting members' competence.

Within each search stage, some searches for optimum values of different parameters of the methods are tried and the results saved for further analysis. It must be emphasized here, that introducing a data transformation can significantly change the task, so after such operation, additional search for optimum parameter values of final decision methods is necessary.

Search for optimal values of parameters may be performed more reasonably, thanks to meta-parameters descriptions, we have allowed for in our system. Such descriptions usually include the information about the scope of sensible values and the type of the parameter changes (linear, exponential etc.). Moreover, a meta-learner can be provided with information about how to efficiently perform the search (e.g. committee decision modules should be tried just after the member-models have been created, to reduce computation time). To avoid repetitions in running adaptive processes we have created a cache system, which when asked again for the same model does not build it but shares the one created earlier (in future we plan a cache system which could save the data to a disk and load from it when necessary, which will allow to take advantage of the cache also between different instances of the system, also running on different machines).

The major difference between our approach and the ones described so far in the literature is that the crucial part of our meta-learning is the heuristic search continuously analyzing the feedback of running different tests. It gives much more possibilities than simple ranking construction. It turned out advantageous in our different data mining competition efforts.

In fact, the main loop of our meta-search may be seen as an infinite procedure, which tries more and more complicated models for given data. After the first model is built, at each time of the search we can get the information about currently best model. Thus, there is no single stop point of our meta-search. We may stop after some pre-defined time, on user request, after obtaining appropriately small error, after no improvement within a time period etc.

We start with some meta-knowledge, which continuously changes according to what we have learnt. First meta-models use only some general meta-knowledge provided by experts, but then the meta-knowledge may be appropriately adjusted and exchanged between different meta-learning methods. It is very important to differentiate between the general knowledge (averaged for all the data sets) and the knowledge in the context of particular data, because they should have different influence on the meta-search.

2.3 Meta-space

By now, our meta-learning approaches have been solving classification problems. The first stage has always concerned testing basic classification algorithm. The set of examined methods included:

- k Nearest Neighbors,
- Naive Bayesian Classifier,
- Support Vectors Machines (with Gaussian or linear kernel),
- SSV decision tree,
- Feature Space Mapping neural network.

The list is consistent with our assumption that methods of different nature should be tested—it contains statistical methods, neural networks, decision trees and SVM (which, with linear kernel, is a linear discriminant method.

For each model we search for optimum values of parameters. For example: in kNN model we can search for optimal number of neighbors to analyze, in SVM we can determine best values of Gaussian dispersion and the C parameter, in SSV decision tree we may try different settings of discrete parameters defining the way the tree is constructed and/or pruned.

In our OCR competition effort [13] the first stage gave unexpected results: simple 5NN classifier significantly outperformed all the other classifiers, though this was not the final result—just the result of the first stage of the pursuit.

As the second stage we usually define testing different data transformations. It is one of the most important parts of meta-learning, because there is plenty of transformations that must be performed, and they may be combined in many different ways, so that it is easy to be entrapped by combinatorial explosion. It is important not only to avoid senseless combinations but even to drive the search into most attractive directions.

Some of the basic transformations we use are:

- different kinds of normalizations (rescaling to $[0, 1]$ or $[-1, 1]$ interval, standardization, the same methods with respect to the data without outliers etc.)
- feature selection methods (based on correlation coefficient, F-rank, SSV criterion and some information theory measures),
- numerous vector selection methods [13],
- discretization and its reverse (converting continuous features to symbolic),
- Principal Components Analysis (PCA), usually accompanied by selection of several PCs.

Even in the case of standardization we have many possibilities. Apart from eliminating outliers, we may consider *per-feature* standardization (the classical approach, where each feature is standardized independently) and *per-dataset* standardization, which results in mean 0 and variance 1 within the values of all features together—it makes more sense for instance in the case of text analysis data, where word occurrences are counted and thus it is advantageous to keep the proportion between counts for different words. The per-dataset standardization was crucial in our participation in the NIPS 2003 contest, where one of the datasets was devoted to text classification (as it turned out after contest adjudication).

Another version of normalization we used in our OCR data analysis. Our models erroneously classified some vectors representing numbers with very easy to understand shape, but with lower pixel intensity than in the case of other numbers. We called the transformation *darkening*, but in fact it can be seen as a *per-vector* normalization, which may be useful also in other tasks (e.g. again in text analysis with word occurrence counts

as features, it is one of possible methods to eliminate the influence of text length on classification).

Our experience with using PCA is quite diverse. In the case of the OCR efforts it was completely useless (as all the feature selection attempts). Actually it is not a surprise, because in the 8x8 pixel images there is no unimportant information—only the corner pixels are less important, but still they are not a noise in the data. On the other hand, PCA was the key to our best model for the Dorothea dataset of the NIPS 2003 competition.

Some meta-learning approaches are based on the idea that datasets which are similar with respect to some statistical information will be best solved by similar methods. In the context of data transformations it is not justifiable, because a dataset before and after a transformation may be completely different. Moreover, very often different classification methods require different data preprocessing to obtain the highest possible accuracy, so their runs on the same form of dataset may be incomparable. And inversely: it is easy to create two datasets with the same types of features, such that one will be perfectly classified by a decision tree and poorly by kNN, and the other with the opposite result.

Since data transformations can not be tested separately, they must go through tests as a whole with the final models (in our cases: classifiers). Thus the basic scenario with a data transformation model looks like the one given in figure 1, described in more detail in the following section. Our system architecture is organized in such a way, that allows meta-search to operate on any parameter of any method within a defined scenario. This applies also to discrete-values parameters like the one in SVM, which determines the way of multi-class classification. So we perform a complex, multidimensional search for optimal values of method parameters. Unfortunately, because of this complexity, it is very difficult to present the results of such a search as a spectacular 2D or even 3D plot.

2.4 Meta-schemes

As mentioned in previous sections, meta-schemes are our fundamental tools for efficient meta-search. They are directed acyclic graphs (DAG) of boxes representing scheme placeholders and particular models, interconnected according to the input–output connections. The scheme placeholders define places in the DAG, where meta-learning algorithms, in their adaptive processes, try to put different hierarchies of models (they need not to be just single models, but also some complex structures which we call *schemes*). We have introduced the meta-schemes to provide an easy way to put restrictions on the search. Such constraints facilitate significant reduction of the space searched through, so that we can eliminate insensible schemes of models, and at the same time define the directions of the search. Thence, the key point is to design such meta-schemes, that the space is significantly reduced, but it still contains the interesting models.

The meta-search algorithm uses the meta-schemes according to their roles. We may define meta-schemes to play the role of classification, data transformation etc. We can nest the meta-schemes, i.e. fill the placeholders in one meta-scheme with an instantiation of another meta-scheme, so there are no limits in complex schemes construction. The possibility of nesting is especially precious, for example when searching for most

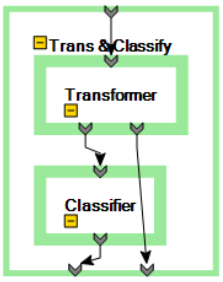

Fig. 1. Basic scenario for data transformation and classification

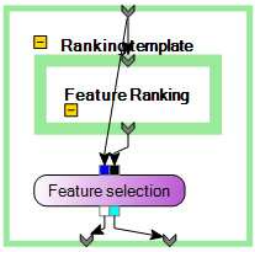

Fig. 2. Typical feature selection transformation

useful data transformation, which may have different length (unknown at the start of the search).

An example of simple meta-scheme is presented in figure 1, where we have two placeholders to be filled while learning: one for data transformation and another one for a classification model. The whole meta-scheme, has one input (where training dataset is expected) and two outputs: one for classification and the other for data preparation before classification. Thus after proper substitutions, it may be used everywhere a classifier is needed.

An example of a complex data transformation is presented in figure 2. The transformation performs feature selection for a data table. It is split into two parts: first a ranking of features is created and then proper selection performed. It is again a meta-scheme, because it contains a placeholder for a feature ranking model. The feature selection part is a precise model here, because given ranking the selection is always performed in the same way. The meta scheme of figure 2 can be put in the placeholder for data transformation in figure 1 (the idea of nesting schemes, mentioned above).

A more complex, but also more practical meta-scheme is presented in figure 3. It is more eligible for meta-search, because it allows to validate the models substituted for the placeholders. The meta-search can perform the substitutions, run the whole scenario by a single command, and check the validation results afterward. The left side figure concerns the configuration time of the “Meta model”. The “Validator” model will validate model configurations composed of a data transformer and a classifier (substituted by the “Meta-model” in runtime). The right side figure presents an iteration of the runtime. The “Meta model” substituted F-score feature selection for the “Transformer” and SVM for “Classifier” and executed the “Validator” which used train-test data distributor to validate the configuration prepared by the “Meta-model” (the details of the validation model are beyond the scope of this article, they can be found in [13]).

2.5 Advanced techniques of meta-learning

Meta-schemes provide very powerful means for meta-search restriction and direction. The task of meta-learning method designer is to define such set of meta-schemes and items to fill placeholders, that allows to avoid spending time on testing insensible model

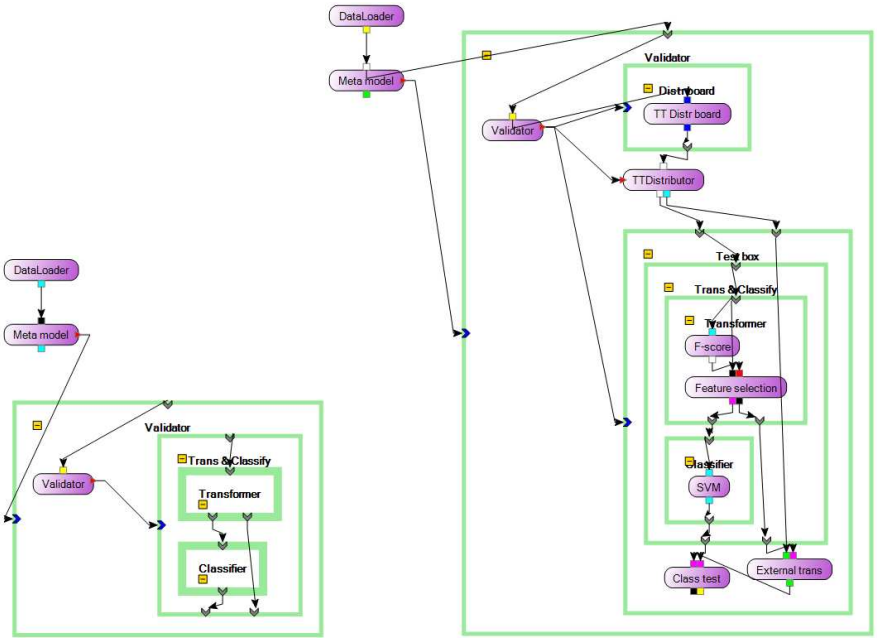

Fig. 3. A model validation meta-scheme (configuration—left side diagram and runtime—right side diagram)

structures and to point out the most promising structures. The task of meta-learning algorithms that use meta-schemes is not only to search for the most accurate solutions, but also to learn from the search experience. Such learning includes:

- Finding the correlations of occurring different items in most accurate results. It will enable to learn which data transformations are most useful for given classification model, to define some areas of model structures successful in similar environment, so that finding out, that a model structure is successful, we can check some other structures which worked in similar circumstances, etc.
- Finding new successful complex structures and converting them into meta-schemes (which we call *meta abstraction*) by replacing proper substructures with placeholders.
- Extracting meta-rules, describing the advantageous directions of the search.
- Depositing the knowledge they gain in a reusable meta-knowledge repository. The possibility to exchange meta-learning heuristics is very precious, because saves much time—otherwise each meta-learning method would have to learn itself, what other meta-learners have already learnt.

3 Summary

We have presented basic ideas and some examples of our meta-learning approaches based on intelligent search. The idea of meta-schemes is very precious tool in defin-

ing heuristics for the search process. The meta search starting with the simplest models and proceeding to more and more complex ones by means of the abstraction levels and Levin complexity control turned out to be successful and promising, since its gates to further development are open, and new directions of advanced meta-learning are evident. We believe, that quite soon such techniques will be more successful, than human driven search for (sub)optimal solutions to many problems addressed to computational intelligence.

References

1. Brazdil, P., Soares, C.: Ranking classification algorithms based on relevant performance information. In Keller, J., Giraud-Carrier, C., eds.: Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination. (2000)
2. Berrer, H., Paterson, I., Keller, J.: Evaluation of machine-learning algorithm ranking advisors. In Brazdil, P., Jorge, A., eds.: Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions, Lyon, France, Springer-Verlag (2000)
3. Brazdil, P., Soares, C., Costa, J.P.D.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning **50**(3) (2003) 251–277
4. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann (June 2000) 743–750
5. Frnkranz, J., Petrak, J.: An evaluation of landmarking variants. In Giraud-Carrier, C., Lavra, N., Moyle, S., Kavsek, B., eds.: Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning. (2001)
6. Y.H., Peng, Falch, P., Soares, C., Brazdil, P.: Improved dataset characterisation for meta-learning. In: The 5th International Conference on Discovery Science, Luebeck, Germany, Springer-Verlag (January 2002) 141–152
7. Kalousis, A., Hilario, M.: Model selection via meta-learning: a comparative study. (2000) 406–413
8. Duch, W., Itert, L.: Committees of undemocratic competent models. In: Proceedings of the Joint Int. Conf. on Artificial Neu-ral Networks (ICANN) and Int. Conf. on Neural Information Processing (ICONIP), Istanbul, Turkey (2003) 33–36
9. Jankowski, N., Grąbczewski, K.: Heterogenous committees with competence analysis. In Nedjah, N., Mourelle, L., Vellasco, M., Abraham, A., Köppen, M., eds.: Fifth International conference on Hybrid Intelligent Systems, Brasil, Rio de Janeiro, IEEE, Computer Society (November 2005) 417–422
10. Stolfo, S., Prodromidis, A., Tselepis, S., Lee, W., Fan, D., Chan, P.: JAM: Java agents for meta-learning over distributed databases. In: Third Intl. Conf. on Knowledge Discovery and Data Mining. (1997) 74–81
11. Prodromidis, A., Chan, P.: Meta-learning in distributed data mining systems: Issues and approaches. In Kargupta, H., Chan, P., eds.: Book on Advances of Distributed Data Mining. AAAI press (2000)
12. Todorovski, L., Dzeroski, S.: Combining classifiers with meta decision trees. Machine Learning Journal **50**(3) 223
13. : One or more papers hidden beacause of the submission policy of avoiding disclosing the authors before reviews.