

Meta-learning as scheme-based search with complexity control

Krzysztof Grąbczewski and Norbert Jankowski

Department of Informatics
Nicolaus Copernicus University
Toruń, Poland
<http://www.is.umk.pl/>
{kgrabcze|norbert}@is.umk.pl

Abstract. Recent years have revealed growing need for efficient meta-learning. For much longer time it has been 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 candidate models. Because testing all possible models is not (and will never 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 examples.

1 Introduction

The need for successful meta-learning is growing. Fortunately, at the same time, advanced learning is getting more and more feasible. For now, searching for as accurate models as possible, has been staying within the domain of human intelligence, but we are never as precise, thorough and systematic as computers can be, so there is no reason why machines could not perform such tasks better than us.

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” has been used in several 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 the space of datasets, where each dataset was described by a number of

quantities corresponding to the types of values contained within the data, some 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 articles devoted to ensemble methods, since building complex structured models can also be seen as a meta-level task. The complex models include different kinds of committees (also the 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 model 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 Goals of meta-learning

From our point of view, meta-learning is the process of learning how to learn, to obtain as good solution to given 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 about being attractive. 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.

Meta-learning aims at finding models which optimize some criteria. 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. Our system provides such openness and has already been used to reach different goals. Here we discuss only classification tasks, but two measures of model attractiveness: classical accuracy and 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

$$\mu - \alpha\sigma, \quad (1)$$

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).

3 The meta-space

By now, our meta-learning approaches have been applied only to classification problems. In such cases, the solution space contains different classification models (simple or complex) built by machines trained on the data obtained with a number of different transformations.

The set of classification algorithms, we usually examine includes:

- 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 need to search through the space of possible values of the parameters. For example: in kNN model we can search for optimal number of neighbors to analyze, in SVM we can determine the 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.

An interesting (because quite unexpected) result, we obtained in the first stage (testing just classification algorithms) of our OCR competition effort [13]: simple 5NN classifier significantly outperformed all the other classifiers, however it was not the final result—after some data transformations we obtained significantly better results (one of them won the contest) and other classifiers performed better than 5NN.

Our data mining experience (like the one with the OCR competition) shows, that proper data transformation is the crucial point of a successful model. Even when dealing with classification problems, it is almost never possible to get optimum results with a classification algorithm alone. Moreover, finding appropriate data transformation is often much more difficult than finding optimal configuration of a classification algorithm, because there are plenty of transformations that may be performed, and they may be combined in many different ways, so that it is easy to be entrapped in a combinatorial explosion. It is important not only to avoid senseless combinations, but also to drive the search into most attractive directions.

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 measures derived from information theory and also some ensemble methods),
- numerous vector selection methods [14],
- 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 Feature Selection Challenge, where one of the datasets was devoted to text classification (as it turned out after contest adjudication).

Another version of normalization was helpful 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.

The set of reasonable data transformations and classification algorithms is not small, even for a single unit exhaustive search in the space of parameters is too expensive and when different methods are combined, the solution space gets so huge that the search must be supported with some intelligence. The following sections address these problems.

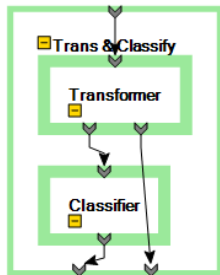


Fig. 1. Basic scenario for data transformation and classification

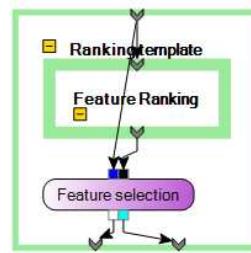


Fig. 2. Typical feature selection transformation

4 Meta-schemes

One of the fundamental ideas of our meta-learning approach is driving the search by means of *meta-schemes*. 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 learning models (they need not to be just single methods, but also some complex hierarchies which we call *schemes*). Restricting the search to model structures compatible with the ones given by the meta-schemes is a way to take advantage of experts meta-knowledge to drive the search process. At the same time, such constraints facilitate significant reduction of time consumed by the search. Thence, the key point is to design such meta-schemes, that the space is significantly reduced, but it still contains interesting models.

We may define meta-schemes to play the role of classification, data transformation etc. (the role is defined by inputs and outputs). 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 useful data transformations, which may have different length (unknown in the beginning of the search).

An example of a simple meta-scheme is presented in figure 1, where we have two placeholders to be filled during learning: one for a data transformation and another one for a classification machine. 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 a 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).

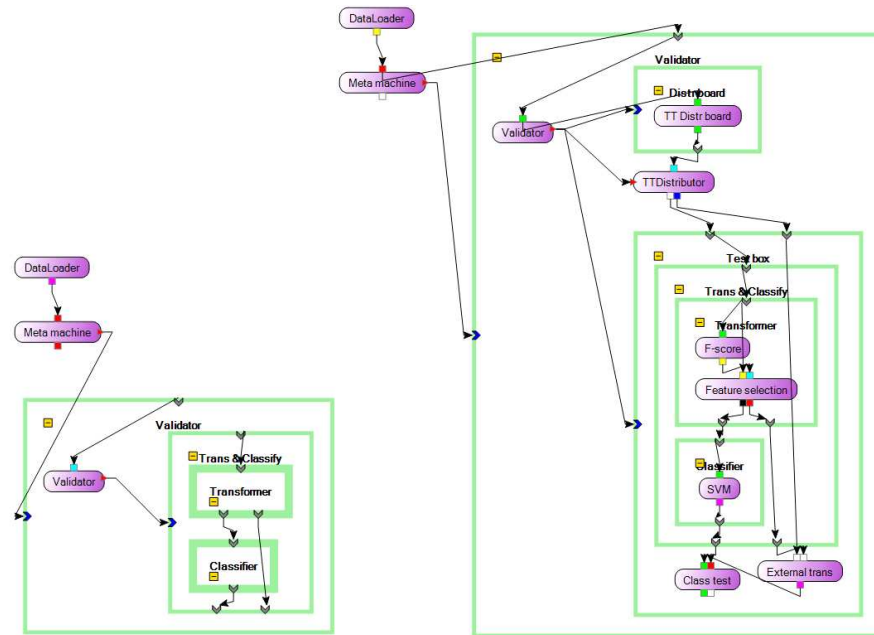


Fig. 3. A meta-scheme for validation (configuration—left side diagram and runtime—right side diagram)

When striving to meta-learning goals, we must not forget about justification of the validation methods we use. Incorrect validation usually leads to overoptimistic (thence useless) results, and provides no real confirmation of generalization abilities of the machine. 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.

The idea of meta-schemes is also very useful in the context of validation. The meta-scheme presented in figure 3 facilitates easy validation of the machines 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 depicts the configuration time of the “Meta machine”. The “Validator” machine is to validate configurations of machines composed of a data transformer and a classifier (substituted by the “Meta machine” in runtime). The right side figure presents an iteration of the runtime. The “Meta machine” 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 machine” (the details of the validation model are beyond the scope of this article, they can be found in [15]).

5 The meta-learning algorithm

Our meta-learning approach is a heuristic search in the pursuit of the optimum model. The search space is restricted by means of meta-scheme based *machine generators* and a *complexity control mechanism* is introduced to reasonably conduct the search.

The search procedure is a single loop in which we validate different machines 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.

The machine generators use constraints defined as meta-schemes to build more and more complex machine architectures (compatible with the meta-schemes) and pass them to the main meta-learning algorithm, when requested.

Machine complexity control is used both to decide the order of validation of candidate machines, and also to avoid long-lasting processes which could block the whole process (sometimes it is not possible to guess the complexity in advance, so a machine which declares low complexity may turn out to be very expensive, and the main meta-learning loop must detect it and brake such a subprocess).

Obviously the methods which are fast should be tested before more time consuming ones and simple models are preferable to complex ones, when their quality does not differ significantly. Thus, the complexity measure must reflect both model structure complexity and machine time complexity. In this context, a comfortable measure is the *Levin complexity*, which is defined as the sum of model (description) length and logarithm of the time of its adaptive method execution (in case of learning machine):

$$L + \log(T). \quad (2)$$

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). In practice we use the Levin complexity augmented by a measure of prediction of the fitness of the resulting model. This extension is also very important, because it lets us ignore some type of machines which usually declare low complexity, but have proven to be unsuccessful, so it is more reasonable to try other, more complex machines but possibly much more suitable. It is also sensible, to try from time to time, some less convincing machine constructions to leave some chance for a surprising invention.

As soon as the first model is built (regardless its optimality) we may put restrictions on new machines (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, are put into the proper place in the queue of candidates, that must be validated. The methods, for which the prediction is not possible, are put to the queue on the basis of a rough complexity prediction, and their execution is properly controlled. Thanks to using Levin-like 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.

Such control of the search process results in testing models (more or less) in the following order:

- simple methods of different types (e.g. classifiers of different nature),
- combinations of different data transformation algorithms (normalizations, feature selection, 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.

Apart from searching for the optimal machines hierarchy, our meta-learning algorithms perform some searches for the optimal values of machines parameters. The two types of search are in fact mixed into a single search process. The results of parameters searches are appropriately stored and then used also in other machine configurations, however it must be emphasized here, that for example adding a data transformation to a machine structure may significantly change the task, so after such operation, additional search for optimum parameter values of final decision methods is necessary, though it can take advantage of the results of previous searches for the same parameters to adjust the density of the search.

Our system architecture includes a unified meta-parameters description system, that allows meta-search to control parameters of any machine available in the system without any knowledge of the internals of the machine. The descriptions usually include the information about the scope of sensible values and the type of the parameter changes (discrete, 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 twice, but shares the one created earlier (in future we plan also a cache system which could save the data to a disk and load from it when necessary—it will allow to take advantage of the cache also between different instances of the system, even running on different machines).

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, if no improvement occurred within a time period etc.

We start with some meta-knowledge, which continuously changes according to what we learn. First meta-machines 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.

6 Advanced techniques of meta-learning

Meta-schemes provide very powerful means for meta-search restriction and direction. The task of meta-learning method designer (a human expert) is to define such set of meta-schemes and items to fill placeholders, that allows to avoid spending time on testing insensible model 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 learning which data transformations are most useful for given classification model, finding some areas of model space with structures successful in similar environment, so that a discovery of a successful model structure, may be followed by testing some other structures which have performed similar in similar circumstances, etc.
- Finding new successful complex structures and converting them into meta-schemes (which we call *meta abstraction*) by replacing proper substructures by 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 experience is very precious, because saves much time—otherwise each meta-learning method would have to learn from scratch instead of taking advantage of what other meta-learners have already gained.

It is important to provide a uniform representation of the meta-knowledge, regardless its source, so that for example the knowledge may be exchanged, 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, etc.

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.

7 Summary

We have presented basic ideas and some examples of our meta-learning approaches based on intelligent search. 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 providing simple rankings of methods and constructing committees of models.

The idea of meta-schemes is very precious tool in defining 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 searches.

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 Neural 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) (2003) 223–249
13. Jankowski, N., Grąbczewski, K.: Handwritten digit recognition — road to contest victory. In: *IEEE Symposium Series on Computational Intelligence (SSCI 2007)*, IEEE (2007) 491–498
14. Jankowski, N., Grochowski, M.: Comparison of instances selection algorithms: I. Algorithms survey. In: *Artificial Intelligence and Soft Computing*, Springer (June 2004) 598–603
15. Grąbczewski, K., Jankowski, N.: Versatile and efficient meta-learning architecture: Knowledge representation and management in computational intelligence. In: *IEEE Symposium Series on Computational Intelligence (SSCI 2007)*, IEEE (2007) 51–58