Meta-learning via Search Combined with Parameter Optimization.

Włodzisław Duch and Karol Grudziński

Department of Informatics, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.
www.phys.uni.torun.pl/kmk

Abstract. Framework for Similarity-Based Methods (SBMs) allows to create many algorithms that differ in important aspects. Although no single learning algorithm may outperform other algorithms on all data an almost optimal algorithm may be found within the SBM framework. To avoid tedious experimentation a meta-learning search procedure in the space of all possible algorithms is used to build new algorithms. Each new algorithm is generated by applying admissible extensions to the existing algorithms and the most promising are retained and extended further. Training is performed using parameter optimization techniques. Preliminary tests of this approach are very encouraging.

1 Introduction.

There is no single learning algorithm that is inherently superior to all other algorithms. This fact is known as the 'no free lunch' theorem [1]. Yet most efforts in the computational intelligence field goes into the improvement of individual methods. For example, in the neural network field model selection efforts are restricted to selection of architectures (number of nodes, each performing the same type of functions) and improvements of the training schemes, while in the decision tree field branching criteria and pruning strategies are discussed. A notable exception in the machine learning field is the multistrategy learning introduced by Michalski [2]. Our own efforts in this direction include the introduction of heterogeneous adaptive systems of decision tree [3] and of different neural networks types [4].

In real world applications a good strategy is to find the best algorithm that works for a given data trying many different approaches. This may not be easy. First, not all algorithms are easily available, for example there is no research or commercial software for some of the best algorithms used in the StatLog project [5]. Second, each program requires usually a different data format. Third, programs have many parameters and it is not easy to master them all. Our "meta-learning" strategy here is to use recently introduced framework for Similarity-Based Methods (SBM) [6] to construct automatically the best model of the given data. A search for the best model in the space of all models that may be generated within SBM framework is performed. Simplest model are created at the beginning and new types of parameters and procedures are added, allowing to explore more complex models. Neural networks increase model complexity by adding the same type of parameters, generating different models within a single method. This is not a good strategy if the bias

of the method (in this case coming from particular type of transfer functions used by the network) does not match the structure of the data. Creating different models of the data using algorithms that have different biases should allow overcome the "no free lunch" theorem and lead to relatively simple models that may be easily understood.

Although the meta-learning approach is quite general and may be used for any association, approximation and unsupervised learning tasks, this paper is focused on methods useful for classification. In the next section the SBM framework is briefly introduced, the third section presents the meta-learning approach used to select the best method, and the fourth section contains our preliminary experiences in analyzing a few datasets. Conclusions and plans for future developments close this paper.

2 A framework for meta-learning

By **an algorithm**, or **a method**, a certain well-defined computational procedure is meant, for example a *k*-NN method or an RBF neural network. **A model** is an instance of a method with specific values of parameters. The SBM framework [6] covers all methods based on computing similarity between the new case and cases in the training set. It includes such well-known methods as the *k*–Nearest Neighbor (*k*-NN) algorithm and it's extensions, originating mainly from machine learning and pattern recognition fields, as well as neural methods such as the popular multilayer perceptron networks (MLP) and networks based on radial–basis functions (RBF).

A function or a procedure to estimate the posterior probability $p(C_i|\mathbf{X};M), i = 1..K$ of assigning vector $\mathbf{X}$ to class C_i, depends on the choice of the model M, that involves various procedures, parameters and optimization methods. Let N be the number of attributes, K be the number of classes, vectors are written in bold face while vector components are in italics. Given a set of objects (cases) $\{\mathbf{O}^p\}$, $p = 1..n$ and their symbolic labels $C(\mathbf{O}^p)$, define useful numerical features $X_j^p = X_j(\mathbf{O}^p), j = 1...N$ characterizing these objects. This preprocessing step involves computing various characteristics of images, spatio-temporal patterns, replacing symbolic features by numerical values etc. Using a function suitable for evaluation of similarity or dissimilarity of objects represented by vectors in the feature space, $D(\mathbf{X},\mathbf{Y})$ create a reference (or prototype) vectors $\mathbf{R}$ in the feature space using the similarity measure and the training set $T = \{\mathbf{X}^p\}$ (a subset of all cases given for classification). The set of reference vectors, similarity measure, the feature space and procedures employed to compute probability define the classification model M.

Once the model has been selected it should be optimized. For this purpose define a cost function $E[T;M]$ measuring the performance accuracy of the system on a training set T of vectors; a validation set V composed of cases that are not used directly to optimize model M may also be defined and performance $E[V;M]$ measuring generalization abilities of the model assessed. Optimize parameters of the model M_a until the cost function $E[T;M_a]$ reaches minimum on the set T or on the validation set $E[V;M_a]$. If the model produced so far is not sufficiently accurate add new procedures/parameters creating more complex model M_{a+1}. If a single model

is not sufficient create several local models $M_a^{(l)}$ and use an interpolation procedure to select the best model or combine results creating ensembles of models. All these steps are mutually dependent and involve many choices described below in some details.

The final classification model M is build by selecting a combination of all available elements and procedures. A general similarity-based classification model may include all or some of the following elements:
$M = \{\mathbf{X}(\mathbf{O}), \Delta(\cdot,\cdot), D(\cdot,\cdot), k, G(D), \{\mathbf{R}\}, \{p_i(R)\}, E[\cdot], K(\cdot), S(\cdot)\}$, where:
$\mathbf{X}(\mathbf{O})$ is the mapping defining the feature space and selecting the relevant features;
$\Delta_j(X_j;Y_j)$ calculates similarity of X_j, Y_j features, $j = 1..N$;
$D(\mathbf{X},\mathbf{Y}) = D(\{\Delta_j(X_j;Y_j)\})$ is a function that combines similarities defined for each attribute to compute similarities of vectors; if the similarity function selected has metric properties the SBM may be called the minimal distance (MD) method.
k is the number of reference vectors taken into account in the neighborhood of $\mathbf{X}$;
$G(D) = G(D(\mathbf{X},\mathbf{R}))$ is the weighting function estimating contribution of the reference vector $\mathbf{R}$ to the classification probability of $\mathbf{X}$;
$\{\mathbf{R}\}$ is a set of reference vectors created from the set of training vectors $T = \{\mathbf{X}^p\}$ by some selection and optimization procedure;
$p_i(\mathbf{R}), i = 1..K$ is a set of class probabilities for each reference vector;
$E[T;M]$ or $E[V;M]$ is a total cost function that is minimized at the training stage; it may include a misclassification risk matrix $R(C_i,C_j), i,j = 1..K$;
$K(\cdot)$ is a kernel function, scaling the influence of the error, for a given training example, on the total cost function;
$S(\cdot)$ is a function (or a matrix) evaluating similarity (or more frequently dissimilarity) of the classes; if class labels are soft, or if they are given by a vector of probabilities $p_i(\mathbf{X})$, classification task is in fact a mapping. $S(C_i,C_j)$ function allows to include a large number of classes, “softening" the labeling of objects that are given for classification.

Various choices of parameters and procedures in the context of network computations leads to a large number of similarity-based classification methods. Some of these models are well known and some have not yet been used. We have explored so far only a few aspects of this framework, describing various procedures of feature selection, parameterization of similarity functions for objects and single features, selection and weighting of reference vectors, creation of ensembles of models and estimation of classification probability using ensembles, definitions of cost functions, choice of optimization methods, and various network realizations of the methods that may be created by combination of all these procedures [6–8].

The k-NN model $p(C_i|\mathbf{X};M)$ is parameterized by $p(C_i|\mathbf{X};k,D(\cdot),\{\mathbf{X}\}\})$, i.e. the whole training dataset is used as the reference set, k nearest prototypes are included with the same weight, and a typical distance function, such as the Euclidean or the Manhattan distance, is used. Probabilities are $p(C_i|\mathbf{X};M) = N_i/k$, where N_i is the number of neighboring vectors belonging to the class C_i. The most probable class is selected as the winner. Many variants of this basic model may be created [6,7].

Neural-like network realizations of the RBF and MLP types are also special cases of this framework.

The SBM framework allows for so many choices that exploring all the choices will be almost impossible. Instead an automatic search for the best model for a given data within the space of all possible models is pursued below.

3 Search for the best model

A search tree in the space of all models M_a for the simplest and most accurate model that accounts for the data requires a reference model that should be placed in the root of the tree. The reference model should be the simplest possible. In the SBM framework the *k*-NN model with k=1 and Euclidean distance function applied to standardized data is a good reference model. Not all extensions may be applied to all models. Instead of creating an expert system based on abstract description of models and checking the conditions for applicability of extensions algorithms, an "interaction matrix" defining possible extensions is defined. Interaction defines how to combine models in order to create more complex models. Consider two extensions of the basic model, one using an optimization of the number of nearest neighbors *k* and the other using attribute selection method. The interaction in the first algorithm says: 'If the attribute selection method is preceding the optimization of *k* in a model chain, optimize *k* with the attributes found by the earlier method'. Interaction in the second algorithm says that if optimization of *k* is followed by the attribute selection, the optimal *k* found earlier should be used to search for attributes. Without interaction the meta-learning algorithm reduces to the single-level ranking of basic models and does not create more complex methods.

Optimization should be done using validation sets (for example in crossvalidation tests) to improve generalization. Starting from the simplest model, such as the nearest neighbor model, qualitatively new "optimization channel" is opened by adding the most promising new extension, a set of parameters or a procedure that leads to greatest improvements. The model may be more or less complex than the previous one (for example, feature selection or selection of reference vectors may simplify the model). If several models give similar results the one with the lowest complexity is selected. For example, feature selection should be prefered over feature weighting. The search in the space of all SBM models is stopped when no significant improvements are achieved by new extensions.

The evaluation function $C(M_l)$ returns the classification accuracy of the model M_l calculated on a validation set or in the crossvalidation test. Let n denote the initial number of possible extensions of the reference model. The model sequence selection algorithm proceeds as follows:

1. Take the initial reference model as the best mode M_b.
2. Repeat until the pool of possible extensions is empty:
3. Create a pool of n initial models, $M = \{M_l\}, l = 1 \ldots n$ applying all extensions to the best mode M_b.

4. Optimize all models in the pool.
5. Evaluate all these models $C(M_l)$ and arrange them in a decreasing order of accuracy $C_a(M_i) \geq C_a(M_j)$ for $i > j$.
6. Select the best model M_b from the M pool as the reference; if several models have similar performance select the one with lowest complexity.
7. If there is no significant improvement stop and return the current best model.
8. Otherwise remove the extension used to create this model from the list of available extensions; set $n = n - 1$.

The interaction between the current "best model" and all possible extension determines whether these extensions are applicable at a given stage. The number of model optimizations is equal to the number of initial extensions n and does not exceed $n(n-1)/2$.

The result of this algorithm is a sequence of models of increasing complexity, without re-optimization of previously created models. This "best-first" algorithm finds a sequence of models that give the highest classification accuracy on validation partition or in crossvalidation tests. In case of k-NN-like models validation partition is rarely used since the leave-one-out calculations are easy to perform; for more complex models crossvalidation calculations are performed. Such approach may actually be preferable because rarely the data sets are sufficiently large to use validation sets, and using only the training set makes comparison with other methods easier.

The algorithm described above is prone to local minima, as any "best-first" or search algorithm. The beam search algorithm for selection of the best sequence of models is more computationally expensive but it has a better chance to find a good sequence of models. Since the SBM scheme allows to add many parameters and procedures, new models may also be created on demand if adding models created so far does not improve results. Some model optimizations, such as the minimization of the weights of attributes in the distance function, may be relatively expensive. Re-optimization of models in the pool may be desirable but it would increase the computational costs significantly. Therefore we will investigate below only the simplest "best-first" sequence selection algorithm, as described above.

4 Numerical experiments

We have performed preliminary numerical tests on several datasets. The models taken into account include optimization of k, optimization of distance function, feature selection, and optimization of the scaled distance functions:

$$D(\mathbf{X},\mathbf{Y})^{\alpha} = \sum_{i=1}^{n} s_i |X_i - Y_i|^{\alpha} \tag{1}$$

In the present implementation of the program α is changed from 0.25 to 10 in 0.25 steps; this covers Euclidean ($\alpha = 2$) and Manhattan ($\alpha = 1$) weighted functions. Two

other distance function, Chebyschev ($\alpha = \inf$) and Canberra,

$$D_C(\mathbf{X},\mathbf{Y}) = \frac{|X_j, Y_j|}{|X_j, -Y_j|} \tag{2}$$

$$D_{Ch}(\mathbf{X},\mathbf{Y}) = \max_{i=1,\ldots,N} |X_i - Y_i| \tag{3}$$

are also included.

Various methods of learning by parameter optimization may be used. We have used the multisimplex method, adaptive simulated annealing (ASA) method [9] and a discretized search methods with progressive decreasing of quantization step (tuning). One of the scaling coefficients s_i should be fixed to 1, since only relative distances play role in the similarity-based methods. Starting parameters are another issue; initial scaling factors may start from zero or from one. Application of simplex or ASA methods may lead to models with relatively large variance; one way to stabilize these models is to create a number of models and use them in a committee [10].

4.1 Monk problems

The artificial dataset Monk-1 [11] is designed for rule-based symbolic machine learning algorithms (the data was taken from the UCI repository [12]). The nearest neighbor algorithms usually do not work well in such cases. 6 symbolic attributes are given as input, 124 cases are given for training and 432 cases for testing. We are interested here in the performance of the model selection procedures.

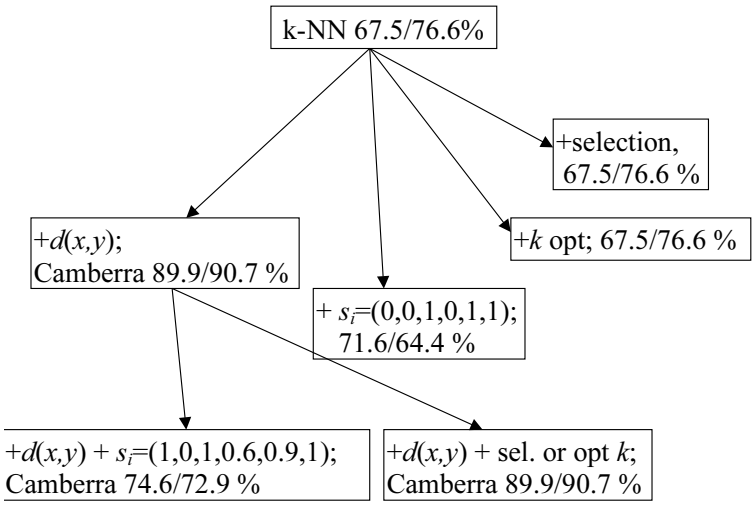

Fig. 1. Search for the best Monk2 data model.

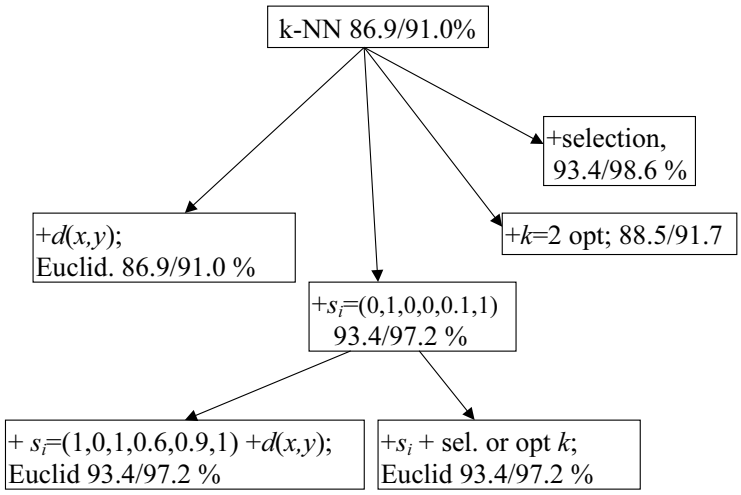

Fig. 2. Search for the best Monk3 data model.

The meta-learning algorithm starts from the reference model, a standard *k*-NN, with $k = 1$ and Euclidean function. The leave-one-out training accuracy is 76.6% (on test 85.9%). At the first level the choice is: optimization of *k*, optimization of the type of similarity function, selection of features and weighting of features. Results are summarized in the Table below. Feature weighting (1, 1, 0.1, 0, 0.9, 0), implemented here using a search procedure with 0.1 quantization step, already at the first level of search for the best extension of the reference model achieves 100% accuracy on the test set and 99.2%, or just a single error, in the leave-one-out estimations on the training set. Additional complexity may not justify further search. Selection of the optimal distance for the weighted *k*-NN reference model achieves 100% on both training and the test set, therefore the search procedure is stopped.

Table 1. Results for the Monk-1 problem with *k*-NN as reference model.

Method	Acc. Train %	Test %
ref = *k*-NN, *k*=1, Euclidean	76.6	85.9
ref + *k*=3	82.3	80.6
ref + Canberra distance	79.8	88.4
ref + feature selection 1, 2, 5	96.8	100.0
ref + feature weights	99.2	100.0
ref = *k*-NN, Euclid, weights	99.2	100.0
ref + Canberra distance	100.0	100.0

In the Monk 2 problem the best combination sequence of models was *k*-NN with Canberra distance function, giving the training accuracy of 89.9% and test

set accuracy of 90.7%. In the Monk 3 case weighted distance with just 2 non-zero coefficients gave training accuracy of 93.4% and test result of 97.2%.

4.2 Hepatobiliary disorders

The data contain four types of hepatobiliary disorders found in 536 patients of a university affiliated Tokyo-based hospital; 163 cases were used as the test data [13]. Each case is described by 9 biochemical tests and a sex of the patient. The class distribution in the training partition is 34.0%, 23.9%, 22.3% and 19.8%. This dataset has strongly overlapping classes and is rather difficult. With 49 crisp logic rules only about 63% accuracy on the test set was achieved [14], and over 100 fuzzy rules based on Gaussian or triangular membership functions give about 75-76% accuracy.

The reference *k*-NN model with k=1, Euclidean distance function gave 72.7% in the leave-one-out run on the training set (77.9% on the test set). Although only the training set results are used in the model search results on the test set are given here to show if there is any correlation between the training and the test results. The search for the best model proceeded as follows:

First level

1. Optimization of *k* finds the best result with *k*=1, accuracy 72.7% on training (test 77.9%).
2. Optimization of the distance function gives training accuracy of 79.1% with Manhattan function (test 77.9%).
3. Selection of features removed feature "Creatinine level", giving 74.3% on the training set; (test 79.1%).
4. Weighting of features in the Euclidean distance function gives 78.0% on training (test 78.5%). Final weights were [1.0, 1.0, 0.7, 1.0, 0.2, 0.3, 0.8, 0.8, 0.0].

The best training result 79.1% (although 77.9% is not the best test result) is obtained by selecting the Manhattan function, therefore at the **second level** this becomes the reference model:

1. Optimization of *k* finds the best result with *k*=1, accuracy 72.7% on training (test 77.9%).
2. Selection of features did not remove anything, leaving 79.1% on the training (test 77.9%).
3. Weighting of features in the Manhattan distance function gives 80.1% on training (final weights are [1.0, 0.8, 1.0, 0.9, 0.4, 1.0, 1.0, 1.0, 1.0]; (test 80.4%).

At the **third level** weighted Manhattan distance giving 80.1% on training (test 80.4%) becomes the reference model and since optimization of *k* nor the selection of features does not improve the training (nor test) result this becomes the final model. Comparison of results on this data set is given below:

Since classes strongly overlap the best one can do in such cases is to identify the cases that can be reliable classified and assign the remaining cases to pairs of classes.

Table 2. Results for the hepatobiliary disorders. Accuracy on the training and test sets.

Method	Training set	Test set
Model optimization	80.1	80.4
FSM, Gaussian functions	93	75.6
FSM, 60 triangular functions	93	75.8
IB1c (instance-based)	–	76.7
C4.5 decision tree	94.4	75.5
Cascade Correlation	–	71.0
MLP with RPROP	–	68.0
Best fuzzy MLP model	75.5	66.3
LDA (statistical)	68.4	65.0
FOIL (inductive logic)	99	60.1
1R (rules)	58.4	50.3
Naive Bayes	–	46.6
IB2-IB4	81.2-85.5	43.6-44.6

4.3 Other data

We have tried the metalearning procedure on the ionosphere data [12]. Unfortunately there was no correlation between the results on the training and on the test set, so any good results in this case must be fortuitous.

The hypothyroid data [12] has 3772 cases for training, 3428 cases for testing, 22 attributes (15 binary, 6 continuous), and 3 classes: primary hypothyroid, compensated hypothyroid and normal (no hypothyroid). The class distribution in the training set is 93, 191, 3488 (92.5%) vectors and in the test set 73, 177, 3178 (92.7%). *k*-NN gives on standardized data slightly more than the base rate, but the search procedure finds first $k = 4$, then Canberra distance function and finally a set of weights for each attribute, leading to 98.89% accuracy on the test set. This is better than the best neural networks although still worse than logical rules for this data set [14].

5 Discussion

Although in this paper meta-learning was applied only to classification problems the SBM framework is also useful for associative memory algorithms, pattern completion, missing values [6], approximation and other computational intelligence problems. Although only a few extensions to the reference *k*-NN model were used the search in the model space automatically created quite accurate models. For hepatobiliary disorders a model with highest accuracy for real medical data has been found. Although the use of a validation set (or the use of the crossvalidation partitions) to guide the search process for the new models should prevent them from overfitting the data, at the same time enabling them to discover the best bias for the data other ways of model selection, such as the minimum description length (cf. [1]), should be investigated.

Similarity Based Learner (SBL) software developed in our laboratory includes many procedures belonging to the SBM framework. Methods implemented so far provide many similarity functions with different parameters, include several methods of feature selection, methods that weight attributes (based on minimization of the cost function or based on searching in the quantized weight space), methods of selection of interesting prototypes in the batch and on-line versions, and methods implementing partial-memory of the evolving system. Many optimization channels have not yet been programmed in our software, network models are still missing, but even at this preliminary stage results are very encouraging.

Acknowledgments: Support by the Polish Committee for Scientific Research, grant no. 8 T11C 006 19, is gratefully acknowledged.

References

1. Duda, R.O., Hart, P.E and Stork, D.G. (2001): Pattern classification. 2nd ed, John Wiley and Sons, New York (2001)
2. Michalski, R.S. (Ed.) (1993): Multistrategy Learning. Kluwer Academic Publishers.
3. Duch, W., and Grąbczewski, K. (2001): Heterogeneous adaptive systems. World Congress of Computational Intelligence, Honolulu, May 2001 (submitted).
4. Duch, W. and Jankowski N. (2001): Transfer functions: hidden possibilities for better neural networks. 9th European Symposium on Artificial Neural Networks (ESANN), Brugge 2001. De-facto publications, pp. 81-94
5. Michie, D., Spiegelhalter, D. J. and Taylor, C.C. (1994): Machine learning, neural and statistical classification. Elis Horwood, London
6. Duch, W., Adamczak, R., and Diercksen, G.H.F. (2000): Classification, Association and Pattern Completion using Neural Similarity Based Methods. Applied Mathematics and Computer Science **10**, 101–120
7. Duch W. (2000): Similarity-Based Methods. Similarity based methods: a general framework for classification, approximation and association, Control and Cybernetics 29 (4), 937-968.
8. Duch W., Grudziński K. (1999): Search and global minimization in similarity-based methods. In: Int. Joint Conference on Neural Networks (IJCNN), Washington, July 1999, paper no. 742
9. Ingberg, L. (1996): Adaptive simulated annealing (ASA): Lessons learned. J. Control and Cybernetics 25, 33-54
10. Grudziński, K., Duch, W. (2001): Ensembles of Similarity-Based Models. Inteligent Information Systems 2001, Advances in Soft Computing, Physica Verlag (Springer), pp. 75-85
11. Thrun, S.B. *et al.* (1991): The MONK's problems: a performance comparison of different learning algorithms. Carnegie Mellon University, Tech.Rep. CMU-CS-91-197
12. Mertz, C. J. and Murphy, P.M. UCI repository of machine learning datasets, http://www.ics.uci.edu/AI/ML/MLDBRepository.html
13. Hayashi, Y. , Imura, A., and Yoshida, K. (1990): Fuzzy neural expert system and its application to medical diagnosis. In: 8th International Congress on Cybernetics and Systems, New York, pp. 54-61
14. Duch, W., Adamczak, R., Grabczewski, K. (2001) A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 12, 277-306