SBL-PM: A Simple Algorithm for Selection of Reference Instances in Similarity Based Methods

Karol Grudziński and Włodzisław Duch

Department of Computer Methods, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.
E-mail: kagru,duch@phys.uni.torun.pl

Abstract. SBL-PM is a simple algorithm for selection of reference instances, a first step towards building a partial memory learner. A batch and on-line version of the algorithm is presented, allowing to find a compromise between the number of reference cases retained and the accuracy of the system. Preliminary experiments on real and artificial datasets illustrate these relations.

1 Introduction

The **SBL** system (Similarity Based Learner) is a set of computer programs encompassing many methods which are based on evaluation of similarity between the case under evaluation and reference cases constructed from the training library. A unified framework for similarity based methods (SBM) has been presented recently [1,2]. Methods belonging to this framework include the *k*–Nearest Neighbor (*k*–NN) algorithm and it's extensions, methods originating from pattern recognition and machine learning (instance based learning, memory based learning), as well as some neural network methods, such as the multi–layer perceptron networks (MLP) or networks based on radial–basis functions (RBF). All these methods may be viewed as examples of similarity based methods based on specific parameterization of the $p(C_i|\mathbf{X};M)$ posterior classification probability, where the model M involves all procedures and parameters that are optimized in a given method.

Within the SBM framework we have implemented many variants of the *k*–NN algorithm with optimization of the number of neighbors *k*, various functions for weighting of their influence, various parameterizations of the similarity (distance) functions, several methods that automatically assign weights to input attributes (based on minimization of the cost function or on searching in the weight space) and methods for selection of the most important attributes [3]. Currently our research is focused on neural network-like implementations of the SBM methods [4] and on various algorithms aiming at speeding up the calculations.

Selection of the reference cases is an important issue in all similarity-based methods. Reducing the size of the training set leads to minimization of the memory requirements and faster classification, usually at slight expense of prediction accuracy on test cases. Eliminating redundant cases and leaving only the most interesting prototypes allows to understand why an unseen case was classified to a particular class by analyzing the prototypes that were selected in the SBM method.

Such analysis may sometimes replace the need for logical analysis of the data, providing an alternative to the rule-based classifiers. SBL-PM, the algorithm proposed in this paper, is a new addition to the set of SBM framework programs. The algorithm is described in the next section. In the third section preliminary empirical tests to evaluate its performance and measure it's ability to reduce the size of a reference set are presented, and the last section concludes this paper.

2 Selection of the Reference Instances

In the simplest case the SBM classification algorithm may include all training cases as reference vectors. For several reasons it is not a good solution:

1. If the training set is very large most of the cases have no influence on classification; including them decreases only the computing performance of the algorithm.
2. If there is noise in data or the training set contains wrongly classified instances decreasing the number of reference vectors may increase the prediction ability of the system on unseen cases.
3. Large number of reference instances do not allow to identify interesting prototypes, making it difficult to understand the structure of the data.
4. If the number of the training instances is quite small it may be worthwhile to include virtual reference cases or to optimize the existing reference cases.

The first group of methods, which should work well for large datasets, is based on clusterization techniques. One has to select a relatively small reference set from the training cases laying near the centers of clusters. Such methods have been implemented in the Feature Space Mapping (FSM) network to select the initial prototypes [5,6] that are further optimized. FSM also belongs to the similarity-based method framework, but its algorithm is aimed at modeling the probability density functions and is more complex than most of the SBL algorithms.

The SBL-PM algorithm proposed here in principle can be used with any classification system, but because of the performance reasons it has been used so far only with the classical *k*–Nearest–Neighbor method. The algorithm is summarized below:

```
1. Set the partial memory of the system (reference set) to the entire training set,
   $\mathcal{R} = \mathcal{T} = \{\mathbf{R}_i\}, i = 1..N$.
2. Set the target accuracy $\Delta$ to $\Delta_1$ obtained from the leave-one-out test on $\mathcal{T}$.
3. Set the lowest accuracy $\Delta_m$ that should be considered.
4. Define the $\delta$ parameter determining steps in which the target accuracy $\Delta$ is lowered, for example $\delta = 0.05$.
   (a) Until $\Delta < \Delta_m$
       i. For i=1 to N
       ii. Select one vector $\mathbf{R}_i$ from $\mathcal{R}$ and set the temporary reference set to
           $\mathcal{R}' = \mathcal{R} - \mathbf{R}_i$.
```

```
        iii. Using the leave-one-out test and the current reference set $R'$ calculate
             the prediction accuracy $A_c$ on the whole training set $T$.
        iv.  If $A_c \geq \Delta$ set $R = R'$.
    (b) Set $A_e(\Delta) = A_c$ to record the accuracy at the end of this step.
    (c) Set $R(\Delta) = R$ to remember reference vectors at this stage.
    (d) Change $\Delta \leftarrow \Delta - \delta$.
5. Select the references obtained for the highest $A_e(\Delta)$.
```

Vectors are sequentially eliminated from the reference set unless the classification accuracy drops below the target accuracy Δ. Since Δ is set to the leave-one-out accuracy Δ_1 on the entire training set T this algorithm should not degrade the results of the *k*-NN classifier – in the worst case it will leave all training vectors as references. The threshold value Δ is lowered in several steps δ, allowing for some degradation of the performance as a penalty for reduction of the reference set. Displaying the function $A_e(\Delta)$ allows to select the optimal value of Δ, depending on our goal. The final reference set should be significantly smaller than the original training set with minimal degradation of the prediction accuracy.

The Δ parameter controls the number of reference cases that remain in the partial memory. We have used cumulative accuracy estimation here. Since the accuracy is decreased in small steps δ important references that may significantly decrease the accuracy, are not removed from the reference set. If the δ steps are not sufficiently small an additional parameter specifying the allowed threshold of A_c decrease due to the removal of a single reference may be specified.

SBL-PM procedure is repeated for several values of Δ to find a good compromise between classification accuracy and the number of reference vectors. If our goal is to maximize performance, estimation of the predicted accuracy $A_e(\Delta)$ should combine results $A_t(\Delta)$ obtained for the rejected vectors, i.e those in the $T - R$ set, and results $A_r(\Delta)$ for the reference vectors R. Accuracy $A_r(\Delta)$ for the reference vectors may only be estimated using the leave-one-out test. The goal of the reference selection algorithm is to obtain a small number of non-redundand reference cases. However, if all N_r references are non-redundant the accuracy $A_r(\Delta)$ may be quite low. Thus for sufficiently small Δ that corresponds to a small number of references, $A_t(\Delta)$ estimation should be used, while for $\Delta \approx \Delta_1$ the $A_r(\Delta)$ estimation is also important.

In the *k*-NN method selection of the number of neighbors *k* has strong influence on the final number of reference vectors and therefore *k* should be set before the selection procedure. Additional parameters (such as the selection and weighting of features or parameterization of the similarity function) may be optimized on the training set after the references are fixed. The order of optimization may be important for performance of the final system. Finding an optimal order of various optimizations in the SBM systems is still an open research issue.

The off-line SBL-PM procedure described above requires an access to all vectors in the training set T in the batch mode. In the on-line version of the method the system has to decide whether a new training case $\mathbf{X}_k$, coming from the input stream, should be added to the reference set (partial memory of past cases). An obvious approach, used for example in the IB2 procedure [7], is to check whether the new instance $\mathbf{X}_k$ is correctly classified using the reference set R created so far, and add $\mathbf{X}_k$ to R only if it is not handled correctly. To make this algorithm resistant to noise one may introduce a "candidate reference" vectors, that are included in R only on the preliminary basis. Candidate reference vectors $\mathbf{R}_c$ are then checked in the batch selection step described above, repeated after a specified number of references is created. In this approach the number of reference vectors is initially growing and when it becomes too large it is reduced. Partial memory of the system tries to match the complexity of the classification model to the complexity of the incoming data. The on-line SBL-PM algorithm looks as follows:

1. Set the maximum number of reference vectors N^r_{max} and the maximum number of training vectors N^t_{max}.
2. Take the first incoming vector $\mathbf{X}_1$ as the first reference $R = \{\mathbf{X}_1\}$ and as the first training vector $T = \{\mathbf{X}_1\}$.
3. Repeat for all incoming vectors $\mathbf{X}_k$:
   (a) Add the incoming vector $\mathbf{X}_k$ to the training set T created so far.
   (b) Determine the class $C(\mathbf{X}_k)$ of this vector using the reference set R created so far.
   (c) If $C(\mathbf{X}_k)$ is not correct add $\mathbf{X}_k$ to the current R.
   (d) If $N_r \geq N^r_{max}$ or $N_t \geq N^t_{max}$, where N_r (N_t) is the number of vectors in R (T), then
      i. Perform the batch step reducing R.
      ii. Empty the training set T.

The SBL-PM algorithm in the on-line version builds a partial memory system, forgetting the references that did not appear for a longer time. The N^r_{max} and N^t_{max} values should be sufficiently large to avoid forgetting important reference cases. N^t_{max} is called the length of the growing epoch – it sets the upper limit for the number of incoming vectors that are evaluated between two batch reductions. For large N^t_{max} this algorithm may become computationally expensive (due to the batch reduction costs), therefore another approach is recommended, allowing for smaller N^r_{max} and N^t_{max} values. References that survive several batch reductions are given an extra importance and should not be removed for a longer period. In the Feature Space Mapping neural network [5,6] a "mass" index is used for each prototype, counting how many vectors are correctly classified thanks to the contribution of this prototype. The same approach may also be used in the on-line SBL-PM algorithm to keep important references.

The advantage of the SBL-PM approach is its simplicity, therefore it may be used as a reference against more sophisticated methods (cf. the GIGA algorithm in which genetic algorithm is used for selection of reference set [8]). The high computational costs is a disadvantage: in the batch version classification of all N training samples T has to be repeated about N times to get the final reference set, while in the on-line version only N_r times. One way to speed up the batch algorithm is to remove groups of vectors rather than single ones. Reduced reference set leads to a lower computational cost during further optimization of the parameters of the SBL system and during actual classification, saving time and memory.

3 Experimental Results

The performance of the batch SBL-PM algorithm for the k-NN method is illustrated first on the Iris example. This classic benchmark data has been taken from the UCI repository [9] and contains 3 classes (Iris Setosa, Virginica and Versicolor flowers), 4 attributes (measurements of leaf and petal widths and length), 50 cases per class. The entire Iris dataset has been shown here (Fig. 1) in two dimensions, x_3 and x_4, which are much more informative the other two (cf. [10]). In Fig 2. the reference set obtained by taking the value of Δ from the leave-one-out test on the entire data and running the SBL-PM procedure for $k = 1$ is displayed. Only 6 reference vectors remained.

For $k = 1$ in the 10-fold stratified cross-validation test repeated 10 times on the Iris dataset the classical k-NN classifier gave 95.8% accuracy, with 0.3% variance. In the same test SBL-PM with $k = 1$ gave 95.3% accuracy (an insignificant decrease), with variance 1.7%. On average 6.7 reference vectors were used, which is only about 4% of the size of the training base (in 10-fold cross-validation the number of training cases is 135). A single Iris Setosa prototype is sufficient to perfectly account for this class.

For $k = 10$ in the same crossvalidation tests standard k-NN gives 97.0% accuracy, with 0.9% variance, while SBL-PM gives 95.3% $\pm$ 1.4% using on average 22.7 cases (about 17% of the training set). For larger k the minimum number of prototype vectors is equal k divided by the number of classes – in this case at least 4 vectors are necessary to have a majority of neighbors from one class. Since our goal is to illustrate the reference selection algorithm rather than obtain the best results we do not present detailed comparisons for the Iris dataset with results obtained by other classification systems. Due to the noisy character of the data the limit in the leave-one-out or crossvalidation tests is about 98% [10].

Another set of experiments was done on the 3 Monk datasets [11]. On this artificial data SBM gives good results (100% of correct answers on the first problem, 85% on problem 2, and over 97% on problem 3) only if feature selection and/or weighting is included [3]. We do not perform feature selection here, illustrating only the effect of selection of references on the performance. The Monk problems are more suitable for rule-based classification systems than for SBM systems that on real, noisy data work frequently better than any other systems (cf. [12]).

For $k = 1$ on Monk1 (2 classes, 124 training cases, 432 in test set) standard k-NN gives 85.9% while SBL-PM for the same value of k gives 70.6%, a significant decrease in prediction ability on unseen cases. The reference set contains only 10 cases, which constitutes only 2.3% of the entire training set. For Monk2 dataset (2 classes, 169 training cases, 432 in test set) standard k-NN for $k = 1$ gives 76.6% on the test set while SBL-PM gives 64.6% with only 18 reference cases (11% of the training set). Also in this case the decrease of prediction ability is significant. For Monk3 dataset (2 classes, 122 training, 432 testing cases) standard k-NN gives 90.1% on test set while SBL-PM gives 85.0% on test using only 11 instances (2.5% of the training set).

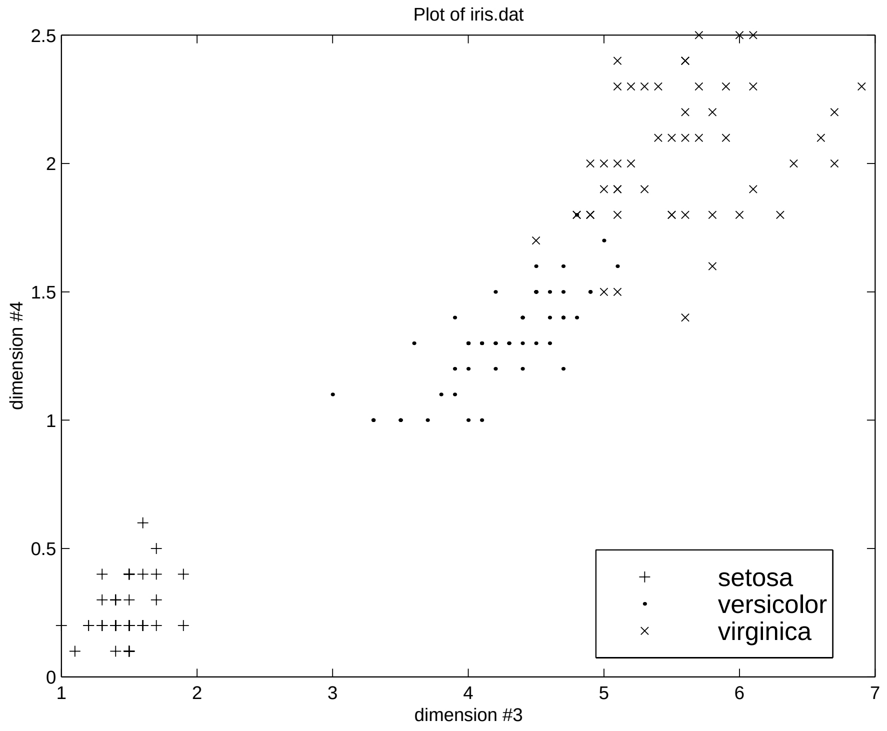

Fig. 1. Original 150 Iris data vectors displayed using the last two features.

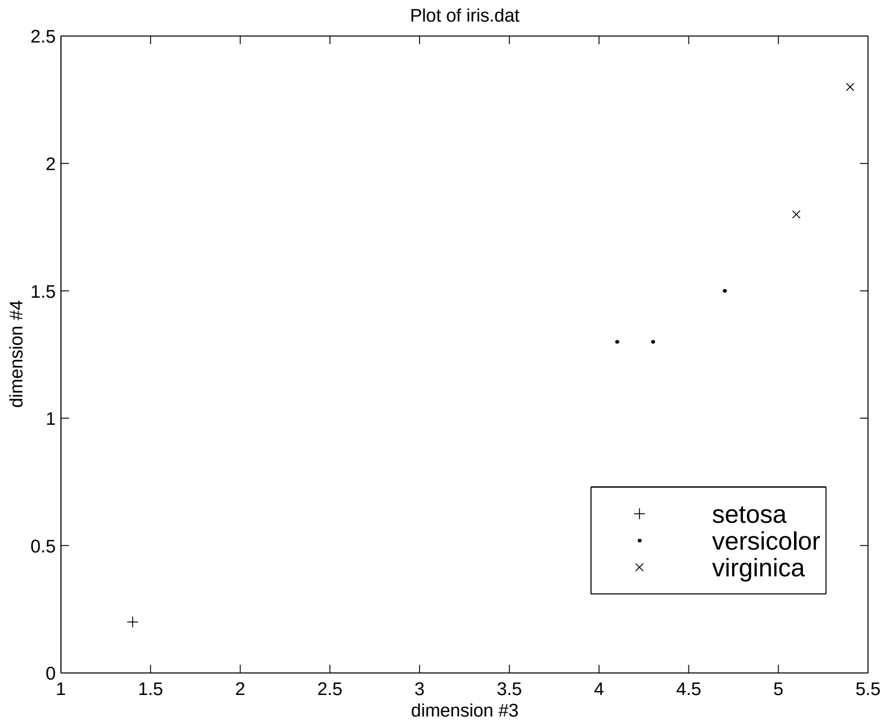

Fig. 2. 6 reference vectors left after the SBL-PM batch procedure.

Poor results of the above experiments are due to the poor leave-one-out results on the training partition. Increasing the value of Δ above the value taken from the leave-one-out procedure improves the results significantly but the algorithm uses more cases in partial memory. Setting the value of Δ to 90% instead of 85.9% SBL-PM on Monk1 dataset gives 80.3% on test set using 36 cases (29%) of the size of training set) and for $\Delta = 95\%$ we obtained 83.1% on test set using 52 cases (42% of the size of the training set). For Monk2 for $\Delta = 80\%$ SBL-PM gave 70.4% on test set using 50 cases (approximately 30% of the size of the training set). For Monk3 for $\Delta = 90\%$ SBL-PM gave 88.4% on test set using 20 cases (16.4% of the size of the training set).

4 Conclusions

A preliminary implementation of the SBL-PM algorithm creating the reference set of cases (partial memory) in similarity-based methods (SBM) has been described. The procedure may be used in the batch or on-line mode, allowing to find a compromise between the number of reference cases retained and the accuracy achieved. It can be combined with other procedures (such as the feature weighting and selection procedures) and parameter optimizations (such as the distance-related parameters) to create SBM system with partial memory. Our goal is to implement mechanisms of different type – optimization of the distance functions, weighting functions, scaling and selection of features, and many other procedures, in one program. This will enable us to search, in space of available models, for the most appropriate model for a given data. The SBM framework is quite rich, containing many possibilities that have not been explored so far. Interaction of different optimization and selection procedures has not been addressed yet. For example, selection of the reference set may be done before, after or simultaneously with selection of the best *k*, selection of features or weighting of features.

The limited empirical evidence presented in previous section indicates that for real data with continuos attributes the number of references may be significantly reduced without loss of accuracy (Iris data), while for artificial data (symbolic Monk problems) the loss of accuracy may be significant. Tests to find out if the combination of the selection of references and attributes will allow to preserve high accuracy are being conducted. Generalization abilities and the degree of the memory resources reduction offered by the SML-PM algorithm should be compared to other partial memory systems like IB2 [7] or AQ-PM [13]. Strategies to determine representative examples from the input stream may follow the IB2 procedure [7] or an on-line variant of clusterization methods used for initialization of neural methods [6]. Optimization of individual vectors in the spirit of LVQ (Learning Vector Quantization) methods is an obvious next step that should be considered. A lot of theoretical and experimental work remains to be done before we will understand in details the optimal way of selection of the reference cases in the SBM framework.

Acknowledgments: Support by the KBN, grant 8 T11F 014 14, is gratefully acknowledged.

References

1. Duch W. (1998) A framework for similarity-based classification methods, Intelligent Information Systems VII, Malbork, Poland, June 1998, pp. 288-291
2. Duch W., Grudziński K. (1999) Search and global minimization in similarity-based methods, Int. Joint Conf. on Neural Networks, Washington, July 1999, paper no. 742
3. Duch W., Grudziński K. (1999) Weighting and selection of features. Intelligent Information Systems VII, Ustroń, Poland, 14-18 June 1999, pp. 32-36
4. Duch W., Grudziński K. The weighted k-NN method with selection of features and its neural realization, 4th Conf. on Neural Networks and Their Applications, Zakopane, May 1999, pp. 191-196
5. Duch, W., Diercksen, G.H.F. (1995) *Feature Space Mapping as a universal adaptive system*, Computer Physics Communication **87**, 341–371
6. Duch W., Adamczak R., Jankowski, N. (1997) Initialization of adaptive parameters in density networks, 3rd Conf. on Neural Networks, Kule, Oct. 1997, pp. 99-104
7. Aha D., Kibler D., Albert M. (1991) Instance-based learning algorithms, Machine Learning **6**: 37-66
8. Fuchs M. (1996) Optimized nearest-neighbor classifiers using generated instances. LSA-96-02E Technical Report, Learning Systems & Applications Group, Univeristy of Kaiserslautern, Germany
9. Mertz C.J., Murphy. P.M. (1999) UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-data-bases.
10. Duch, W., Adamczak, R., Grąbczewski, K. (in print) Methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks.
11. Thrun S.B. *et al.* (1991) The MONK's problems: a performance comparison of different learning algorithms. Carnegie Mellon University, CMU-CS-91-197
12. Duch, W., Adamczak, R., Grąbczewski, K., Żal G., Hayashi, Y. (1999) Fuzzy and crisp logical rule extraction methods in application to medical data. Computational Intelligence and Applications – Springer Studies in Fuzziness and Soft Computing, Vol. 41, ed. P.S. Szczepaniak, pp. 593-616
13. Michalski R. (1999) AQ-PM: A System for Partial Memory Learning. Intelligent Information Systems VII, Ustroń, Poland, June 1999, pp. 70-79