IMPROVED MEMORY-BASED CLASSIFICATION

Wlodzislaw Duch, Rafal Adamczak and Norbert Jankowski
Department of Computer Methods, Nicholas Copernicus University
Grudziadzka 5, 87-100 Toruñ, Poland
e-mail: duch,raad,norbert@phys.uni.torun.pl

Abstract: Memory-based methods and memory-based reasoning are powerful classification methods but for a large number of data items they are quite expensive. The number of memorized data items is reduced if some fuzziness in the representation of the data is introduced. Memory-based techniques are extended to deal with non-point objects in feature spaces. The effectiveness of this approach is demonstrated in a number of examples.

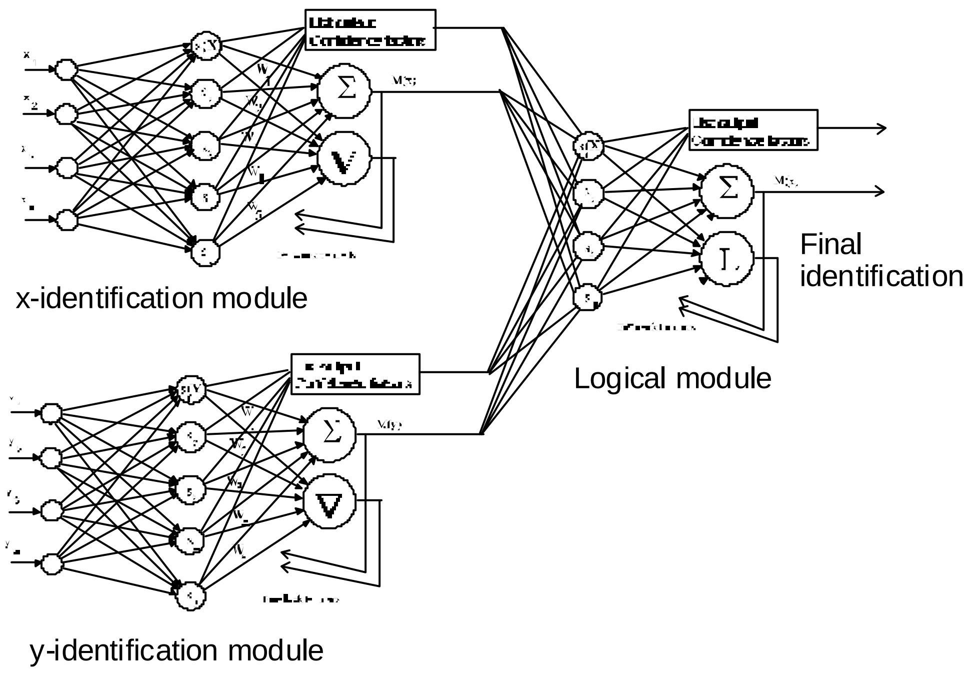
1. Introduction

Memory-based methods (MBM) belong to the family of the nearest-neighbor-like methods useful for classification. Memory-based reasoning (MBR), a particular version of the memory-based methods, has been applied [1] to the typical benchmark database tests as well as to the object recognition and classification of free text samples taken from a very large databases. There is no learning phase and no adaptive or learning parameters to manipulate. The results are better or comparable to those obtained by neural networks. In general memory based methods seem to be better suited to industrial, large scale applications than neural methods.

The disadvantages of the memory based methods are: large number of good examples is needed (generalization and interpolation is not as good as in neural networks), there is no data compression, leading to large computational demands and large memory requirements and making these methods unsuitable for on-line learning. Clearly a combination of memory-based methods (developed by the pattern recognition community) and adaptive systems (developed by the neural networks community) is desired. One way to improve MBM is to use modified Value Difference Metric [1]. Another approach, explored in this paper, is based on forming prototypes or smoothing description for a number of facts clustered near each other, instead of storing all training facts. Neural networks that use localized functions, such as RBF networks, are capable of such a description. However, their classification decisions are based only on the output values of the network, not on the distance between the data and the clusters. The theory behind RBF networks does not help to see the problem as that of a geometrical description of complex shapes in multidimensional input spaces and does not distinguish between the local generalization based on interpolation, and the probabilistic, nearest neighbor generalization based on distances from the input data to the boarders of clusters.

2. Feature Space Mapping approach

The Feature Space Mapping (FSM) neurofuzzy network [2] is based on fuzzy memory traces. Changing the parameters of FSM allows for trading the memory requirements and computational demands for accuracy of data description. The incoming data vectors $\mathbf{I}(t)$ define features (via pre-processing, if necessary) of internal representations $\mathbf{X}_i(t)$ suitable for classification. A coordinate system based on these features $\{\mathbf{X}_i\}$ defines a multidimensional feature space. Since these features may be of different types (for example, coming from different sensors) partial classifications are performed in local feature spaces while the final classification is performed in the space that is highest in the hierarchy. The system learns by creating and modifying memory traces (clusters) in these spaces. Memory traces are described using memory function $FSM(\mathbf{X})$ as the fuzzy areas in the feature space where the function has non-zero values. Local maxima of the memory function are prototypical representations of the (fuzzy) training data. Such system may learn in



Two recognition and one reasoning module of the FSM system. Recognition modules specialize in classification of different type of data and contribute to the final classification performed by the reasoning module. Each module computes the output function as well as the gradient necessary to find the nearest clusters.

supervised as well as unsupervised mode and may use fuzzy reasoning in the process of classification or decision making.

In the unsupervised mode the system remembers the incoming data vectors and tries to form clusters in the data space. There are several problems in forming reliable clusters usually solved by placing many small clusters in the data space (cf. RCE or NLS classifiers [3]). A vector that lies at the border of a real cluster and that appears early in the training sequence may initialize a new cluster, leading to a large final number of clusters or slowing down the learning process. To avoid such situation we have developed an algorithm for **initial clusterization**. This algorithm is independent of any particular network realization. The total complexity of the data representation (number of network nodes) is restricted in the initialization step by assuming the maximum number of clusters allowed. The position and size of clusters for each class are estimated independently in each dimension. The data is renormalized to a unit section and viewed at a low resolution (by dropping the significant digits using the integer arithmetics). At a resolution of *k* bins we count the number of data vectors in each bin. Several adjacent bins with non-zero vectors in them form a cluster in one dimension, defining positions and initial cluster sizes. The resolution is increased up to the noise level (defined in a global way for the input data by the user) or until the total number of initial clusters reaches allowed value. Suppose that the cluster starting from bin *i* takes *l* units of size *s*. Then its position and size in the original scale is

$$C_p = x_{\min} + s(i + \tfrac{l}{2});\; C_\sigma = \tfrac{1}{2} ls;\; s = \left| x_{\max} - x_{\min} \right| / k$$

Each of these one-dimensional clusters may be a projection of many separate *n*-dimensional clusters. In the initialization phase vectors are read and a loop over all dimensions started. In the first dimension the vector will belong to some cluster C_1^i, in the second dimension to C_2^j (this already defines a two-dimensional cluster). and in the n-th dimension to C_n^k. Each *n*-dimensional cluster is characterized by a chain of lower-dimensional clusters, $C_1^i \to C_2^j \to ... \to C_n^k$, therefore this initialization procedure creates a classification tree. In the leafs of this tree the number of vectors belonging to the *n*-dimensional cluster are counted, determining the initial inertia of this cluster. If the number of clusters at any level exceeds the maximum allowed the search is terminated and results from more coarse-grained initialization are taken. Finally, once all clusters are found distances between them are computed and those clusters that are closer than a specified threshold are merged into one. This step allows to take into account skewed distributions that are not recognized as single clusters by the search algorithm that effectively divides the input space into parallelepipeds. So far the clusters are not represented by any special function, their positions and sizes are memorized as numerical values. This rough description of the initial data for simple distributions of clusters may already be sufficient for reliable classification, but usually it is followed by a more precise representation of the full data set (in case of on-line learning by continuos tuning). In the FSM system this is done by adapting parameters of the network function:

$$FSM(\mathbf{X}) = \sum_{i=1}^{m} W_i * G(\mathbf{X}; \mathbf{D}_i, \mathbf{s}_i)$$

Localized functions G realized by the *FSM* network nodes are parametrized by the positions at which they are centered and their size or degree of fuzziness. In particular gaussian-like functions or products of pairs of sigmoids may be taken. The values describing the initial clusters are used to set initial parameters of the *FSM* network and the data is read at full resolution. Various mechanism improving the representation of the data are used (see [2] for details). Classification in neural networks is based on the value of the output. Localized description of the input data may leave large regions of the input space where the network output is effectively zero. On the other hand delocalized functions, such as sigmoidal functions used in the popular MLP networks, create arbitrary boundaries between classes. Although the network pretends that it knows it may actually extrapolate far beyond any reliable values. In memory based systems using k-NN situation is slightly similar, although if the distance from the nearest memory trace is large the system may warn the user that the reliability of the answer is low. In FSM the representation of the data is compressed by clustering, therefore the k-NN approach should not be used directly for centers of the clusters, because in such a case the data that is close to the border of a large cluster would be wrongly assigned to a small cluster whose center is closer. FSM networks compute gradients and use them to find the cluster with “largest influence” on a given position in the data space. If gradients and the network outputs are too small the fuziness of the network nodes is temporarily increased until they become sufficiently large to point towards the most important cluster in the neighborhood. To save time in the applications described below we have used gradients only when the value of the *FSM* function was quite small, otherwise the class of the most active node was selected.

The improvement over memory based classification results from clusterization of data and calculation of gradients, allowing to extend the idea of nearest neighbors to those clusters that have the largest influence on a given spot of the input space. It is equally easy to extend classification procedure to a larger number of nearest neighbors by switching off the cluster found as first and repeating the same procedure to find next neighbor.

3. Results

The data were taken from the UCI machine learning repository [4]. All tests have been performed 10 times each and randomized whenever necessary. The average error rates are given below.

Glass Identification Database created by Forensic Science Service contains information about 7 type of glasses that may be identified by 10 chemical and physical properties. A total of 214 samples are provided, out of which we have selected randomly 21 vectors as the test base and took the rest as the training base. After initialization the network had 7 nodes and accuracy of 88%. We have improved this accuracy to 96% by adding 12 new nodes, achieving accuracy of 94.5% on the test set. Further increase of accuracy for the training set to 100% resulted in worse generalization and decreasing accuracy to 88.5% on the test set. For this database the accuracy of a rule-based system (BEAGLE system), the nearest-neighbor algorithm and discriminant analysis is 82%, 83% and 74%, respectively.

Ionosphere data from the Space Physics Group at John Hopkins University contains classification of radar returns from the ionosphere. It has 351 vectors, with 34 attributes, divided into two classes. In this case 200 training vectors and 151 test vectors were taken. Initialization gave only two clusters and the FSM network working with accuracy of 77%. Subsequent learning increased the number of nodes to 59 for 96%, to 76 for 98% and to 100 for 100% accuracy on the training set. On the test set the accuracy achieved was 91.5%, 92.8% and 92.2% respectively. The nearest neighbor method reaches an accuracy of 92.1% in this case while best backpropagation attempts gave 96%.

The Shuttle dataset from NASA contains 9 numerical attributes, 43500 training vectors and 14500 test vectors. There are 6 classes. Initialization gives 7 network nodes and 88% accuracy. Increasing accuracy on the training set to 94%, 96% and 98% leads to a total of 15, 18 and 25 nodes and accuracies on the test set of 95.5%, 97.8% and 98.5%. Backpropagation network reached an accuracy of 95.5% on the training set. K-NN is very slow in this case, requiring all 43500 vectors as memory traces, but quite accurate, reaching on training and test set 99.6%. Although our results are slightly worse they are achieved by memorizing 25 nodes only.

Letter Image Recognition Data contains 16-dimensional description of 26 English language letters (statistical moments and edge counts). Out of 20000 vectors the last 5000 were selected as test cases. In this case much larger number of network nodes were generated: for 92% of accuracy 717 and for 94% 1074. Accuracy of classification on the test set was 88.2% and 88.4%. Other methods are not doing much better in this case, with backpropagation network achieving only 68% accuracy on the training and 67% on the test set. Interestingly best k-NN is reported with 93.2% accuracy, so it seems that our description of the data clusters is not able in this case to capture all the details of the data distribution.

DNA database contains data on windows of 60 DNA base-pairs, coded by 3 bits for a total of 180 attributes. In the middle of this window 3 types of boundaries are possible (intron to exon, exon to intron or neither). There are 2000 training vectors and 1186 test vectors. The network was trained to up to 98% of accuracy on the training set, leading to a relatively large number of 1250 nodes and achieving 92.0% accuracy on the test set. This is slightly better than the 91.2% accuracy achieved by backpropagation and much better than 85.4% accuracy of k-NN classifier.

Galaxies data were obtained from ESO-LV catalog and compared with the results of backpropagation network reported in [5]. The goal is to achieve automatic classification comparable to that of human experts. There were 2 classes (early type and late type galaxies), with 13 attributes each, 1700 vectors used for training and 3517 for testing. Backpropagation achieved 89.6% accuracy while our results are similar for 90% accuracy on the training set (43 nodes), but they are easily improved to 98% accuracy on the training set and 93.2% on the test set, in which case 384 nodes were created.

In summary we see a significant improvement over the memory based methods: the number of nodes to be memorized is much smaller, therefore classification is much faster and, comparing to such classifiers as k-NN, the accuracy is usually increased.

Acknowledgment: Support by the Polish Committee for Scientific Research, grant 8T11F 00308, is gratefully acknowledged.

References:

[1] D.L. Waltz, in: M.A. Arbib, Editor, The Handbook of Brain Theory and Neural Networks (MIT Press 1995), pp. 568-570; R.H.Creecy, B.M. Masand, S.J.Smith, D.L. Waltz, Comm. of the ACM 35 (1992) 49-64

[2] W. Duch, G.H.F. Diercksen, Comp. Phys. Comm. **87** (1995) 341-371; W. Duch, Neural Network World **4** (1994) 645-654

[3] P.D. Wasserman, *Advanced methods in neural networks* (van Nostrand Reinhold, 1993)

[4] P.M. Murphy, D.W. Aha, *UCI Repository of machine learning databases*. Univ. of California, Irvine 1994; http://www.ics.uci.edu/~mlearn/MLRepository.html

[5] O. Lahav, A. Naim, L.Sodre Jr. and M. C. Storrie-Lombardi, Institute of Astronomy, Cambridge, Technical Report CB3 OHA (1995)