Search and global minimization in similarity-based methods.

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

Abstract.

The class of similarity based methods (SBM) covers most neural models and many other classifiers. Performance of such methods is significantly improved if irrelevant features are removed and feature weights introduced, scaling their influence on calculation of similarity. Several methods for feature selection and weighting are described. As an alternative to the global minimization procedures computationally efficient best–first search methods are advocated. Although these methods can be used with any SBM classifier they have been tested using the k-NN method, since it is relatively fast and for some databases gives excellent results. A few illustrative examples show significant improvements due to the feature weighting and selection.

Introduction.

MANY neural, pattern recognition and machine learning methods developed in the past use explicitly or implicitly similarity measures during the training and classification process (although our focus here is on classification the same reasoning may also be applied to regression and pattern completion problems). Empirical evaluation of 20 statistical, neural and machine learning classification methods made within the *StatLog* project [1] showed that the simplest version of the nearest neighbor method, using a single neighbor only, achieved best results in classification of 4 out of 22 datasets, and close to the best results for another 3 datasets, i.e. in about 1/3 of all problems considered (see also a more recent comparison of different methods by Rohwer and Morciniec [2]). It is now clear that no single classification model will outperform all other models for all datasets. Therefore a useful strategy is to have a common framework generating many classification models and to select empirically the best method for a given dataset. If we could systematically increase the complexity of classification models, finding an optimal combination of parameters and procedures should result in best (or close to the best) performance in almost all cases.

A general framework for similarity-based methods has been presented recently [3], [4], [5]. Except for classical minimal distance methods, such as the *k*-NN method, many popular neural network models, such as the Radial Basis Functions (RBFs), Multilayer Perceptrons (MLPs), Learning Vector Quantization (LVQ), may be presented in this form. Classification probability $p(C_i|\mathbf{X};M)$ is determined by parameters and procedures defining the classification model M, including cost functions, similarity functions (usually distance functions), number of reference centers included in similarity evaluations, size of the $\mathbf{X}$ neighborhood considered, weighting functions estimating contribution of reference vectors $\mathbf{R}^p$ to the classification probability, procedures for selection of the set of reference vectors, procedures for feature selection and feature scaling and several other factors. Minimization of a cost function using gradient procedures is possible only if the parameterization of $p(C_i|\mathbf{X};M)$ allows for gradient computations; otherwise non-gradient minimization methods should be used. Since these methods are computationally very demanding search-based procedures should be considered as an alternative.

The SBM framework enables systematic generation of many classification models of increasing complexity, including new neural network models. The best method for a given problem should be selected automatically from methods belonging to the SBM framework, starting with the simplest method (such as *k*-NN) and adding more complex features until no gain in performance is observed. For each type of classification model there is an optimal data-dependent complexity, but if the optimal model obtained at a given stage is enhanced in qualitatively different way (i.e. by optimization of different types of parameters rather than adding more parameters of the same type) better performance may still be achieved. For example an optimal number of neurons in Gaussian-based RBF network may be determined by cross-validation, but the results may improve further if dispersions of all Gaussian are optimized independently, and improve even further if additional parameters related to rotation of Gaussian functions are provided or if the distance function used by Gaussian transfer functions is changed from Euclidean to a more general one.

An important aspects of all SB methods concerns selection and weighting of features. Memory based algorithms degrade their performance when irrelevant or noisy features are used. Many methods of feature selection and weighting have been developed so far. Wettschereck and Aha [6], [7] propose a five-dimensional framework to characterize different methods of feature weighting. We have developed and tested several new feature selection methods useful for *k*-NN and other SB methods. They are based either on the global minimization of cost functions [8] or on search techniques. In the framework of Wettschereck and Aha they use feedback, do not change feature representation, use global weights, do not use task specific knowledge and perform both feature selection and feature weighting.

In the next section a brief description of the SBM framework is presented. The third section deals with new algorithms for feature selection and scaling, presenting global minimization and search based methods. Empirical results are discussed in the fourth section and in the fifth section neural realization of the nearest neighbor method is presented. A short summary is presented in the last section.

A framework for the similarity-based methods.

IN THE SBM framework the probability of classification $p(C_i|\mathbf{X};M)$, where $\mathbf{X}$ is a vector of an unknown class and M describes the classification model used (values of all parameters and procedures employed), is determined using the information provided in the similarity measure $D(\mathbf{X},\mathbf{R}^j)$, where $\{\mathbf{R}^j\}, j=1..N_t$ is a set of class-labeled $C(\mathbf{R}^j)$ training vectors. A general model of an adaptive system used for classification should include at least the following elements: $M=\{D(\cdot;r),G(D(\cdot),k),\{\mathbf{R}^j\},\{p(\mathbf{R})\},E[\cdot],K(\cdot)\)$, where:
$D(\cdot;r)$ is a function (usually based on a distance function) or a table used to compute similarities and r is the maximum size of the neighborhood considered;
$G(D(\mathbf{X},\mathbf{R}^j),k)$ is the weighting function estimating contribution of the reference vector $\mathbf{R}^j$ to the classification probability and k is the number of reference vectors taken into account in the neighborhood of $\mathbf{X}$;
$\{\mathbf{R}^j\}$ is the set of reference vectors created from the set of training vectors $\{\mathbf{X}^i\}$ by some computational procedure;
$p(\mathbf{R})$ is the set of class probabilities for each reference vector (confidence in the *a priori* knowledge, useful for clearing the data);
$E[\cdot]$ is the total cost function minimized at the training stage;
$K(\cdot)$ is the kernel function, determining for a given training example the influence of error on the total cost function.

In addition an adaptive system may include several classification models M_l and an interpolation procedure to select between different models or combine results of a committee of models. Various procedures for selection of features, minimization algorithms, architectures used for network computation add more flexibility to the SBM framework. The number of output classes may either be relatively small or it may be a continuos number, in which case classification probabilities $p(C_i|\mathbf{X};M)$ become continuos functions. The cost function is:

$$E(\{\mathbf{X}\};R,M) = \sum_i \sum_{\mathbf{X}} R(C_i,C(\mathbf{X}))H(p(C_i|\mathbf{X};M),S(C_i,C(\mathbf{X}))) \quad (1)$$

where $i=1\ldots N_c$ runs over all classes and $\mathbf{X}$ over all training vectors, $C(\mathbf{X})$ is the true class of the vector $\mathbf{X}$ and function $H(\cdot)$ is monotonic and positive, often a quadratic function. The elements of the risk matrix $R(C_i,C_j)$ are proportional to the risk of assigning the C_i class when the true class is C_j (for example, $R(C_i,C_j)=1-\delta_{ij}$). The similarity function $S(C_i,C_j)$ in the simplest case is equal to δ_{ij}, while for continuos number of classes it may be proportional to $1/(R(C_i,C_j)+1)$, or defined in a problem-dependent way.

M specifies all adaptive parameters and variable procedures of the classification model that may affect the cost function. Regularization terms aimed at minimization of the complexity of the classification model are frequently added to the cost function, helping to avoid the overfitting problems. If $H(\cdot)$ is a quadratic function of

$$H(\mathbf{X};M)=\sum_j \left(\max_i p(C_i|\mathbf{X}^j;M)-\delta(C_i,C(\mathbf{X})\right)^2 \quad (2)$$

standard mean square error function is recovered. To minimize the leave-one-out error the sum runs over all training examples $\mathbf{X}^j$ and the model used to specify the classifier should not contain the $\mathbf{X}^j$ vector in the reference set while $p(C_i|\mathbf{X}^j)$ is computed. In local regression based on the minimal distance approaches [9] the error function is usually

$$E(\mathbf{X};M)=\sum_j K(D(\mathbf{X}^j,\mathbf{R}))(F(\mathbf{X}^j;M)-Y^j)^2 \quad (3)$$

where Y^j are the desired values for $\mathbf{X}^j$ and $F(\mathbf{X}^j;M)$ are the values predicted by the model M. Here the kernel function $K(d)$ measures the influence of the reference vectors $\mathbf{R}$ on the total error. For example, if $K(d)$ has a sharp high peak around $d=0$ the function $F(\mathbf{X};M)$ will fit the values corresponding to the reference input vectors almost exactly and will make large errors for other values. This is not the same as the weighting function $G(d)$ which is used to estimate the influence of reference vectors on the final result. In classification problems kernel function will determine the size of the neighborhood around the known cases in which accurate classification is required.

Distance functions provide natural similarity measures in cases in which we may define them. Calculation of distance is most often based on Euclidean metric for continuos inputs and Hamming metric for binary inputs. Additional parameters that may be optimized are either global (for all data)

or local (for each reference vector). Minkowski's metric involves one global parameter, exponent α. Scaling factors are useful global parameters – for Minkowski's distance:

$$D(\mathbf{X},\mathbf{Y};s)^{\alpha} = \sum_{i}^{N} s_i d(\mathbf{X}_i,\mathbf{Y}_i)^{\alpha} \tag{4}$$

where $d(\cdot)$ function is used to estimate similarity at the feature level and in the simplest case is $|X_i - Y_i|$. If all contributions $|s_i d(X_i,Y_i)|$ for some input feature i are small the feature may be eliminated. To facilitate elimination of features that are not useful for classification the cost function may include additional penalty term, such as the sum of s_i^2 (the weight of the most important feature should be fixed at 1). There are many methods that attempt to determine scaling factors without adaptation (cf. recent review [7]).

Combination of sigmoidal functions offers a good parameterization and easy network realization:

$$d(\mathbf{X}_i,\mathbf{Y}_i) = \sum_{j} \alpha_{ij}\sigma(\beta_{ij}|\mathbf{X}_i - \mathbf{Y}_i| - \gamma_{ij}) \tag{5}$$

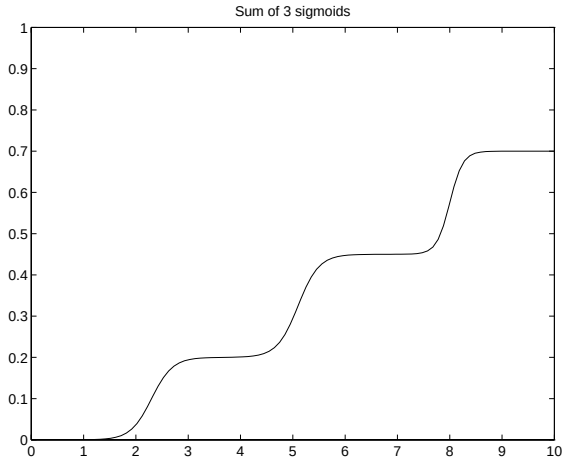

Fig. 1. Sum of 3 sigmoidal functions provides a useful distance function that minimizes in-class and maximizes between class distances.

Such functions allow to minimize in-class and maximize between class distances (as it is done in the Fisher discrimnant analysis, cf. [1]), and may be used in classification models that provide sharp, hyperrectangular decision borders, similar to logical systems [10], [11].

Feature selection and weighting methods.

SIMILARITY based algorithms faced with many features that are not necessary for predicting the desired output may perform quite poorly. We have developed several methods for feature selection and weighting, based on variants of "the best first" search strategy [14]. A simple test for such methods is to add a class number (or a function of a class number) to the list of features. Good selection methods should pick out this feature as the most important, rejecting all other features as superfluous and giving classification accuracy of 100%. This simple test should be used for all feature selection or feature weighting methods. In the description of our algorithms we assume that a training set is used for feature selection or determination of weights. If separate training and test sets are not provided than the training/test sets are obtained from stratified crossvalidation procedure.

The **feature ranking** algorithm proceeds as follows: 1) Parameters are optimized and the accuracy is checked using all features. 2) Accuracy with each feature turned off is determined in crossvalidation tests done on the training part; average change in accuracy and variance of this change for each feature is noted. 3) Ranking of all features is done on the basis of accuracy changes. 4) Accuracy of classification is checked adding features, starting from a single most significant feature, i.e. the one that, when removed, created the largest decrease of accuracy. Hill climbing is used at this stage – only those features that, when added to the growing list of features, contribute to significant increase of accuracy (significance is estimated using the calculated variance for each feature) are accepted on the final list of relevant features. Using crossvalidation for feature ranking makes this model rather robust. For N features not more than $2N$ evaluation steps are required.

The **feature dropping** algorithm is similar although more costly: 1) Parameters are optimized and the accuracy is checked using all features. 2) Features are turned off one after another and the leave-one-out test (for k-NN) or stratified crossvalidation test is performed on the training file to determine the change in classification accuracy. 3) Feature leading to the highest accuracy improvement (after the feature is dropped) on the training file is selected as the least important and removed from the input set. 4) The whole procedure is repeated $N-1$ times until only one feature is left in the input set, presumably the most important one.

Although this algorithm may be used with any method it is particularly easy to use with the k-NN algorithm because computational costs of the leave-one-out test are in this case rather low: apart from the value of k there is no learning and the cost of each step is equal to the cost of classification of all training vectors. If the cross-validation calculations are performed the leave-one-out test is done on the training partition of the dataset. The number of the leave-one-out evaluations in the worst case is $N \cdot (N+1)/2 - 1$. The sequence of features that should be removed to achieve the highest classification accuracy on a training file is selected after all $N-1$ steps are performed. It may happen that at some level there is no improvement and the feature which leads to minimal degradation is selected. If the number of features is large and there is no improvement for several

levels the procedure may be stopped, because there is little chance that results will improve at a later stage. The peak performance is usually observed after all irrelevant features are dropped.

This method is more expensive but allows to take the interactions between different features into account, while the feature ranking algorithm works well only if features are approximately independent of each other. Standardization of the input vectors before the feature selection procedure is usually desired. The feature dropping method is acceptable if the number of features is not too high, otherwise it may be too slow. In such a case the feature ranking method should be used, or the feature dropping selection may be performed using the subset of features with highest rank, for example those that significantly decrease accuracy when dropped. After the final selection of features parameters of the classification model are reoptimized again and results checked on the test set.

Feature selection is the special case of feature weighting, i.e. calculation of optimal weighting factors s_i in the metric function Eq. (4). If the cost function is taken as the number of classification errors (i.e. it is discontinuous) only non-gradient minimization procedures are applicable. We have used the inexpensive simplex method (local procedure), an adaptive simulated annealing (ASA) global method and multisimplex global minimization method [12], [13]. The local simplex method usually requires less than 100 evaluations and is the fastest but results have large variance. It may be used to determine whether optimization of feature weights works for a particular database. For a dataset having separate training file ASA or multisimplex method converge to similar results in all calculations (the advantage of global minimization). However, global optimization methods are expensive and may require a large number of epochs (evaluations of errors for all training vectors) for convergence. They rarely find solutions which are as good as those obtained by feature selection, showing that the minimization problem is really diffcult. Binary weights obtained from feature selection may be used as the starting parameters for minimization methods ensuring that results after weighting are at least as good as from the feature selection.

To decrease computational costs we have used search strategies [14] for feature weighting. Real-valued weights s_i should be quantized, either with fixed precision (for example 0.1) or precision that is steadily increased during the progress of a search procedure. Three search schemes were implemented, aiming at speeding up calculations and finding better and simpler weights. They are called here $S0$, $S1$ and the weight-tuning method. The last algorithm is used to search for optimal weighting factors, starting from a solution obtained by other methods. In each case the weight of the most important feature is arbitrarily fixed at 1 and all other weights changed (they should be positive, at least for distance function Eq. 4).

The $S0$ algorithm is 'a mirror image' of the $S1$ algorithm: in $S0$ features are added, starting from a single one, and in $S1$ features are dropped, starting from all features. $S1$ algorithm usually works better and therefore it is described below. 1) In the initial ranking of the features all weighting factors are set to one. 2) Evaluation with a single feature turned off (weighting factor $s_i = 0$) is made for $i = 1 \ldots N$. Thus the ranking is done in the same way as in the feature dropping selection method. The most important feature has a fixed value of the weighting factor $s_k = 1$ and the optimal weighting factor for the second feature s_l in the ranking is determined by the search procedure. The classifier's performance is evaluated with $s_l = m\Delta \geq 0$ (the default is $\Delta = 0.05$) and the remaining weighting factors are all fixed (initially at 1, later at their optimal values). 3) The search is repeated for feature that has the next-highest rating, with optimal values of weighting factors for higher ranked features kept fixed and lower-ranked features left at 1. 4) After determination of all weighting factors performance of the classifier is evaluated on the test set. The total number of evaluations on the training set is on the order of N/Δ. The method corresponds to quantized version of the line search optimization procedure.

The **weight-tuning method** is used to tune the weighting factors already found by some other procedure (either by a minimization method or one of the search methods). Weighting factors are changed without initial ranking, either sequentially or in a random order. Weights are changed for every feature by adding or subtracting a constant value $s_i \leftarrow s_i \pm \delta$ where δ is a parameter given by the user (by default $\delta = 0.5$). If a change of the weighting factor leads to an improvement of the classification accuracy the factor is updated, otherwise it remains unchanged. After the last weighting factor is checked in this way the δ parameter is divided by 2 and the whole procedure repeated. The algorithm is terminated if the difference in classification accuracy during two subsequent iterations is smaller than a given threshold.

Illustrative results

Memory-based methods, such as k-NN, work well if the number of training vectors is sufficiently large. However, for the well known hypothyroid dataset [15], despite the high number of training cases (3772) results of the k-NN method are quite poor. For non-standardized data the nearest neighbor rule gives results that are below the majority rate! After standardization of data, selection of k and selection of the distance measure, k-NN performs only slightly better than the majority classifier (92.7% test vectors for normal thyroid), giving classification accuracy of 94.4% on the test set (3428 cases). This is much worse than most other algorithms, including neural networks and logical rules [11]. Af-

ter applying the feature dropping algorithm from 21 features present in the original dataset only 4 remain (f3, f8, f17, f21), increasing the classification accuracy on the test set by nearly 3% to 97.9% (a significant improvement since the number of cases in the test set is large). Applying weighting method $S1$ and tuning the weighting factors we were able to increase the classification accuracy further to 98.1%. This is probably the best result for this database obtained so far with the minimal distance method although still significantly worse than the result obtained with logical rules or decision trees [11]. Further improvement of the *k*-NN results is possible using the step-like distance functions Eq. (5), increasing the Minkowsky exponent in Eq. (4) to large values and reducing the number of reference vectors.

Cleveland Heart data was also taken from the UCI [15] repository. 6 of the 303 cases contain missing attributes, therefore they are usually removed in most tests, leaving 297 vectors. There are 13 attributes (4 continuos, 9 nominal), and two classes (healthy or sick), with 164 (54.1%) healthy, and 139 (45.9%) cases with various degrees of heart problems.

Although Wettschereck and Aha [6] use untypical testing procedure (30% of data is used for tests and the rest for training) while we use the standard 10-fold crossvalidation (averaged 10 times to obtain variance) it should be noted that their feature selection methods have always decreased the accuracy of the *k*-NN classification. On the other hand our feature dropping algorithm selected 3 features (thal, ca, cp or features 13, 12 and 3) as the most useful, giving 80.1±0.2% (note the small variance). These are the same features as those used in logical rules [11]. After adding weights to the 3 selected features averaged results are 83.9±0.5%, very close to the best results, 84.2%±0.4% obtained with only a single feature dropped, and quite close to 83.6%±1.0% obtained with all features used (note the large variance for all features).

Even better results were obtained on this dataset with the feature ranking algorithm: although for all features dropping was accompanied with decrease of accuracy for 6 features this decrease was rather small, within the variance ±1% error bounds. Sequentially adding features from the most important (*thal*, giving 4% change) to the least important showed a peak with 7 features used. After re-optimization of *k* and running the stratified 10-fold crossvalidation tests 10 times to obtain estimation of variance an accuracy of 85.1±0.5%, better than obtained with most statistical, machine learning and neural methods [16].

A small medical **appendicitis dataset** [18] contains only 106 vectors, 8 attributes, two classes, and is rather difficult to classify. The feature ranking method has found features 1, 3 and 7 as the most important features, the same as we have selected before using our logical rule extraction methods. Using all features the accuracy of 87.8±1.1% was achieved,

TABLE I
RESULTS FOR THE CLEVELAND HEART DISEASE DATASET.

Method	Accuracy %	Reference
IncNet	90.0	[17]
k-NN, k=28, 7 features	85.1±0.5	this paper
Linear Discriminant Anal.	84.5	[16]
Fisher LDA	84.2	[16]
k-NN, k=16	84.0±0.6	this paper
FSM, Feature Space Mapping	84.0	this paper
Naive Bayes	83.4	[16]
Bayes (pairwise dependent)	83.1	[16]
LVQ	82.9	[16]
k-NN, k=27, Manhattan	82.8±0.6	this paper
MLP+backprop	81.3	[16]
CART (decision tree)	80.8	[16]
Quadratic Discriminant Anal.	75.4	[16]
LFC, ASI, ASR decision trees	74.4-78.4	[16]

while after the selection it increased to 89.3±0.5% in the 10-fold stratified crossvalidation tests. This is the best result for this dataset that we know of, comparable to the best results obtained in the leave-one-out test [18] for this dataset. The data is quite noisy, as can be seen on a Sammon (multidimensional scaling) plot shown in Fig. 2. MLP crossvalidation results are below 84%, RBF are even worse.

Another interesting dataset is the **hepatobiliary disorders** data obtained from Tokyo Dental and Medical University [11] (536 cases, including 163 test cases, 9 features, 4 classes). *k*-NN with Manhattan distance function gives 77.9% accuracy for this dataset which is already much better than other methods [11] (for example MLP trained with RPROP gives accuracy that is below 70%). After applying feature selection method 4 features were removed (features 2, 5, 6 and 9), increasing accuracy to 79.1%. Using weighted *k*-NN methods it was possible to increase the accuracy further to 82.8%. These results are significantly better than for all other classifiers applied to this data (including IB2-IB4, FOIL, LDA, DLVQ, C4.5, FSM, Fuzzy MLP and K* methods). This data has also been analyzed by Mitra, De and Pal [19] using a knowledge-based fuzzy MLP system with results on the test set in the range from 33% to 66.3%, depending on the actual fuzzy model used.

Conclusions

SIMILARITY based methods require selection of features and determination of the feature weight factors for good performance. Several methods for feature selection and weighting have been introduced here. Search-based techniques were used instead of minimization both for feature selection and weighting. Preliminary empirical tests described

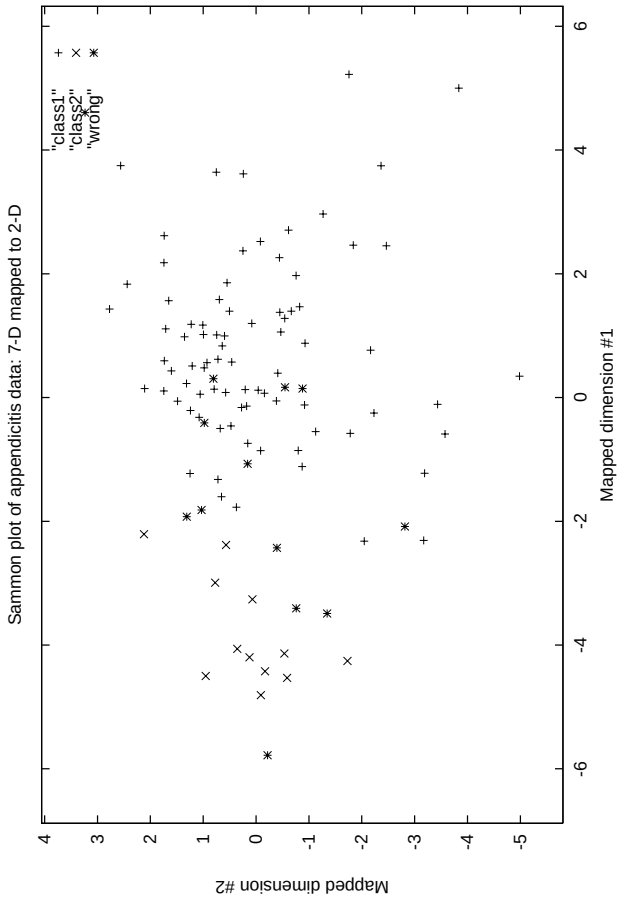

Fig. 2. Appendicitis - the best 2-dimensional representation of 7-dimensional vectors preserving their relative distances.

in the previous section show that using the straightforward *k*-NN method significant improvements of results obtained with reduced feature set or with weighted features are obtained. Other tests that we have performed indicate that for more than half of the *Statlog* datasets [1] feature selection and weighting makes *k*-NN results better than that of any other classifiers used in this project. The same feature selection and weighting methods may be used to improve performance of any similarity-based classifier.

A natural network realization of *k*-NN method introduced very recently [20] leads to a classification model with more adaptive parameters and should allow to improve the results even further. It is applicable to problems with an infinite number of output classes and may take into account costs of misclassifications. We have implemented many features of the general framework for SBM, such as a variety of distance functions, selection of reference vectors and weighting of their contributions. However, even if such more sophisticated similarity-based methods are used performing feature selection and weighting at the level of simple *k*-NN method may be the only practical solution due to the computational efficiency.

Acknowledgments

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

References

[1] D. Michie, D.J. Spiegelhalter and C.C. Taylor, *Machine learning, neural and statistical classification*. Elis Horwood, London 1994

[2] R. Rohwer and M. Morciniec, *A Theoretical and Experimental Account of n-tuple Classifier Performance*, Neural Computation 8 (1996) 657–670

[3] W. Duch, *Neural minimal distance methods*, Proc. 3-rd Conf. on Neural Networks and Their Applications, Kule, Poland, Oct. 14-18, 1997; W. Duch, K. Grudziński, *A framework for similarity-based methods.* Second Polish Conference on Theory and Applications of Artificial Intelligence, Łódź, 28-30 Sept. 1998, pp. 33-60;

[4] W. Duch, K. Grudziński, G.H.F. Diercksen, *Minimal distance neural methods.* Proc. of IJCNN'98, pp. 1299-1304.

[5] W. Duch, R. Adamczak, G.H.F. Diercksen, *Neural Networks in non-Euclidean spaces.* Neural Processing Letters (in print)

[6] D. Wettschereck and D.W. Aha, *Weighting Features*. In: 1st Int. conf. on Case-based Reasoning (ICCBR-95), Lisbon, Portugal: Springer-Verlag; D. Wettschereck, D.W. Aha and T. Mohri, *A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms*, Artificial Intelligence Review 11 (1997) 273-314

[7] D.W. Aha, *Feature weighting for lazy learning algorithms*. In: H. Liu and H. Motoda (Eds.) Feature Extraction, Construction and Selection: A Data Mining Perspective. Norwell MA: Kluwer, 1998.

[8] W. Duch, *A framework for similarity-based classification methods*, Intelligent Information Systems VII, Malbork, Poland, 15-19.06.1998, pp. 288-291

[9] C.G. Atkenson, A.W. Moor and S. Schaal, *Locally weighted learning*, Artificial Intelligence Review (submitted)

[10] W. Duch, R. Adamczak, K. Grąbczewski, G. al, *Hybrid neural-global minimization method of logical rule extraction.* Journal of Advanced Computational Intelligence (in print)

[11] W. Duch, R. Adamczak, K. Grąbczewski, G. al, Y. Hayashi, *Fuzzy and crisp logical rule extraction methods in application to medical data.* Fuzzy Systems in Medicine, Springer 1999 (in print)

[12] L. Ingberg, *Adaptive simulated annealing (ASA): Lessons learned*, J. Control and Cybernetics 25 (1996) 33-54

[13] H.V. Gupta, K. Hsu, S. Sorooshian, *Superior training of artificial neural networks using weight-space partitioning*, Proc. of ICNN'97, Houston, June 1997, pp. 1919-1923

[14] L. Kanal, V. Kumar (Eds), Search in Artificial Intelligence (Springer Verlag 1988)

[15] C.J. Mertz, P.M. Murphy, UCI repository, http://www.ics.uci.edu/pub/machine-learning-databases.

[16] B. Ster and A. Dobnikar, Neural networks in medical diagnosis: Comparison with other methods. In: A. Bulsari et al., eds, Proc. Int. Conf. EANN'96, pp. 427-430, 1996.

[17] N. Jankowski N and V. Kadirkamanathan, Statistical control of RBF-like networks for classification. In: 7th Int. Conf. on Artificial Neural Networks (ICANN'97), Lausanne, Switzerland, 1997, pp. 385–390

[18] S.M. Weiss, I. Kapouleas, *An empirical comparison of pattern recognition, neural nets and machine learning classification methods.* In: J.W. Shavlik and T.G. Dietterich, *Readings in Machine Learning*, Morgan Kauffman Publ, CA 1990

[19] Mitra S, De R, Pal S. *Knowledge based fuzzy MLP for classification and rule generation*, IEEE Transactions on Neural Networks 8 (1997) 1338-1350

[20] W. Duch, K. Grudziński, *The weighted k-NN with selection of features and its neural realization.*, Proc. 4-th Conf. on Neural Networks and Their Applications, Zakopane, Poland, May 18-22, 1999 (in print)