Minimal distance neural methods.

Włodzisław Duch[a], Karol Grudziński[a] and Geerd H.F. Diercksen[b]
[a]Department of Computer Methods, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland; e-mail: duch,kagru@phys.uni.torun.pl
[b]Max-Planck Institute of Astrophysics, 85740-Garching, Germany,
e-mail: GDiercksen@mpa-garching.mpg.de

Abstract— **A general framework for minimal distance methods is presented. Radial Basis Functions (RBFs) and Multilayer Perceptrons (MLPs) neural networks are included in this framework as special cases. New versions of minimal distance methods are formulated. A few of them have been tested on a real-world datasets obtaining very encouraging results.**

I. Introduction

Classification is one of the most important applications of neural systems. The accuracy of 24 neural-based, pattern recognition and statistical classification systems has been compared recently on 11 large datasets by Rhower and Morciniec [1]. No consistent trends have been observed in the results of this large-scale study. For each classifier one may find a real-world dataset for which the results will be excellent and another one for which the results will be quite bad. In real world applications a good strategy is to find the best classifier that works for given data. Frequently simple methods, such as the nearest neighbor methods or n-tuple methods [1], are among the best. These methods have natural neural realizations. Recently one of us [2] has presented a general framework for minimal distance (MD) methods. This framework is extended here and results of tests for a few new versions of the MD methods are presented.

Some of the simplest classification algorithms applicable to pattern recognition problems are based on the k-nearest neighbor (k-NN) rule [3]. This approach is so important that in artificial intelligence it is referred to as the instance based learning, memory based learning or case based learning. Each training data vector is labeled by the class it belongs to and is treated as a reference vector. During classification k nearest reference vectors to the unknown (query) vector $\mathbf{X}$ are found, and the class of vector $\mathbf{X}$ is determined by a 'majority rule'. The probability of assigning a vector $\mathbf{X}$ to class C_i is $p(C_i|\mathbf{X}) = N_i/k$. In the simplest case of $k = 1$ only the nearest neighbor determines the class of an unknown vector, i.e. $p(C_i|\mathbf{X})$ =0 or 1. The asymptotic error rate of the k-NN classifier in the limit of large k and a large number of reference vectors becomes equal to the optimal Bayesian values [3]. In practice the number of reference vectors is relatively small and small values of k work better.

Because the k-NN method is so simple it is frequently used as a standard reference for other classificators. One problem is the computational complexity of the actual classification, demanding for n reference vectors calculation of $\sim n^2$ distances and finding the k smallest distances. Although Laaksonen and Oja [4] claim that "For realistic pattern space dimensions, it is hard to find any variation of the rule that would be significantly lighter than the brute force method" various hierarchical schemes of partitioning the data space or hierarchical clusterization schemes are quite effective. The search for the nearest neighbors is easily paralelizable and the training time (selection of optimal k) is relatively short. Nearest neighbor methods are especially suitable for complex applications, where large training datasets are available. They are also used in case-based expert systems, an alternative to the rule-based systems (cf. [5]).

Only one neural model proposed so far is explicitly based on the nearest neighbor rule: the Hamming network [6] computes the Hamming distances for binary patterns and finds the maximum overlap (minimum distance) with the prototype vectors, thus realizing the 1-NN rule. Although other minimal distance methods presented here have natural neural-network type realizations we will concentrate more on the presentation of general framework and on the testing of specific methods derived from this framework rather than on the network implementation issues, since at this initial stage the implementation is of secondary importance. In the next section the general framework for MD methods is presented and relations between neural and MD methods explained. In the third section results of tests on real datasets are given. A short discussion is presented in the last section.

II. A framework for minimal distance methods

The problem of classification is stated as follows: given a set of class-labeled training vectors $\{\mathbf{X}^p, \mathbf{C}(\mathbf{X}^p)\}, p = 1..K$, where $\mathbf{C}(\mathbf{X}^p)$ is the class of $\mathbf{X}^p$, and a vector $\mathbf{X}$ of an unknown class, use the information provided in the distance $d(\mathbf{X}, \mathbf{X}^p)$ to estimate the probability of classification $p(C_i|\mathbf{X}; M)$, where M describes the classification model used (parameter values and procedures employed). A general model of an adaptive system used for classification may include all or some of the following:
$M = \{k, d(\cdot; r), G(d(\cdot)), \{\mathbf{D}^n\}, E[\cdot], K(\cdot))$, where

k is the number of reference vectors taken into account in the neighborhood of $\mathbf{X}$;
$d(\cdot; r)$ is the metric function used to compute distances;
r is the maximum size of the neighborhood considered;
$G(d(\mathbf{X}, \mathbf{X}^p))$ is the weighting function estimating contribution of the reference vector $\mathbf{X}^p$ to the classification probability;
$\{\mathbf{D}^n\}$ is the set of reference vectors created from the set of training vectors $\{\mathbf{X}^p\}$ by some procedure;
$E[\cdot]$ is the total cost function minimized at the training stage;
$K(\cdot)$ is a kernel function, scaling the influence of the error, for a given training example, on the total cost function.
Various selections of parameters and procedures in the context of network computations lead to different versions of MD neural methods.

Before analyzing the details of this general framework the relations between neural networks and MD methods are explained. Threshold neurons compute distances in a natural way. If the input signals $\{X_i = \pm 1\}$ and the weights $\{W_i = \pm 1\}$ are binary, neurons with N inputs and the threshold θ realize the following function:

$$\Theta(\sum_{i}^{N} W_i X_i - \theta) = \begin{cases} 0 & \text{if } ||\mathbf{W} - \mathbf{X}|| > \theta \\ 1 & \text{if } ||\mathbf{W} - \mathbf{X}|| \leq \theta \end{cases} \quad (1)$$

where the norm $||\cdot||$ is defined by the Hamming distance. One can interpret the weights of neurons in the first hidden layer as addresses of the reference vectors in the input space. Threshold neurons are activated by inputs falling into a hard sphere of radius θ centered at $\mathbf{W}$. By changing the binary into real values and the threshold into sigmoidal transfer functions, for inputs and weights normalized to $||\mathbf{X}|| = ||\mathbf{W}|| = 1$, a soft activation of neurons by input vectors that are close to $\mathbf{W}$ on the unit sphere is realized. In general the output function of neurons is:

$$\begin{aligned} \sigma(\mathbf{W} \cdot \mathbf{X}) &= \sigma\left(\frac{1}{2}(||\mathbf{W}||^2 + ||\mathbf{X}||^2 - ||\mathbf{W} - \mathbf{X}||^2)\right) = \\ & \sigma(I_{max} - d(\mathbf{W}, \mathbf{X})) \end{aligned} \quad (2)$$

For normalized input vectors sigmoidal functions (or any other monotonically growing transfer functions) simply evaluate the influence of the reference vectors $\mathbf{W}$ depending on their distance $d(\mathbf{W}, \mathbf{X})$, on the classification probability $p(C_i|\mathbf{X}; \{\mathbf{W}, \theta\})$. As a function of a distance the output function $\sigma(I_{max} - d(\mathbf{W}, \mathbf{X}))$ monotonically decreases, reaching the value of 0.5 at $d(\mathbf{W}, \mathbf{X}) = I_{max}$. For normalized weights and inputs $\sigma(1 - d(\mathbf{W}, \mathbf{X})) \in [0.5, \sigma(1)]$, with $0 \leq d(\mathbf{W}, \mathbf{X})) \leq 1$, i.e. only a part of the range of the sigmoidal function is used. For normalized $\mathbf{X}$ but arbitrary $\mathbf{W}$ the range of the sigmoid argument lies in the $[-|\mathbf{W}|, +|\mathbf{W}|]$ interval. A unipolar sigmoid has a maximum curvature around ± 2.4, therefore smaller weights of the norm mean that the network operates in an almost linear regime. Regularization methods add penalty terms to the error function forcing the weights to become small and thus smoothing the network approximation to the training data.

In MLP networks sigmoidal functions are used to estimate the influence of weight vectors according to the distance between weight and training vectors, combining many such estimations to compute the final output. In RBF networks Euclidean distance functions $d(\mathbf{X}, \mathbf{D}^n) = ||\mathbf{X} - \mathbf{D}^n||$ and exponential $\exp(-d^2)$ weighting functions are used. By changing the distance function in equation (2) new types of neural networks may be defined. The interpretation of neural classifiers as a special MD adaptive systems is worth investigating. Combining different weighting and distance functions with additional parameters a large number of possible classification systems is obtained. Several variants of the MD methods are presented below.

k-NN networks.

In the simplest version $p(C_i|\mathbf{X}; M)$ is parameterized by $p(C_i|\mathbf{X}; k, d(\cdot), \{\mathbf{X}^n\})$, leading to the k-NN method in which the whole training dataset is used as the reference set. The distance function $d(\mathbf{X}, \mathbf{X}^n)$ is a hard-sphere metric function with a variable radius such that exactly k neighboring vectors $\mathbf{X}^n$ fall inside the sphere. The type of the metric function $d(\cdot)$ and k are the only parameters that should be optimized. For k=2 the training vector near the class border may have nearest vectors from two different classes. The error on the training set, equal to zero for $k = 1$, grows for $k > 1$ but may decrease for larger values of k. The leave-one-out test is recommended to optimize k using the training set only. This test is easy to perform in the k-NN method since there is no learning phase, unless the metric function is parameterized. For the L-class problem selecting $k = 1, L + 1, 2L + 1, ...$ avoids the ties but is a severe restriction on the value of k. Ties may be resolved either by: a) rejecting cases in which ties occur, b) adding one or more extra neighboring vectors until the tie is broken, c) decreasing the number of neighboring vectors d) randomly braking the tie and e) giving probabilities instead of yes-no decisions. Details of the k-NN procedure are rarely given in practice and it is not clear which of the five possibilities is used. In our experience adding more vectors to break a tie is preferable although differences in classification accuracy are sometimes negligible.

Neural realization of the 1-NN rule for binary patterns is afforded by the Hamming network [6]. This network is significantly simplified if more complex output network nodes are allowed. For normalized vectors the output unit should determine from which hidden node the maximum input is received and transfer the class label of this node to the output.

r-NN algorithm.

Instead of enforcing exactly k neighbors the radius r may be used as an adaptive parameter. The number

of classification errors, or the probability of classification $p(C_i|\mathbf{X};r) = N_i/\sum_j N_j$, is then optimized using the leave-one-out method or a validation set. The hard sphere transfer functions should be used in the network realization of this algorithm. r-NN may reject some vectors $\mathbf{X}$ if no reference vectors fall into the r radius of $\mathbf{X}$ or if equal probability of classification for several classes is obtained. To avoid such problems r is increased until a unique classification is done.

Introduction of variable radiuses r_i for each reference vector instead of one universal radius in the input space improves the method further increasing the number of adaptive parameters significantly. Development along this line leads to the Restricted Coulomb Energy (RCE) classifier introduced by Reilly, Cooper and Elbaum [7] which may be treated as the hard limit approximation of the Gaussian-based RBF network. If no neighbors are found around the training vector $\mathbf{X}$ new spheres (virtual reference vectors) are added, with largest radius such that the new sphere does not overlap with the spheres of other classes. If the new training vector falls into the range of a sphere of a wrong class the radius of this sphere is shrinked to leave the vector outside of the sphere. Positions of the spheres are not optimized in the RCE algorithm (this would lead in the direction of LVQ algorithms [4]), but voting methods for the committees of classifiers were used with success [8]. The number of radiuses r_i may be reduced by using only a few independent values in selected input space areas. One could also optimize components of one radius (i.e. not just a total distance but separate distances for individual input features), but this would give the same result as optimization of the metric function described below. To reduce the number of parameters variable radiuses should be attached only to the centers of clusters. To assure smooth transition between different regions of the input space interpolation of the r values from the nearest cluster centers is recommended.

Soft weighting the k-NN and r-NN algorithms.

Nearest reference vectors should influence probabilities of classification more strongly than those laying further. Changing the hard sphere transfer functions into smoother functions allows to include such weights. The favorite fuzzy logic membership function is **the conical radial function**: it has zero value outside the radius r and grows linearly to 1 inside this radius. Classification probability is calculated by the output node using the formula:

$$\begin{aligned} p(C_i|\mathbf{X};r) &= \frac{\sum_{n\in C_i} G(\mathbf{X};\mathbf{D}^n,r)}{\sum_n G(\mathbf{X};\mathbf{D}^n,r)}; \\ G(\mathbf{X};\mathbf{D},r) &= \max\left(0, 1-\frac{d(\mathbf{X},\mathbf{D})}{r}\right) \end{aligned} \tag{3}$$

Here $G(\mathbf{X};\mathbf{D},r)$ plays the role of a weight estimating contribution of reference vector placed at distance $d(\mathbf{X},\mathbf{D})$. In the soft r-NN algorithm the r parameter is optimized. Radial Basis Function (RBF) networks using Gaussian or inverse multiquadratic transfer functions are a particular example of the soft weighting MD algorithm. Other useful weighting functions include the combination of two sigmoidal functions: $\sigma(||\mathbf{X}-\mathbf{D}^n|| - r) - \sigma(||\mathbf{X}-\mathbf{D}^n|| + r)$. Variable r equal to the distance to the k-th neighbor may be used as the weighting function for the vectors inside this radius. If r_k is the distance to the k-th neighbor and $r_k \geq r_i, i = 1..k-1$ then a conical weighting function $G(d) = 1 - d/\alpha r_k, \alpha > 1$ has values $G(0) = 1$ and $G(r_k) = 1 - 1/\alpha$; for large α the cone is very broad and all vectors receive the same attention; for α approaching 1 the furthest neighbor has weight approaching zero. MD classifier with optimized α can not be less accurate than k-NN. The effect of weighting is more pronounced for larger k values.

Parameterization of distance measures

Calculation of distance is most often based on Euclidean metric for continuos inputs and on Hamming metric for binary inputs. Additional parameters that may be optimized are either global (the same in all input space) or local (different for each reference vector). Minkowski's metric involves one parameter α. Scaling factors are very useful parameters – for Minkowski's distance:

$$d(\mathbf{A},\mathbf{B};g)^\alpha = \sum_i^N s_i|A_i - B_i|^\alpha \tag{4}$$

Euclidean metric corresponds to $\alpha = 2$, which is completly isotropic, and Manhattan metric to $\alpha = 1$, which is less sensitive in directions along the axis than to the directions between the axis. In the simplest RBF version with Gaussian functions only one parameter – dispersion – is optimized. Independent optimization of all N dispersion components has the same effect as optimization of the scaling factors s_i in soft-weighted r-NN method with Euclidean metric. Scaling is the simplest way of pre-processing the attributes. Mahalanobis distance is obtained by applying a particular linear transformation to the input vectors. Alternatively, a metric tensor $G_{ij} = G_{ji}$ is introduced, providing $N(N+1)/2$ adaptive parameters:

$$d(\mathbf{A},\mathbf{B};\mathbf{G})^2 = \sum_{i,j}^N G_{ij}(A_i - B_i)(A_j - B_j) \tag{5}$$

Calculation of distances may also be parameterized in a different way around each reference vector, providing a large number of adaptive parameters. Local coordinate systems with their origin placed at the reference vectors may provide either local scaling factors or local metric tensors. A simple way to select features useful for classification and at the same time to lower the complexity of the classification model is to add to the cost function an additional penalty term, such as the sum of all s_i^2 or

G^2_{ij}. Features for which the product of the scaling factors $s_i \max_{jk} |X_i^{(j)} - X_i^{(k)}|$ is small may be deleted without significant loss of accuracy – after additional optimization of the scaling factors accuracy may even increase.

In memory-based reasoning the Modified Value Difference Metric (MVDM) has gained popularity [5]. The distance between two N-dimensional vectors $\mathbf{A}, \mathbf{B}$ with discrete (for example symbolic) elements, in a K class problem, is computed using conditional probabilities:

$$d(\mathbf{A}, \mathbf{B}) = \sum_j^N \sum_i^K (p(C_i|A_j) - p(C_i|B_j)) \qquad (6)$$

where $p(C_i|A_j)$ is estimated by calculating the number $N_i(A_j)$ of times feature A_j occurred in vectors belonging to class C_i, and dividing it by the number of times feature A_j occurred for any class. We can also define a "value difference" for each feature j as $d_v(A_j, B_j) = \sum_i^K (p(C_i|A_j) - p(C_i|B_j))$ and compute $d(\mathbf{A}, \mathbf{B})$ as a sum of value differences over all features. Metric is defined here via a data-dependent matrix with the number of rows equal to the number of classes and the number of columns equal to the number of all attributes. Generalization for continuos values requires a set of probability density functions $p_{ij}(x)$, with $i = 1..K, j = 1..N$ or some discretization procedure.

Any adaptive system may provide a distance function for MD methods. For example [9], a typical MLP network may be trained on the differences of pairs of vectors $\{A_i - B_i\}$, giving at the output the distance between the classes $||C(\mathbf{A}) - C(\mathbf{B})||$. The output of the neural network is then used in k-NN or other MD procedure. A better way is to input to MLP both A and $\mathbf{A} - \mathbf{B} = \{d^i(A_i) - d^i(B_i)\}$ vectors, where $d^i(\cdot)$ is a set of attribute pre-processing functions (for example, scaling factors). A similarity function (it does not have all properties required from the distance function, for example it is not symmetric) $d(\mathbf{A} - \mathbf{B}; \mathbf{A})$, smoothly changing between different regions of the input space, is obtained iteratively: for each training vector k nearest neighbors are selected using initial similarity estimation, and after the first epoch the process is repeated using the new similarity function. In general one should create a distance function that minimizes in-class distances and maximizes between-class distances.

Active selection of reference vectors.

In MD method one should avoid large number of reference vectors. Reducing the number of vectors in the reference set leads to models of lower complexity and helps to improve generalization capabilities of the classification system. K-means, dendrograms or other clusterization techniques are used to select a relatively small number of initial reference vectors close to the cluster centers. Classification accuracy is checked on the remaining set (using k-NN or r-NN algorithm) and each wrongly classified vector is moved from the training to the reference set. Variants of this approach my use a validation set to determine best candidates for the reference set.

An alternative approach that does not require initial clusterization starts from the whole training set and removes those vectors that have all k nearest vectors from the same class. These vectors are far from cluster borders and all new vectors in their neighborhood will be anyway unambiguously classified. This approach leads to a "hollow" cluster representation. Here one may start with a large number of neighbors k' to remove vectors near the centers of clusters first and reduce it to k in a few steps. An interesting algorithm to select good reference vectors is: run over all vectors $\mathbf{X}$, determine k nearest vectors from each class different than $C(\mathbf{X})$ and move these vectors to the reference set. This algorithm leaves only vectors near the class borders.

Parameterization of reference vectors

Active selection of reference vectors may leave only a small subset of training vectors. Further optimization of their positions should decrease the training error. The reference vector $\mathbf{D}$ in the neighborhood of the training vector $\mathbf{X}$ should be updated as follows:

$$\mathbf{D} \leftarrow \mathbf{D} + \eta \delta_{\pm}(C(\mathbf{X}), C(\mathbf{D}))(\mathbf{X} - \mathbf{D}) \qquad (7)$$

Here η is the learning rate, slowly decreasing during training, and $\delta_{\pm}$ is $+1$ if the class $C(\mathbf{X}) = C(\mathbf{D})$ or -1 otherwise. Various rules for moving centers $\mathbf{D}$ are used: moving only the nearest neighbor, moving all k neighbors by the same amount, using distance-dependent η etc. [4]. One can also optimize a subset of vectors, for example only those that are close to the center of clusters.

Virtual Support Vectors (VSV) may be added to the reference set to improve classification rates. The simplest approach is to interpolate between existing training vectors and add VSV between neighboring vectors belonging to different classes. In cases when data clusters belonging to different classes are well separated VSV should help to shift decision borders improving generalization. A minimum threshold value for the distance between vectors of different classes is used to prevent creation of VSV for overlapping vector distributions.

Selection of the kernel function and the error function The choice of kernel function in the measure of classification error should also be discussed. In local regression based on the MD approaches [10] the error function is simply

$$E(\mathbf{X}; M) = \sum_p K(d(\mathbf{X}^p, \mathbf{X}^{ref}))(F(\mathbf{X}^p; M) - y^p)^2 \qquad (8)$$

where y^i are the desired values for $\mathbf{X}^i$ and $F(\mathbf{X}^i; M)$ are the values predicted by the model M; here the kernel

function $K(d)$ measures the influence of the reference vectors 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 quite the same as the weighting function $G(d)$ which is used to estimate the distance. In classification problems kernel function will determine the size of the neighborhood around the known cases in which accurate classification is required.

The cost function is either a classification error (as for the hard-distance case) or – since continuos output values are provided – minimization of risk for overall classification:

$$E_R(\mathbf{X}; M) = \sum_{i,p} R(C_i, C(\mathbf{X}^p)) \left[p(C_i|\mathbf{X}^p; M) - \delta(C_i, C(\mathbf{X}^p)) \right]^2 \tag{9}$$

where the sum runs over all training vectors $\mathbf{X}$, $C(\mathbf{X})$ is the true class of vector $\mathbf{X}^p$, $R(C_i, C_j)$ is the risk matrix, and M specifies parameters of the classifier. To minimize the leave-one-out error the sum runs over all training examples $\mathbf{X}^p$ and the model used to specify the classifier should not contain the $\mathbf{X}^p$ vector in the reference set while $p(C_i|\mathbf{X}^p)$ is computed.

III. Results

The framework for the minimal distance methods described in the previous section leads to many different methods of classification. The k-NN approach is frequently one of the best among many popular statistical, neural and machine learning classification methods. Most MD methods presented here should improve the k-NN results. We have tested so far only a few simplest choices on the real and artificial data. k-NN results reported here are based on increasing the number of neighbors until a tie is broken. Manhattan distance function was used in all studies ($\sum_i |X_i - Y_i|$) although in some cases optimized Minkovsky metric gives better results. Both raw and standardized data (zero mean, unit variance) were used, since standardization does not always lead to improvement. The optimal value of k was found first and then adaptive simmulated annealing [11] or a multistart simplex minimization method was used to find the scaling factors and other parameters.

Most of the data were taken from the UCI repository [12] of the Machine Learning Databases, where more detailed description may be found. Medical data include appendicitis (106 vectors, 8 attributes, two classes, obtained from S. Weiss), Wisconsin breast cancer (699 cases, 9 attributes, two classes), Cleveland heart disease (303 cases, 13 attributes, two classes), hepatitis (155 vectors, 19 attributes, two classes) and a larger hypothyroid dataset (described in more details below). All medical datasets strongly benefited from normalization. The missing values of features have been replaced by the mean for their class (this is not always the best procedure).

Detailed comparison of the scaled k-NN data with other approaches shows that even for small datasets results belong to the best reported so far. PVM, Cart, MLP and Bayes results are taken from [13], C-MLP2LN from [14], RIAC and C4.5 from [15] and FSM [16] are our own. Unfortunately we have only a few results of the the leave-one-out tests to compare with. k-NN result for appendicitis is very good and has been significantly improved by scaling using ASA minimization, while the conical-weighted r-NN result, 85.9% is at the MLP level. For the Wisconsin breast cancer data the best results were obtained by FSM, with scaled k-NN results only slightly worse. Tests using 10-fold crossvalidation give accuracies worse by as much as 1.5%. For the hepatitis data scaled results with ASA optimization are excellent. Scaled results for the Cleveland heart disease data are also quite good, although the much better result obtained by FSM shows that more than one distance function should be used in this case (FSM uses clusters of different sizes in different regions of the input space).

The hypothyroid dataset was created from real medical tests screening for hypothyroid problems [12]. Since most people were healthy 92.5% of cases belong to the normal group, and 7.5% of cases belonging to the primary hypothyroid or compensated hypothyroid group. A total of 3772 cases are given for training (results from one year) and 3428 cases for testing (results from the next year). Many neural classifiers have been tried on this data, giving accuracies up to 98.5% on the test set. Schiffman et.al. [17] optimized about 15 MLPs trained with different variants of backpropagation and cascade correlation algorithms and performed tedious genetic optimizations on many MLP network architectures. The best results of this study [17] (genetic optimization, local adaptation rates) reach 98.5% accuracy on the test set, while logical rules derived using C-MLP2LN method [14] reach 99.36% accuracy. The best k-NN results give only 97.0% and were obtained for $k = 6$ with scaled metric optimized using multisimplex method (ASA is more reliable but more expensive). Since the number of reference vectors is rather large these results are rather disappointing. We have tried to use soft weighting with the conical function and optimized radius but the results were not significantly better. The failure of the minimum distance method with global parameters for this case is surprising and requires further study.

Non-medical datasets included the ionosphere (350 cases, 34 attributes, 2 classes), satimage (4435 cases, 36 attributes, 6 classes), sonar (208 cases, 60 attributes, 2 classes) and vovel (528 training and 462 test cases, 10 attributes, 11 classes) data. For this data scaling factors were optimized with multisimplex method only. For the ionosphere we have the RIAC accuracy of 94.6% and C4.5 of 94.9% which are significantly worse than our results. For satimage slightly better results (90.5%) were obtained from conical soft-weighting of the number of neighbors and for vowel the r-NN method gives 57.8% accuracy, but in both

TABLE I

The appendicitis, Wisconsin breast cancer data, hepatitis and the Cleveland heart data.

Dataset and method	Leave-one-out %
The appendicitis data	
Bayes rule (statistical)	83.0
CART, C4.5 (dec. trees)	84.9
MLP+backpropagation	85.8
RIAC (prob. inductive)	86.9
9-NN	89.6
PVM, C-MLP2LN (logical rules)	89.6
1-NN + ASA	94.3
Wisconsin breast cancer data	
RIAC	95.0
C4.5 (dec. tree)	96.0
3-NN	97.0
scaled 4-NN + ASA	97.9
FSM	98.3
The hepatitis data	
MLP+backprop	82.1
CART	82.7
LDA	86.4
9-NN	90.3
FSM	93.6
Scaled 1-NN + ASA	98.1
The Cleveland heart data	
CART	80.8
MLP+backprop	81.3
C-MLP2LN	82.5
7-NN	83.2
LDA	84.5
Scaled 4-NN + ASA	88.1
FSM	96.3

cases scaling improves the accuracy even more (of course one can combine scaling with soft-weighting and r-NN). We have also tried the artificial data using the three monk problems. For the scaled 3-NN method in the first Monk problem 100% accuracy was obtained, for the second problem 85.9% (in this case MLP obtain 100%) and for the third problem 97.7%, which is optimal in this case (the error is due to some noise added to the data).

IV. Summary and discussion

General conclusions that one may draw from these preliminary results are: scaling of individual features is very important and can bring substantial gains in accuracy as well as reduce the number of features (small scaling factors). Selection of a fixed number of neighbors works usually better than optimization of one radius in which the number of neighbors is counted. If the optimal number of neighbors is small weighting procedures do not contribute significantly to accuracy. Better results are probably achieved if local weighting functions are introduced, similarly as in the RBF, where adaptation of individual dispersions is of great importance, or if the α-optimized soft weighting is performed.

TABLE II

Results for non-medical datasets.

database	**k**	k-**NN %**	**scaled** k-**NN %**
ionosphere	3	96.7	98.7
satimage	2	90.3	91.4
sonar	1	93.3	95.2
vowel	9	56.5	62.1

The minimal distance point of view leads to a fruitful framework in which many methods are accommodated. We have found very few methods in the literature that try to improve upon the simple k-NN scheme. Hastie and Tibshirani [18] write about adaptive k-NN classification from the linear discriminant point of view, advocating the use of several local metrics in different areas of the input space, instead of just one. Friedman [19] proposed an interesting way of adapting the metric based on a tree-structure interactive partitioning of the data. Laaksonen and Oja [4] proposed to improve the k-NN reference vectors using LVQ techniques. Atkenson, Moor and Schaal [10] discuss locally weighted regression techniques, minimal distance methods with various metric and kernel functions applied to approximation problems.

All these proposals may be accommodated in the general framework presented here. Both MLP and RBF networks may be seen as particular examples of neural MD methods. Identification of the best combination of procedures and adaptive parameters should allow for improvement of results achieved by the k-NN as well as neural classifiers. Many possibilities to create fuzzy k-NN models remain also to be explored. Performance of various methods described here (as well as any other classification methods) depends on the nature of the data given for classification and remains a subject of further empirical study. Our preliminary results for most of the datasets tried are the best obtained so far by any method. Bearing in mind that so far we have tested only a few simplest methods our results, even for small datasets, are very encouraging.

Acknowledgments: Support of W.D. and K.G. by the Polish Committee for Scientific Research, grant 8T11F 00308, is gratefully acknowledged.

References

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

[2] W. Duch, Neural minimal distance methods, Proc. 3-rd Conf. on Neural Networks and Their Applications, Kule, Poland, Oct. 14-18, 1997

[3] P.R. Krishnaiah, L.N. Kanal, eds, Handbook of statistics 2: classification, pattern recognition and reduction of dimensionality (North Holland, Amsterdam 1982)

[4] J. Laaksonen, E. Oja, Classification with Learning k-Nearest Neighbors. In: *Proc. of ICNN'96*, Washington, D.C., June 1996, pp. 1480-1483.
[5] D.L. Waltz, Memory-based reasoning, in: M. A. Arbib, ed, *The Handbook of Brain Theory and Neural Networks* (MIT Press 1995), pp. 568–570
[6] R.P. Lippmann, An introduction to computing with neural nets, *IEEE Magazine on Acoustics, Signal and Speech Processing* 4 (1987) 4–22; P. Floreen, The convergence of Hamming memory networks, *Trans. Neural Networks* 2 (1991) 449–457
[7] D.L. Reilly, L.N. Cooper, C. Elbaum, A neural model for category learning, *Biological Cybernetics* 45 (1982) 35–41
[8] P.D. Wasserman, Advanced methods in neural networks (van Nostrand Reinhold 1993)
[9] D.K.Y. Chiu, F.E. Kavanaugh, the ck-Nearest Neighbor distance Network: a network using class boundary feature distances, ICONIP'97, New Zealand, Nov.1997, pp. 535-538
[10] C.G. Atkenson, A.W. Moor and S. Schaal, Locally weighted learning, *Artificial Intelligence Review* (submitted, 1997)
[11] L. Ingberg, *Adaptive simulated annealing (ASA): Lessons learned,* J. Control and Cybernetics 25 (1996) 33-54
[12] C.J. Mertz, P.M. Murphy, UCI repository, http://www.ics.uci.edu/pub/machine-learning-databases.
[13] 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
[14] W. Duch, R. Adamczak, K. Grąbczewski, Extraction of crisp logical rules using constrained backpropagation networks. *ICANN'97*, Houston, 9-12.6.1997, pp. 2384-2389, Logical rules for classification of medical data using ontogenic neural algorithm, *EANN'97*, Stockholm, 16-18.06.1997, pp. 199-202
[15] H.J. Hamilton, N. Shan, N. Cercone, RIAC: a rule induction algorithm based on approximate classification, Tech. Rep. CS 96-06, Regina University 1996
[16] W. Duch, G.H.F. Diercksen, Feature Space Mapping as a universal adaptive system, *Comp. Phys. Comm.* 87 (1995) 341-371
[17] W. Schiffmann, M. Joost, R. Werner, Comparison of optimized backpropagation algorithms, *ESANN '93*, Brussels 1993, pp. 97-104; Synthesis and Performance Analysis of Multilayer Neural Network Architectures, Tech. Rep. 15/1992, available in `neuroprose` as schiff.gann.ps.Z
[18] T. Hastie, R. Tibshirani, Discriminant adaptive nearest neighbor classification, *IEEE PAMI* 18 (1996) 607-616
[19] J. H. Friedman, Flexible metric nearest neighbor classification, *Technical Report, Dept. of Statistics, Stanford University* 1994