Int. J. Appl. Math. Comput. Sci., 2000, Vol.10, No.4, 747–766

CLASSIFICATION, ASSOCIATION AND PATTERN COMPLETION USING NEURAL SIMILARITY BASED METHODS

WŁODZISŁAW DUCH*, RAFAŁ ADAMCZAK*,
GEERD H.F. DIERCKSEN**

A framework for Similarity-Based Methods (SBMs) includes many classification models as special cases: neural networks of the Radial Basis Function type, Feature Space Mapping neurofuzzy networks based on separable transfer functions, Learning Vector Quantization, variants of the k nearest neighbor methods and several new models that may be presented in a network form. Multilayer Perceptrons (MLPs) use scalar products to compute a weighted activation of neurons, combining soft hyperplanes to provide decision borders. Distance-based multilayer perceptrons (D-MLPs) evaluate the similarity of inputs to weights offering a natural generalization of standard MLPs. A cluster-based initialization procedure determining the architecture and values of all adaptive parameters is described. Networks implementing SBM methods are useful not only for classification and approximation, but also as associative memories, in problems requiring pattern completion, offering an efficient way to deal with missing values. Non-Euclidean distance functions may also be introduced by normalization of the input vectors in an extended feature space. Both the approaches dramatically influence the shapes of decision borders. An illustrative example showing these changes is provided.

Keywords: neural networks, classification, association, pattern recognition

1. Introduction

Multilayer perceptrons (MLPs) trained with backpropagation method (BP) are certainly the most popular among all neural techniques (Bishop, 1995). When applied to classification or approximation problems, MLPs use sigmoidal functions to provide soft hyperplanes dividing the input space into separate regions. MLPs are therefore similar to the statistical discriminant techniques or the Support Vector Machines (SVM) (Cortes and Vapnik, 1995), although a combination of soft sigmoids allows for representation of more complex, nonlinear decision borders. This is usually considered to be a strength of the MLP model, although in the cases when sharp decision

* Nicholas Copernicus University, Department of Computer Methods, ul. Grudziądzka 5, 87–100 Toruń, Poland, e-mail: {duch,raad}@phys.uni.torun.pl

** Max-Planck Institute of Astrophysics, 85740-Garching, Germany, e-mail: GDiercksen@mpa-garching.mpg.de

borders are needed it may also become its weakness. For example, classification borders conforming to a simple logical rule $x_1 > 1 \wedge x_2 > 1$ are easily represented by two hyperplanes, but there is no way to represent them accurately using soft sigmoidal functions used in MLPs. This problem is especially evident if regularization terms are added to the cost function, enforcing small values of weights. As a result, for some datasets no change in the learning rule or network architecture will improve the accuracy of neural solutions. A good real-world example is the hypothyroid dataset, for which the best optimized MLPs still give about 1.5% of error (Schiffman *et al.*, 1993) while logical rules reduce it to 0.64% (Duch *et al.*, 2000). Most research on neural networks is concentrated on architectures and learning rules, but the selection of neural transfer functions may be crucial to network performance (Duch and Jankowski, 1999).

Another problem with MLP models concerns selection of architecture and initialization of adaptive parameters. Constructive neural algorithms (Skapura, 1996) may help to find architectures that roughly match complexity of the data analyzed, but constructive models may also end up in suboptimal architectures. Genetic algorithms applied to the selection of architectures do not guarantee good solutions and are computationally very demanding (Yao 1993). Missing inputs are especially difficult to handle since filling the unknown features with the most frequently appearing values may lead to poor results.

MLPs are widely used for classification and approximation problems, while many interesting problems involve pattern completion and association. Associative memory models are usually based on recurrent networks. It would be very interesting to accomplish a similar task using feedforward MLP networks. MLPs, SVMs (Cortes and Vapnik, 1995) and other methods based on discriminant analysis, perform mappings that are rather difficult to interpret. Proponents of the logical rule-based machine learning methods consider it to be the biggest drawback of neural networks, limiting their applications in safety-critical fields such as medicine. Similarity-Based Methods (SBMs), e.g., the k-nearest neighbor (k-NN) method, retrieve the relevant context for each query presented to the classification system, providing some interpretation and estimating the probability of different class assignments. Such an interpretation is also possible for the Radial Basis Function (RBF) networks using Gaussian or other localized functions, or the Learning Vector Quantization (LVQ) method based on optimization of reference vectors. It may seem that such an interpretation is not possible for MLPs since they belong to the discriminant, rather than to memory-based techniques. One way to obtain an interpretation of MLP decisions is to study the transition from MLPs to networks performing logical operations (Duch *et al.*, 1998). Although discriminant methods and prototype methods seem to be quite different, in fact the two approaches are deeply connected. A single hyperplane discriminating vectors belonging to two classes can be replaced by two prototypes, one for each class. For N prototypes one can generate $N(N-1)/2$ pair-wise discriminating hyperplanes providing piece-wise linear approximation to the decision borders.

All these shortcomings of the MLP networks are overcome here. Recently, a general framework for Similarity-Based Methods (SBMs) used for classification has been presented (Duch, 1997). It is briefly presented in the next section, and several examples of well-known and new neural methods derived using this framework are presented. In particular, the Distance-Based Multilayer Perceptrons (D-MLPs, cf. Duch *et al.*, 1999) are introduced, improving upon the traditional approach by providing more flexible decision borders, using information about the structure of the data derived from clusterization procedures and enabling a prototype-based interpretation of the results. Symbolic values used with probabilistic distance functions allow us to avoid an ad hoc procedure to replace them with numerical values. The SBM perspective allows us to initialize all D-MLP network parameters starting from some standard clusterization procedure and thus using information that can be easily obtained from the data.

A simple procedure to change D-MLP models into associative memories and to use them in pattern completion problems is described in Section 4. As a result, missing values are handled in an efficient way. Finally, to avoid writing computer programs for the backpropagation method for each type of the distance function, a simple transformation of the input data is proposed, allowing for a distance-based interpretation. An illustration of this method on the Iris data is presented in Section 6. The paper is finished with a short discussion.

2. Neural Methods from a Similarity-Based Perspective

The classification problem (the same reasoning may also be applied to regression and pattern completion problems) is stated as follows: Given a set of N_t class-labeled training vectors $\{\mathbf{R}^j, \mathrm{C}(\mathbf{R}^j)\}$, $j = 1, \ldots, N_t$, where $\mathrm{C}(\mathbf{R}^j)$ is the class of $\mathbf{R}^j$, and given a vector $\mathbf{X}$ of an unknown class, use the information provided in the similarity measure $D(\mathbf{X}, \mathbf{R}^j)$ to estimate the probability $p(C_i|\mathbf{X}; M)$ that the vector $\mathbf{X}$ belongs to one of the classes C_i, $i = 1, \ldots, N_C$, called further the probability of classification (Bishop, 1995). Some classifiers predict only binary, 0 or 1, probabilities, recommending only one class as the winner, but if distributions of data vectors belonging to different classes overlap (which is usually the case), binary approximation is not justified. While calculating probabilities $p(C_i|\mathbf{X}; M)$ a classification model M is used, described by the values of all parameters and procedures employed. A general similarity-based model of an adaptive system used for classification should include at least the following elements:

$$M = \{\{\mathbf{R}^j\}, D(\cdot), G(D(\cdot)), k, E[\cdot], K(\cdot), \mathcal{R}(\cdot|\cdot)\},$$

where $\{\mathbf{R}^j\}$ is the set of reference vectors created from the set of training vectors $\{\mathbf{X}^i\}$ by some procedure; $D(\cdot)$ is a similarity function (frequently, a distance function) parameterized in various ways, or a table used to compute similarities; $G(D(\mathbf{X}, \mathbf{R}))$ is a weighting function estimating the contribution of the reference vector $\mathbf{R}$ to the classification probability; k is the number of reference vectors taken into account in the neighborhood of $\mathbf{X}$; $E[\cdot]$ is the total cost function optimized during training; it may include regularization terms and may depend on a kernel function $K(\cdot)$, scaling the influence of the error, for a given training example, on the total cost function, using a risk matrix $\mathcal{R}(C_i|C_j)$ that estimates the costs of assigning wrong classes.

The cost function that minimizes the risk for the overall classification is

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

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 $\mathcal{R}(C_i, C_j)$ are proportional to the risk of assigning the C_i class when the true class is C_j, and in the simplest case $\mathcal{R}(C_i, C_j) = 2 - \delta_{ij}$ or $\mathcal{R}(C_i, C_j) = 1 + |i - j|$ is taken (strictly speaking, a unit matrix is added here to the usual risk matrix). M specifies all the 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 the $\max_i(p(C_i|\mathbf{X}; M) - \delta(C_i, C(\mathbf{X}))$, a standard mean square error (MSE) function is recovered.

An adaptive system may include several such models M_l, $l = 1, \ldots, N_M$ and an interpolation procedure to select between different models or average results over many models generated using the same data sampled in different ways. Such averaging with boosting procedures for selection of training vectors leads to creation of stable and accurate classifiers (Breiman, 1996), sometimes called 'classifier committees' or 'classifier ensembles'. Simple averaging, or a linear combination of several models is used most frequently

$$P(C_i|\mathbf{X}; M) = \sum_{l=1}^{N_M} W_l \, p(C_i|\mathbf{X}; M_l). \quad (2)$$

A least-squares minimization (LSM) procedure is used to determine W_l coefficients. The models are usually of the same type (e.g., MLP neural networks), although it may be of some advantage to have diverse models that specialize in correct classification of different areas of the input space. Creating ensembles, one should use all information available. Since we know for which training vectors R_k each model makes an error, it seems reasonable to use this information in creating an ensemble. The coefficients of the linear combination should depend on the distance between X and those regions $R_{l,k}$ of the feature space where model M_l works poorly. Therefore,

$$P(C_i|\mathbf{X}; M) = \sum_{l=1}^{N} \sum_k W_l D(X, R_{l,k}) p(C_i|\mathbf{X}; M_l) \quad (3)$$

should be a good choice. The same LMS optimization procedures are used here as in the previous case. After renormalization, estimates of classifcation probabilities are obtained:

$$p(C_i|\mathbf{X}; M) = P(C_i|\mathbf{X}; M) \Big/ \sum_j P(C_j|\mathbf{X}; M). \quad (4)$$

Many pattern recognition, machine learning and neural network models are special cases of this SBM framework. One way to use this framework is to start with the simplest model and turn on various optimization parameters and procedures, for example starting from the simplest k-NN and optimizing the number of neighbors, distance function parameters, soft weighting, feature selection, number and position of reference vectors. Each step towards a more complex model decreases the bias of the classifier, but may increase its variance (Breiman, 1996), therefore after each step the model should be validated and only if the greater complexity is justified by a higher accuracy, more complex models should be accepted, otherwise a different type of optimization should be used.

2.1. RBF and LVQ-Like Methods

In RBF networks Euclidean distance functions $D(\mathbf{X},\mathbf{R}^j) = ||\mathbf{X}-\mathbf{R}^j||$ are assumed and a radial, for example Gaussian $G(D) = \exp(-D^2)$ weighting functions are used. Essentially, RBF is a minimal distance soft weighted method with no restrictions on the number of neighbors, reference vectors $\mathbf{R}^j$ that are near influence probabilities of classification more than those that are far. The SBM framework suggests that there is nothing special about this choice of the distance function and the weighting function. The simplest suitable weighting function is the *conical radial function*: zero outside the radius σ and $1 - D(\mathbf{X},\mathbf{R})/\sigma$ inside this radius. The classification probability is calculated by the output node using the formula:

$$p(C_i|X;\sigma) = \frac{\sum\limits_{j\in C_i} G\big(D(\mathbf{X};\mathbf{R}^j),\sigma\big)}{\sum\limits_{j} G\big(D(\mathbf{X};\mathbf{R}^j),\sigma\big)}, \tag{5}$$

with

$$G(D(\mathbf{X};\mathbf{R}^j),\sigma) = \max\left(0, 1 - D(\mathbf{X},\mathbf{R}^j)/\sigma\right). \tag{6}$$

Here $W(D) = G(D(\mathbf{X},\mathbf{R}^j);\sigma)$ is the weight associated with the distance D. The reference vectors outside of the σ radius have no influence on the classification probability, while their influence inside this radius depends linearly on the distance D. Combining this weighting with the restriction on the number of neighbors leads to the weight $W(D) = \max(0, 1 - D/\alpha r_k)$, where r_k is the distance to the k-th neighbor and α is an adaptive parameter optimized on the test set.

More sophisticated versions of this algorithm include optimization of the shape of $G(D;\sigma)$ weighting functions using additional parameters. One example is a combination of two sigmoidal functions $\sigma(||\mathbf{X}-\mathbf{R}^j|| - b) - \sigma(||\mathbf{X}-\mathbf{R}^j|| - b')$, providing a larger area in which the weighting factor is essentially constant. Another example is the hyperbolic weighting scheme:

$$p(C|X;M) = \frac{\sum\limits_{j} \delta(C(X),C)/\big(D(\mathbf{X},\mathbf{R}^j)+\varepsilon\big)}{\sum\limits_{j} 1/\big(D(\mathbf{X},\mathbf{R}^j)+\varepsilon\big)}, \tag{7}$$

where ε is a small positive number.

In the Gaussian classifier (Krishnaiah and Kanal, 1982) or in the original RBF network only one parameter σ was optimized (Wasserman, 1993). Optimization of the positions of the reference centers $\mathbf{R}^j$ leads to the LVQ method (Kohonen, 1995) in which the training set is used to define the initial prototypes and the minimal distance rule to assign the classes. The Restricted Coulomb Energy (RCE) classifier (Reilly *et al.*, 1982) uses hard-sphere weighting functions. The Feature Space Mapping model (FSM) is based on separable, rather than radial weighting functions (Duch and Diercksen, 1995). All these models are special cases of the general SBM framework.

An important problem with the localized description of the data by RBFs and similar methods concerns the representation of oblique probability distributions of the classes. Only very recently a method to create oblique probability distributions in N-dimensional spaces using only N parameters has been described (Duch and Jankowski, 1999). Oblique decision borders in the SBM are obtained by rotation of the local coordinate system in which the distances are computed. It is sufficient to use a rotation matrix with scaling factors $R_{ii} = s_i$ on the diagonal and rotation parameters $R_{ii+1} = \beta_i$ as the only off-diagonal element.

2.2. D-MLP Model

Threshold neurons compute distances in a natural way. If the input signals $\mathbf{X}$ and the weights $\mathbf{W}$ are $(\pm,1,\ldots,\pm,1)$ vectors, a neuron with N inputs and a threshold θ realizes the following function:

$$\Theta\Big(\sum_i^N W_i X_i - \theta\Big) = \begin{cases} 0 & \text{if } ||\mathbf{W}-\mathbf{X}|| > (N-\theta)/2, \\ 1 & \text{if } ||\mathbf{W}-\mathbf{X}|| \le (N-\theta)/2, \end{cases} \tag{8}$$

where the $||\cdot||$ norm is defined by the Hamming distance (it counts the number of mismatches for binary strings). One can interpret the weights of neurons in the first hidden layer as addresses of the reference vectors in the input space and the activity of the threshold neuron as an activation by inputs falling into a hard sphere of radius $(N-\theta)/2$ centered at $\mathbf{W}$. Changing binary into real values and the threshold into sigmoidal neurons for the inputs normalized to $||\mathbf{X}|| = ||\mathbf{W}|| = 1$ leads to a soft activation of the neurons by input vectors close to $\mathbf{W}$ on a unit sphere. The Hamming neural network (Lippmann, 1987) is actually a neural realization of the nearest neighbor method for a single neighbor and binary vectors.

In general, treating $\mathbf{W}$ and $\mathbf{X}$ as vectors and the activation as a scalar product $\mathbf{W}\cdot\mathbf{X}$, the activation of a neuron is written as:

$$\mathbf{W}\cdot\mathbf{X} = \frac{1}{2}\left(||\mathbf{W}||^2 + ||\mathbf{X}||^2 - ||\mathbf{W}-\mathbf{X}||^2\right). \tag{9}$$

For normalized input vectors the sigmoidal functions (or any other monotonically growing transfer functions) may therefore be written in the form:

$$\sigma(\mathbf{W}\cdot\mathbf{X}+\theta) = \sigma\big(d_0 - D(\mathbf{W},\mathbf{X})\big), \tag{10}$$

where $D(\mathbf{W},\mathbf{X})$ is proportional to the squared Euclidean distance between $\mathbf{W}$ and $\mathbf{X}$ and $d_0 = 1/2 + 1/2||\mathbf{W}||^2 + \theta$. The normalization $||\mathbf{X}|| = 1$ is necessary to avoid the dependence of d_0 on $\mathbf{X}$. The sigmoidal function evaluates the influence of the reference vectors $\mathbf{W}$ on the classification probability $p(C_i|\mathbf{X};\{\mathbf{W},\theta\})$. It plays a role of the weight function $G(D) = \sigma(d_0 - D(\mathbf{W},\mathbf{X}))$, monotonically decreasing, with flat plateau for small distances, reaching the value of 0.5 for $D(\mathbf{W},\mathbf{X}) = d_0$ and approaching zero for larger distances. For normalized $\mathbf{X}$ but arbitrary $\mathbf{W}$, the range of the sigmoid argument lies in the $[\theta - |\mathbf{W}|, \theta + |\mathbf{W}|]$ interval. A unipolar sigmoid has a maximum curvature around ± 2.4, therefore small thresholds and weights mean that the network operates in an almost linear regime. Regularization methods add penalty terms to the error function forcing the weights and thresholds to become small and thus smoothing the network approximation.

From the SBM point of view, in MLP networks sigmoidal functions are used to estimate the influence of the weight vectors according to the distance between the weight and training vectors. By changing the distance function in (10) from the square of the Euclidean distance to some other distance measures, new types of neural networks, called the D-MLP networks, are defined. Another possibility is to write the weighted product in the form

$$\sigma(\mathbf{W}\cdot\mathbf{X}) = \sigma\left(\frac{1}{4}(||\mathbf{W}+\mathbf{X}||^2 - ||\mathbf{W}-\mathbf{X}||^2)\right). \tag{11}$$

The D-MLP networks simply replace the square of the Euclidean distance in the equation above or in eqn. (10) by Minkovsky's or other types of norms. The network with the nodes computing $\sigma(d_0 - D(\mathbf{W},\mathbf{X}))$ is trained like a standard MLP, using the backpropagation method (Bishop, 1995). The backpropagation procedure requires derivatives of the distance functions, but for Minkovsky's and other popular distance functions they are easily derived.

In eqn. (10) the parameter d_0 should be treated as an adaptive parameter only if $\mathbf{X}$ is normalized. This can always be done without loss of information if one or more additional components are added to the vector, extending the feature space by at least one dimension. In particular, taking $x_r = \sqrt{R^2 - ||\mathbf{X}||^2}$, where $R \geq \max_X ||X||$, amounts to a projection of the data on a unit hemisphere with radius R (a more sophisticated projection is described in (Duch *et al.*, 1997)). If a non-Euclidean norm is used, the sphere changes its shape (see Sec. 5).

2.3. Other Examples of Neural Methods Derived from SBM Framework

The non-Euclidean D-MLP networks described above are only one of many methods that may be derived from the SBM framework. Adapting the similarity function to minimize the in-class distance variance and to maximize the between-class variance, a non-linear version of Fisher's discrimination analysis is obtained. Combination of sigmoidal functions offers an interesting parametrization for non-linear feature transformations, enabling a simple network realization:

$$\rho_i(\mathbf{X}_i) = \sum_j \alpha_{ij}\sigma\big(\beta_{ij}(\mathbf{X}_i - \mathbf{X}_{i,min}) - \gamma_{ij}\big), \tag{12}$$

$$d(\mathbf{A}_i,\mathbf{B}_i) = \big|\rho_i(\mathbf{A}_i) - \rho_i(\mathbf{B}_i)\big|, \tag{13}$$

where α_{ij}, β_{ij}, γ_{ij} are adaptive parameters that are optimized to increase the accuracy of classification. Such distance functions have not been used in practice so far. They allow for automatic discretization of data, starting from soft sigmoidal slopes and increasing the values of β_{ij} to account for sharp decision borders necessary for some classifcation tasks. An additional advantage is the ability to interpret the outcome of classification using logical rules, since the decision borders in the limit of high slopes will have hyperrectangular shapes.

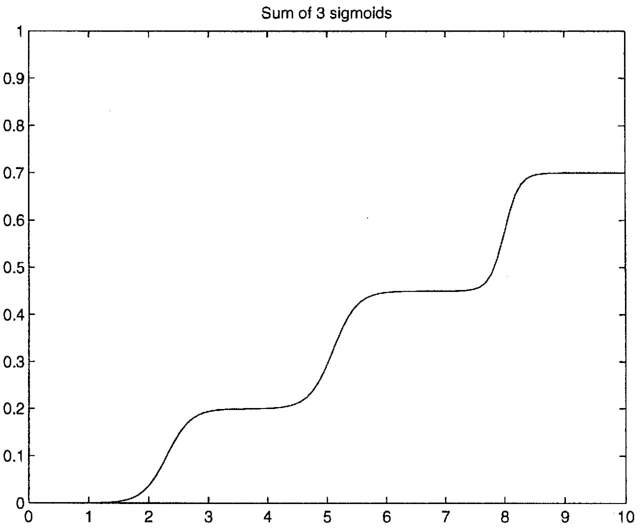

Fig. 1. Sum of three sigmoidal functions provides a useful distance function allowing one to minimize in-class and to maximize between-class distances, defining a non-linear version of Fisher's discrimination. The horizontal axis shows $\mathbf{X}_i - \mathbf{X}_{i,\min}$ and the vertical axis the transformed values $\rho_i(\mathbf{X}_i)$.

A network of nodes computing such distances can be used for classification or prediction as any other neural network. It can also be used for extraction of logical rules from data, either fuzzy rules or (in the limit of high slopes) crisp logical rules.

Another interesting possibility is to use a neural network to learn the most appropriate similarity and weighting function. The advantage of such an approach is that instead of having a global coordinate system with a single distance function, local coordinate systems smoothly changing at different points of the feature space are defined, providing an optimal distance and weighting in different regions of the input space. Combining the k-NN approach with the neural distance function is certainly worth trying.

Discrimination and cluster-based methods are deeply connected. A single discrimination hyperplane (used in the linear discrimination method or provided by MLP neurons) may be replaced by a fixed reference vector (for example a cluster center) and an adaptive reference vector. In the Statlog project comparing 20 classification methods the simplest nearest neighbor method appeared as the top one in about one third of all cases (Michie *et al.*, 1994). The network realization described below is a generalization of the k-NN method and should improve the results on the remaining problems.

The network has hidden nodes computing the distances $D(\mathbf{X}-\mathbf{R})$, where $\mathbf{R}$ are reference (training) vectors. The k nodes with the smallest distances output their class labels $h_j(\mathbf{X};\mathbf{R}) = C_i$ and the remaining nodes output $h_j(\mathbf{X};\mathbf{R}) = 0$. The classes are numbered as $i = 1, \ldots, N_C$. The output layer computes the probabilities using the formula

$$P(C_i|\mathbf{X};M) = \sum_j \mathbf{W}_{ij} \cdot h_j(\mathbf{X}), \tag{14}$$

$$p(C_i|\mathbf{X};M) = \frac{P(C_i|\mathbf{X};M)}{\sum_j P(C_j|\mathbf{X};M)}. \tag{15}$$

The weight W_{ij} between the hidden node R_j belonging to class C_j and the output node computing the probabilities for class C_i is initially equal to $W_{ij} = (1-\mathcal{R}(C_i,C_j))/C_j$, where the elements of the risk matrix $0 \le \mathcal{R}(C_i,C_j) \le 1$ in the simplest k-NN are replaced by δ_{ij}. Thus each vector that belongs to the k nearest ones or that falls into the r radius of $\mathbf{X}$ and is of the C_j class, contributes to the probability of the C_i class a value of $1-\mathcal{R}(C_i,C_j)$. The structure of the network is shown in Fig. 2. For the cost function that should be optimized one may take

$$E(\mathbf{W};M) = \sum_{\mathbf{X},i} \left(p(C_i|\mathbf{X};M) - \delta(C_i,C(\mathbf{X}))\right)^2, \tag{16}$$

where the model M includes k, weights $\mathbf{W}$ and distance-related quantities as parameters. If the number of classification errors should be minimized, binary 0, 1 output probabilities are taken, provided for example by the winner-takes-all neural procedure. Binary probabilities should be used with global minimization or search-based methods, since gradient-based methods cannot be used in this case. The output weights, initialized to $W_{ij} = (1-\mathcal{R}(C_i,C_j))/C_j$, may be treated as adaptive parameters. Introduction of soft weighting $G(D(\,\cdot\,))$ allows us to use gradient optimization methods. For many datasets (especially for images (Michie *et al.*, 1994)) this simple network should outperform MLPs and other classification models, since the results should be at least as good as the k-NN results.

A single neuron provides a discrimination hyperplane which may be replaced by one reference vector. The position of this reference vector should be adapted to the data. If different exponents in the Minkovsky distance functions $D^\alpha(\mathbf{X},\mathbf{Y})$ are used, the decision borders may be drastically changed (cf. Figs. 3 and 6). Using one prototype $\mathbf{R}_i$ per class (i.e. one hidden node), the class membership is decided by the discriminant function

$$z(\mathbf{X}) = W_1 D(\mathbf{X},\mathbf{R}_1) - W_2 D(\mathbf{X},\mathbf{R}_2) - \theta, \tag{17}$$

where θ is a threshold. The three adaptive parameters, W_1, W_2, θ and the positions of two prototype vectors provide very flexible decision borders in the two-class prob-

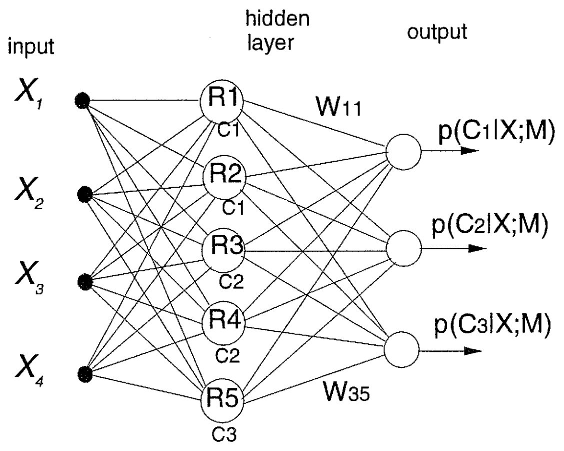

Fig. 2. Network generalization of the k-NN method. The hidden nodes compute the distances to the reference vectors and return k values of the class labels associated with the nodes, while the output nodes compute probabilities.

lems. If more reference vectors are required, the output node computes discriminating function sums over the prototypes for each class:

$$z(\mathbf{X}) = \sum_{i\in C_1} W_i D(\mathbf{X},\mathbf{R}_i) - \sum_{i\in C_2} W_i D(\mathbf{X},\mathbf{R}_i) - \theta. \tag{18}$$

Scaling the whole sum, instead of influences of individual reference vectors, is a simple way to reduce the number of adaptive parameters used by the system. One option that we are investigating is to use simple gradient optimization for weights and thresholds, and search based techniques for nonlinear scaling parameters.

A similarity of such a neural realization of the nearest neighbor method to the RBF model with radial coordinate functions should be noted. If the number of neighbors is not restricted, the two methods are identical.

The SBM point of view on neural networks not only allows us to define many new methods, but also leads to novel applications such as pattern completion or associative memory recall. A natural cluster-based initialization described below determines all the parameters of D-MLP networks.

3. Initialization of the Network

The D-MLP network uses normalized vectors, adding one extra dimension if necessary, projecting the data on a hemisphere. The network should be initialized taking the centers of clusters in the extended $N + 1$ dimensional space as weights $\mathbf{W}$, and taking $d_0 = D(\mathbf{W}, \mathbf{X}^b)$, where $\mathbf{X}^b$ is a vector at the border of a given cluster. To define clusters we have tried (Duch *et al.*, 1997) dendrograms and decision trees, but other clusterization methods may also be used for initialization (Krishnaiah and Kanal, 1982). Using weighted activations the contribution of a center of an input data

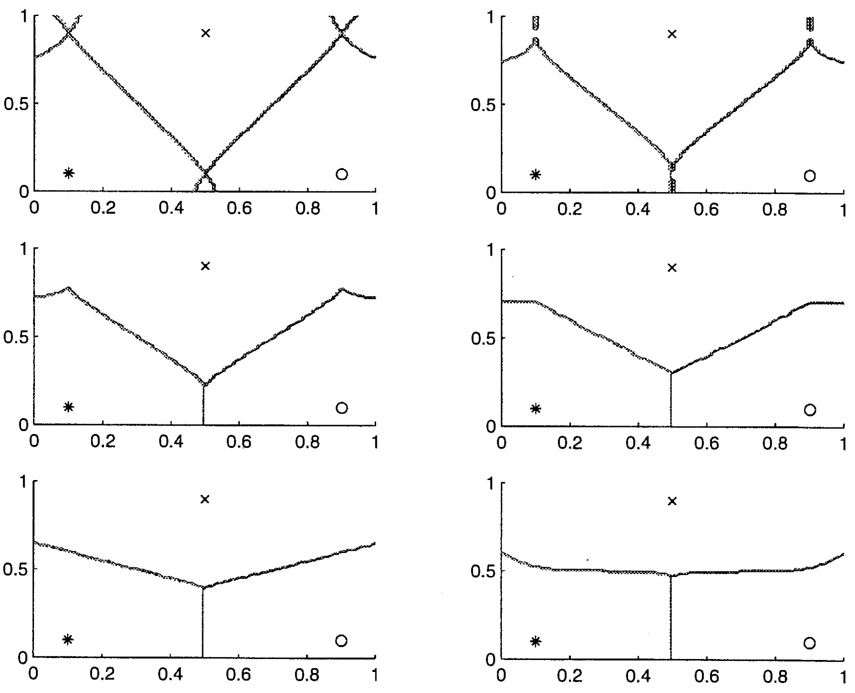

Fig. 3. Decision borders for various exponents of Minkovsky's distance function in the nearest neighbor method for $\alpha = 0.1, 0.3, 0.7, 1, 2, 8$. All weights are identical.

cluster $\mathbf{C}$ laying on the unit sphere is $\mathbf{W}\cdot\mathbf{C}$. The largest activation is obtained when the weights $\mathbf{W}$ point in the same direction as the center $\mathbf{C}$. The sigmoidal function $\sigma(\mathbf{C}\cdot\mathbf{X}-\theta) = (1+\exp((-\mathbf{C}\cdot\mathbf{X}+\theta)/T))^{-1}$, where T determines the slope, has the largest gradient in the direction of $\mathbf{W}=\mathbf{C}$. The value $\sigma(0) = 0.5$ is obtained at a θ distance from the origin of the coordinate system. Since the $\mathbf{C}$ vector is normalized, $\theta = 1$ places the contours for the 0.5 value tangentially to the unit hypersphere. The contours for lower values $\sigma(\mathbf{C}\cdot\mathbf{X}-\theta) < 0.5$ cut segments of the hypersphere in which the value of $\sigma(\mathbf{C}\cdot\mathbf{X}-\theta)$ is constant.

A parameter which is rarely changed in MLPs is the slope of sigmoidal functions. It defines the area which exerts influence on the performance of each node. If the slope is too high, the area in which the sigmoidal function is not approximately constant is small and only a few training vectors have a chance to influence the gradient-based learning procedures. If it is too low, then all functions strongly overlap and there is no possibility to create sharp decision borders. Normalization of the weights $\mathbf{W}$ is equivalent to a local change of the slope:

$$\begin{aligned}(\mathbf{W}\cdot\mathbf{X}+\theta)/T &= \left(\frac{\mathbf{W}}{||\mathbf{W}||}\cdot\mathbf{X}+\frac{\theta}{||\mathbf{W}||}\right)\frac{||\mathbf{W}||}{T}\\ &= (\mathbf{W}'\cdot\mathbf{X}+\theta')/T' = \left(d_0' - D(\mathbf{W}',\mathbf{X})\right)/T',\end{aligned} \tag{19}$$

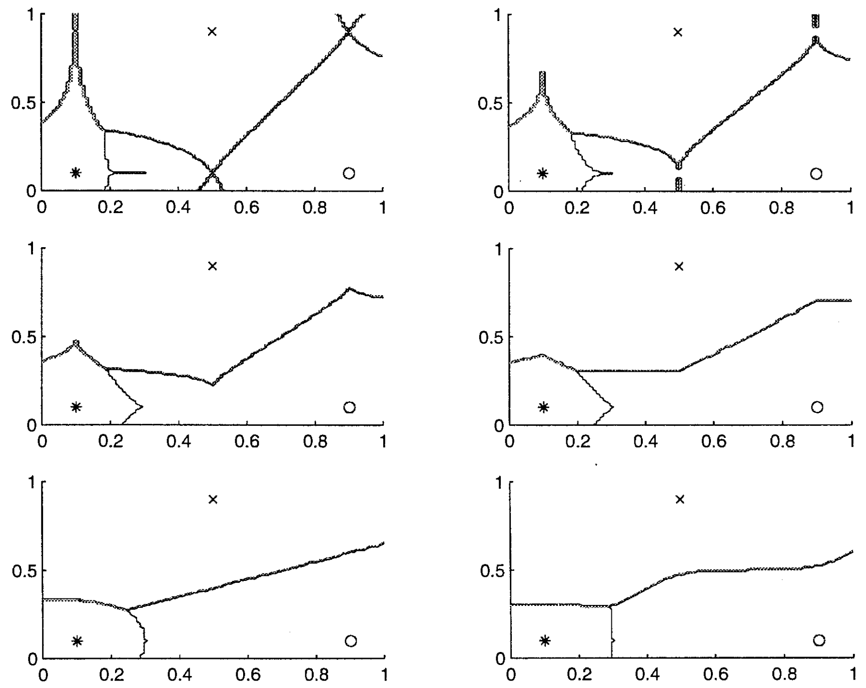

Fig. 4. Decision borders for various exponents of Minkovsky's distance function in the nearest neighbor method for $\alpha = 0.1, 0.3, 0.7, 1, 2, 8$. The weight of the first prototype is three times larger than other weights.

where the primed quantities T', θ' and $\mathbf{W}'$ are divided by the norm $||\mathbf{W}||$. Thus as long as the slopes of the transfer function may change, without loss of generality both $\mathbf{X}$ and $\mathbf{W}'$ may be normalized. A useful variability range of the sigmoid is between its maximum curvature points, which for $T = 1$ are between $\Delta(T) = \pm 2.4$. If the variability range is assumed to be $1/10$ of the cluster size, i.e. $\Delta(T) = \pm d_0/10$, then setting $T \approx d_0/24$ will be appropriate. After such an initialization, training the network is usually quite short.

In practice, one may take a dendrogram, starting from few top largest clusters, initializing the network with the number of neurons equal to the number of these clusters, and training the network. If the complexity of the network is too low, the results on the training set will be poor. More clusters should be taken into account—in the case of dendrograms one should break those clusters that are inhomogenous in the first place. A more complex network is initialized and trained, until the results on the training set will be satisfactory.

In the XOR case, the input vectors for class =T are $(0,1), (1,0)$ and for class =F they are $(0,0), (1,1)$. The mean for each feature is 0.5 and after shifting and renormalizing the vectors are $\mathbf{C}_1 = (-1,+1)/\sqrt{2}$, $\mathbf{C}_2 = (+1,-1)/\sqrt{2}$ for class T and $(-1,-1)/\sqrt{2}$, $(+1,+1)/\sqrt{2}$ for class F. Selecting one of the classes for output,

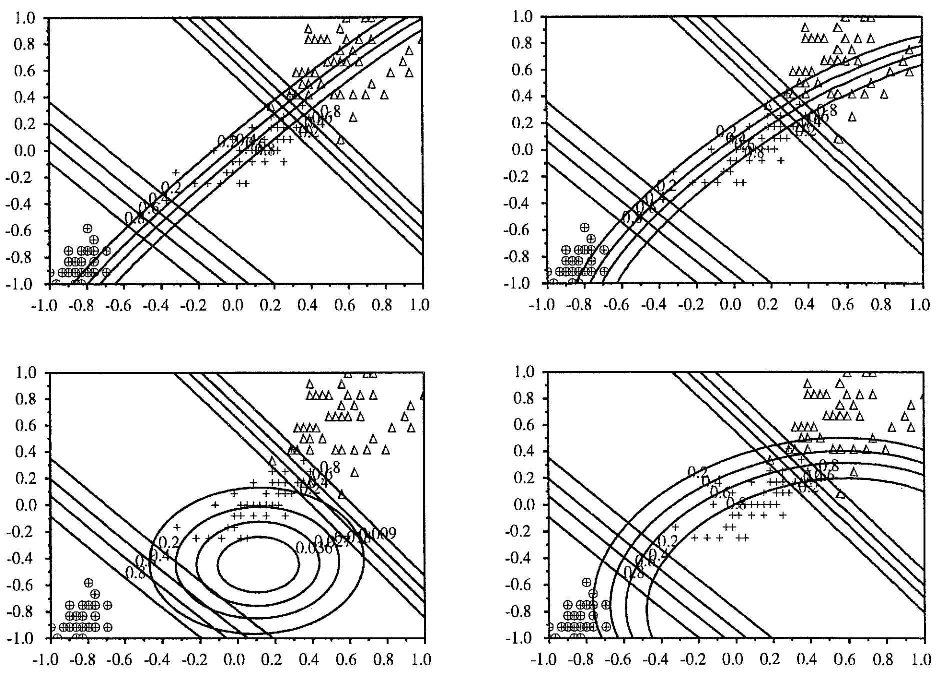

Fig. 5. Shapes of decision borders in the Iris case for the network without the hidden layer (3 output neurons, 3 inputs), with a growing W_3 weight.

for example class T, initial weights for the first neuron are given by $\mathbf{C}_1$ and for the second neuron by $\mathbf{C}_2$, while the hidden to output layer weights are all $+1$. This is the correct and the simplest solution to the XOR problem found without any optimization of the network. For more complex examples of this type of initialization see (Duch *et al.*, 1997).

Since the architecture of the MLP network in the extended space is completely determined by the initialization procedure (the clusterization method used determines all parameters) and the training is short due to a good starting point, many distance functions can be tried on a given problem.

4. Pattern Completion, Associative Memory and Missing Values

The methods belonging to the SBM framework, such as the nearest neighbor method, may be used as associative memories in a natural way. Any part of the input vector $X = (X_d, X_u)$ may be used to find nearest neighbors in the subspace of defined input values X_d. The undefined part X_u is predicted interpolating the values of nearest neighbors for the dominating class. Optimization of parameters for classification in the X_d subspace should only improve the results but frequently the same k-NN model works well in subspaces.

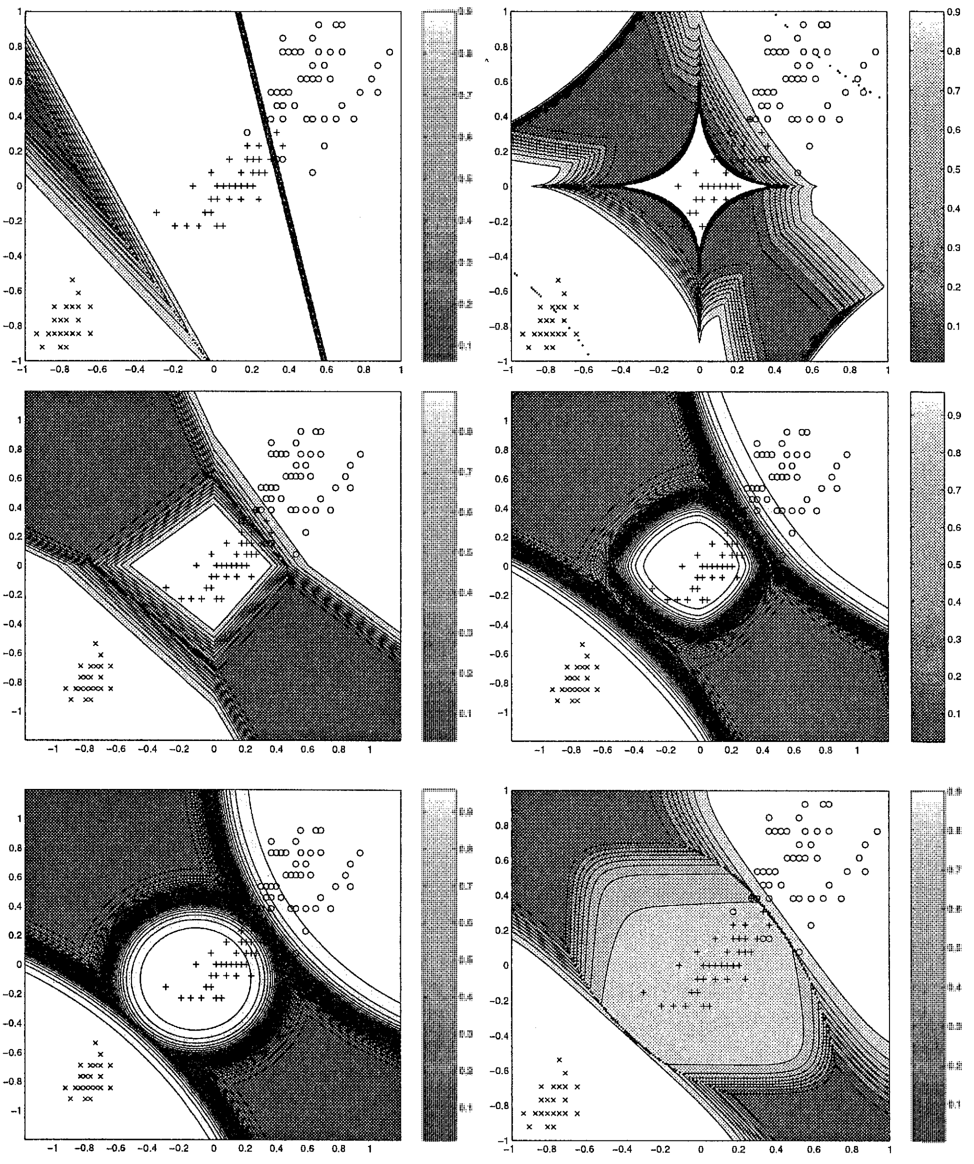
Fig. 6. Shapes of decision borders in the Iris case for standard MLP (3 neurons) and an MLP network in the extended space with vectors renormalized using the Minkovsky metric, $\alpha = 0.5, 1.0, 1.5, 2.0$ and 7.0.

Pattern completion may be implemented in several ways. In many cases vectors with missing values are removed from the training set or some averaged or most frequent values are inserted. In this way useful information is thrown out or inappropriate information is introduced. For example, the echocardiogram data from the UCI repository (Mertz and Murphy, 2000), contains 132 vectors, 12 attributes of which only 1–9 are useful, the second being the class. 15 values of the attribute 6 are missing, 11 values for attribute 7, etc. If the attributes with the missing values are ignored, 10-fold stratified crossvalidation tests give the 87.8% accuracy using on the average 24 neurons of the FSM network (Duch and Diercksen, 1995) (FSM is based on a constructive algorithm, therefore different numbers of neurons may be created in different crossvalidations), while inserting averages over all classes decreased the accuracy to 85.5% (with 20 neurons), and inserting a new value that does not appear in the data, such as -100, decreased the accuracy to 81.5% (using 22 neurons).

The same behavior has been observed for the Hepatitis dataset taken from the same source. The data contains 155 vectors, 18 attributes, 13 of them are binary, the other having integer values. The last attribute has 67 missing values, the attribute 16 has 29 missing values, etc. 10-fold crossvalidation tests ignoring missing values give the 79.9% accuracy using on the average 19 neurons, inserting averages over all classes 81.0% (with 12 neurons) and inserting -100 gives the lowest accuracy 79.1% (with 16 neurons).

Suppose that two-dimensional data vectors are clustered around (1.0,1.0) and (2.0,2.0), with the first cluster containing twice as many vectors as the other. Suppose now that the second feature is missing in the training vector X with $x_1 = 1.9$. If X neighbors in the x_1 subspace, around the given $x_1 = 1.9$ value are found, interpolating the missing x_2 value will give an approximately correct answer (around 2.0) while using the most frequent values or averaged values will give an incorrect guess (around 1.0). In many applications a hierarchical approach to data collection is taken: initial tests allow us to make a hypothesis, followed by specific tests that confirm it or not. The challenge is to discover such a hierarchical classification. In statistics analysis of independent surveys in which some questions are not answered by some respondents and some questions are not asked in some surveys it is known as the 'multiple imputation' problem (Rubin, 1996), but the assumptions about normal distributions used in this theory may not be valid. Another approach is described below.

In the first step, the missing features in the training vectors should be completed. The information contained in the training vectors with missing features is then used to improve the classification model. The probability of the unknown values X_u is calculated by maximization of

$$p(X_u|X_d;M) = \max_{u',i} p\big(C_i|(X_{u'},X_d);M\big), \tag{20}$$

i.e., searching for the maximum of the probability given by the model M in the subspace of undefined features, with a fixed point in the X_d subspace. If a single missing feature is sought, one dimensional maximization or a search procedure in the range of admissible values for X_u is carried out. The initial model M is prepared using either training vectors that have all features defined, or—if most vectors contain missing values—a largest subset of training vectors is found with the largest number of the same input features defined. For example, if only a few vectors with all values are given, but a large number contains just a single missing value X_u, the initial classification model should be based on a reduced number of features. The model is then retrained using vectors containing the extra feature X_u and the missing values of this feature imputed to the remaining vectors. At each step one may check if it is worth to include a new feature and to perform pattern completion. If the results in crossvalidation tests get worse, the feature should be dropped.

For strongly interacting features the problem of initial feature selection and the order of imputing/adding features suffers from combinatorial explosion. There is no guarantee that an optimal model will be found. In practice, network computations make the whole search procedure rather simple, since after a reasonable initial model is created, maximization in eqn. (20) does not involve costly multidimensional searches, but can be performed analytically or by evaluating the excitation level of network nodes. Moreover, since networks offer an analytical representation of computed probabilities, integration using statistical sampling techniques is easily performed. Using an FSM network and the method based on eqn. (20) for the two datasets mentioned above we have obtained for the echocardiogram the 90.2% accuracy using only 18 neurons, and for the hepatitis the 83.4% accuracy using only 10 neurons, a significantly better result than by other methods.

5. Normalization of Input Vectors in Non-Euclidean Spaces

The parameter d_0 should be treated as an adaptive parameter only if $\mathbf{X}$ is normalized. This may always be done without loss of information if one or more additional components are added to the vector, extending the feature space by at least one dimension. Taking $X_r = \sqrt{R^2 - ||\mathbf{X}||^2}$, where $R \geq \max_X ||\mathbf{X}||$, amounts to a projection of the data onto a unit hemisphere with radius R. In general, vectors $(\mathbf{X}, X_r)$ may be normalized, $||(\mathbf{X}, X_r)||_D = 1$, using the metric defined by the distance function $D(\mathbf{X}, \mathbf{R})$.

The distance function may be heterogeneous, using Minkovsky's metric for numerical features and probabilistic metric functions for symbolic features. Minkovsky's distance with the scaling factors is

$$D(\mathbf{A}, \mathbf{B}; s)^\alpha = \sum_i^N s_i d(\mathbf{A}_i, \mathbf{B}_i)^\alpha. \tag{21}$$

The $d(\cdot)$ function is used to estimate the similarity at the feature level and in the simplest case it is equal to $|A_i - B_i|$. For large α this metric changes the sphere into a soft cuboid, for $\alpha = 1$ it becomes a pyramid and for $\alpha < 1$ it has a hypocycloidal shape. Instead of deriving the backpropagation equations for the transfer functions with non-Euclidean distances, one may achieve a similar result using a standard MLP network with x_r determined by the normalization condition using the desired metric.

In memory-based reasoning the Modified Value Difference Metric (MVDM) has gained some popularity (Wilson and Martinez, 1997). The distance between two N-dimensional vectors A, B with discrete (nominal, symbolic) elements, in a K class problem, is computed using the conditional probabilities:

$$D_\alpha(A, B) = \sum_j^N \sum_i^K |p(C_i|A_j) - p(C_i|B_j)|^\alpha, \tag{22}$$

where $p(C_i|A_j)$ is estimated by calculating the number $N_i(A_j)$ of times the value A_j of the feature j occurred in vectors belonging to class C_i, and dividing it by the number of times A_j occurred for any class. A 'value difference' for each feature j is defined as $d_V^\alpha(A_j, B_j) = \sum_i^K |(p(C_i|A_j) - p(C_i|B_j))|^\alpha$. It allows us to compute $D_V(\mathbf{A}, \mathbf{B})$ as a sum of value differences over all features. The distance 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 attribute values. Generalization for continuous values requires a set of probability density functions $p_{ij}(x)$, with $i = 1, \dots, K$, $j = 1, \dots, N$.

Using the VDM type of metrics leads to problems with calculation of gradients, therefore another method is advocated here. Replacing symbolic features by vectors of $p(C_i|A_j)$ probabilities (with the dimension equal to the number of classes times the number of different symbolic values the feature takes) allows us to reproduce the MVDM distances using numerical values of vector components. Many other types of metric functions exist (Wilson and Martinez, 1997) and their performance should be empirically verified. Several alternative extensions of the input space may be considered, for example adding one or more features $X_r = D(\mathbf{X}, \mathbf{R})$ equal to the distance of a given vector $\mathbf{X}$ to some fixed vector $\mathbf{R}$ a parabolic projection is made.

It may be of some advantage to increase the separation of the clusters projected on the hypersphere. It is impossible to make such a projection on the whole hypersphere without violating topological constraints. In the one-dimensional case with $X \in [-1, +1]$ the (X, X_r) vector should not make a full circle when X is changed from -1 to $+1$ because the two extreme vectors $X = \pm 1$ will then be identical. An optimal separation for three vectors with the lengths $||X||, ||X|| + \Delta, ||X|| + 2\Delta$ is to place them in corners of an equilateral triangle, for example at angles $0, \pm 120°$. One can search for the best input preprocessing treating it as a rigorous optimization problem, or just use polar coordinates to shift some upper hemisphere vectors to the part of the lower hemisphere. A much simpler approach is to rescale all vectors to get their Euclidean norms ≤ 1, use the norm $||X||$ mapping it to points on a circle: $(\sin\frac{\pi}{3}(4-5||X||), \cos\frac{\pi}{3}(4-5||X||))$. These points for $0 \leq ||X|| \leq 1$ are within the angle $-\pi/3$ and $4\pi/3$. The first factor, $\sin\frac{\pi}{3}(4-5||X||)$ is used to rescale components of the vector $\mathbf{X}$, while the second factor is taken as an extra X_r component. The extended vectors $||(\mathbf{X}^j, X_r^j)||_D$ are renormalized using the metric function $D(\cdot)$, placing them on a unit sphere defined by this metric.

6. Illustrative Example

The influence of non-Euclidean distance functions on the decision borders is illustrated here on the classical Iris flowers dataset, containing 50 cases in each of the three classes. The flowers are described by 4 measurements (petal and sepal width and length). Two classes, Iris virginica and Iris versicolor, overlap, and therefore a perfect partition of the input space into separate classes is not possible. An optimal solution (from the point of view of generalization) contains three errors and is obtained using only two of the four input features (x_3 and x_4), therefore it is easy to display and only those two features have been left in the simulations described below.

A standard MLP solution is obtained with 2 input, 4 hidden and 3 output neurons, with a total of 27 adaptive parameters. One discriminating plane per class for the smallest and the largest flowers (setosa and virginica) is needed and two planes are needed to separate the vectors of the versicolor class. To increase the accuracy and speed up learning, in the final phase of learning only the vectors near the class borders were presented to the network. The selection algorithm loops over all vectors and for a given vector $\mathbf{X}$ it finds k (for example $k = 10$) nearest vectors belonging to a different class than $\mathbf{X}$. These vectors are written to a new training file providing a description of the border region. This method of training leads to sharper and more accurate decision borders, as seen in the first drawing of Fig. 6.

An additional input feature was added and the 3-dimensional vectors normalized using various Minkovsky distance measures. The network was initialized taking the normalized weights equal to the centers of the three clusters. In the extended feature space the same accuracy is achieved using only 3 input and 3 output neurons without the hidden layer, for a total of 12 adaptive parameters. Using such a network architecture with the squared Euclidean metric and the weight $W_3 = 0$ for the third X_3 component, only two classes are separated. The transition process from the planar to circular decision borders is shown in Fig. 5 (clockwise, from top left). In the learning process W_3 grows, curving the borders for the vectors near the center of the drawing.

The data were standardized and rescaled in order to fit them inside a square with ± 1 corners. An additional feature was added and the 3-dimensional vectors normalized using various Minkovsky distance measures. The network was initialized taking the normalized weights equal to the centers of the three clusters. In the extended feature space only 3 neurons are necessary. In Fig. 6 dramatic changes in the shapes of decision borders for the Minkovsky metric are observed. Using the squared Euclidean metric in $\sigma(d_0 - D(\mathbf{X}, \mathbf{R}))$ transfer functions a standard MLP solution is obtained. The Euclidean case corresponds to circular decision borders, the city block metric $\alpha = 1$ gives sharp, diamond-like shapes, for large α almost rectangular decision borders are obtained (an approximation using logical rules is in this case straightforward) while for small α hypocycloidal shapes are created. Since smooth transition between these cases is made, α should be treated as an adaptive parameter. For the Iris data the optimal solutions (3 errors) were recovered for all values of $\alpha \geq 0.8$. A smooth transition between these cases is made by changing α and retraining the network. For other datasets we have found significant improvements in the accuracy for the optimized α.

7. Discussion

Building robust neural networks to use them as decision trees, is still a challenge. New neural models described here, such as the D-MLP networks with non-Euclidean distance functions, are one of many realizations of similarity based methods. The non-Euclidean transformation of the input vectors leads to very flexible shapes of neural network decision borders without any change in the standard computer programs. The training times are short since a good initialization procedure based on clusterization techniques determines the weights, thresholds and slopes of all neurons. The number of neurons in the network defined in the extended space may also decrease, as has been observed in the Iris example. A new method to treat symbolic values and a new training procedure using only the vectors close to the decision borders have been used here. Since the training is fast, many different metric functions may be tried before selecting (using crossvalidation tests) the best model. Networks with the activation given by eqn. (10) or (11) have not been implemented yet, but such models seem to be quite promising.

The change in the shapes of decision borders has been accomplished before by adding a new type of units to neural networks. For example, Ridella *et al.* (1997) used circular units in their Circular Backpropagation Networks. A different type of circular units was used by Kirby and Miranda (1996). In their implementation two sigmoidal units are coupled together and their output is restricted to lie on a unit circle. Dorffner (1994) proposed conic section transfer functions as a unified framework for MLP and RBF networks. Straight lines and ellipses are special cases of conic sections. The method presented here may be treated as a generalization of the circular or conical unit methods. It is not restricted to MLP neural networks, but can be used with any neural network and any classifier.

Neural models derived from the SBM framework solve not only classification, but also pattern completion and associative memory problems. An additional advantage of the approach outlined here is the understanding of what these networks have really learned in terms of the prototypes (weights) and the weighted distances from these prototypes. A challenge for the neural networks is combinatorial productivity and extraction of knowledge from complex datasets. First steps in this direction have already been done (Duch and Diercksen, 1995; Duch *et al.*, 2000).

Acknowledgments

Support by the State Committee for Scientific Research, under grant No. 8T11C 006 19, and by the German-Polish grant IBMBF-KBN POL-040-98 is gratefully acknowledged.

References

Bishop C. (1995): *Neural Networks for Pattern Recognition.* — Oxford: Clarendon Press.

Breiman L. (1996): *Bagging predictors.* — Machine Learn., Vol.24, pp.123–140.

Cortes C. and Vapnik V. (1995): *Support vector networks.* — Machine Learn., Vol.20, pp.273–297.

Dorffner G. (1994): *A unified framework for of MLPs and RBFNs: Introducing conic section function networks.* — Cybern. Syst., Vol.25, pp.511–554.

Duch W. (1997): *Neural minimal distance methods.* — Proc. 3rd Nat. Conf. *Neural Networks and Their Applications*, Kule, Poland, pp.183–188.

Duch W. and Diercksen G.H.F. (1995): *Feature space mapping as a universal adaptive system.* — Comp. Phys. Communic, Vol.87, pp.341–371.

Duch W. and Jankowski N. (1999): *Survey of neural transfer functions.* — Neural Comput. Surveys, Vol.2, pp.163–213.

Duch W., Adamczak R. and Diercksen G.H.F. (1999): *Neural networks in non-Euclidean spaces.* — Neural Process. Lett., Vol.10, No.3, pp.201–210.

Duch W., Adamczak R. and Grąbczewski K. (1998): *Extraction of logical rules from backpropagation networks.* — Neural Process. Lett., Vol.7, No.3, pp.211–219.

Duch W., Adamczak R. and Grąbczewski K. (2000): *Methodology of extraction, optimization and application of crisp and fuzzy logical rules.* — IEEE Trans. Neural Networks (in print).

Duch W., Adamczak R. and Jankowski N. (1997): *Initialization and optimization of multilayer perceptrons.* — Proc. 3rd Nat. Conf. *Neural Networks and Their Applications*, Kule, Poland, pp.105–110.

Kirby M.J. and Miranda R. (1996): *Circular nodes in neural networks.* — Neural Computat., Vol.8, No.2, pp.390–402.

Kohonen T. (1995): *Self-Organizing Maps.* — Berlin: Springer-Verlag.

Krishnaiah P.R. and Kanal L.N. (Eds.) (1982): *Handbook of Statistics 2: Classification, Pattern Recognition and Reduction of Dimensionality.* — Amsterdam: North Holland.

Lippmann R.P. (1987): *An introduction to computing with neural nets.* — IEEE Mag. Acoust. Signal Speech Process., Vol.4, pp.4–22.

Mertz C.J. and Murphy P.M. (2000): *UCI repository of machine learning databases,* `http://www.ics.uci.edu/pub/machine-learning-data-bases`.

Michie D., Spiegelhalter D.J. and Taylor C.C. (1994): *Machine Learning, Neural and Statistical Classification.* — London: Elis Horwood.

Reilly D.L., Cooper L.N. and Elbaum C. (1982): *A neural model for category learning.* — Biol. Cybern. Vol.45, pp.35–41.

Ridella S., Rovetta S. and Zunino R. (1997): *Circular Backpropagation Networks for Classification.* — IEEE Trans. Neural Networks, Vol.8, No.1, pp.84–97.

Rubin D.B. (1996): *Multiple imputation after 18+ years.* — J. Amer. Statist. Assoc., Vol.91, pp.473–489.

Schiffman W., Joost M. and Werner R. (1993): *Comparison of optimal backpropagation algorithms.* — Proc. *ESANN'93*, Brussels, pp.97–104.

Skapura D.M. (1996): *Building Neural Networks.* — Reading, MA: Addison-Wesley.

Wasserman P.D. (1993): *Advanced methods in neural networks.* — New York: Van Nostrand Reinhold.

Wilson D.R. and Martinez T.R. (1997): *Improved heterogenous distance functions.* — J. Artif. Intell. Res., Vol.6, pp.1–34.

Yao X. (1993): *A review of evolutionary neural networks.* — Int. J. Intell. Syst., Vol.8, pp.539–567.