NEURAL NETWORKS IN NON-EUCLIDEAN SPACES.

Włodzisław Duch (duch@phys.uni.torun.pl)* and Rafał Adamczak (raad@phys.uni.torun.pl)
Department of Computer Methods, Nicholas Copernicus University, Grudziądzka 5, 87-100 Toruń, Poland

Geerd H.F. Diercksen (gdiercksen@mpa-garching.mpg.de)
Max-Planck Institute of Astrophysics, 85740-Garching, Germany

Abstract. Multilayer Perceptrons (MLPs) use scalar products to compute weighted activation of neurons providing decision borders using combinations of soft hyperplanes. The weighted fun-in activation function may be replaced by a distance function between the inputs and the weights, offering a natural generalization of the standard MLP model. Non-Euclidean distance functions may also be introduced by normalization of the input vectors into an extended feature space. Both approaches influence the shapes of decision borders dramatically. An illustrative example showing these changes is provided.

Keywords: neural networks, learning, minimal distance methods, similarity-based methods, machine learning, interpretation of neural functions, classification

1. Introduction

Multilayer perceptrons (MLPs) trained with backpropagation method (BP) are certainly the most popular neural technique [1]. Applied to classification problems MLPs provide soft hyperplanes dividing the input space into separate regions. MLPs are therefore similar to the statistical discriminant techniques, although soft sigmoids allow for representation of more complex, nonlinear decision borders. This is usually considered to be a strength of the MLP model, although in cases when sharp decision 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. Increasing the slopes of sigmoidal functions to improve representation of such decision borders around the (1,1) point leads to problems with learning by backpropagation, or by any other gradient-based method, since the volume of the input space in which sigmoids change rapidly (and thus gradients are non-zero) is rapidly shrinking. In the limit sigmoidal functions become step-functions but gradient techniques like backpropagation cannot be used to make this transition. As a result for some datasets no change in learning rule or network architecture will improve the accuracy of neural

* Support by the Polish Committee for Scientific Research, grant 8T11F 014 14, and by the Max-Planck Foundation is gratefully acknowledged.

solutions. A good real-world example is the hypothyroid dataset, for which the best optimized MLPs still give about 1.5% of error [2] while logical rules reduce it to 0.64% (since 3428 cases are provided for testing this is a significant improvement). 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 [3].

Recently a general framework for similarity-based classification methods (SBMs) has been presented [4]. Investigation of connections between neural network and similarity-based methods leads to a number of new neural network models. In particular the distance-based MLP (D-MLP) networks are obtained by replacing the weighted activation with a square of Euclidean distance [5]. Such networks improve upon the traditional approach by providing more flexible decision borders and by enabling a prototype-based interpretation of the results. Since the use of distance functions (instead of weighted activation) in neural network models is a novel idea it is described in the next section. In the third section transformation of the input data to the extended feature space is proposed, enabling the use of non-Euclidean distance functions in the standard MLP backpropagation programs without the need for coding the new transfer functions and their derivatives. The fourth section shows how to determine the architecture and parameters of the network, including the slopes for each neuron. An illustration of this method on the Iris data is presented for pedagogical purposes in the fifth section. The paper is finished with a short discussion.

2. Distance functions in neural networks

The classification problem (the same reasoning may also be applied to regression and pattern completion problems) is stated as follows: given a set of class-labeled training vectors $\{\mathbf{R}^j, \mathbf{C}(\mathbf{R}^j)\}$, $j = 1..N_t$, where $\mathbf{C}(\mathbf{R}^j)$ 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 of classification $p(C_i|\mathbf{X}; M)$, where M describes the classification model used (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:
$M = \{\{\mathbf{R}^j\}, D(\cdot), G(D(\cdot)), k, E[\cdot]\}$, 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 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 depends upon 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 $\mathcal{R}(C_i|C_j)$ of assigning wrong classes. An adaptive system may include several such models M_l and an interpolation procedure to select between different models or average results of a committee of models.

In RBF networks Euclidean distance functions $D(\mathbf{X},\mathbf{R}^j) = ||\mathbf{X}-\mathbf{R}^j||$ are assumed and 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 distance function and the weighting function. Any distance function $D(\mathbf{X},\mathbf{R})$ and the weighting function $G(D)$ may be used to create neural network. In the Gaussian classifier [6] or in the original RBF network only one parameter, dispersion, was optimized [7]. Optimization of the positions of the reference centers $\mathbf{R}^j$ leads to the LVQ method [8] in which the training set vectors are used to define the initial prototypes and the minimal distance rule is used to assign the classes. The Restricted Coulomb Energy (RCE) classifier [9] uses a hard-sphere weighting functions. The Feature Space Mapping model (FSM) is based on separable, rather than radial weighting functions [10]. Very recently a method to create oblique probability distributions in N-dimensional space using only N parameters has been described [3].

MLPs and other networks using discriminant functions are also special cases of general SBM framework. 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, neuron with N inputs and the threshold θ realizes the following function:

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

where $||\cdot||$ norm 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 and the activity of threshold neuron as activation by inputs falling into a hard sphere of radius θ centered at $\mathbf{W}$. The Hamming neural network [11] is actually a neural realization of the nearest neighbor method for a single neighbor and binary inputs. Changing binary into real values and threshold into sigmoidal neurons for inputs normalized to $||\mathbf{X}|| = ||\mathbf{W}|| = 1$ leads to soft activation of neurons by input vectors close to $\mathbf{W}$ on a unit sphere. In general the activation of a neuron is written as:

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

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

$$\sigma(\mathbf{W}\cdot\mathbf{X}+\theta)=\sigma(d_0-D(\mathbf{W},\mathbf{X})) \quad (3)$$

where $D(\mathbf{W},\mathbf{X})$ is the square of Euclidean distance between $\mathbf{W}$ and $\mathbf{X}$ and the $1/2$ factor is absorbed in the sigmoid's slope. This function evaluates the influence of the reference vectors $\mathbf{W}$ on the classification probability $p(C_i|\mathbf{X};\{\mathbf{W},\theta\})$. To avoid loss of information during normalization of input vectors an additional component X_r is added, a new feature measuring the difference between the largest norm and the norm of original input vectors.

Transfer function $f(D(\mathbf{W},\mathbf{X}))=\sigma(d_0-D(\mathbf{W},\mathbf{X}))$ decreases monotonically as a function of distance, with flat plateau for small distances D, 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 sigmoid arguments belong to the $[\theta-|\mathbf{W}|,\theta+|\mathbf{W}|]$ interval. A unipolar sigmoid has its maximum curvature around ± 2.4, therefore smaller thresholds and norms of the weights 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.

From the similarity-based point of view MLP networks use sigmoidal functions to estimate the influence of weight vectors according to distance between the weight and the training vectors, combining many such estimations to compute the final output. Changing the distance function in equation (3) from the square of the Euclidean distance to some other distance measures new types of neural networks, called D-MLP networks [5], are defined. Another possibility is to write the weighted product in the form:

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

Euclidean norms may be replaced by Minkovsky or other type of norms. Backpropagation procedure requires derivatives of the distance functions, but for Minkovsky and other popular functions they are easily provided. Generalized Minkovsky's distance with the scaling factors is given by:

$$D(\mathbf{A},\mathbf{B};s)^\beta=\sum_i^N s_i d(\mathbf{A}_i,\mathbf{B}_i)^\alpha \quad (5)$$

where $\beta=\alpha$ is usually taken. The $d(\cdot)$ function is used to estimate similarity at the feature level and in the simplest case $d(\mathbf{A}_i,\mathbf{B}_i)=|\mathbf{A}_i-\mathbf{B}_i|$. For $\alpha=\beta=2$ the vectors $||\mathbf{A}||=1$ are on the unit sphere, for large α the sphere is changed into a soft cuboid, for $\alpha=1$ it has pyramidal and for $\alpha<1$ hypocycloidal shape.

Thus using non-Euclidean distance activation functions changes the shape of decision borders completely, from the usual hyperplanes ($\beta = 1,\ \alpha = 2$ and $W_r = 0$ for the weight corresponding to the X_r component) to spherical, cuboidal or hypocycloidal. Derivation of the backpropagation equations for $\sigma(d_0 - D(\mathbf{X}, \mathbf{W}))$ functions with generalized Minkovsky distances is straightforward but requires extensive modification of standard MLP software. In the next section a simpler way of using the non-Euclidean distance functions is introduced.

3. 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 on 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. In memory-based reasoning the Modified Value Difference Metric (MVDM) has gained popularity [12]. The distance between two N-dimensional vectors $\mathbf{A}, \mathbf{B}$ with discrete (nominal, symbolic) elements, in a K class problem, is computed using conditional probabilities:

$$D_V^{\alpha}(A, B) = \sum_j^N \sum_i^K \left| p(C_i|A_j) - p(C_i|B_j) \right|^{\alpha} \tag{6}$$

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 to compute $D_V(\mathbf{A}, \mathbf{B})$ as a sum of value differences over all features. 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 continuos values requires a set of probability density functions $p_{ij}(x)$, with $i = 1..K, j = 1..N$.

Using 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 dimension equal to the number of classes times the number of different symbolic values the feature takes) allows to reproduce MVDM distances using numerical values of vector components. Many other types of metric functions exist [12] 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 3 vectors with the length $||X||, ||X|| + \Delta, ||X|| + 2\Delta$ is to place them in corners of 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. 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: $\left(\sin\frac{\pi}{3}(4-5||X||), \cos\frac{\pi}{3}(4-5||X||)\right)$. 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. 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.

4. Initialization of the network

The network should be initialized taking the centers of clusters in the extended space as $\mathbf{W}$ and taking $d_0 = D(\mathbf{W}, \mathbf{X}^b)$, where $\mathbf{X}^b$ is a vector at the border of the given cluster (we have tried [13] dendrograms and decision trees but other clusterization methods may also be used for initialization [6]). Using weighted activation the contribution of a center of an input data 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 0.5 value tangentially to the unit hypersphere. 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 has an influence on 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:

$$\begin{aligned}(\mathbf{W}\cdot\mathbf{X}+\theta)/T &= (\frac{\mathbf{W}}{||\mathbf{W}||}\cdot\mathbf{X}+\frac{\theta}{||\mathbf{W}||})||\mathbf{W}||/T \\ =(\mathbf{W}'\cdot\mathbf{X}+\theta')/T' &= (d_0'-D(\mathbf{W}',\mathbf{X}))/T' \end{aligned} \quad (7)$$

Thus without loss of generality both $\mathbf{X}$ and $\mathbf{W}'$ may be normalized. No special learning for the slopes is required. 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 size of the cluster, i.e. $\Delta(T) = \pm d_0/10$ then setting $T \approx d_0/24$ will be appropriate. After such initialization training of the network is usually quite short.

In the XOR case the input vectors for class = T are $(0,1),(1,0)$ and for the class = F 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, 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 for the XOR problem found without any optimization of the network! For more complex examples of this type of initialization see [13]. 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 may be tried on a given problem.

5. Pedagogical illustration

The influence of input renormalization (using non-Euclidean distance functions) on the shapes of decision borders is illustrated below on the classical Iris flowers dataset, containing 150 cases divided into 3 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 3 errors [14] and may be obtained using only two of the four input features (x_3 and x_4), therefore results are easy to display and only those two features have been left in simulations described below.

A standard MLP solution is obtained with 4 hidden neurons and 3 output neurons. One discriminating plane per class of the smallest and the largest

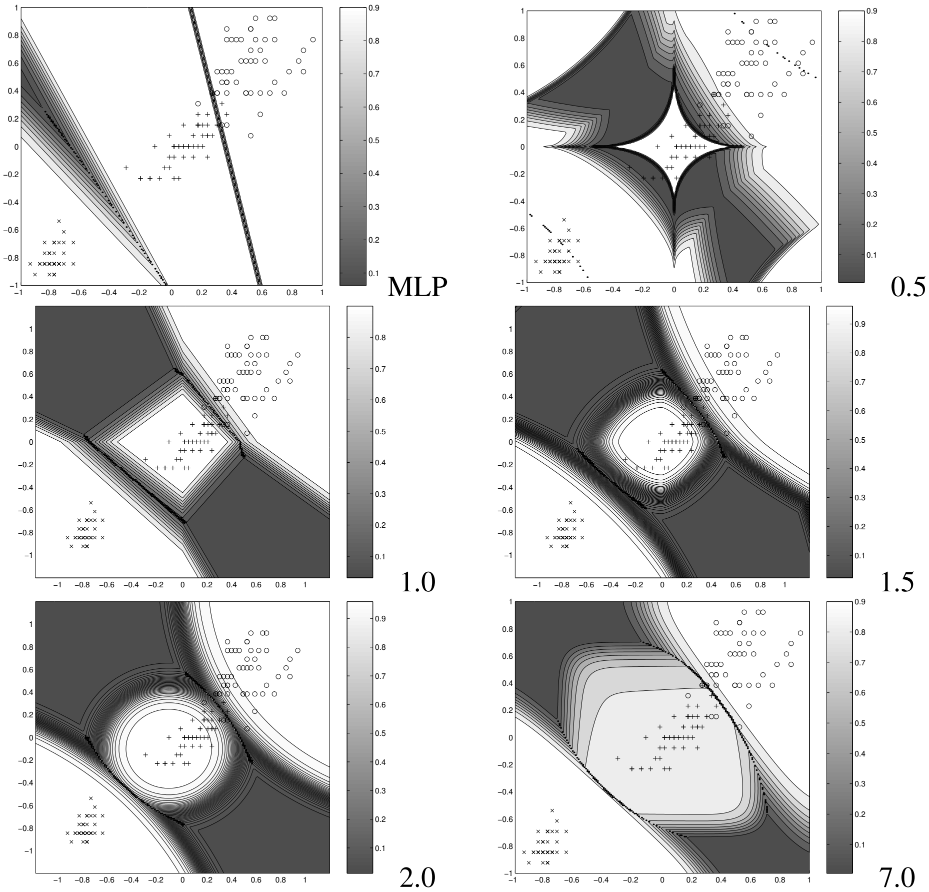

Figure 1. Shapes of decision borders in the Iris case for standard MLP (4 neurons, 2 inputs) solution and for MLP (3 neurons, 3 inputs) using the data vectors renormalized with Minkovsky metric, $\alpha = 0.5, 1.0, 1.5, 2.0$ and 7.0.

flowers (setosa and virginica) is needed and two planes to separate the vectors of the versicolor class. To increase accuracy and speed up the 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 $\mathbf{X}$ 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.

The data has been standardized and rescaled to fit inside a square with ± 1 corners. An additional input feature has been added and the 3-dimensional vectors normalized using various Minkovsky distance measures. The network has been initialized taking the normalized weights that are equal to the centers of the three clusters. In the extended feature space the same accuracy is achieved using only 3 hidden neurons without an output layer. In Fig. 1 dramatic changes in the shapes of decision borders for Minkovsky metric are observed. Using squared Euclidean metric with $\sigma(d_0 - D(\mathbf{X},\mathbf{R}))$ transfer functions and $W_3 = 0$ the standard MLP solution is obtained. Euclidean case corresponds to circular decision borders, the city block metric $\alpha = 1$ gives sharp, romboidal 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 solution (3 errors) has been recovered for all values of $\alpha \geq 0.8$, but for other datasets we have found significant improvements of accuracy for optimized α.

6. Discussion

Non-Euclidean transformation of 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 weights, thresholds and slopes of all neurons. The number of neurons and the complexity of the network defined in extended space is usually smaller comparing to the standard MLPs needed to obtain similar accuracy on the original data, 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 described here. Since the training is fast many different metric functions may be tried before selecting (using crossvalidation tests) the best model. Networks with activation given by Eq.(3) or (4) have not yet been implemented but such models seem to be quite promising.

The change of the shapes of decision borders has been accomplished before by adding new type of units to neural networks. For example, Ridella *et al.* [15] used circular units in their Circular Backpropagation Networks. Different type of circular units have been used by Kirby and Miranda [16] – in their implementation two sigmoidal units are coupled together and their output is restricted to lie on a unit circle. Dorffner [17] proposed conic section transfer functions as a unified framework for MLP and RBF networks. Straight lines and ellipses are special cases of conic sections. Non-Euclidean metrics have been used to characterize the manifold characterizing Boltzman machines and EM algorithms by Amari [18], but his approach is completely different than ours. The method presented here may be treated as a generalization of the circular or conical unit method. It is not restricted to MLP neural networks, but can be used with any neural network and any classifier.

An additional advantage of our approach is the understanding of what the network has really learned in terms of the prototypes (weights) and sigmoidally weighted distances from these prototypes. Many ideas presented in this paper are now being tested empirically.

References

1. Bishop C, *Neural networks for pattern recognition*. Clarendon Press, Oxford, 1995
2. W. Schiffman, M. Joost, R. Werner, Comparison of optimized backpropagation algorithms, Proc. of ESANN'93, Brussels 1993, pp. 97-104
3. W. Duch, N. Jankowski, New neural transfer functions, Applied Mathematics and Computer Science 7 (1997) 639-658
4. W. Duch, Neural minimal distance methods, Proc. 3-rd Conf. on Neural Networks and Their Applications, Kule, Poland, Oct. 14-18, 1997, pp. 183-188; W. Duch, K. Grudzinski, G.H.F. Diercksen, Minimal distance neural methods. World Congress of Computational Intelligence, May 1998, Anchorage, Alaska, IJCNN'98 Proceedings, pp. 1299-1304
5. W. Duch, R. Adamczak, G.H.F. Diercksen, Distance-based multilayer perceptrons, In: Computational Intelligence for Modelling Control and Automation. Neural Networks and Advanced Control Strategies. Ed. M. Mohammadian, IOS Press, Amsterdam, pp. 75-80
6. P.R. Krishnaiah, L.N. Kanal, eds, Handbook of statistics 2: classification, pattern recognition and reduction of dimensionality (North Holland, Amsterdam 1982)
7. P.D. Wasserman, Advanced methods in neural networks (van Nostrand Reinhold 1993)
8. T. Kohonen, Self-organizing maps (Berlin, Springer-Verlag 1995).
9. D.L. Reilly, L.N. Cooper, C. Elbaum, A neural model for category learning, Biological Cybernetics 45 (1982) 35–41
10. W. Duch, G.H.F. Diercksen, Feature Space Mapping as a universal adaptive system, Comp. Phys. Communic. **87** (1995) 341-371
11. 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
12. D.R. Wilson, T.R. Martinez, Improved heterogenous distance functions. J. Artificial Intelligence Research 6 (1997) 1-34
13. W. Duch, R. Adamczak, N. Jankowski, Initialization and optimization of multilayered perceptrons, 3rd Conf. on Neural Networks and Their Applications, Kule, Poland, October 1997, pp. 105-110
14. Duch W, Adamczak R, Grąbczewski K, Żal G, Hybrid neural-global minimization method of logical rule extraction, Journal of Advanced Computational Intelligence (in print)
15. S. Ridella, S. Rovetta, R. Zunino, Circular Backpropagation Networks for Classification, *IEEE Trans. Neural Networks* 8 (1997) 84–97
16. M.J. Kirby, R. Miranda, Circular Nodes in Neural Networks, *Neural Computations* 8 (1996) 390-402
17. G. Dorffner, A Unified Framework for of MLPs and RBFNs: Introducing Conic Section Function Networks, *Cybernetics & Systems* 25 (1994) 511-554
18. S-i. Amari, Information geometry of the EM and em algorithms for neural networks, Neural Networks 8 (1995) 1379-1408