# Taxonomy of neural transfer functions

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

**Abstract.**

The choice of transfer functions may strongly influence complexity and performance of neural networks used in classification and approximation tasks. A taxonomy of activation and output functions is proposed, allowing to generate many transfer functions. Several less-known types of transfer functions and new combinations of activation/output functions are described. Functions parameterize to change from localized to delocalized type, functions with activation based on non-Euclidean distance measures, bicentral functions formed from pairs of sigmoids are discussed.

## 1 INTRODUCTION.

From the probabilistic point of view [1] adaptive systems should approximate the density of joint probability $p(X,Y)$ or the posterior probability $p(Y|X)$ of input-output values. Modification of adjustable internal parameters $W$ allows neural networks to learn an arbitrary vector mapping from the space of inputs $X$ to the space of outputs $Y = F_W(X)$. The current focus in neural network research is on learning algorithms and architectures of networks based on sigmoidal or radial basis functions. In approximation theory many basis functions are used (cf. [2]). Neural networks with sigmoidal or Gaussian functions are universal approximators [3] but are they optimal from the point of view of complexity and difficulty of their training? For some datasets a large (and hard to train) network using sigmoidal functions may be needed for tasks that could be solved with a small (and easy to train) network using other transfer functions.

Consider [4] a classification problem in $N$ dimensions, with vectors belonging to the class $C_1$ inside the unit sphere and $C_2$ outside of this sphere. A single multivariate Gaussian function with $2N$ adaptive parameters (specifying the center and dispersions in each dimension) will solve this task after a short training. To capture any bound region in $N$-dimensional space a multilayer perceptron (MLP) needs $N$ hyperplanes forming a simplex and one additional neuron to smooth the combined output of these neurons. At least $N^2+N$ parameters are needed in this case, making the training process much more difficult. Consider now the $C_1$ data vectors contained in the corner of the coordinate system bound by the $(1,1,...,1)$ plane and $C_2$ vectors outside. A single hyperplane with $N+1$ parameters is sufficient for perfect classification. One Gaussian function placed in the center of the region and $N+1$ Gaussians in the corners, using at least $2N(N+2)$ adaptive parameters, still provide a poor approximation here. Improved learning algorithms or network architectures will not change the relative complexity of solutions as long as the decision borders provided by the transfer functions remain spherical (as in the first example) or planar (as in the second example). Thus we conclude that the type of transfer functions used is an important (although badly neglected) issue. In this paper a taxonomy of transfer functions is presented and several new types of transfer functions proposed.

## 2 TAXONOMY OF TRANSFER FUNCTIONS

*The activation function* determines the total signal a neuron receives. The inner product (IP) function (fan-in function) is most common, $I(\mathbf{x}) = \sum W_i x_i$, with $W_0 = \theta$ (threshold) and $x_0 = 1$. *The output function* $o(I)$ takes the scalar activation as input and returns scalar output values. Typically a squashing function is used to keep the output values within specified bounds. The composition of the activation and the output function is called the *transfer function* $o(I(\mathbf{x}))$. It is defined in the $N$-dimensional *input space*, called also *the parameter space*. The transfer function is *local* if its values are significantly different from zero (i.e. $|o(I(\mathbf{x}))| > \varepsilon$ for some small $\varepsilon$) in a finite area of the input space; otherwise the function is *non-local*.

Statistical methods of classification may be divided into two broad groups: methods based on discrimination, using hyperplanes or other hypersurfaces for tessellation of the input space, and methods based on clusterization, in which

similarities are calculated using some kind of a distance measure. Therefore there are three main choices for the activation functions (Fig. 1):

- The **inner product** $I(\mathbf{x};\mathbf{w}) \propto \mathbf{w}^T \cdot \mathbf{x}$ (as in the MLP networks).
- The **distance-based** activation $D(\mathbf{x};\mathbf{t}) \propto ||\mathbf{x}-\mathbf{t}||$, used to calculate similarity of $\mathbf{x}$ to a prototype vector $\mathbf{t}$.
- A **combination** of the two activations, $A(\mathbf{x};\mathbf{w},\mathbf{t}) \propto \alpha\, \mathbf{w}^T \cdot \mathbf{x} + \beta\, ||\mathbf{x}-\mathbf{t}||$,

In each case either the final scalar value of activation or the vector components of this activation are used. The distance-based form of activation is usually taken in the scalar form $D(\mathbf{x},\mathbf{t})$, but some output functions may use the vector components $D_i \propto (x_i - t_i)^2$. The $D(\mathbf{x},\mathbf{t})$ distance function does not have to be Euclidean. The Minkovsky's distance function is: $D_{Mb}(\mathbf{x},\mathbf{y};\mathbf{b})^\alpha = \sum_i^N d(x_i,y_i)^\alpha / b_i$, where $b_i$ are scaling factors and the $d(\cdot)$ function is used to estimate similarity at the feature level and in the simplest case $d(x_i,y_i) = |x_i - y_i|$. Mahalanobis distance, more general quadratic distance function and many correlation factors are also suitable for metric functions, for example Camberra, Chebychev, Kendall's Rank Correlation or Chi-square distance [5]. All these function may replace the Euclidean distance used in the definition of many transfer functions. For symbolic features the Modified Value Difference Metric (MVDM) is useful [6]. A *value difference* for feature $j$ of two $N$-dimensional vectors $\mathbf{x},\mathbf{y}$ with discrete (symbolic) elements, in a $C$ class problem, is:

$$d_V^q(x_j,y_j) = \sum_i^C \left| p(C_i|x_j) - p(C_i|y_j) \right|^q \tag{1}$$

where $p(C_i|x_j)$ is estimated by calculating the number $N_i(x_j)$ of times the value $x_j$ of the feature $j$ occurred in vectors belonging to class $C_i$, and dividing it by the number $N(x_j)$ of times $x_j$ occurred for any class. $d_V^q(x_j,y_j)$ is used as $d(\cdot)$ in the Minkovsky distance function for symbolic attributes, combined with other distance functions for numerical attributes. Many variants of MVDM have been discussed in the literature and may be used in neural network activation functions [5, 6]. To avoid problems with calculation of gradients instead of using VDM-type metrics directly one may use numerical input vectors obtained by replacing symbolic and discrete attributes with $p(C_i|x_j)$ probabilities. Resulting numerical vectors have the number of components equal to the number of different symbolic values times the number of classes.

The square of the inner product activation function is a quadratic form. Treating all coefficients of this form as independent and transforming it into a canonical form:

$$I^2(\mathbf{x};\mathbf{w}) \sim D^2(\mathbf{x};\mathbf{t},\mathbf{a}) = \sum_i^N a_i (x_i' - t_i)^2 \tag{2}$$

where the new variables $x_i'$ are linear combinations of the original variables $x_i$. If all parameters $a_i$ are positive, $a_i = 1/b_i^2$, then an Euclidean distance function is obtained, providing hyperellipsoidal contours of constant values. Squared fan-in activation function is used in the *Lorentzian* transfer functions, providing a window-type non-localized surfaces of constant density [5].

Identity function is used as the output function in linear networks. The radial coordinate transfer function $||\mathbf{x}-\mathbf{t}||$ has a distance-based activation and identity output function. Most activation functions are unbounded and output functions are used as squashing functions. There are three major choices (Fig. 2):

- Sigmoidal non-local functions.
- Functions localized around a single center.
- Semi-centralized functions that have several centers.

Systematic combination of activation and output functions leads to many useful transfer functions, some of which have already been introduced in the literature (cf. review [5]). Linear transfer functions, combination of the scalar inner product (IP) activation with identity output function, are frequently used for input and output layers. Combining IP with the step function or multi-step function, transfer functions of logical neurons are obtained. They provide decision borders in the form of hyperplanes rotated by the $W_{ij}$ coefficients, dividing the input space into polyhedral areas.

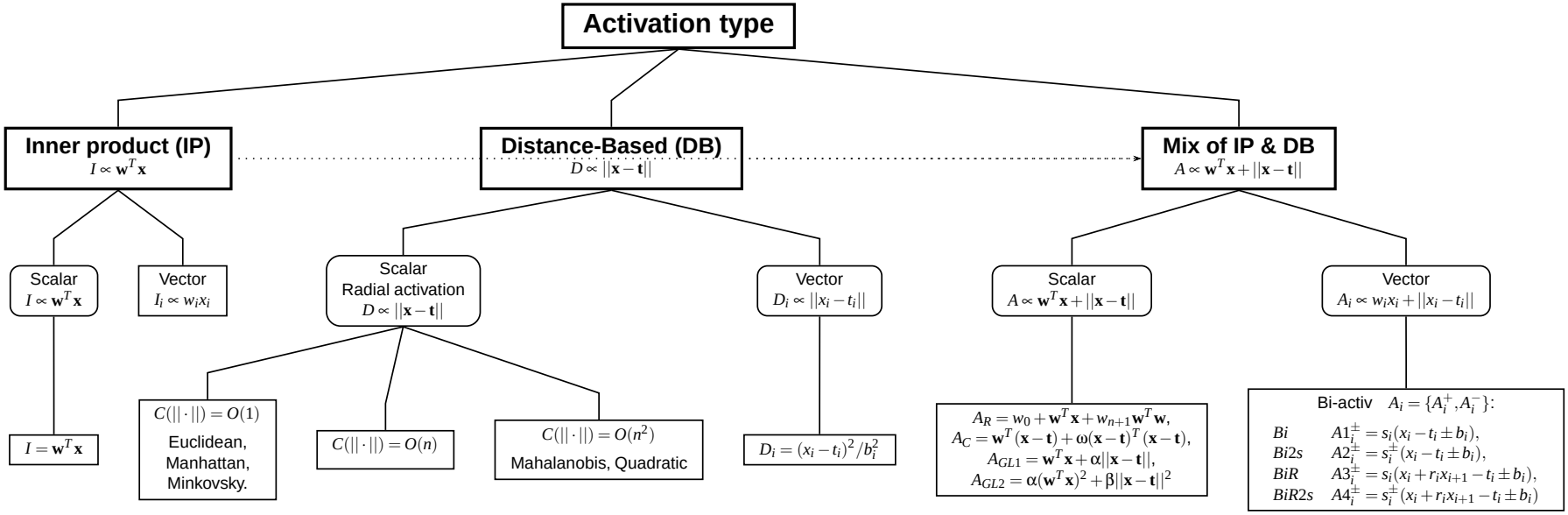


Figure 1: Taxonomy of activation functions. $C(||\cdot||)$ is the number of adaptive parameters of $||\cdot||$ norm.

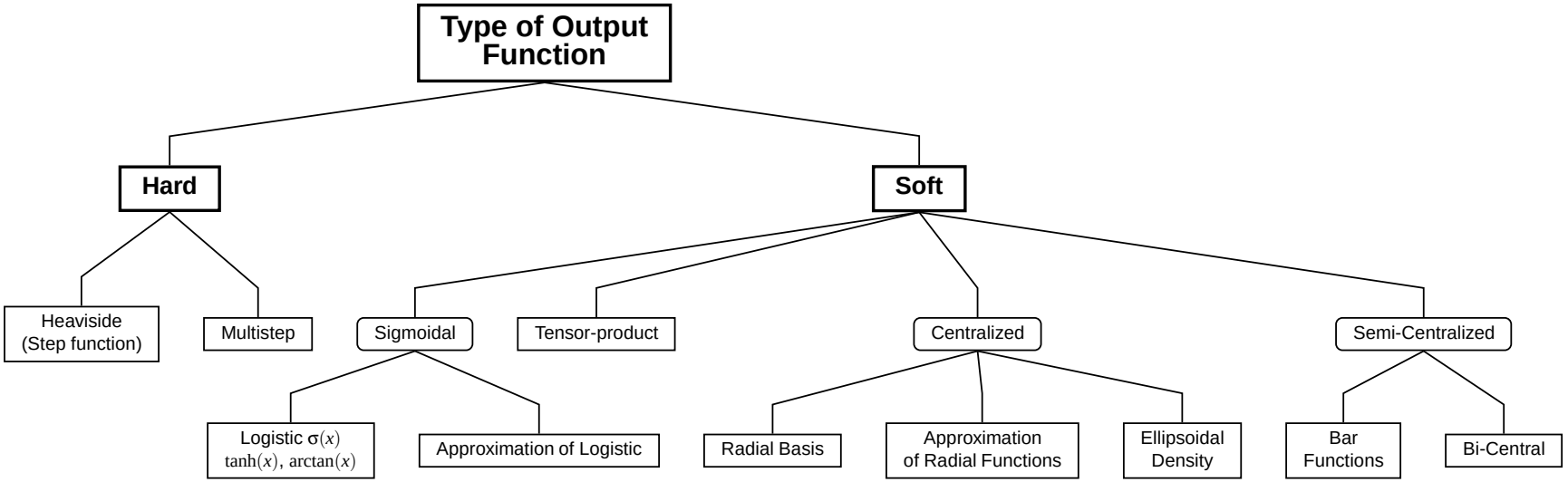


Figure 2: Taxonomy of output functions.

The *graded response neurons* used most often in the literature are based on the soft *sigmoidal output* functions, for example logistic functions $\sigma(I/s) = 1/\left(1+e^{-I/s}\right)$, combined with scalar IP activation. These transfer functions are non-local. Sigmoidal output functions allow to construct wide-margin classifiers, smoothing out many shallow local minima in the total output function of the network. The usefulness of sigmoidal transfer functions seems to be related to the usefulness of Gaussian functions in statistics. Sigmoidal transfer function appear in expressions on probability $p(x > a)$ of measurements $x$ which have Gaussian-type uncertainties [7]. Since observations are not quite accurate, instead of a number $y$ a Gaussian distribution $G_y = G(y;\overline{y},s_y)$ centered around $\overline{y}$ with dispersion $s_y$ should be given. This distribution may be treated as a membership function of a fuzzy number $G_y$. The cumulative distribution $p(x-\overline{y}) = \int_{-\infty}^{x} G(y;\overline{y},s_y)dy$ may be approximated by $\sigma((x-\overline{y})/T)$, where $T = \sqrt{2}s_y/2.4$, with accuracy better than 0.02 for all $x$. The $p(x-\overline{y})$ distribution may be interpreted as the probability of a certain decision rule $R_x(z) =$True iff $z \leq x$ is true, i.e. $p(R_x|G_y) = p(x-\overline{y})$. The Gaussian assumption for the uncertainty of inputs is equivalent to the *soft trapeziodal* membership functions of the logical rules used with the sharply defined inputs. On the other hand starting with sigmoidal functions instead of the erf function is equivalent to the assumption that the uncertainties are given by $\sigma((x-\overline{y})/T)(1-\sigma((x-\overline{y})/T))$, approximating the Gaussian function within a few percent.

Logistic functions may be replaced by the error (erf) function, arcus tangent or the hyperbolic tangent functions. Since calculation of exponents is much slower than simple arithmetic operations, other functions of sigmoidal shape may be useful to speed up computations, for example $s_1(I;s) = I(\mathrm{sgn}(I)I - s)/(I^2 - s^2)$ with derivative $s_1'(I;s) = s/(I + \mathrm{sgn}(I)s)^2$ (other approximations are presented in [5]).

The scalar distance-based activation (DB) may be used with the step or multistep functions, defining hard sphere type of transfer functions. Such spheres with fixed radius are used in the Restricted Coulomb Energy classifiers [8]. In the $k$-nearest neighbor ($k$-NN) method spheres with variable radius, covering exactly $k$ vectors, are used. Combining sigmoidal output functions with scalar DB activation leads to "soft" versions of these methods. Logistic function $G_1(D) = 1 - \sigma(D^2) = 1/(1+\exp(D^2))$, hyperbolic tangent $G_2(D) = 1 - \tanh(D^2)$ function and their approximations have ellipsoidal contours with Gaussian-like shapes and may be used in RBF networks. For $N$-dimensional input space each ellipsoidal unit uses $2N$ adaptive parameters. Using the Mahalanobis distance function $D_M(\mathbf{x};\mathbf{t})$ with (symmetric) covariance matrix $\Sigma$ rotation of hyperellipsoids is introduced, but the number of adaptive parameters per each function becomes large. Simpler units giving approximately ellipsoidal densities are also useful (cf. [5]).

Radial functions were used in approximation theory [2] and in pattern recognition under different names (cf. Gaussian classifiers and potential function approach, [10]) since a long time. RBFs take the scalar DB activation $D = ||\mathbf{x} - \mathbf{t}||$ and one of the radial basis output functions or their approximations. Except for $o(I) = I$ (identity function) multiquadratic functions $s_m(r;b) = \sqrt{b^2 + r^2}$, nonlocal general multiquadratics, thin-plate spline functions and other non-local radial output functions are used [9]. Among localized radial basis functions *Gaussian functions* are unique since for all distance functions that may be presented as a sum of independent components they are separable, $G(r,b) = \exp(-D^2/b^2)$. Inverse quadratic function $G_3(D) = 1/(1+D^2)$ or quartic function have similar shape but are not separable. In the $N$-dimensional case a center is described by $N$ coordinates $\mathbf{t}_i$. Together with the scaling factors $b_i$ (dispersions for Gaussians) $2N$ adaptive parameters for each neural unit are provided.

Vector form of DB activation in combination with one-dimensional Gaussian output functions is called *Gaussian bar functions* [11] $\bar{G}(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{v}) = \sum_{i=1}^{N} v_i \exp(-(x_i - t_i)^2/b_i^2)$ and has been introduced to enable feature selection. $3N$ adjustable parameters are needed per processing unit. Except for a single maximum around center $\mathbf{t}$ in $N$-dimensions these functions involve Gaussians in $N-1$ dimensional subspaces. For some data a smaller number of bar functions than multidimensional Gaussians is sufficient to reach similar accuracy. Combination of vector DB activation with sigmoidal functions creates *sigmoidal bar functions* that have not been used so far: $\bar{\sigma}(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{v}) = \sum_{i=1}^{N} v_i/(1+\exp(-(x_i-t_i)^2/b_i^2)) = \sum_{i=1}^{N} v_i \sigma((x_i-t_i)^2/b_i^2)$.

Mixing the inner product and distance-based activation leads to the most interesting *universal transfer functions* that may become local or nonlocal, depending on the values of their adaptive parameters. Linear terms used to calculate $I(\mathbf{x};\mathbf{w},\theta)$ activation and quadratic terms used in Euclidean distance measures combined together create functions that for some parameters are localized, and for other parameters non-localized. Ridella *et al*. [12] use circular units in their Circular Backpropagation Networks. The output function is a standard sigmoid while the activation function contains one quadratic term $w_{N+1}\sum_{i=1}^{N} x_i^2$ and may also be presented in the form of a distance function with:

$$A_R(\mathbf{x};\mathbf{w}) = d_c(\mathbf{x};\mathbf{c}) = (||\mathbf{x}-\mathbf{c}||^2 - \theta)w_{N+1} \tag{3}$$

where $c_i = -w_i/2w_{N+1}$ and $\theta = \left(\sum_{i=1}^{N} w_i^2/4w_{N+1}^2 - w_0\right)/w_{N+1}$. These units perform very well in the standard backpropagation network, providing an optimal solution in classification problems [12].

Dorffner [13] proposed *conic section* transfer functions as a unified framework for the MLP and RBF networks. Straight lines and ellipses are special cases of conic sections. From geometrical considerations Dorffner proposes a combination of IP and DB activation functions $A_C(\mathbf{x};\mathbf{w},\mathbf{t},W_D) = I(\mathbf{x}-\mathbf{t};\mathbf{w}) + W_D D(\mathbf{x}-\mathbf{t})$. This activation is then composed with the standard sigmoidal function to produce the conical transfer function.

Many other combinations of fan-in and distance functions could also serve as universal transfer functions. For example, $\exp(\alpha I^2 - \beta D^2)$ or the approximated Gaussian combined with the Lorentzian function also provide an interesting universal transfer function, $C_{GL1}(\mathbf{x};\mathbf{w},\mathbf{t},\alpha,\theta) = 1/\left(1 + (I(\mathbf{x};\mathbf{w}) + \alpha D(\mathbf{x};\mathbf{t}))^2\right)$, or $A_{GL2}$ activation (Fig. 1). The $\alpha$ parameter scales the relative importance of the linear, non-localized terms. The number of adaptive parameters in this case is equal to $2N+1$ (no scaling factors in distance function) or $3N+1$ (separate distance scaling factors for each dimensions). Unfortunately these functions are not separable.

Sigmoidal functions may be combined into a *window* type localized functions in several ways. Two simple window-type functions are constructed as the difference of two sigmoids, $\sigma(x) - \sigma(x-\theta)$ or the product of pairs of sigmoidal functions $\sigma(x)(1-\sigma(x))$ for each dimension. One may prove that after normalization these two forms are identical [5]. This type of transfer functions are very flexible, producing decision regions with convex shapes, suitable for classification. Product of $N$ pairs of sigmoids has the following general form:

$$Bi(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s}) = \prod_{i=1}^{N} \sigma(e^{s_i}\cdot(x_i - t_i + e^{b_i}))\ (1 - \sigma(e^{s_i}\cdot(x_i - t_i - e^{b_i}))) \tag{4}$$

Shape adaptation of the $Bi(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s})$ contours is possible by shifting centers $\mathbf{t}$, rescaling $\mathbf{b}$ and $\mathbf{s}$. Radial basis functions are defined relatively to only one center $||\mathbf{x}-\mathbf{t}||$. Here components of two centers are used, $t_i \pm e^{b_i}$, therefore these functions are called *bicentral*. Product form leads to well-localized convex contours of bicentral functions. Exponentials $e^{s_i}$ and $e^{b_i}$ are used instead of $s_i$ and $b_i$ parameters to prevent oscillations during the learning procedure. The number of adjustable parameters per processing unit is in this case $3N$. Localized bicentral functions may be extended to the semi-localized universal transfer functions by adding two parameters:

$$SBi(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s}) = \prod_{i=1}^{N} (\alpha + \sigma(e^{s_i}\cdot(x_i - t_i + e^{b_i})))\ (1 - \beta\sigma(e^{s_i}\cdot(x_i - t_i - e^{b_i}))) \tag{5}$$

This function does not vanish for large $|x|$, for $\alpha = 0$, $\beta = 1$ it is identical to the bicentral localized functions while for $\alpha = \beta = 0$ each component under the product turns into the usual sigmoidal function. For each unit semi-local functions *SBi* have $3N+2$ parameters or $5N$ parameters (if different $\alpha_i$ and $\beta_i$ are used in each dimension). Another possibility to control bicentral function's contours is to use independent slopes $s, s'$ in both factors. For small slopes $s_i$ and/or $s_i'$ the bicentral function may delocalize or stretch to *left* and/or *right* in any dimension. This allows creation of such contours as half-infinite channel, half-hyper ellipsoidal, soft triangular, etc. Although the costs of using this function is a bit higher than of the bicentral function (each function requires $4N$ parameters) more flexible decision borders are produced. Approximations to sigmoids may be used to avoid calculations of exponents.

Individual rotation of contours provided by each unit is possible in all dimensions with just $N$ additional parameters per neuron [5]. Bicentral functions with rotations (as well as multivariate Gaussian functions with rotation) have been implemented so far only in two neural network models, the Feature Space Mapping [14] and the IncNet [15]. Separability of the bicentral functions enables analysis of each dimension or a subspace of the input data independently. This is very important in classification when some of the features are missing. Bicentral transfer functions may also be used for *logical rule extraction* using the Feature Space Mapping network (FSM network, a model similar to RBF but based on separable functions) [16]. Sufficiently large values of the slopes are needed to change the bicentral functions into rectangular functions. Early results obtained with these functions are presented in [17].

## 3 Conclusions

Most neural networks are based on either sigmoidal or Gaussian transfer functions. In the *functional link* networks of Pao [18] a combination of various functions, such as polynomial, periodic, sigmoidal and Gaussian functions are used. We have presented a taxonomy of different transfer functions used in neural network models and proposed several new combinations of activation and output functions suitable for this purpose. Functions presented in this paper are more suitable as neural transfer functions – although we have not proved this most of these functions should lead to a

similar convergence behavior as given by the sigmoidal or Gaussian functions, i.e. in mutidimensional spaces much more attractive than given by polynomials.

According to the Statlog report [19] comparing many classification methods on 22 real-world datasets results of RBF and MLP are different. In some cases crossvalidation errors were twice as large using MLP than RBF while for other data the situation was reversed. Although one may always argue that better initialization, learning and architectures will reduce the difference, at least part of that difference may be attributed to different shapes of decision borders (hyperplane vs. ellipsoidal) provided by their transfer functions. Unfortunately little is known about advantages and disadvantages of different transfer functions presented in this paper. Many of them have never been used in neural networks so far (due to the lack of space only selected transfer functions were presented here) and almost none are available in public domain neural software. There is a tradeoff between flexibility of a single processing unit, increasing with the number of adjustable parameters, and the number of units needed to solve the problem, decreasing with flexibility of individual units. The complexity of the training process of the whole network should be minimized, but what is the optimal balance between the number and the complexity of units is not known. We believe that neural transfer functions deserve more attention.

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

## REFERENCES

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] N. Dyn, "Interpolation and approximation by radial and related functions", in C. K. Chiu et al, editors, *Approximation Theory VI*. Academic Press, San Diego, 1989.

[3] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators", *Neural Networks* 2: 359–366, 1989.

[4] T. Hastie, R. Tibshirani, "Discriminant adaptive nearest neighbor classification", *IEEE PAMI* 18: 607–616, 1996.

[5] W. Duch, N. Jankowski, "Survey of neural transfer functions", *Neural Computing Surveys* 2: 163–213, 1999.

[6] D. R. Wilson, T. R. Martinez, "Improved heterogeneous distance functions", *Journal of Artificial Intelligence Research* 6: 1–34, 1997.

[7] W. Duch, R. Adamczak, K. Grąbczewski, "Methodology of extraction, optimization and application of logical rules", in *Intelligent Information Systems VIII*, Ustroń, Poland, 1999, pp. 22–31.

[8] L.N. Cooper D.L. Reilly, C. Elbaum, "A neural model for category learning", *Biolog. Cybern.* 45: 35–41, 1982.

[9] T. Poggio, F. Girosi, "Network for approximation and learning", *Proceedings of the IEEE* 78: 1481–1497, 1990.

[10] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, 1972.

[11] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks", *Neural Computation* 3: 246–257, 1991.

[12] S. Ridella, S. Rovetta, and R. Zunino, "Circular backpropagation networks for classification", *IEEE Transaction on Neural Networks* 8: 84–97, 1997.

[13] G. Dorffner, "A unified framework for MLPs and RBFNs: Introducing conic section function networks", *Cybernetics and Systems* 25: 511–554, 1994.

[14] W. Duch and G. H. F. Diercksen, "Feature space mapping as a universal adaptive system", *Computer Physics Communications* 87: 341–371, 1995.

[15] N. Jankowski, V. Kadirkamanathan, "Statistical control of RBF-like networks for classification", in *7th Int. Conf. on Artificial Neural Networks*, Lausanne, Switzerland, Oct. 1997. Springer-Verlag, pp. 385–390.

[16] W. Duch, R. Adamczak, K. Grąbczewski, "Extraction of logical rules from backpropagation networks", *Neural Processing Letters* 7: 1–9, 1998.

[17] W. Duch, N. Jankowski, "New neural transfer functions", *J. of Applied Mathematics and Computer Science* 7: 639–658, 1997.

[18] Y-H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, MA, 1989.

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