Flexible Transfer Functions with Ontogenic Neural Networks

Norbert Jankowski[*]

Department of Computer Methods
Nicholas Copernicus University
ul. Grudziądzka 5
87-100 Toruń, Poland

Abstract

Transfer functions play a very important role in learning process of neural systems. This paper presents new functions which are more flexible than other functions commonly used in artificial neural networks. The latest improvement added is the ability to rotate the contours of constant values of transfer functions in multi-dimensional spaces with only $N-1$ adaptive parameters. Rotation using full covariance matrices requires N^2 parameters. These functions have biases and slopes separable in each dimension for each neuron, completely independent in multi-dimensional spaces. Therefore they are dimensionally separable — each dimension may be excluded independently at any time.

As the neural network model for testing the performance of these new transfer functions the Incremental Network (IncNet) was chosen. These networks are similar to radial basis function (RBF) networks and resource allocating networks. The architecture of IncNet is the same as the architecture of RBF networks, but the structure (the number of hidden nodes) changes during the learning process according to certain statistical criterion that controls growth and pruning of network connections.

Preliminary results show superior performance of the new transfer functions comparing with gaussian functions often used by RBF networks and other models.

1 Introduction

The research on the artificial neural networks (ANNs) can be divided to local and global optimization. Both have the same goal which is finding the unknown mapping between the input and output space for given data sets $\mathcal{S} = \{\langle \mathbf{x}_1, y_1\rangle, \ldots, \langle \mathbf{x}_n, y_n\rangle\}$, where $\langle \mathbf{x}_i, y_i\rangle$ is input – output pair ($\mathbf{x}_i \in \mathcal{R}^N, \quad y_i \in \mathcal{R}$). The underlying mapping $F(\cdot)$ can be written as

$$F(\mathbf{x}_i) = y_i + \eta, \qquad i = 1, \ldots, n \tag{1}$$

η is a zero mean white noise with variance σ^2_{ns}.

Artificial neural networks use many different architectures and many different transfer functions. The problems considered in this paper will deal rather with the local learning than global. Especially the local and semi-local transfer functions will be described. For extensive review of other transfer function see [Duch and Jankowski, 1999].

The best known local learning models are the radial basis function networks [Powell, 1987; Broomhead and Lowe, 1988; Dyn, 1989; Poggio and Girosi, 1990], adaptive kernel methods and local risk minimization [Bottou and Vapnik, 1992; Vapnik, 1995; Girosi, 1998].

The *Radial Basis Function* (RBF) networks [Powell, 1987; Poggio and Girosi, 1990; Lowe, 1989] were designed as a solution to an approximation problem in multi–dimensional spaces. The typical form of the RBF network can be written as

$$f(\mathbf{x}; \mathbf{w}, \mathbf{p}) = \sum_{i=1}^{M} w_i G_i(||\mathbf{x}||_i, \mathbf{p}_i) \tag{2}$$

where M is the number of neurons in hidden layer, $G_i(||\mathbf{x}||_i, \mathbf{p}_i)$ is the i-th Radial Basis Function, $\mathbf{p}_i$ are adjustable parameters such as centers, biases, etc., depending on $G_i(||\mathbf{x}||_i, \mathbf{p}_i)$ function. Commonly used radial basis functions are Gaussian functions, the nonlocal radial coordinates, general multiquadratics, and thin-plate spline functions defined as

$$\begin{aligned}
h_1(\mathbf{x}; \mathbf{t}, b) &= e^{-||\mathbf{x}-\mathbf{t}||^2/b^2} && (3)\\
h_2(\mathbf{x}; \mathbf{t}) &= ||\mathbf{x}-\mathbf{t}|| && (4)\\
h_3(\mathbf{x}; \mathbf{t}, b) &= (b^2 + ||\mathbf{x}-\mathbf{t}||^2)^{-\alpha}, \quad \alpha > 0 && (5)\\
h_4(\mathbf{x}; \mathbf{t}, b) &= (b^2 + ||\mathbf{x}-\mathbf{t}||^2)^{\beta}, \quad 0 < \beta < 1 && (6)\\
h_5(\mathbf{x}; \mathbf{t}, b) &= (b||\mathbf{x}-\mathbf{t}||)^2 \ln(b||\mathbf{x}-\mathbf{t}||) && (7)
\end{aligned}$$

In contrast to many *artificial neural networks* (ANNs) including well known *multi-layered perceptron* (MLP)

[0]In *Technical report, Computational Intelligence Lab*, DCM NCU, Toru, Poland, 1999
e-mail:Norbert.Jankowski@phys.uni.torun.pl,
http://www.phys.uni.torun.pl/˜norbert

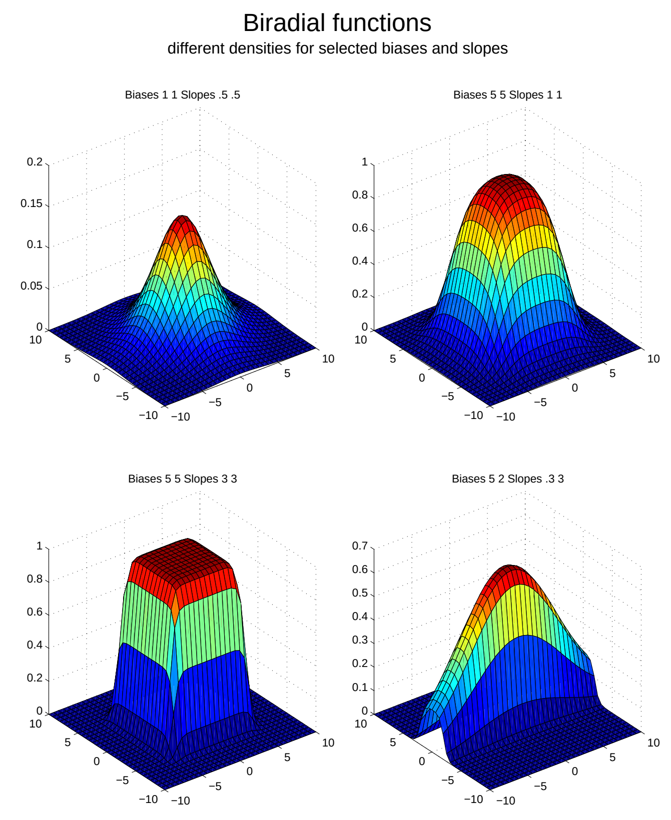

Figure 1: A few shapes of the biradial functions in 2D (Eq. 8).

networks the RBF networks and local risk minimization methods have well known mathematical properties. RBF networks are universal aproximator [Hartman *et al.*, 1990; Park and Sandberg, 1991]. Girosi and Poggio [Poggio and Girosi, 1990] proved the existence and uniqueness of best approximation for regularization and RBF networks.

2 Transfer Functions

Sigmoidal functions may be combined into a *window* type localized functions in several ways. Perhaps the simplest is to take the difference of two sigmoids, $\sigma(x) - \sigma(x-\theta)$. One may also use products of pairs of sigmoidal functions $\sigma(x)(1-\sigma(x))$ for each dimension. 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 (see Fig. 2):

$$Bi(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s}) = \prod_{i=1}^{N} \sigma(e^{s_i}\cdot(x_i - t_i + e^{b_i})) \quad (1-\sigma(e^{s_i}\cdot(x_i - t_i - e^{b_i}))) \tag{8}$$

where $\sigma(x) = 1/(1+e^{-x})$. The first sigmoidal factor in the product is growing for increasing input x_i while the second is decreasing, localizing the function around t_i. Shape adaptation of the density $Bi(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s})$ is possible by shifting centers $\mathbf{t}$, rescaling $\mathbf{b}$ and $\mathbf{s}$. Radial basis functions are defined relatively to only one center $||x-t||$. Here two centers are used, $t_i + e^{b_i}$ and $t_i - e^{b_i}$, therefore these functions are called biradial. Product form leads to well-localized convex densities of biradial 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 (learning becomes more stable).

The number of adjustable parameters per processing unit is in this case $3N$. Dimensionality reduction is possible as in the *Gaussian bar case* [Hartman *et al.*, 1990; Park and Sandberg, 1991], but more flexible density shapes are obtained, thus allowing to reduce the number of adaptive units in the network.

Biradial functions with independent slopes. Localized biradial functions may be extended to the semi-localized universal transfer functions using independent slopes in the pair of sigmoids:

$$Bi2s(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s}) = \prod_{i=1}^{N} \sigma(e^{s_i}\cdot(x_i - t_i + e^{b_i})) \; (1- \sigma(e^{s'_i}\cdot(x_i - t_i - e^{b_i}))) \tag{9}$$

Using small slope s_i and/or s'_i the biradial function may delocalize or stretch to *left* and/or *right* in any dimension. This allows to get such contours of transfer functions as half-infinite channel, half-hyper ellipsoidal, soft triangular, etc. Although the cost of using this function is a bit higher than of the biradial function (it uses $4N$ parameters for each neuron), more flexible density contours are produced.

Biradial functions with rotation. The biradial functions proposed above contain $3N$ parameters per

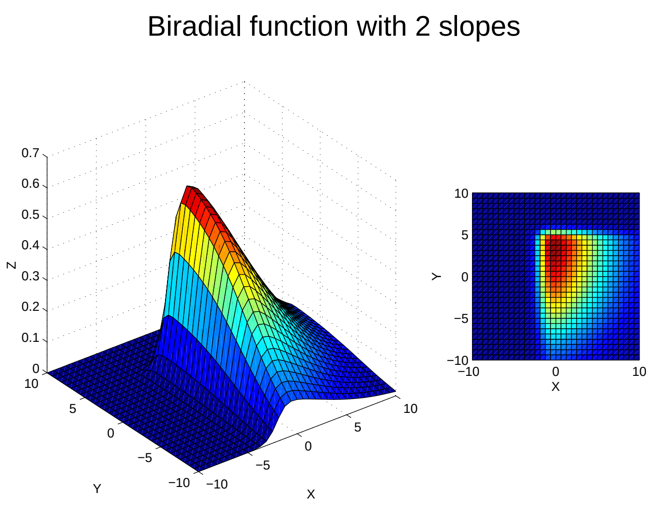

Figure 2: Biradial functions with two slopes (Eq. 9).

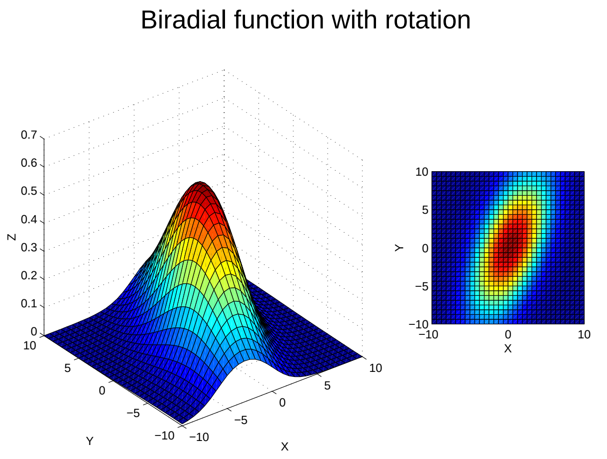

Figure 3: Biradial functions with rotation (Eq. 10).

one unit and are quite flexible in representing various probability densities. Semi-biradial functions and biradial functions with independent slopes provide local and non-local units in one network. Next step towards even greater flexibility requires individual rotation of densities provided by each unit. Of course one can introduce a rotation matrix operating on the inputs $\mathbf{Rx}$, but in practice it is very hard to parameterize this $N \times N$ matrix with $N-1$ independent angles (for example, Euler's angles) and to calculate the derivatives necessary for backpropagation training procedure. Rotated densities in all dimensions may be obtained in two ways using transfer functions with just $N-1$ additional parameters per neuron. In the first approach (for the second see [Duch and Jankowski, 1999; 1997]) product form of the combination of sigmoids is used (see Fig. 3)

$$C_P(\mathbf{x};\mathbf{t},\mathbf{t}',\mathbf{R}) = \prod_i^N \Big(\sigma(\mathbf{R}_i\mathbf{x}+t_i) - \sigma(\mathbf{R}_i\mathbf{x}+t'_i)\Big) \quad (10)$$

$$SC_P(\mathbf{x};\mathbf{t},\mathbf{t}',\mathbf{p},\mathbf{r},\mathbf{R}) = \prod_i^N \Big(p_i \cdot \sigma(\mathbf{R}_i\mathbf{x}+t_i) + r_i \cdot \sigma(\mathbf{R}_i\mathbf{x}+t'_i)\Big) \quad (11)$$

where $\mathbf{R}_i$ is the i-th row of the rotation matrix $\mathbf{R}$ with the following structure:

$$\mathbf{R} = \begin{bmatrix} s_1 & \alpha_1 & 0 & & \cdots & 0 \\ 0 & s_2 & \alpha_2 & 0 & & \\ & & & & & \\ \vdots & & & \ddots & & \vdots \\ & & & & s_{N-1} & \alpha_{N-1} \\ 0 & \cdots & & & 0 & s_N \end{bmatrix} \quad (12)$$

If $p_i = 1$ and $r_i = -1$ then SC_P function is localized and gives similar densities as the biradial functions (except for rotation). Choosing other values for the p_i and r_i parameters non-local transfer functions are created.

3 Incremental Network

In the 1991 Platt published the article on the *Resource–Allocating Network* [Platt, 1991]. The RAN network is an RBF-like network that starts from empty hidden layer and grows when the following criteria are satisfied:

$$\mathbf{y}_n - f(\mathbf{x}_n) = e_n > e_{min}; \qquad ||\mathbf{x}_n - \mathbf{t}_c|| > \epsilon_{min} \quad (13)$$

e_n is equal the current error, $\mathbf{t}_c$ is the nearest center of a basis function to the vector $\mathbf{x}_n$ and e_{min}, ϵ_{min} are some experimentally chosen constants. The growing network can be described by

$$\begin{aligned} f^{(n)}(\mathbf{x},\mathbf{p}) &= \sum_{i=1}^{k-1} w_i G_i(\mathbf{x},\mathbf{p}_i) + e_n G_k(\mathbf{x},\mathbf{p}_k) \\ &= \sum_{i=1}^{k} w_i G_i(\mathbf{x},\mathbf{p}_i) \end{aligned} \quad (14)$$

where $\mathbf{p}_k$ includes centers $\mathbf{x}_n$ and other adaptive parameters which are set up with some initial values. If the growth criteria are not satisfied the RAN network uses the LMS algorithm to estimate adaptive parameters.

Although LMS algorithm is faster than *Extended Kalman Filter* (EKF) algorithm [Candy, 1986], the EKF algorithm was chosen because it exhibits fast convergence, uses lower number of neurons in hidden layer [Kadirkamanathan and Niranjan, 1992; 1993] and gives some *tools* which would be useful in control of the growth and pruning process.

RAN network using EKF learning algorithm (RAN-EKF) was proposed by [Kadirkamanathan and Niranjan, 1993]. The previous version of the IncNet [Kadirkamanathan, 1994] is a RAN-EKF network with statistically controlled growth criterion. For more exhaustive description of ontogenic neural networks see [Fiesler, 1994].

The EKF equations can be written as follows:

$$\begin{aligned} e_n &= y_n - f(\mathbf{x}_n;\mathbf{p}_{n-1}) \\ \mathbf{d}_n &= \frac{\partial f(\mathbf{x}_n;\mathbf{p}_{n-1})}{\partial \mathbf{p}_{n-1}} \\ R_y &= R_n + \mathbf{d}_n^T \mathbf{P}_{n-1}\mathbf{d}_n \\ \mathbf{k}_n &= \mathbf{P}_{n-1}\mathbf{d}_n / R_y \\ \mathbf{p}_n &= \mathbf{p}_{n-1} + e_n \mathbf{k}_n \\ \mathbf{P}_n &= [\mathbf{I} - \mathbf{k}_n \mathbf{d}_n^T]\mathbf{P}_{n-1} + Q_0(n)\mathbf{I} \end{aligned} \quad (15)$$

The suffixes $n-1$ and n denotes the priors and posteriors. $\mathbf{p}_n$ consists of all adaptive parameters: weights, centers, biases, etc. To prevent too quick convergence of the EKF, which leads to data overfitting, the $Q_0\mathbf{I}$ adds a *random* change, where Q_0 is scalar and $\mathbf{I}$ is the identity matrix.

Novelty Criterion: Using methods which estimate during learning covariance of uncertainty of each parameter, the network output uncertainty can be determined and the same criterion as in previous version of IncNet [Jankowski and Kadirkamanathan, 1997a; 1997b;

Kadirkamanathan, 1994] may be used. Then the hypothesis for the statistical inference of model sufficiency is stated as follows:

$$\mathcal{H}_0: \quad \frac{e^2}{\mathrm{Var}[f(\mathbf{x};\mathbf{p})+\eta]} = \frac{e^2}{\sigma_y^2(\mathbf{x})+\sigma_{ns}^2} < \chi^2_{n,\theta} \qquad (16)$$

where $\chi^2_{n,\theta}$ is θ% confidence on χ^2 distribution for n degree of freedom. $e = y - f(\mathbf{x};\mathbf{p})$ is the error (see Eq. 15). If this hypothesis is not satisfied the current model is not sufficient and should change. $R_y = \mathrm{Var}[f(\mathbf{x};\mathbf{p})+\eta]$ (part of EKF) estimates the total uncertainty in the expected output and the null hypothesis can be written as:

$$\mathcal{H}_0: \quad e_n^2/R_y < \chi^2_{n,\theta} \qquad (17)$$

If hypothesis $\mathcal{H}_0$ is satisfied then IncNet continues learning using the EKF algorithm. Otherwise, a new neuron $(M+1)$-th should be added with some initial parameters. For Gaussian functions $G_{M+1}(\cdot)$ these parameters are: $w_{M+1} := e_n$, $\mathbf{t}_{M+1} := \mathbf{x}_n$, $b_{M+1} := b_0$, $\mathbf{P}_n := \left[\begin{array}{cc} \mathbf{P}_n & 0 \\ 0 & P_0\mathbf{I} \end{array}\right]$, where e_n is the error for given input vector $\mathbf{x}_n$, b_0 and P_0 are some initial values for bias (depending on a given problem) and covariance matrix elements (usually 1).

Pruning: As a result of the learning process a neuron can become completely useless and should be pruned. Assume the structure of vector $\mathbf{p}_n$ and the covariance matrix as:

$$\mathbf{p}_n = [w_1, \ldots, w_M, \ldots]^T \qquad \mathbf{P} = \left[\begin{array}{cc} \mathbf{P}_w & \mathbf{P}_{wv} \\ \mathbf{P}_{wv}^T & \mathbf{P}_v \end{array}\right] \qquad (18)$$

where $\mathbf{P}_w$ is a matrix of correlations between weights, $\mathbf{P}_{wv}$ between weights and other parameters, $\mathbf{P}_v$ **only** between others parameters (excluding all weights).

Then by checking the inequality $\mathcal{P}$ (Eq.19) we can decide whether to prune or not and also we know that the neuron for which value L was obtained has smallest saliency and should be pruned.

$$\mathcal{P}: \ L/R_y < \chi^2_{1,\vartheta} \qquad L = \min_i w_i^2/[\mathbf{P}_w]_{ii} \qquad (19)$$

where $\chi^2_{n,\vartheta}$ is ϑ% confidence on χ^2 distribution for one degree of freedom.

Neurons are pruned if the saliency L is too small and/or the uncertainty of the network output R_y is too big.

Many pruning methods were described in the last decade. Pruning leads to the removal of the network connections and unnecessary neurons, but frequently many neurons contribute to decision borders that could be represented by smaller network without decreasing of accuracy. Therefore one should merge two [Jankowski, 1998] neurons(or even more – it may be more complicated computationally) keeping the current shape of the decision surface unchanged as much as possible. Two neurons can be replaced by a sigle one if the ratio:

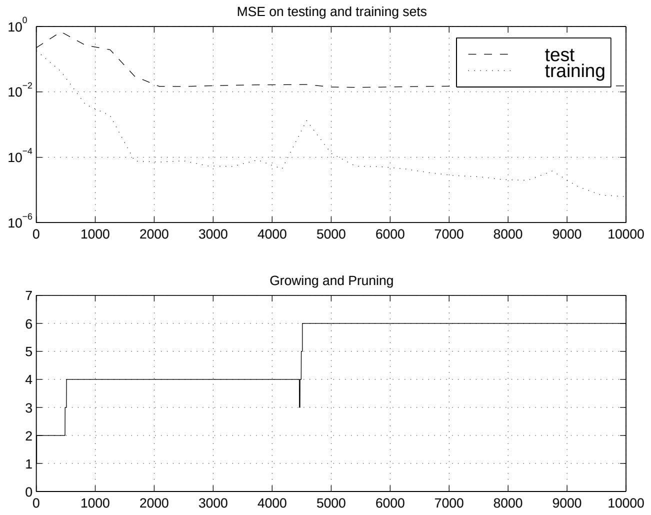

Figure 4: The MSE error on training and testing sets, and below number of neurons in hidden layer was shown.

$$\frac{\int_{\mathbf{d}\subseteq\mathcal{D}^n} |\phi_i(\mathbf{x}) + \phi_j(\mathbf{x}) - \phi_{new}(\mathbf{x})|\, d\mathbf{x}}{\int_{\mathbf{d}\subseteq\mathcal{D}^n} |\phi_i(\mathbf{x}) + \phi_j(\mathbf{x})|\, d\mathbf{x}} < \alpha \qquad (20)$$

is smaller than some confidence parameter α.

Unfortunately merging of neurons has not been used yet in the simulations described below.

4 Results

Gabor and Girosi functions are approximation benchmark. These functions were used previously by Girosi *et al.* [Girosi *et al.*, 1993]:

$$f_{gab}(x,y) = e^{-||\mathbf{x}||^2}\cos(.75\pi(x+y)) \qquad (21)$$

$$f_{gir}(x,y) = sin(2\pi x) + 4(y-0.5)^2 \qquad (22)$$

The learning is difficult because only 20 points were provided for learning from uniformly distributed interval $[-1,1]\times[-1,1]$ for Gabor function (Eq. 21) and from $[0,1]\times[0,1]$ interval for additive function (Eq. 22). 10,000 points were used in testing phase from the same intervals[1]. The table 1 describes the models[2] and the MSE errors of those models.

Although the IncNet model is not always the best one, it is the best on average, adapting more flexibly (see table 1). Remember that building the network one should choose the transfer function using only the training error.

[1]Methods of preparation of training and testing data and methods of results comparing are the same as used by previous authors.

[2]Models 1 to 8 originally published by Girosi, Jones and Poggio in [Girosi *et al.*, 1993].

Model1	$\sum_{i=1}^{20} c_i[e^{-\left(\frac{(x-x_i)^2}{\sigma_1}+\frac{(y-y_i)^2}{\sigma_2}\right)} + e^{-\left(\frac{(x-x_i)^2}{\sigma_2}+\frac{(y-y_i)^2}{\sigma_1}\right)}]$	$\sigma_1 = \sigma_2 = 0.5$
Model2	$\sum_{i=1}^{20} c_i[e^{-\left(\frac{(x-x_i)^2}{\sigma_1}+\frac{(y-y_i)^2}{\sigma_2}\right)} + e^{-\left(\frac{(x-x_i)^2}{\sigma_2}+\frac{(y-y_i)^2}{\sigma_1}\right)}]$	$\sigma_1 = 10, \sigma_2 = 0.5$
Model3	$\sum_{i=1}^{20} c_i[e^{\frac{(x-x_i)^2}{\sigma}} + e^{-\frac{(y-y_i)^2}{\sigma}}]$	$\sigma = 0.5$
Model4	$\sum_{\alpha=1}^{7} b_\alpha e^{-\frac{(x-t_x^\alpha)^2}{\sigma}} + \sum_{\beta=1}^{7} c_\beta e^{-\frac{(y-t_y^\beta)^2}{\sigma}}$	$\sigma = 0.5$
Model5	$\sum_{\alpha=1}^{n} c_\alpha e^{-(\mathbf{W}_\alpha \cdot \mathbf{X} - t_\alpha)^2}$	
Model6	$\sum_{i=1}^{20} c_i[\sigma(x-x_i) + \sigma(y-y_i)]$	
Model7	$\sum_{\alpha=1}^{7} b_\alpha \sigma(x - t_x^\alpha) + \sum_{\beta=1}^{7} c_\beta \sigma(y - t_y^\beta)$	
Model8	$\sum_{\alpha=1}^{n} c_\alpha \sigma(\mathbf{W}_\alpha \cdot \mathbf{X} - t_\alpha)$	

Additive function — MSE train/test									
IncNet Rot	**IncNet**	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8
.00000133	.00000232	.000036	.000067	.000001	.000001	.000170	.000001	.000003	.000743
0.000859	.000082	.011717	.001598	.000007	.000009	.001422	.000015	.000020	.026699
Gabor function — MSE train/test									
.000006	.000025	.000000	.000000	.000000	.345423	.000001	.000000	.456822	.000044
0.015316	0.025113	.003818	.344881	67.9523	1.22211	.033964	98.4198	1.39739	.191055

Table 1: Comparison of different models, based on different transfer functions with IncNet network using biradial and biradial with rotation functions for approximation of two functions Eq. (21) and Eq. (22)

The table shows that rotation of biradial functions works well and can help in learning. Biradial functions with rotation can estimate more complex unknown mapping and the costs of $N-1$ additional adaptive parameters is not too big. For Gabor function Eq. (21) IncNet used 4 neurons using biradial functions Eq. (8) and 6 neurons using biradial function with rotation Eq. (10). Note that IncNet with biradial neurons which use more than 4 neurons loss the generalization on testing set. For additive Girosi function Eq. (22) 8 biradial neurons Eq. (8) and the same number of functions of biradial with rotation Eq. (10) were used. Note that the IncNet controls the growth and pruning during the learning process, see Fig. 4. In the first phase of learning IncNet adds and removes neurons to find the final structure — the growth and pruning is looking for optimal structure continuously during the learning.

Sugeno function. The second benchmark problem concerns an approximation of Sugeno function [Sugeno and Kang, 1988] defined as

$$f(x, y, z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \tag{23}$$

Results using IncNet with biradial and biradial with rotation transfer functions were compared to other results presented by Sugeno [Sugeno and Kang, 1988], Kosiński [Kosiński and Weigl, 1995], and Horikawa *et al.* [Horikawa *et al.*, 1992] (see table 2). Although this function is frequently used for testing the approximation capabilities of adaptive systems, there is no standard procedure to select the training points and thus the results are rather hard to compare. For training 216 points from $[1, 6]$ interval and 125 points for testing from $[1.5, 5.5]$ interval were randomly chosen. All tests were performed using the same (if possible) or similar initial

Model	APE_{TRS}	APE_{TES}
GMDS model Kongo	4.7	5.7
Fuzzy model 1 Sugeno	1.5	2.1
Fuzzy model 2 Sugeno	0.59	3.4
FNN Type 1 Horikawa	0.84	1.22
FNN Type 2 Horikawa	0.73	1.28
FNN Type 3 Horikawa	0.63	1.25
M - Delta model	0.72	0.74
Fuzzy INET	0.18	0.24
Fuzzy VINET	0.076	0.18
IncNet	**0.119**	**0.122**
IncNet Rot	**0.053**	**0.061**

Table 2: Approximation of Sugeno function.

parameters. The *Average Percentage Error* (APE) was used as a measure of approximation error:

$$APE = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{f(\mathbf{x}_i) - y_i}{y_i} \right| * 100\% \tag{24}$$

Final networks had maximally 11 neurons in the hidden layer.

5 Conclusions

Results presented above show that biradial with rotation and biradial functions used with incremental network work very efficiently. The final networks have quite high generalization, and the structure of the networks are controlled online by statistical criteria. Biradial transfer functions may estimate many different probability densities with good generalization in efficient framework. Biradial functions with rotation definitely improve estimation of complex densities using just $4N-1$ parameters per neuron (where N is dimension of input space).

References

[Bottou and Vapnik, 1992] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[Broomhead and Lowe, 1988] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[Candy, 1986] J. V. Candy. *Signal processing: The model based approach.* McGraw-Hill, New York, 1986.

[Duch and Jankowski, 1997] W. Duch and N. Jankowski. New neural transfer functions. *Journal of Applied Mathematics and Computer Science*, 7(3):639–658, 1997.

[Duch and Jankowski, 1999] W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, (2):163–212, 1999.

[Dyn, 1989] N. Dyn. Interpolation and approximation by radial and related functions. In C. K. Chiu, L. L. Schumaker, and J. D. Watts, editors, *Approximation Theory VI.* Academic Press, San Diego, 1989.

[Fiesler, 1994] E. Fiesler. Comparative bibliography of ontogenic neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*, 1994.

[Girosi *et al.*, 1993] F. Girosi, M. Jones, and T. Poggio. Priors stabilizers and basis functions: From regularization to radial, tensor and additive splines. Technical report, MIT, Cambridge, Massachusetts, 1993.

[Girosi, 1998] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, August 1998.

[Hartman *et al.*, 1990] E. J. Hartman, J. D. Keeler, and J. M. Kowalski. Layered neural networks with gaussian hidden units as universal approximations. *Neural Computation*, 2(2):210–215, 1990.

[Horikawa *et al.*, 1992] S. Horikawa, Takeshi Furuhashi, and Yoshiki Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Transactions on Neural Networks*, 3(5):801–806, September 1992.

[Jankowski and Kadirkamanathan, 1997a] N. Jankowski and V. Kadirkamanathan. Statistical control of RBF-like networks for classification. In *7th International Conference on Artificial Neural Networks*, pages 385–390, Lausanne, Switzerland, October 1997. Springer-Verlag.

[Jankowski and Kadirkamanathan, 1997b] N. Jankowski and V. Kadirkamanathan. Statistical control of growing and pruning in RBF-like neural networks. In *Third Conference on Neural Networks and Their Applications*, pages 663–670, Kule, Poland, October 1997.

[Jankowski, 1998] N. Jankowski. Controlling the structure of neural networks that grow and shrink. In *Second International Conference on Cognitive and Neural Systems*, Boston, USA, May 1998.

[Kadirkamanathan and Niranjan, 1992] V. Kadirkamanathan and M. Niranjan. Application of an architecturally dynamic network for speech pattern classification. *Proceedings of the Institute of Acoustics*, 14:343–350, 1992.

[Kadirkamanathan and Niranjan, 1993] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975, 1993.

[Kadirkamanathan, 1994] V. Kadirkamanathan. A statistical inference based growth criterion for the RBF networks. In *Proceedings of the IEEE. Workshop on Neural Networks for Signal Processing*, 1994.

[Kosiński and Weigl, 1995] W. Kosiński and M. Weigl. Mapping neural networks and fuzzy inference systems for approximation of multivariate function. In E. Kącki, editor, *System Modeling Control, Artificial Neural Networks and Their Applications*, volume 3, pages 60–65, Łódź, Poland, May 1995.

[Lowe, 1989] D. Lowe. Adaptive radial basis function nonlinearities, and the problem of generalization. In *1st IEE International Conference on Artificial Neural Networks*, pages 171–175, London, UK, 1989.

[Park and Sandberg, 1991] J. Park and I. W. Sandberg. Universal approximation using radial basis function networks. *Neural Computation*, 3(2):246–257, 1991.

[Platt, 1991] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225, 1991.

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

[Powell, 1987] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation of Functions and Data*, pages 143–167, Oxford, 1987. Oxford University Press.

[Sugeno and Kang, 1988] M. Sugeno and G. T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28, 1988.

[Vapnik, 1995] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.