Approximation and classification with RBF-type Neural Networks using flexible local and semi-local transfer functions[1]

Norbert Jankowski[2]
Department of Computer Methods
Nicholas Copernicus University

Abstract:
Structure of incremental neural network (IncNet) is controlled by growing and pruning to match the complexity of training data. Extended Kalman Filter algorithm and its fast version is used as learning algorithm. Bi-radial transfer functions, more flexible than other functions commonly used in artificial neural networks, are used. The latest improvement added is the ability to rotate the contours of constant values of transfer functions in multidimensional spaces with only $N-1$ adaptive parameters. Results on approximation benchmarks and on the real world psychometric classification problem clearly shows superior generalization performance of presented network comparing with other classification models.

1 INTRODUCTION

Artificial Neural Networks (ANN) are used to many different kinds of problems such as classification, approximation, pattern recognition, signal processing, prediction, feature extraction, etc. Most of them are solved with ANN by learning of the 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

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

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

Building a network which preserves information with complexity matched to training data, using an architecture which is able to grow, shrink, and using flexible transfer functions to estimate complex probability density distributions, is the goal of this paper.

The best known local learning models are the radial basis function networks (RBF) [12, 11, 2], adaptive kernel methods and local risk minimization [5]. The RBF networks were designed as a solution to an approximation problem in multi–dimensional spaces. The typical form of the RBF network can be written as

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

[1]In 4th Conference on Neural Networks and Their Applications, pages 77-82, Zakopane, Poland, May 1999.
[2]E-mail: Norbert.Jankowski@phys.uni.torun.pl, WWW pages http://www.phys.uni.torun.pl/~norbert

where M is the number of neurons in the 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 which is usually a Gaussian ($e^{-||x-\mathbf{t}||^2/b^2}$), multi-quadratics or thin-plate spline function.

The RAN network [10] is an RBF-like network that grows when the following criteria are satisfied:

$$y_n - f(\mathbf{x}_n) = e_n > e_{min} \qquad ||\mathbf{x}_n - \mathbf{t}_c|| > \epsilon_{min} \tag{3}$$

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.

2 LEARNING ALGORITHM

Extended Kalman Filter (EKF) was used as learning algorithm [3] because it exhibits fast convergence, uses lower number of neurons in the hidden layer [8] and gives some *tools* which are useful for control of the growth and pruning of the network. The algorithm computes the following quantities:

$$\begin{array}{rclrcl} e_n &=& y_n - f(\mathbf{x}_n; \mathbf{p}_{n-1}) & \mathbf{d}_n &=& \frac{\partial f(\mathbf{x}_n;\mathbf{p}_{n-1})}{\partial \mathbf{p}_{n-1}} \\ R_y &=& R_n + \mathbf{d}_n^T \mathbf{P}_{n-1} \mathbf{d}_n & \mathbf{k}_n &=& \mathbf{P}_{n-1}\mathbf{d}_n / R_y \\ \mathbf{p}_n &=& \mathbf{p}_{n-1} + e_n \mathbf{k}_n & \mathbf{P}_n &=& [\mathbf{I} - \mathbf{k}_n \mathbf{d}_n^T]\mathbf{P}_{n-1} + Q_0(n)\mathbf{I} \end{array} \tag{4}$$

Tthe suffixes $n-1$ and n denote the priors and posteriors. $\mathbf{p}_n$ consists of all adaptive parameters: weights, centers, biases, etc.

Fast EKF: The fast version of the EKF learning algorithm was introduced in [7]. Because the covariance matrix $\mathbf{P}_n$ can be computationally expensive some simplifications are applied. Assuming that correlations between parameters of different neurons are not very important we can simplify the matrix $\mathbf{P}_n$ to block-diagonal structure $\widetilde{\mathbf{P}}_n$ which consists of matrix $\widetilde{\mathbf{P}}_n^i$, $i = 1 \dots M$. Those diagonal elements represent correlations of adaptive parameters of the i-th neuron. For a given problem $\mathcal{P}$ the complexity of matrix $\mathbf{P}_n$ is $O(M^2)$, and matrix $\widetilde{\mathbf{P}}_n$ just $O(M)$ (M is the number of neurons). Using this approximation the fast EKF is defined by:

$$\begin{array}{rclrcl} e_n &=& y_n - f(\mathbf{x}_n; \mathbf{p}_{n-1}) & \mathbf{d}_n^i &=& \frac{\partial f(\mathbf{x}_n;\mathbf{p}_{n-1})}{\partial \mathbf{p}_{n-1}^i} \\ R_y &=& \multicolumn{4}{l}{R_n + \mathbf{d}_n^{1\,T} \widetilde{\mathbf{P}}_{n-1}^1 \mathbf{d}_n^1 + \cdots + \mathbf{d}_n^{M\,T} \widetilde{\mathbf{P}}_{n-1}^M \mathbf{d}_n^M} \\ \mathbf{k}_n^i &=& \widetilde{\mathbf{P}}_{n-1}^i \mathbf{d}_n^i / R_y & \mathbf{p}_n^i &=& \mathbf{p}_{n-1}^i + e_n \mathbf{k}_n^i \\ \widetilde{\mathbf{P}}_n^i &=& [\mathbf{I} - \mathbf{k}_n^i \mathbf{d}_n^{i\,T}]\widetilde{\mathbf{P}}_{n-1}^i + Q_0(n)\mathbf{I} & & & i = 1, \dots, M \end{array} \tag{5}$$

Novelty Criterion: Using methods which estimate covariance of uncertainty of each parameter during learning, the uncertainty of network output can be determined. The following novelty criterion is used:

$$\mathcal{H}_0 : \quad \frac{e_n^2}{R_y} = \frac{e^2}{\mathrm{Var}[f(\mathbf{x};\mathbf{p}) + \eta]} < \chi^2_{n,\theta} \tag{6}$$

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. If this hypothesis is not satisfied the current model is not sufficient and the network should grow.

Pruning Criterion: Checking the inequality $\mathcal{P}$ given below it is possible to decide whether to prune the network or not. It allows also to select the neuron for which L value has smallest saliency and the neuron should be pruned.

$$\mathcal{P}: \; L/R_y < \chi^2_{1,\vartheta} \qquad\qquad L = \min_i w_i^2/[\mathbf{P}_w]_{ii} \tag{7}$$

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.

Bi-radial Transfer Functions: Sigmoidal functions may be combined into a *window* type localized functions in several ways, for example by taking the difference of two sigmoids, $\sigma(x) - \sigma(x - \theta)$ or product of pairs of sigmoidal functions $\sigma(x)(1 - \sigma(x))$ for each dimension. These transfer functions are very flexible, producing decision regions with convex shapes, suitable for classification. Product of N pairs of sigmoids $\sigma(x) = 1/(1 + e^{-x})$ has the following general form:

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

Biradial functions with rotation: The biradial functions proposed above contain 3N parameters per one node and are quite flexible in representing various probability densities. 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 back-propagation training procedure (see Fig. 1).

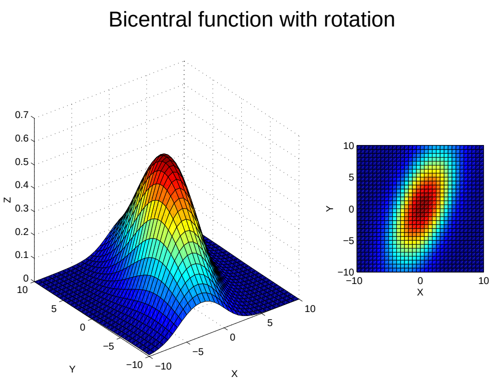

Figure 1: Biradial functions with rotation (Eq. 9).

$$C_P(\mathbf{x}; \mathbf{t}, \mathbf{t}', \mathbf{R}) = \prod_{i}^{N} \left(\sigma(\mathbf{R}_i \mathbf{x} + t_i) - \sigma(\mathbf{R}_i \mathbf{x} + t_i') \right) \tag{9}$$

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

$$\mathbf{R} = \begin{bmatrix} s_1 & \alpha_1 & & 0 \\ & \ddots & \ddots & \\ & & s_{N-1} & \alpha_{N-1} \\ 0 & & & s_N \end{bmatrix} \tag{10}$$

For other biradial transfer function extensions see [6, 4].

Classification using IncNet. Independed IncNet networks are constructed for each class for a given problem. Each of them receives input vector x and 1 if index of i-th sub-network is equal to desired class number, otherwise 0. The output of i-th network defines how much a given case belongs to i-th class. *Winner takes all* strategy is used to decide the final class for a case. Figure on the right presents the structure of IncNet network for classification. Note that each of the sub-networks learns separately (which helps in parallelisation of the algorithm) and its final structure tries to match the complexity for i-th class, not for all classes (structure of each sub-network is usually different).

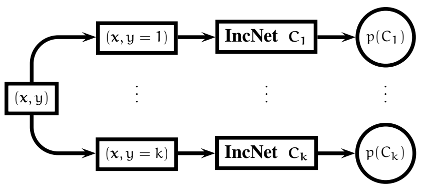

Figure 2: IncNet network for classification.

3 ILLUSTRATIVE RESULTS

Sugeno function. The first benchmark problem concerns an approximation of Sugeno function defined as $f(x,y,z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2$

Results using the IncNet model with biradial, and biradial with rotation, transfer functions were compared to other results presented by Sugeno, Kosiński, and Horikawa [9](Table 1). 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 parameters. The *Average Percentage Error* (APE) was used as a measure of approximation error $APE = 1/N \sum_{i=1}^{N} |(f(x_i) - y_i)/y_i| * 100\%$. Final networks had at most 11 neurons in the hidden layer.

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 1: Approximation of Sugeno function.

Psychometric data. In the *real world* problem psychometric data problem each case (person) is assigned a personality type using the data from Minnesota Multiphasic Personality Inventory (MMPI) test. The MMPI test is one of the most popular psychometric tests designed to help in the psychological diagnoses. MMPI test consists of over 550 questions. Using the answers from each MMPI test 14 numerical factors are computed (by some arithmetic operations) forming the intermediate basis (**not** the final hypothesis) for the diagnosis.

Is it possible to build a model, which will perform automatic assignment of a given person to one of personality type basing on a set of well diagnosed examples? To solve this question several data sets were collected and classified by psychologists. In this article two of those sets have been considered, the first with 27 classes and the second with 28 classes. Each case has 14 features determined from over 550 questions of MMPI test. Some classes concern men, and other women only. Each case can be classified as normal or belong to a disease such as neurosis, psychopathy, schizophrenia, delusions, psychosis, etc. Data sets consists of 1027 and 1167 examples respectively for 27 and 28 classes sets. Figure 3 shows the learning of one single class, displaying the changes of accuracy and the number of neurons.

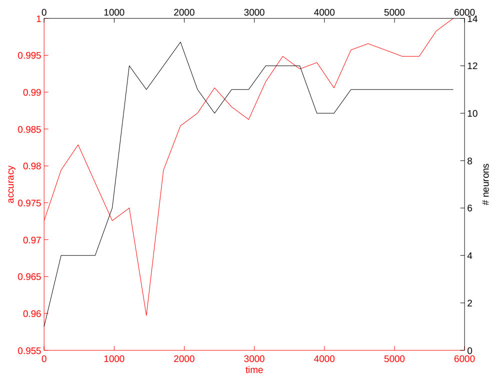

Figure 3: Curves shows the accuracy and the number of neurons through the learning process of a single class. [1 unit of time is a single learning pair presentation.]

In Tables 2 and 3 comparison of generalization for IncNet, FSM [1] and C4.5 is shown. In Table 2 the overall performance is presented and in Table 3 the generalization after dividing the whole set into training and testing sets for 10% + 90% and 5% + 95% learning. Figure 4 shows the confusion matrix (on the left). It clearly shows that there are just a few errors after the classification. On the right side of the same figure the analysis of errors from the same learning process are presented. Edges of each line shows the target (left) and desired output values for given case (person). Note that most slopes of the error-lines are small, meaning that a given case is not clear and can **not** be assigned to a single class. More over, most of these errors are not really errors because they may indeed correspond to two classes.

Model	**Overall test for**	
	27 classes	**28 classes**
C 4.5	93.67	93.06
FSM Rule Opt.	97.57	96.91
IncNet	**99.22**	**99.23**

Table 2: Accuracy of different classification models in an overall test.

Model	**27 classes set**				**28 classes set**			
	10% test		**5% test**		**10% test**		**5% test**	
	TRS	TES	TRS	TES	TRS	TES	TRS	TES
FSM		91.59						
IncNet	99.03	93.14	98.77	96.08	98.95	93.10	98.29	94.83

Table 3: Accuracy of different classification models. 10% (or 5%) test means that 10% (or 5%) of examples are used as testing set and 90% (or 95%) as training set.

4 CONCLUSIONS

Results presented above show that biradial transfer functions used with the incremental network work very efficiently. The final network show high generalization, and the structure

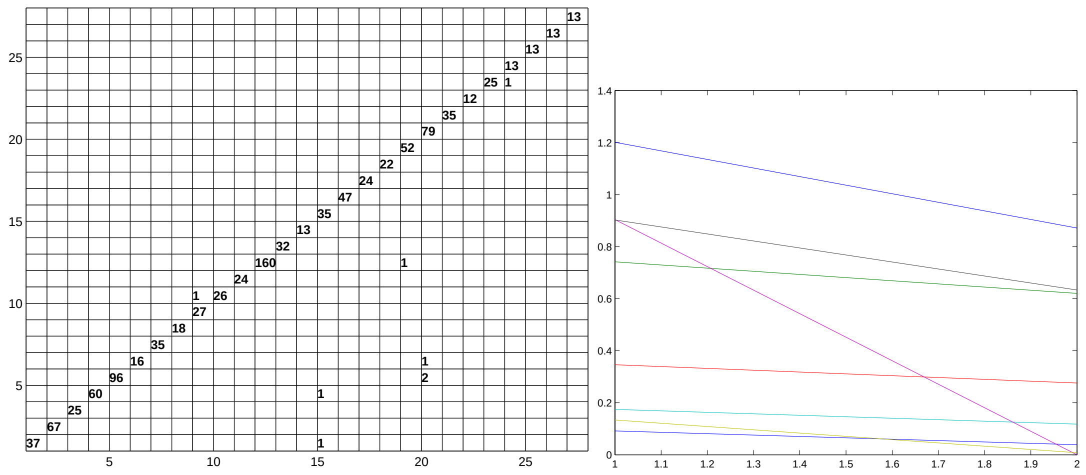

Figure 4: The confusion matrix for the 27 classes set (left). Comparison of network output and desired output values.

of the networks is 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). Such networks may be used successfully for real world problems.

REFERENCES

[1] R. Adamczak, W. Duch, and N. Jankowski. New developments in the Feature Space Mapping model. In *Third Conference on Neural Networks and Their Applications*, pages 65–70, Kule, Poland, Oct. 1997.

[2] C. M. Bishop. Improving the generalization properties of radial basis function neural networks. *Neural Computation*, 3(4):579–588, 1991.

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

[4] W. Duch and N. Jankowski. Survey of neural transfer functions. *Neural Computing Surveys*, 7, 1999. (submitted).

[5] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6), Aug. 1998.

[6] N. Jankowski. *Ontogenic neural networks and their applications to classification of medical data*. PhD thesis, Department of Computer Methods, Nicholas Copernicus University, Toruń, Poland, 1999. (in preparation).

[7] 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, Oct. 1997.

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

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

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

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

[12] 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.