# STATISTICAL CONTROL OF GROWING AND PRUNING IN RBF-LIKE NEURAL NETWORKS

**Norbert Jankowski**[1] **Visakan Kadirkamanathan**[2]

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

Department of Automatic Control
and Systems Engineering
The University of Sheffield
Mappin Street Sheffield
S1 3JD, UK

**Abstract:** *Incremental Net Pro* (IncNet Pro) with local learning feature and statistically controlled growing and pruning of the network is introduced. The architecture of the net is based on RBF networks. *Extended Kalman Filter* algorithm and its new fast version is proposed and used as learning algorithm. IncNet Pro is similar to the *Resource Allocation Network* described by Platt in the main idea of the growing. The statistical *novel criterion* is used to determine the growing point. IncNet Pro use pruning method similar to *Optimal Brain Surgeon* by Hassibi, but based on Extended Kalman Filter algorithm. The Bi-radial functions are used instead of radial basis functions to obtain more flexible network.

## 1 Introduction

The *Radial Basis Function* (RBF) networks [21, 20, 18, 1] were designed as a solution to an approximation problem in multi–dimensional spaces. The typical form of the RBF network can be written as

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

where $M$ is the number of the 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 which is usually choosed as a Gaussian ($e^{-||\mathbf{x}-\mathbf{t}||^2/b^2}$), multi-quadratics or thin-plate spline function[3]. In contrastopposition to many *artificial neural networks* (ANNs) including well known *multi-leyered perceptrons* (MLPs) networks the RBF networks have well mathematical properties. Girosi and Poggio [9, 20] proved the existence and uniqueness of best approximation for regularization and RBF networks. In the 1991 Paltt published the article on the *Resource–Allocating Network* [19]. The RAN network is an RBF-like network that grows when criteria are satisfied:

$$\mathbf{y}_n - f(\mathbf{x}_n) = e_n > e_{min}; \qquad ||\mathbf{x}_n - \mathbf{t}_c|| > \epsilon_{min} \tag{2}$$

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

$$f^{(n)}(\mathbf{x},\mathbf{p}) = \sum_{i=1}^{k-1} w_i G_i(\mathbf{x},\mathbf{p}_i) + e_n G_k(\mathbf{x},\mathbf{p}_k) = \sum_{i=1}^{k} w_i G_i(\mathbf{x},\mathbf{p}_i) \tag{3}$$

where $\mathbf{p}_k$ includes centers $\mathbf{x}_n$ and others 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 free parameters. Although LMS algorithm is faster than *Extended Kalman Filter* (EKF) algorithm [2] we decided to used EKF algorithm because it exhibits fast convergence, use lower number of neurons in hidden layer [13, 14] and gives some *tools* which would be useful in control of the growth and pruning process.

[3]For a interesting review of many other transfer function see [4].

**The Goal of IncNet Pro** The main goal of our researche was to build a network which would be able to preserve information in as complex net as the data shown to the network during the learning time. The IncNet Pro tries to solve the above task in 4 ways: • ESTIMATION: The typical learning process by EKF algorithm. • GROWING: If the *novelty criterion* is satisfied then a new neuron is added to the hidden layer. • DIRECT PRUNING: IncNet algorithm checks whether or not a neuron should be pruned. If yes, then the neuron with the smallest saliency is removed. • BI-RADIAL FUNCTIONS: The Bi-radial transfer function estimate more complex density of input data through using the separate biases and separate slopes in each dimension and for each neuron.

Similar work has been done taken in recent years by several authors, but it was quite rare to combine growing and pruning in one network, which is quite important for optimal generalization of the network. Weigend, Rumelhart & Huberman [24] described weight-decay, pruning neurons with smallest magnitude of weights. LeCun et al. [17] described more effective pruning method, *Optimal Brain Damage*. Hassibi in 1993 [10] published the *Optimal Brain Surgeon* algorith, which works without assumption used by LeCun that the Hessian matrix is near diagonal.

RAN network using EKF learning algorithm (RAN-EKF) was proposed by [14]. The M-RAN net [25] is based on RAN-EKF with pruning based on removing neurons with smallest normalized output from hidden layer. The previous version of the IncNet [12] is a RAN-EKF network with statistically controlled growth criterion. Another very good example, derived from MLP network, is the *Cascade–Correlation* algorithm[5, 6]. *Feature Space Mapping* (FSM) system is the system which joins two strategies: growing and pruning, see [3] for more information. For more exhaustive description of ontogenic neural network see [7].

## 2 The IncNet Pro Framework

**EKF:** We decided to use the EKF algorithm [2] because it can estimate not only the parameters (weights, biases, etc.), but also some others values such as the output of the network for a given input vector, uncertainty in the expected output ($R_y$), matrix $\mathbf{P}$ which represents the uncertainty in the estimated parameters $\mathbf{p}_n$. All of this estimates carry useful information. The EKF equations can be written as follows:

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

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 (sometimes decreasing to small values around 0.00001) and $\mathbf{I}$ is the identity matrix.

**Fast EKF:** Covariance matrix $\mathbf{P}_n$ can be quite large for real data because its size is the square of the total number of adaptive parameters. Assuming that correlations between parameters of different neurons are not very important we can simplify the matrix $\mathbf{P}_n$ assuming block-diagonal structure:

$$\widetilde{\mathbf{P}}_n = \begin{bmatrix} \widetilde{\mathbf{P}}_n^1 & 0 & \cdots & 0 \\ 0 & \widetilde{\mathbf{P}}_n^2 & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & \widetilde{\mathbf{P}}_n^M \end{bmatrix} \tag{5}$$

where $\widetilde{\mathbf{P}}_n^i$ consists of correlations of adaptive paramiters of $i$-th neuron.

Let $m$ be the number of adaptive parameters per neuron and $M$ the number of neurons. The size of matrix $\mathbf{P}_n$ is $m \cdot M \times m \cdot M$, but matrix $\widetilde{\mathbf{P}}_n$ has $m^2 M$ elements not equal to zero. 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 constant in $\mathcal{P}$)! Using this approximation the fast version of the EKF is:

$$\begin{aligned}
e_n &= y_n - f(\mathbf{x}_n; \mathbf{p}_{n-1}) \\
\mathbf{d}_n^i &= \frac{\partial f(\mathbf{x}_n; \mathbf{p}_{n-1})}{\partial \mathbf{p}_{n-1}^i} \\
R_y &= R_n + \mathbf{d}_n^{1\,\mathrm{T}} \widetilde{\mathbf{P}}_{n-1}^1 \mathbf{d}_n^1 + \cdots + \mathbf{d}_n^{M\,\mathrm{T}} \widetilde{\mathbf{P}}_{n-1}^M \mathbf{d}_n^M \qquad i = 1, \ldots, M \\
\mathbf{k}_n^i &= \widetilde{\mathbf{P}}_{n-1}^i \mathbf{d}_n^i / R_y \\
\mathbf{p}_n^i &= \mathbf{p}_{n-1}^i + e_n \mathbf{k}_n^i \\
\widetilde{\mathbf{P}}_n^i &= [\mathbf{I} - \mathbf{k}_n^i \mathbf{d}_n^{i\,\mathrm{T}}] \widetilde{\mathbf{P}}_{n-1}^i + Q_0(n)\mathbf{I}
\end{aligned} \tag{6}$$

In all investigation in the next sections we can use either the EKF or the Fast EKF.

**Novelty Criterion:** Using methods which estimate during learning covariance of uncertainty of each parameter, the network output uncertainty can be determined and use the same criterion as in previous version of IncNet [12] may be used. Then the hypothesis for the statistical inference of model sufficiency is stated as follows:

$$\mathcal{H}_0 : \quad \frac{e^2}{\mathrm{Var}[f(\mathbf{x}; \mathbf{p}) + \eta]} = \frac{e^2}{\sigma_y^2(\mathbf{x}) + \sigma_{ns}^2} < \chi^2_{n,\theta} \tag{7}$$

where $\chi^2_{n,\theta}$ is $\theta$% confidence on $\chi^2$ distribution for $n$ degree of freedom. $e = y - f(\mathbf{x}; \mathbf{p})$ is the error (see Eq. 4). If this hypothesis is not satisfied the current model is not enough 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:

$$\mathcal{H}_0 : \quad e_n^2 / R_y < \chi^2_{n,\theta} \tag{8}$$

If hypothesis $\mathcal{H}_0$ is satisfied then IncNet continues learning using the EKF (or Fast 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: $w_{M+1} := e_n$, $\mathbf{t}_{M+1} := \mathbf{x}_n$, $b_{M+1} := b_0$, $\mathbf{P}_n := \begin{bmatrix} \mathbf{P}_n & 0 \\ 0 & P_0 \mathbf{I} \end{bmatrix}$, 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 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:

$$\mathbf{p}_n = [w_1, \ldots, w_M, \ldots]^{\mathrm{T}} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{P}_w & \mathbf{P}_{wv} \\ \mathbf{P}_{wv}^{\mathrm{T}} & \mathbf{P}_v \end{bmatrix} \tag{9}$$

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.10) 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.

$$\mathcal{P}:\ L/R_y < \chi^2_{1,\vartheta} \qquad\qquad L = \min_i w_i^2/[\mathbf{P}_w]_{ii} \tag{10}$$

where $\chi^2_{n,\vartheta}$ is $\vartheta$% confidence on $\chi^2$ 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:** To obtain greater flexibility the bi-radial transfer functiona [4] źre used instead of Gaussians. These functiona are build from products of pairs of sigmoidal functions for each variable and produce decision regions for classification of almost arbitrary shapes.

$$\mathrm{Bi}(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s}) = \prod_{i=1}^{N} \sigma(e^{s_i}\cdot(x_i - t_i + e^{b_i}))(1-\sigma(e^{s_i}\cdot(x_i - t_i - e^{b_i}))) \tag{11}$$

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 $\mathrm{Bi}(\mathbf{x};\mathbf{t},\mathbf{b},\mathbf{s})$ is possible by shifting centers $\mathbf{t}$, rescaling $\mathbf{b}$ and $\mathbf{s}$, see Fig. 1. The number of adjustable parameters per processing unit is in this case (excluding weights $w_i$) $3N$. Dimensionality reduction is possible as in the *gaussian bar case* [4], but we can obtain more flexible density shapes, thus reducing the number of adaptive units in the network. Exponentials $e^{s_i}$ and $e^{b_i}$ are used instead of $s_i$ and $b_i$ to prevent oscillations during learning procedure (learning becomes more stable).

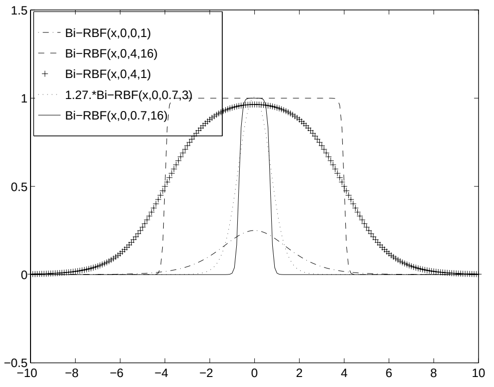

**Figure 1:** A few shapes of the bi-radial functions in two dimensions.

## 3 Results

We have applied th IncNet network to a few well known approximation and classification benchmarks.

**Hermite function approximation.** The first problem is to approximate the Hermite function:

$$f_{her}(x) = 1.1(1 - x + 2x^2)\exp(-1/2x^2) \tag{12}$$

The training data consists of 40 random points in $[-4, 4]$ interval, and test data consists of 100 uniformly distributed points from the same interval.

In two dimensional case for simple problems (such as Hermite function) we may obtain similar results using IncNet with either Gaussian or Bi-radial nodes. Adding pruning also makes rathe small difference (see fig. 2). To solve this problem the network has used just two neurons, with

RMSE[4] at the end of learning process equal to 0.009. Comparition of IncNet Pro with other versions is presented in the table on the right. By IncNet S.G. we mean IncNet with Gaussian nodes and one adaptive parameter, weight (see [12]); RAN-EKF is a RAN net with EKF as learning algorithm (see [12, 14]); RAN is a net described in [19]. All nets were running for 800 iterations (not epochs).

| RMSE errors | | | |
|---|---|---|---|
| IncNet Pro | IncNet S.G. | RAN-EKF | RAN |
| 0.009 | 0.054 | 0.09 | 0.15 |

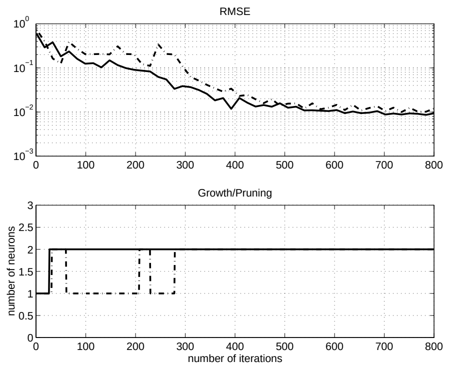

Figure 2: Comparison of IncNet with Gaussian (dashdot line) and Bi-radial (solid line) nodes on 2D approximation problem of Hermite function.

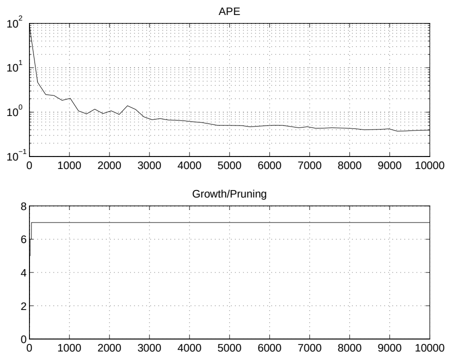

Figure 3: Approximation of Sugeno function.

**Sugeno function approximation.** Now is commonly used in benchmarks tests [23, 16]:

$$f_{sug}(x, y, z) = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \tag{13}$$

216 training points were randomly chosen from $[1, 6]$ interval for all variables, and 125 testing points from $[1.5, 5.5]$ interval. For this function good generalization of IncNet algorithm is found (see the table on the right – for reference of presented methods see[15, 23, 11, 16]). $APE_{TRS}$ and $APE_{TES}$ are the average percentage error on training and test data respectively, defined by: $APE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{f(\mathbf{x}_i)-y_i}{y_i}\right| * 100\%$. Final network had 7 neurons in hidden layer, and the time of learning was about 1000 seconds. Calculations were made on Sun Sparc Station 10.

The convergence of the learning process was presented in Fig. 3. For fast version of IncNet Pro the $APE_{TRS}$ is 0.51 and $APE_{TES}$ is 0.51, and the time has been reduced to about 500 seconds. Learning with Gaussian transfer functions was much more difficult and less stable.

| 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 |
| **IncNet Pro** | **0.45** | **0.39** |
| Fuzzy INET | 0.18 | 0.24 |
| Fuzzy VINET | 0.076 | 0.18 |

[4]To compare some results we will use the *root mean square error* (RMSE) (root of MSE)) defined as follows: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2}$

**Gabor and Girosi functions** are another approximation benchmark considered here [8]:

$$f_{gab}(x,y) = e^{-||\mathbf{x}||^2}\cos(.75\pi(x+y)) \tag{14}$$

$$f_{gir}(x,y) = sin(2\pi x) + 4(y-0.5)^2 \tag{15}$$

For learning only 20 points were used from uniformly distributed interval $[-1,1]\times[-1,1]$ for Gabor function (Eq. 14) and from $[0,1]\times[0,1]$ interval for additive function (Eq. 15). 10,000 points were used in testing phase. The tables[5] describe the MSE errors of different data models.

| | | |
|---|---|---|
| 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** | Model 1 | Model 2 | Model 3 | Model 4 | Model 5 | Model 6 | Model 7 | Model 8 |
| .0013 | .000036 | .000067 | .000001 | .000001 | .000170 | .000001 | .000003 | .000743 |
| .0079 | .011717 | .001598 | .000007 | .000009 | .001422 | .000015 | .000020 | .026699 |

| **Gabor function — MSE train/test** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| .000000 | .000000 | .000000 | .000000 | .345423 | .000001 | .000000 | .456822 | .000044 |
| 0.0298 | .003818 | .344881 | 67.9523 | 1.22211 | .033964 | 98.4198 | 1.39739 | .191055 |

Although the IncNet model is not always the best one it is quite good on average, adapting more flexibly. For Gabor function IncNet used 5 neurons and for additive Girosi function used only 3 neurons.

In Fig. 4 we can see that in the first phase of learning pruning was used quite often. This example shows the power of online pruning method – unnecessary neurons are killed as soon as possible. In this way the learning time reduces sometimes significiently.

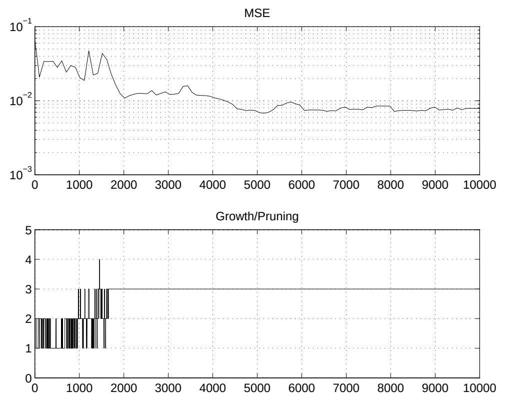

**Figure 4:** Hopeful pruning.

[5]Models 1 to 8 originally published by Girosi, Jones and Poggio in [8].

**The two-spiral problem.** [6]

is quite hard problem for many neural networks, especially when the network should discover not only the *spiral*, but also discover the architecture of the net. The data consists of two sets (training and testing) with 194 patterns each for two spirals (see figure 5).

After 10,000 iterations (it took about 1 hour on Sun 10/41) we got result which fits 192 points of 194 (99%) for training set and 191 (98.5%) for the test set. Final net has 79 neurons and it was build using Fast-EKF algorithm with IncNet – for this benchmark Fast-EKF accelerates computation 50 times!

RBF net with Bi-radial transfer function is able to get lower classification error but using more neurons in the hidden layer (about 100). There are other nets which are able to solve the two-spiral problem too, for example one of the best is an MLP using a global optimization algorithm by Shang and Wah [22]. Their network is able to get 100% correct results for hte training set but never more than 95.4% for the test set. Although it used only 6 neurons, it takes about 200 minutes to train.

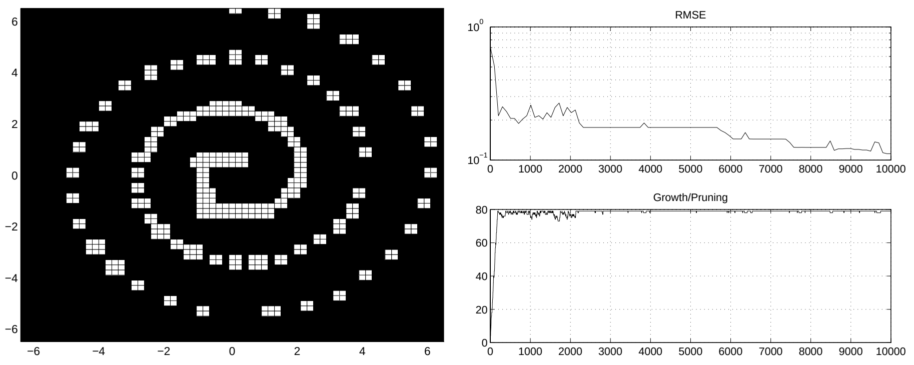

Figure 5: Two-spiral problem.

## 4 Conclusions

The IncNet network is able to control the complexity of its structure by growing and pruning. It is also important that it is direct control. In spite of incremental feature of algorithm, the *pruning time* is determined by theoretical criterion — not in random time moment or by checking the error on the whole training/test data set. Other advantage of the direct pruning is saving of the time of computation. Nearly all parameters of the network are controlled automatically by EKF algorithm, the rest is very similar for different benchmark problems (excluding the biases and slopes, which define the *resolution* of data).

Another positive feature of IncNet Pro is the capacity of uniform generalization. In many benchmarks (see section 3) the errors on testing and training sets are much more similar than for other networks.

In some of classification problems it would be usehul to add the possibility of merging two neurons which can be replaced by another neuron without loss of information, for example using

[6]Benchmark data can be found at: http://www.cs.cmu.edu/afs/cs/project/connect/bench/.

the criterion:

$$\int_{d \subseteq \mathcal{D}} \left( G_i(\mathbf{x}) + G_j(\mathbf{x}) - G_{new}(\mathbf{x}) \right) < \alpha$$

where $G_i$ and $G_j$ are neurons-candidates to be replaced by $G_{new}$ with $\alpha$ confidence.

**Acknowledgments:** I am garetful to prof. Włodzisław Duch for many valuable commants.

## References

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

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

[3] W. Duch and G. H. F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1994.

[4] W. Duch and N. Jankowski. New neural transfer functions. *Jour. of Applied Math. and Computer Science*, 1997. (in print).

[5] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, 1990.

[6] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Feb. 1990.

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

[8] 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.

[9] F. Girosi and T. Poggio. Networks and the best approximation property. AI Lab. Memo, MIT, 1989.

[10] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993.

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

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

[13] 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.

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

[15] T. Kondo. Revised GMDH algorithm estimating degree of the complete polynomial. *Trans. Soc. Instrument and Control Engineers*, 22:928–934, 1986.

[16] W. Kosinski and M. Weigl. Mapping neural networks and fuzzy inference systems for approximation of multivariate function. In E. Kacki, editor, *System Modelling Control'95*, pages 60–65, Łódź, May 1995.

[17] Y. LeCun, J. Denker, S. Solla, R. E. Howard, and L. D. Jackel. Optimal brain damage. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems II*. Morgan Kauffman, 1990.

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

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

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

[21] 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.

[22] Y. Shang and W. Wah. Global optimization for neural network training. *IEEE Computer*, 29, 1996.

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

[24] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Back–propagation, weight elimination and time series prediction. In *Proceedings of the 1990 Connectionist Models Summer School*, pages 65–80. Morgan Kaufmann, 1990.

[25] L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computations*, 9:461–478, 1997.