European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 3, no. 3-4, 2008, pp. 221-230
ISSN 1790–8140

Approximation Capabilities of Neural Networks

C. Enăchescu[1]

Department of Computer Science,
Faculty of Sciences and Letters,
Petru Maior University of Tirgu Mureş,
540088, Romania

Received 7 March, 2007; accepted in revised form 10 July, 2008

Abstract: Algorithm-based computers are programmed, i.e., there must be a set of rules which, a priori, characterize the calculation that is implemented by the computer. Neural computation, based on neural networks, solve problems by learning, they absorb experience, and modify their internal structure in order to accomplish a given task. In the learning process, the available information is usually divided into two categories, examples of function values or training data and prior information, e.g. smoothness constraint, or other particular properties [3]. From the learning point of view, the approximation of a function is equivalent with the learning problem of a neural network. In this paper we want to show the capabilities of a neural network to approximate arbitrary continuous functions and to build a practical neural network to approximate a continuous function. We have made some experiments in order to confirm the theoretical results.

Keywords: Neural networks, Approximation, Learning

Mathematics SubjectClassification: 92B20, 65D99, 68T05

PACS: 02.60.Ed, 07.05.Mh

1. Introduction

Neural computing represents an alternative computational paradigm to the algorithmic one (based on a programmed instruction sequence). Neural computation is inspired by knowledge from neuroscience, though it does not try to be biologically realistic in details [12].

Today's research in neural computation is largely motivated by the possibility of constructing artificial neural networks. Artificial neural networks are simplified models of the biological neural networks.

An artificial neuron has several inputs, n, say. Each input x_i arrives from another neuron (*dendrites*). Each input has a connection strength associated with it. For a given neuron, call it neuron j, the connection strength on the input coming from neuron i is written w_{ji}. An activation function f will be will be applied to the sum of weighted inputs and an output y_j will be generated (*axon*). The artificial neuron performs the following operations:

- a summation of weighted inputs: $\sum_i w_{ji} \cdot x_i$ (1)
- a non-linear thresholding of this sum: $f\left(\sum_i w_{ji} \cdot x_i\right)$ (2)

[1] E-mail: ecalin@upm.ro

- the output of this neuron, *j*, is: $y_j = f\left(\sum_i w_{ji} \cdot x_j\right)$ (3)

We will deal with neural networks organized in layers, where the information is transmitted forward from the first layer until the last layer. These types of feed forward multilayer neural networks are called MLP (Multi Layer Perceptrons [6]).

MLP learn in a supervised manner [4]. Learning represents the process where the input patterns are presented repeatedly and the weights are adjusted according to a learning algorithm, which in this supervised case, takes into consideration the quantitative difference between the target output and the current output. The supervised learning of the neural network uses a training set has the following form:

$$T = \{(\mathbf{x}_i, \mathbf{z}_i) \mid i = 1,2,\ldots,N\} \tag{4}$$

$z_i \in \boldsymbol{R}^m$ is the *m*-dimensional target vector that is provided by a trainer. $N \in \boldsymbol{N}$ is a constant that represents the number of samples used for training. Usually the training set T is obtained from a probabilistic known distribution or from another active procedure [6].

Using the probabilistic distribution the trainer selects a certain input vector x_i, and provides the appropriate target vector z_i. The learning algorithm will compute the difference between the output generated by the neural network y_i and the desired target vector z_i, which will represent the error signal:

$$e_i = y_i - z_i, i = 1,2,\ldots,N \tag{5}$$

The signal error is used to adapt the synaptic weight w_{ji} using a gradient descendent strategy [10]:

$$w_{ji} = w_{ji} + \eta \frac{\partial E}{\partial w_{ji}} \tag{6}$$

where $\eta \in (0,1)$ is the learning rate, controlling the descent slope on the error surface which is corresponding to the *learning error* function E_l:

$$E_l = \frac{1}{2}\sum_{i=1}^{N}(y_i - z_i)^2 \tag{7}$$

A set $Q \subset T$ of samples which haven't been used in the training phase will be used to measure the ability of the neural network to generalize. The error E_g will represent the generalization error:

$$E_g = \frac{1}{2}\sum_{y,z \in Q}(y - z)^2 \tag{8}$$

2. Neural Networks versus Approximation Theory

The approximation theory deals with the problem of approximating or interpolating a continuous, multivariate function $f : \boldsymbol{R}^n \to \boldsymbol{R}^m$ by an approximating function F_W having a fixed number of parameters $W = (w_1, w_2, \ldots, w_p) \in \boldsymbol{R}^p$. For a choice of a specific approximating function F, we want to find the set W of parameters that provides the best possible approximation of f on the set of examples.

From this point of view, the approximation of a function is equivalent with the learning problem for a neural network.

In this paper we want to show the capabilities of a neural network to approximate arbitrary continuous functions. In the same time we will analyze the "best approximation property" of the neural networks and build a practical neural network to approximate continuous functions.

2.1. Neural Networks are Universal Approximators

The questions which arise is whether MLP neural networks are in fact inherently limited to approximate only some special class of functions or they are universal approximators. This approach prompts a natural problem that has to be solved if we want to construct a multilayer feed forward neural network [13]. Given a function f, is it possible to approximate this given f function with a given ε precision?

In the literature we can find many important results concerning the capability of the neural networks to be universal approximators [4], [9], [11], [14], [17], [19], [21].

The approximation problem can be formulated formally as:
if $f : X \subseteq R^n \to R^m$ is a continuous function, d is an Euclidean distance, and $F_W : R^n \to R^m$ is an approximation function that depends continuously on $W \in \boldsymbol{P}$, determine the parameters W^*, such that:

$$d\left[F_{W^*}(x), f(x)\right] \le d\left[F_W(x), f(x)\right], (\forall) W \in P \tag{9}$$

A solution to this problem, if it exists, is said to be *a best approximation*. Typical results deal with the possibility, given a network, of approximating any continuous function arbitrarily well. In mathematical terms this means that the set of functions that can be computed by the network is dense. We can built neural networks such the corresponding set of approximating function is dense in C[$\boldsymbol{R}$]. We will consider the following general neural network:

- an input layer with n input neurons;
- a hidden layer with N hidden neurons and the activation functions H_i;
- an output layer having only one output neuron[2] with the identity function $1_x(x) = x$ as activation function;
- the input neurons are connected to the hidden neurons by the weights $W_i \in \boldsymbol{R}^n$;
- the hidden neurons are connected to the output neuron by the weight $k_i \in \boldsymbol{R}$;

The class of approximating functions corresponding to it is:

$$\boldsymbol{F} = \{ f \in C[U] \left| f(x) = \sum_{i=1}^{N} k_i \cdot H_i(x; W_i), U \subseteq R^n, H \in C[U], N \in \mathbf{N} \right\} \tag{10}$$

The Stone-Weierstrass [2] theorem can be used to show that certain network architectures possess the universal approximation capability.

Theorem (Stone-Weierstrass): Let domain $\boldsymbol{U}$ be a compact space of n dimensions, and let $\boldsymbol{F}$ be a set of continuous real-valued functions on $\boldsymbol{U}$, satisfying the following criteria:
(C1) Identity Function: The constant function f(x) = 1 is in $\boldsymbol{F}$;
(C2) Separability: For any two points $x_1 \neq x_2$ in $\boldsymbol{U}$, there is a function $f \in \boldsymbol{F}$ such that $f(x_1) \neq f(x_2)$;
(C3) Algebraic closure: if $f, g \in \boldsymbol{F}$, then fg, $af + bg \in \boldsymbol{F}$ $(\forall)$ a,b$\in \boldsymbol{R}$; Then $\boldsymbol{F}$ is dense in C[$\boldsymbol{U}$], the set of continuous functions on $\boldsymbol{U}$.
In other words, $(\forall)\ \varepsilon > 0$ and $(\forall)\ g \in$ C[$\boldsymbol{U}$] $(\exists) f \in \boldsymbol{F}$ such that $|g(x) - f(x)| < \varepsilon$.

2.2. Neural Networks and the Best Approximation Theory

We can give an informal formulation of the approximation problem: given a function $f \in \boldsymbol{F}$, and $\boldsymbol{A} \subseteq \boldsymbol{F}$, find the element $a \in \boldsymbol{A}$ that is the "closest" to f.

In order to give a mathematical meaning to the approximation problem, we will introduce some definitions [8]:

Definition 1: Given $f \in \boldsymbol{F}$ and $\boldsymbol{A} \subseteq \boldsymbol{F}$, we call the distance of f from $\boldsymbol{A}$ as $d(f, A) = \inf\limits_{a \in A} \|f - a\|$

Definition 2: If there exist $\inf\limits_{a \in A} \|f - a\|$ and there exist $a_0 \in \boldsymbol{A}$ such that $\|f - a_0\| = d(f, A)$, then a_0 is said to be a best approximation to f from $\boldsymbol{A}$.

Definition 3: A set $\boldsymbol{A}$ is called an existence set if, to each $f \in \boldsymbol{F}$ there is at least one best approximation to f from $\boldsymbol{A}$.

[2] Because a mapping f: $R^n \to R^m$ can be computed by m mappings f_i: $R^n \to R$, i=1,2,...,m it is sufficient to focus on networks with one output neuron only.

Definition 4: A set $\boldsymbol{A}$ is called an uniqueness set if, to each $f \in \boldsymbol{F}$ there is at most one best approximation to f from $\boldsymbol{A}$.

Definition 5: A set $\boldsymbol{A}$ is called a Cebysev set[3] if, to each $f \in \boldsymbol{F}$ there is one and exactly one best approximation to f from $\boldsymbol{A}$.

The precise formulation of the approximation problem is: given $f \in \boldsymbol{F}$ and $\boldsymbol{A} \subseteq \boldsymbol{F}$ find a BA (Best Approximation) to f from $\boldsymbol{A}$.

Following the ideas presented in [8] we will show some simple properties of the existence sets, and to try to apply these results to network architectures.

Proposition 1: Every existence set is closed.

Proposition 2: Let $\boldsymbol{A}$ be a compact set in a metric space $\boldsymbol{F}$. Then $\boldsymbol{A}$ is an existence set.

We will try to apply these simple results to MLP neural networks. From the approximation theory point of view, a MLP neural network is a representation of a set $\boldsymbol{A}$ of parametric functions, and the learning algorithm corresponds to the search of the BA to some target function f from $\boldsymbol{A}$.

Since the approximating functions associated with MLP networks with one hidden layer does not have the BA property [8], it is natural to ask whether it is possible to build neural networks which posses this property.

The answer is positive, and is based on the regularization theory [23], the connection between this theory and the neural networks being presented in [21].

Regularization techniques typically impose smoothness constraints on the approximating set of functions. It can be argued that some form of smoothness is necessary to allow meaningful generalization in approximation type problems. A similar argument can also be used in the case of classification where smoothness involves the classification boundaries rather than the input-output mapping itself.

Our use of regularization, which follows the classical technique, introduced by Tikhonov [22], [23], identifies the approximating function as the minimum of a cost functional that includes an error term $\frac{1}{2}\sum_i (z_i - y_i)^2$ and a smoothness functional $\frac{1}{2}\Phi[f]$, usually called a stabilizer.

In the Bayesian interpretation [6] of regularization the stabilizer corresponds to a smoothness prior, and the error term to a model of the noise in the data (usually Gaussian and additive).

Suppose that the training set $T = \{(x_i, z_i) \mid i = 1,2,\ldots,N\}$ has been obtained by random sampling of a function f, belonging to some space of functions $\boldsymbol{X}$ defined on $\boldsymbol{R}^n$, in the presence of noise, and suppose we are interested in recovering the function f, or an estimate of it, from the set of data $\boldsymbol{T}$. This problem is clearly ill-posed [18], since it has an infinite number of solutions. In order to choose one particular solution we need to have some a priori knowledge of the function that has to be reconstructed. The most common form of a priori knowledge consists in assuming that the function is smooth, in the sense that two similar inputs correspond to two similar outputs.

The main idea underlying regularization theory is that the solution of an ill-posed problem can be obtained from a variational principle [22], which contains both the data and prior smoothness information. Smoothness is taken into account by defining smoothness in such a way that lower values of the functional correspond to smoother functions.

[3] A Cebysev set is a existence set and a uniqueness set.

Since we look for a function that is simultaneously close to the data and also smooth, it is natural to choose as a solution of the approximation problem the function that minimizes the following functional [5]:

$$H[f] = \frac{1}{2}\sum_i (y_i - z_i)^2 + \frac{1}{2}\lambda\Phi[f] \tag{11}$$

where λ is a positive number that is usually called the regularization parameter. The first term is enforcing closeness to the data, and the second smoothness, while the regularization parameter controls the trade off between these two terms. It can be shown that, for a wide class of functionals, the solutions of the minimization of the functional (11) all have the same form [9].

We first need to give a more precise definition of what we mean by smoothness and define a class of suitable smoothness functional. We refer to smoothness as a measure of the oscillatory behavior of a function. Therefore, within a class of differentiable functions, one function will be said to be smoother than another one if it oscillates less. If we look at the functions in the frequency domain, we may say that a function is smoother than another one if it has less energy at high frequency (smaller bandwidth) [10].

First high-pass filtering the function, and then measuring the power, that is the L_2 norm, of the result, can measure the high frequency content of a function. In formulas, this suggests defining smoothness functional of the form:

$$\Phi[f] = \int_{\mathbf{R}^n} d\mathbf{s} \frac{\left|\tilde{f}(\mathbf{s})\right|^2}{\tilde{G}(\mathbf{s})} \tag{12}$$

where ~ indicates the Fourier transform, $\frac{1}{\tilde{G}}$ is some positive function that falls to zero as $\|s\| \to \infty$ (so that $\frac{1}{\tilde{G}}$ is an high-pass filter) and for which the class of functions such that this expression is well defined is not empty [20].

Depending the choice we make with the function *G*, the regularization term $\frac{1}{2}\Phi[f]$ can have or not a void null space. Therefore we can define an equivalence relation on the set of functions which are different relative to an element of the null space of the regularization term. The function that minimizes the functional (11) has the following form [21]:

$$f(\mathbf{x}) = \sum_i w_i G(\mathbf{x};\mathbf{x}_i) + p(\mathbf{x}) \tag{13}$$

where $p(\mathrm{x})$ is a term belonging to the null space of the regularization term $\frac{1}{2}\Phi[f]$.

We make the following notation: $\{\mu_\alpha\}_{\alpha=1}^k$ is a base of the *k*-dimensional null space of the regularization term $\frac{1}{2}\Phi[f]$, and d_α real constants, we have the following general solution:

$$f(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x};\mathbf{x}_i) + \sum_{\alpha=1}^{k} d_\alpha \mu_\alpha(\mathbf{x}) \tag{14}$$

In practical application we can consider classes of stabilizers with void null space. Therefore, without reducing the generality of the solution of the minimization of functional (11), we can consider the following practical solution [6], [21]:

$$f(\mathbf{x}) = \sum_i w_i G(\mathbf{x};\mathbf{x}_i) \tag{15}$$

The solution (15) of the variational problem (11) has a simple interpretation in terms of a MLP neural network with one hidden layer [5]. Let's analyze the architecture of the neural network that corresponds to the solution function (15):

The architecture of the neural network corresponds to a neural network of the MLP type with one hidden layer:

- The input layer contains *n* input neurons, *n* representing the dimensionality of the input space $\mathbf{x}_i = \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right) \in \boldsymbol{R}^n$.

- The hidden layer having a number of hidden neurons equal to the dimension of the training set $T = \{(\mathbf{x}_i, f(\mathbf{x}_i)) | i = 1,2,\ldots,N\}$. The activation functions of the hidden neurons are the Green functions $G(\mathbf{x} - \mathbf{x}_k)$ [4]. The dimension of the hidden layer can be reduced using an unsupervised clustering algorithm [6];
- The output layer contains one single output layer having as activation function a linear function or a special weighted functions of the output values generated by the neurons in the hidden layer [1];

Synaptic weights:

- The weights between the input layer and the hidden layer are included in the form of the activation functions of the hidden neurons. These weights are not explicitly presented in the mathematical equation (15);
- The vector $\mathrm{w} = (w_1, w_2, \ldots, w_N)$ represents the weights between the hidden layer and the output layer.

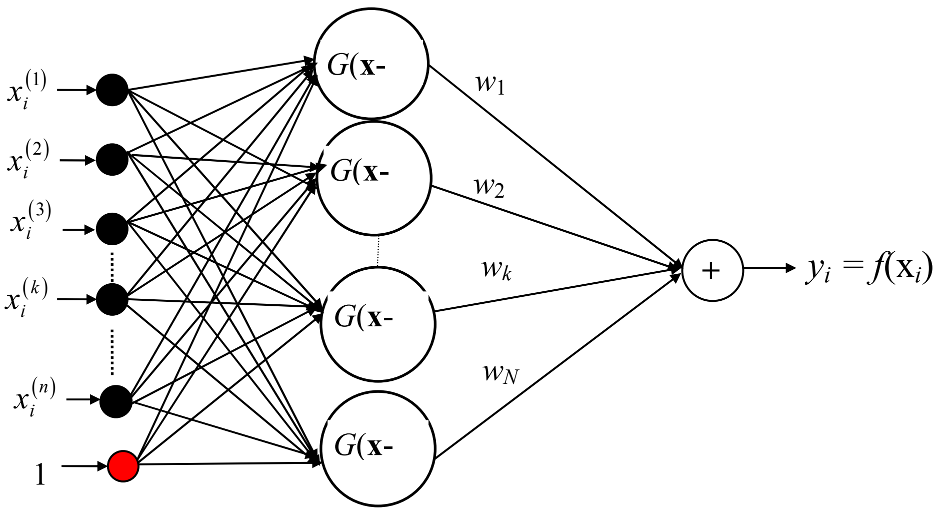

Figure 1: Architecture of a neural network that corresponds to the solution function (15).

In some particular cases, the unknown parameters which correspond to the weight vector *w*, can be calculated using a straight forward computation [10]:

$$w = (G + \lambda I)^{-1} z \tag{16}$$

In practice, the computation (15) is unrealistic due to the requirement that the matrix $(G + \lambda I)^{-1}$ should exist and because of the high computational complexity of the necessary calculations. For this reason, to avoid this high computational complexity, a dimensional reduction should be performed, with the goal to reduce the numbers of neurons in the hidden layer [6]. After this dimensional reduction, the weight vector *w* can be computed using a supervised learning strategy of Back Propagation type (gradient descendent) [12].

In the particular case of RBF (Radial Basis Function), we have $G(\|x - x_i\|) = e^{-\left(\frac{\|x_i - x\|}{\sigma_i}\right)^2}$, and the solution of the approximation problem is then a linear superposition of radial Green's functions "centered" on the data points [16].

The approximated solution belongs to the following subset of $C[U]$:

$$\Omega = \left\{ f \in C[U] \middle| f(x) = \sum_{i=1}^{N} k_i G(x; x_i), k_i \in R \right\} \tag{17}$$

Our intention is to prove that the set Ω has the BA property [8]. For this we will need to state the following property:

Proposition 3: The set Ω is an existence set.

In this way we are able to build a neural network which posses the BA property. We will have a neural network with an architecture like the one is presented in Figure 1, taking as activation functions for the neurons in the hidden layer $H_i = G(x;x_i)$, $i = 1, 2, ..., N$.

3. Network Structure and Learning

From the practical perspective the above solution is not feasible, because when the number of data points becomes large; the complexity of the network may become too high, being proportional to the number of data points.

So we will apply the following strategy:

- we will consider an approximation of the function $f(x) = \sum_{i=1}^{N} k_i G(x;x_i)$, with the Gaussian radial basis function:

$$f^*(x) = \sum_{i=1}^{K} k_i e^{-\left(\frac{\|m_i - x\|}{\sigma_i}\right)^2} \tag{18}$$

where: $K << N$;

- m_i is the mean value of the data inputs x_i in a local neighborhood and is often referred to as the i-th centre or mean; These m_i values can be imagined to be centers of n-dimensional balls of radius σ_i that define regions of influence.
- σ_i is the receptive field width associated with the i-th region.

The learning algorithm for determining the parameters of the approximator f^* is accomplished in two steps:

Step1: K-means clustering algorithm [6] is used to cluster the input data and thereby determine the centers m_i for the hidden neurons. The objective in the clustering phase is to locate K centers (corresponding to K hidden neurons). Once the centers are determined, the widths σ_i are set using the nearest-neighbor heuristic: σ_i is set to the Euclidean distance from m_i to m_j , where m_j is the centre closest to m_i (and distinct).

Step2: the weights k_i are trained using gradient descendent [10].

In this scheme the BA property is preserved while the computational complexity has been reduced with respect to the exact solution of the regularization problem.

4. Simulations and experiments

In order to study the properties of the RBF networks obtained as a theoretical result, we have implemented this type of neural network and we have studied the learning capabilities and the generalization capabilities.

We have taken in consideration as target function, to be approximated, the following function:

$$F(x) = \frac{1}{x^2 + 0.01} + 100x + 20\sin(x) \tag{19}$$

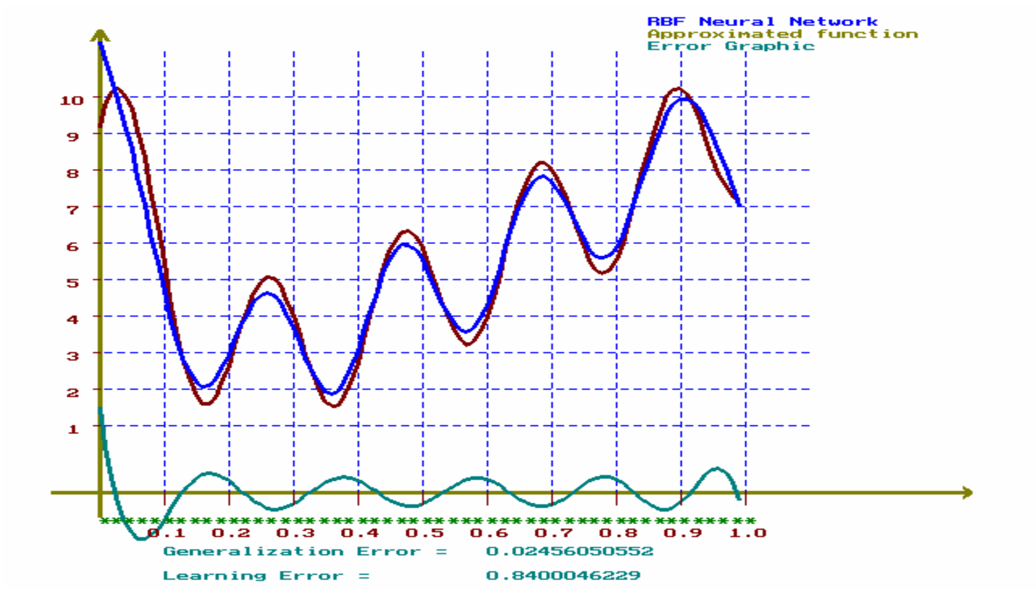

Figure 2: Simulation of the learning process: training set with 50 examples, target function (19), 5.000 epochs. Results $E_l = 0.8400046229$, $E_g = 0.02456050552$.

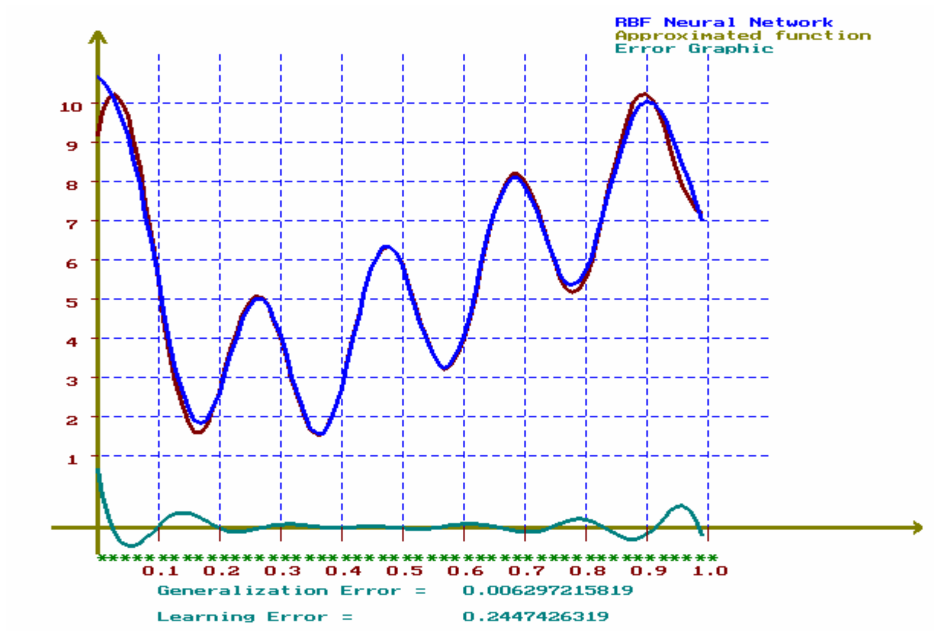

Figure 3: Simulation of the learning process: training set with 50 examples, target function (19), 15.000 epochs. Results $E_l = 0.244744266319$, $E_g = 0.006297215819$.

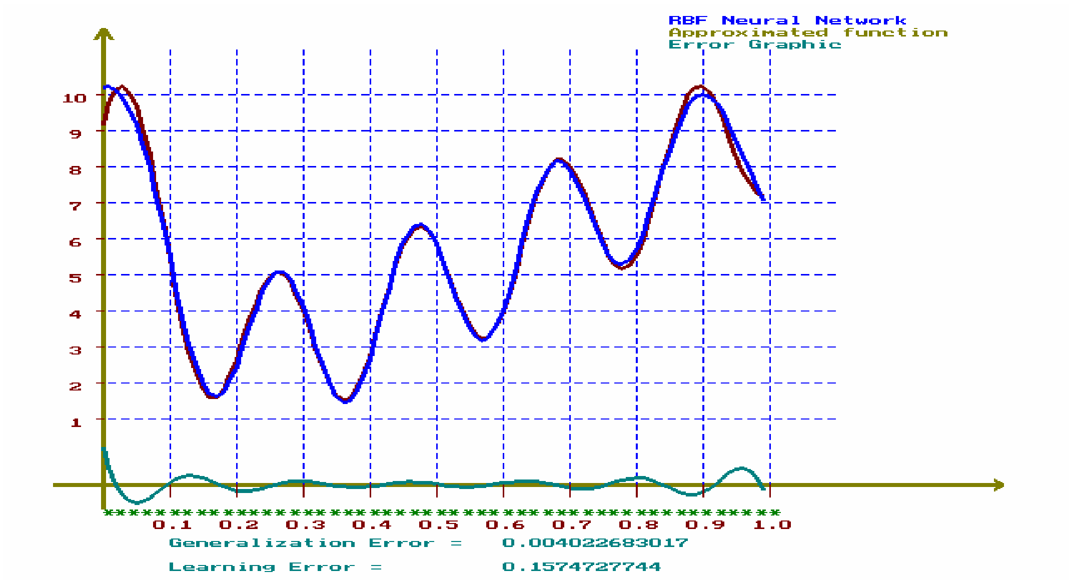

Figure 4: Simulation of the learning process: training set with 50 examples, target function (19), 25.000 epochs. Results $E_l = 0.1574727744$, $E_g = 0.004022683017$.

Some of the simulations regarding the learning capabilities of the RBF neural network, which is equivalent with the process of approximating the function (19), using as examples 50 points uniform distributed on the [0,1] interval.

Nr. of epochs	***E_l Learning error***	***E_g Generalization error***	Activation function
5000	0.8400046229	0.02456050552	Gaussian
10000	0.38664092	0.010000231	-
15000	0.244744266319	0.006297215819	-
25000	0.1574727744	0.004022683017	-
5000	1.219456601	0.1661223741	multi-quadratic inverse function
10000	1.200011187	0.1680013331	-
15000	1.119925620	0.1565290011	-
25000	0.956610098	0.1112432100	-
5000	2.745756980	0.3488366621	multi-quadratic function
10000	2.001883509	0.2761120911	-
15000	1.870343121	0.1135634901	-
25000	1.100087623	0.9999234376	-

Table1: Results of the simulations, describing number of epochs, E_l and E_g and the activation function.

References

[1] G. Bugmann, *Note on the use of Weight-Averaging Output Nodes in RBF-Based Mapping Nets*. Research Report CNAS-96-02, Centre for Neural and Adaptive Systems, University of Plymouth (1996).

[2] N.E. Cotter, *The Stone-Weierstrass Theorem and Its Application to Neural Networks*. IEEE Transactions on Neural Networks, 4, 290-295 (1990).

[3] C. Enăchescu, D, Rădoiu, O. Adjei, *Learning strategies using prior information*. IJICIS-International Journal of Intelligent Computing and Information Science, Vol. 5, Nr. 1, pp. 381-393 (2005).

[4] C. Enăchescu, *Neural Networks as approximation methods*. International Conference on Approximation and Optimization Methods, ICAOR'96, Babes-Bolyai University, Cluj-Napoca (1996).

[5] C. Enăchescu, *Properties of Neural Networks Learning.* 5th International Symposium on Automatic Control and Computer Science, SACCS'95, Vol.2, 273-278, Technical University (1995).

[6] C. Enăchescu, *Neural Computing*. Ph.D. Thesis, Babes-Bolyai University, Cluj-Napoca (1997).

[7] K., Funahashi, *On the approximate realization of continuous mappings by neural networks*. Neural Networks, 2, 183-192 (1989).

[8] F. Girosi, T. Pogio, *Networks and the Best Approximation Property*. Biological Cybernetics, 63, 169-176 (1990).

[9] F. Girosi, M. Jones, T. Poggio, Priors, *Stabilizers and Basis Functions: from regularization to radial, tensor and additive splines*. M.I.T, A.I. Memo No. 1430 (1993).

[10] S. Haykin, *Neural Networks. A Comprehensive Foundation*. IEEE Press, MacMillian, (1994).

[11] R. Hecht-Nielsen, *Kolmogorov Mapping Neural Network Existence Theorem*. IEEE International Conference on Neural Networks, 3, 11-13 (1987).

[12] J. Hertz, A. Krogh, R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Co. (1992).

[13] K. Hornik, M. Stinchcombe, H. White, *Multilayer Feedforward Networks Are Universal Approximators*. Neural Networks, 2, 359-366 (1989).

[14] B. Irie, S. Miyake, *Capabilities of Three-Layers Perceptrons*. IEEE International Conference on Neural Networks, 1, 641-648 (1988).

[15] L.V. Kantorovich, G. P. Akilov, *Functional Analysis*, 2nd edition, Oxford: Pergamon (1982).

[16] A. Kolmogorov, *On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition*. Doglady Akademii Nauk SSSR, 144, 679-681 (1957).

[17] V.Y. Kreinovich, *Arbitrary Nonlinearity Is Sufficient to Represent All Functions by Neural Networks: A Theorem.* Neural Networks, 4, 381-383 (1991).

[18] V. Kurkova, *Learning from Data as an Inverse Problem*. In COMPSTAT 2004 – Proceedings on Computer statistics (J. Antoch Ed.), 1377-1384, Heidelberg: Phisica-Verlag / Springer (2004).

[19] V. Kurkova, *Supervised Learning as an Inverse Problem*. Research Report ICS-2004-960, Institute of Computer Science, Prague (2004).

[20] C.A. Micchelli, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions.* Constr. Approx., Vol. 2, 11-22 (1986).

[21] T. Pogio, F. Girosi, *Networks for Approximation and Learning*. Proceedings of the IEEE, Vol. 78, 9, 1481-1497, Sept. (1990).

[22] A.N. Tichonov, V.A. Arsenin, *Solutions of Ill-posed Problems*. Washington, DC: W.H. Winston (1977).

[23] A.N. Tikhonov, *Solution of incorrectly formulated problems and regularization method.* Soviet Math. Dokl., Vol. 4, 1035-1038 (1963).