STUDIA UNIV. "BABEŞ-BOLYAI", INFORMATICA, Volume I, Number 2, October 1996

ACTIVE LEARNING FOR IMPROVING THE PERFORMANCES OF NEURAL NETWORKS

CĂLIN ENĂCHESCU

Abstract. Neural networks learn, they absorb experience, and modify their internal structure in order to accomplish a given task. For a neural network you do not need to find "rules" that characterise a given task. From this point of view, the approximation of a function is equivalent with the problem of training a neural network. In other words, to approximate a function is equivalent to synthesise an associative memory that generates the appropriate output when an input is presented at the input layer and generalises correctly when a new input is presented at the input layer. In the classical forms of supervised learning, the training set is chosen according to some known or random given distribution. The trainer is a passive agent in the sense that he is not able to interact with the training set in order to improve the performances of the neural networks learning. We will investigate some possibilities that allow the trainer to become active and we will analyse the performances of such supervised learning.

1. Introduction

The main feature of the neural computing is the learning capability. Learning is a general concept that must be studied from a mathematical perspective. In this paper we will focus on supervised learning, where we have a "trainer" that provides the desired responses for the neuronal network, when an input is presented to the neural network. In this way we will have a training set $T = \{(\mathbf{x}_i, \mathbf{z}_i) | i = 1, 2, \ldots, N\}$, where $\mathbf{x}_i$ is the input and $\mathbf{z}_i$ is the desired output of the neural network provided by the trainer.

Usually, the trainer has a passive role, being capable to indicate only the desired output values $\mathbf{z}_i$, without having any role in indicating where should be chosen the training samples. In the training process we encounter regions where the learning process is "difficult" or "easy" related to some error measure. In order to improve the performances of the supervised learning process of a neural network, the trainer should give a bigger attention to the "difficult" regions, by

Received by the editors: July 23, 1997.

1991 *CR Categories and Descriptors.* I.2.6 [Artificial Intelligence]: Learning – connectionism and neural nets.

choosing more training samples from that regions. In this way the trainer's roles become more active, having a dynamic influence in the learning process.

2. Mathematical aspects of supervised learning

The supervised learning of the neural networks uses a training set has the following form:

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

where $\mathbf{x}_i \in \mathbf{R}^n$ is the n-dimensional input vector, and $\mathbf{z}_i \in \mathbf{R}^m$ is the m-dimensional target vector that is provided by a trainer. $N \in \mathbf{N}$ is a constant that represents the number of training samples. Usually the training set T is obtained from a probabilistic known distribution. In the classical supervised learning strategy [7] the trainer is a static agent. Using the probabilistic distribution he selects a certain input vector $\mathbf{x}_i$, and provides the appropriate target vector $\mathbf{z}_i$. The learning algorithm will compute the difference between the output generated by the neural network $\mathbf{y}_i$ and the desired target vector $\mathbf{z}_i$, which will represent the error signal:

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

The signal error is used to adapt the synaptic weights w_{ji} using a gradient descendent strategy [7]:

$$w_{ji} = w_{ji} + \eta \frac{\partial E}{\partial w_{ji}} \tag{3}$$

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

$$E = \frac{1}{2} \sum_{i=1}^{N} (y_i - z_i)^2 \tag{4}$$

Let us consider a physical phenomena described by a vector $\mathbf{x} \in \mathbf{R}^n$ which corresponds to a set of independent random variables, and a real[1] number $z \in \mathbf{R}$ that represents a dependent variable. Let us consider that we have N distinct measurements (observations) of variable $\mathbf{x}$:

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_N \tag{5}$$

[1] A neural network with m output neurons can be considered as m distinct neural networks with only one output neuron. This is why it is allowed to consider, without loosing the generality, a neural network with a single output neuron instead of a neural network with m output neurons. In conclusion, we are allowed to consider, when it is necessary $y, z \in \mathbf{R}$ instead of $\mathbf{y}, \mathbf{z} \in \mathbf{R}^m$.

And a corresponding set of scalars z:

$$z_1, z_2, z_3, \ldots, z_N \tag{6}$$

Usually we do not have enough information about the exact relationship that exist among the variable $\mathbf{x}$ and z. For this reason we will consider a relation represented by the following equation:

$$z = f(\mathbf{x}) + \varepsilon \tag{7}$$

where f is a function which depends on the variable $\mathbf{x}$, and ε is the error represented by a random variable. The error ε represents the mistake made in order to estimate the existing functional relation between variables $\mathbf{x}$ and z. Equation (7) is a known statistical model, named regressive model. Using statistical relations [22], we are now able to express the function f of the regressive model as:

$$f(\mathbf{x}) = E[z|\mathbf{x}] \tag{8}$$

where $E[z|\mathbf{x}]$ represents the conditional statistical average, namely, we will have on average the target value z if he has a particular realisation of variable $\mathbf{x}$. In particular, if the functional relation between variables $\mathbf{x}$ and z is known precisely, then we can consider in the regressive model the ideal case $\varepsilon = 0$.

A neural network is a physical mechanism to implement what we have stated in the regressive model: the prediction of z on the bases of $\mathbf{x}$. This main goal is achieved by encoding the information content in the training set (1) in the synaptic weights w_{ji}. It is quite clear that from the neural computing point of view $\mathbf{x}$ represents the input vector presented at the input layer, and z represents the target vector that we wish to obtain at the output layer of the neural network.

We will note with $\mathbf{w}$ the synaptic weights vector of the neural network that is supposed to approximate the regressive model (7). By applying the input vector to the input layer of the neural network, and by propagating it through the output layer we can write the following equation [2]:

$$y = F(\mathbf{x}, \mathbf{w}) \tag{9}$$

Because the training set $T = \{(\mathbf{x}_i, z_i)|i = 1, 2, \ldots, N\}$ contains also the target vectors z provided by the trainer it is clear the equivalence with the supervised learning paradigm. For this reason the modification of the synaptic weights is done using an iterative process, as a response to the error signal (2).

The supervised learning algorithm will optimise the following error function, in respect with the synaptic weights $\mathbf{w}$ of the neural network:

$$\mathbf{E}(\mathbf{w}) = \frac{1}{2}E[e^2] = \frac{1}{2}E[(z - y)^2] = \frac{1}{2}E[(z - F(\mathbf{x}, \mathbf{w}))^2] \tag{10}$$

This mathematical background related to supervised learning would not be complete if we do not outline the fact that a neural network is a universal approximator of any continuos functions [3, 4, 8, 9]. The architecture of such a

neural network is equivalent to a multylayer perceptron having three layers (an input layer, an output layer and a hidden layer). The activation function of the neurons in the hidden layer must be any non-polynomial function and the activation function of the neuron (neurons) in the output layer can be a linear function or an average function of the output values of the hidden neurons [10]. From this point of view a neural network is an approximation scheme that permits us to approximate at any accuracy any continuos function, provided we have an enough number of hidden neurons [1, 5, 8]. The approximation is obtained through the supervised learning process, which is based on an iterative modification of the synaptic weights of the neural network. Presenting repetitively the training set we will be capable to attain a good generalisation power with the neural network.

3. Active supervised learning

In our presentation of the supervised learning we have outlined the equivalence between the statistical regressive model and the supervised learning of a feedforward neural network. The trainer has a passive role of in the learning process being a simple recipient of passive information about the target function (function to be approximated). 7 We want to determine, if we can consider a more active role for the trainer in the learning process, so, instead of giving only the target vector z for a specific input vector $\mathbf{x}$, to try to indicate which input vector should be selected from the training set, in order to improve the learning capabilities of the neural network, which is equivalent to approximate better with the neural network F the target function f. We can consider that for a specific target function f we have some areas where the function is more "difficult" to be learned (approximated) and so the trainer should choose more examples in order to reduce the approximation error.

In conclusion, we can speak about "difficult" and "easy" regions where the target function is approximated. A "difficult" region will be considered a region with a high approximation error and an "easy" region will be considered a region with a low approximation error (close to zero). This definition is not a very rigorous one because we did not establish the limit that delimits "high approximation error" from "low approximation error". We will see in the next pages that these definitions are not so important, because from the learning algorithm that we will consider, "high approximation error" will be considered the maximum error obtained on the regions which constitute the definition domain of the target function f.

It is obvious that our active learning is based on a fundamental assumption: the trainer is allowed to choose his own examples in order to accomplish the task of approximating the target function as well it is possible. In this assumption

the trainer should be capable to decide which are the "difficult" regions to approximate the target function and to pick up a learning sample from that "difficult" region.

In our analyse we will compare passive supervised learning with the active supervised learning, keeping unchanged the other parameters that influence the learning process. In this approach the only difference between passive and active learning consists only in the way the learning examples are chosen. Another goal of our paper will be to develop a general frame for choosing the examples for the approximation of real functions. We will make also some simulations in order to prove the validity of the theoretical results presented here.

We will need to introduce the following terms:

• $\mathcal{F}$ the set of functions defined on set D with values in set Y, where $Y \subseteq \mathbf{R}$.

$$\mathcal{F} = \{f : D \subseteq \mathbf{R}^n \to Y \subseteq \mathbf{R}\} \tag{11}$$

The target function f that should be approximated by an approximation scheme (a neural network) belongs to the set $\mathcal{F}$ of functions.

• The training (learning) set T is composed by pairs of elements:

$$T = \{(\mathbf{x}_i, z_i) | \mathbf{x}_i \in D, z_i = f(\mathbf{x}_i), i = 1, 2, \ldots, N\} \tag{12}$$

• H is an approximation scheme. This means that H does not contain only a set of functions defined on the set D with values in the set Y, but also the algorithm what the trainer is using to choose the approximator function $F \in H$, based on the learning set T. In other words we will denote by H a couple $< H, A >$, where H is a the set of functions from where we will chose the approximator function F, and A is an algorithm which has as input the learning set T, and generates at the output the approximator function $F \in H$.

• d_C will represent a metric that measures how good is the approximation made by the trainer. More precisely, the metrics d_C measures the error on a subset $C \subseteq D$. This metrics will have the following properties:

  • $\forall C_1, C_2 \subset D, C_1 \subset C_2, d_{C_1}(f_1, f_2) \leq d_{C_2}(f_1, f_2)$;

  • $d_D(f_1, f_2)$ is the distance between two functions on the whole definition set D; it represents the basic criterion to measure the quality of approximation.

• C represents a partition of domain D. We will suppose that all the data points from domain D, which will be chosen to approximate the target function f, partition the domain D in a number of disjoint sets $C_i \in C, \bigcup_{i=1}^{N} C_i = D$.

The trainer's main objective can be stated as follows: operating with an approximation scheme H, based on the learning set T, obtain the approximator function $F \in H$ of the target function f.

In the literature the most widely used criteria to measure the performance of the learning algorithms is the PAC criteria (**P**robably **A**pproximately **C**orrect)

[14].

4. Algorithms for active learning

In the previous paragraphs we have introduced the concepts of passive supervised learning. As we have mentioned, in this case of passive supervised learning, the trainer is supposed to choose his training samples according to some probabilistic distribution defined on domain D. If the passive learning process will be successful, the neural network implemented to learn the training samples will correspond to an approximator function F, so that we have to obtain the relation $d_C(F, f) < \varepsilon$ with a probability greater than $1 - \delta$.

As an alternative, in the active learning the training will have the possibility to choose according to some strategy the training examples from the domain D where the target function f is defined. At a certain moment in learning process, the training set will contain some valuable information about the target function f that has to be approximated by the means of the neural network. Particularly, the training set contains information about the "difficult regions" to be learned, where there is a "high" approximation error. Of course, the trainer will choose more examples in this "difficult region", in order to decrease the total approximation error. In conclusion, we have to develop a learning strategy that will be active in the sense that the trainer can decide which will be the next training sample in the learning process.

First, let us establish the mathematical arguments that describe the mechanism of active learning.

Considering the domain D, the trainer can access all the data from the following general training set:

$$T = \{(\mathbf{x}_i, z_i) | \mathbf{x}_i \in D, z_i = f(\mathbf{x}_i), i = 1, 2, \ldots, N\} \tag{13}$$

The approximation schema H (the neural network), after the learning process, will generate an approximator function $F \in H$, using a learning algorithm A that corresponds in the best way to the training set.

We will use also the following notations:

- $C = \{C_1, C_2, \ldots, C_p\} C_i \subset D, i = 1, \ldots, p_N$[2], a partition of domain D;

$$\mathcal{F}_T = \{f \in \mathcal{F} | f(\mathbf{x}_i) = z_i, \forall (\mathbf{x}_i, z_i) \in T\} \tag{14}$$

The functions belonging to the class of functions $\mathcal{F}_T$ are the functions that are passing through the points of the training set T. Evidently, the target function is a member of the set $\mathcal{F}_T$.

[2]The number p of regions in which the domain D is partitioned by N points depends on the specific geometry of domain D. For example, if D is a real interval then $p = N + 1$

We are now capable to define the following error criteria [11]:

$$e_C(H, T, \mathcal{F}) = \sup d_C(F, f), f \in \mathcal{F}_T \tag{15}$$

The meaning of this error is very important. $e_C(H, T, \mathcal{F})$ measures the maximum error of the approximation schema (neural network) on the region C. This error is dependant on the training set and on the class of functions to which the target function belongs. As we can see it does not depend directly on the target function (function to be approximated), but we cannot forget that this dependency is already captured in the training set, so the target function is present in the above equation.

In this moment we have a certain measurement schema of the uncertainty on different regions of the domain D. In other words, we have now the possibility to define what is a "difficult region" for learning. From now on, we will consider a **difficult region for learning** a region C_i that has the biggest error according to equation (16).

In this way, we have a natural approach for active learning:

Choose the next training sample from the difficult region for learning.

Let us suppose we define the procedure that gives us the possibility to choose the next training sample from the most difficult region for learning with $\mathcal{P}$. This procedure can be very simple:

Procedure $\mathcal{P}$: Choose the sample as the gravity centre of region C_i that is the most difficult region for learning.

Of course this procedure can be adapted to the wishes of the trainer and to the particular form of the target function f.

If we are approximating a one dimensional real function, as we have seen before a region is an interval $C_i = [x_i, x_{i+1}]$, then the next training sample will be:

$$x_{new} = \frac{x_i + x_{i+1}}{2} \tag{16}$$

We have now a possible active strategy for supervised learning. Let us suppose that at one moment the trainer has obtained the new training sample $\mathbf{x}_{new} \in D$. The next thing the trainer will want to know will be the value of the target function in this point. This value will belong to the following data set:

$$\mathcal{F}_T(\mathbf{x}) = \{f(\mathbf{x}) | f \in \mathcal{F}_T\} \tag{17}$$

If the requested value is $z \in \mathcal{F}_T(\mathbf{x})$, in this moment the trainer has a new supervised training pair $(\mathbf{x}_{new}, z)$ which can be added to the existing training set

T, obtaining a new training set:

$$T^* = T \cup (\mathbf{x}_{new}, z) \tag{18}$$

The approximation schema H can now reconsider the approximator function F^*, on the basis of the new training set T^*. We have:

$$e_C(H, T^*, \mathcal{F}) = \sup(F^*, f), f \in F_{T^*} \tag{19}$$

In this moment the error $e_C(H, T^*, \mathcal{F})$ represents the highest possible error related to the new training set T^*. When we choose the new training sample $\mathbf{x}_{new} \in D$ we do not know if we have the necessary information about the value of the target function in this point. A possible strategy to avoid this problem is to choose the "worst case", namely the value which is producing the highest error if $\mathbf{x}_{new} \in D$ is the new training sample.

With this approach, the total error on domain D will be:

$$\sup_{z \in \mathcal{F}_T(\mathbf{x})} e_D(H, T^*, \mathcal{F}) = \sup_{z \in \mathcal{F}_T(\mathbf{x})} e_D(H, T \cup \{\mathbf{x}_{new}, z\}, \mathcal{F}) \tag{20}$$

Our intention is to obtain the training sample that minimises the maximal error. In this respect the new training sample should be chosen according to the following formula:

$$x_{new} = arg \min_{x \in D} \sup_{z \in \mathcal{F}_T(\mathbf{x})} e_D(H, T \cup \{\mathbf{x}, z\}, \mathcal{F}) \tag{21}$$

Using this strategy we are now able to define the following algorithm for active supervised learning. This algorithm gives the trainer the necessary tool to choose the optimal training samples that will improve the supervised learning performances of the approximation schema (neural network).

Step 1: $j := 1$. Choose the first training sample $(\mathbf{x}_j, z_j)$ according to procedure $\mathcal{P}$.

Step 2: Based on the new training example, partition domain D in the regions $C_1, C_2 \ldots, C_{p_j}$.

Step 3: Compute the errors e_{C_i}, for every $i = 1, 2, \ldots, p_j$.

Step 4: Suppose at **Step j** domain D is partitioned in the regions $C_1, C_2, \ldots, C_{p_j}$. According to procedure P we will choose in the most difficult region for learning the new training point $\mathbf{x}_{new} \in D$. Let us consider the new training sample $(\mathbf{x}_{j+1}, z_j) := (\mathbf{x}_{new}, z)$.

IF $e_D(H, T, \mathcal{F}) < \varepsilon$ **THEN**
{
    $N := j$;
    exit;
}
ELSE
{
    $j := j + 1$;

GOTO Step 2;

}

An important calculation is made in our algorithm to obtain the error over the entire domain $e_D(H, T, \mathcal{F})$. This error represents a measure of the highest possible error made by the approximation schema (neural network) in order to approximate a target function from $\mathcal{F}$, using the training set T. If we want an independent approximation schema we have to minimise the error $e_D(H, T, \mathcal{F})$ relative to all the possible approximation schemes:

$$\inf_H(H, D, F) \tag{22}$$

5. Experiments, simulations and conclusions

The learning process was carried out in two phases [4, 13]:
- Unsupervised learning phase [3] in order to determine the following unknown parameters: $\mathbf{t_i} \in \mathbf{R}^n$ the centres of the clusters of the input data and σ_i the radius of the clusters.
- Supervised learning phase in order to determine the synaptic weights $\mathbf{k_i} \in \mathbf{R}$.

The supervised learning phase was done using three different types of training:
1. **Random passive** - the training set was generated randomly from the domain D.
2. **Uniform passive** - the training set was generated using a uniform distribution on domain D.
3. **Active** - the training set was determined using the active learning algorithm presented earlier in this paper.

The experiments were made in order to approximate the following target function:

$$f : [0, 1] \to \mathbf{R}, f(x) = \left(x - \frac{1}{3}\right)^3 + \frac{1}{27} \tag{23}$$

The training set generated by one of the three methods was presented repeatedly in epochs of 1000, 5000 and 10.000 times.

The rulers situated in the bottom of each figure represent the distribution of the training points.

One can observe in Figures 3 that correspond to the active supervised learning the way in which the training samples are distributed. The difficult regions for learning are those where the training points have a higher density, and in our case these regions correspond to the regions where the target function has a higher slope. The regions where the target function it is easy to be approximated the trainer needs just a few examples. These are the easy regions for learning, and in our case for these regions correspond a very slow slope.

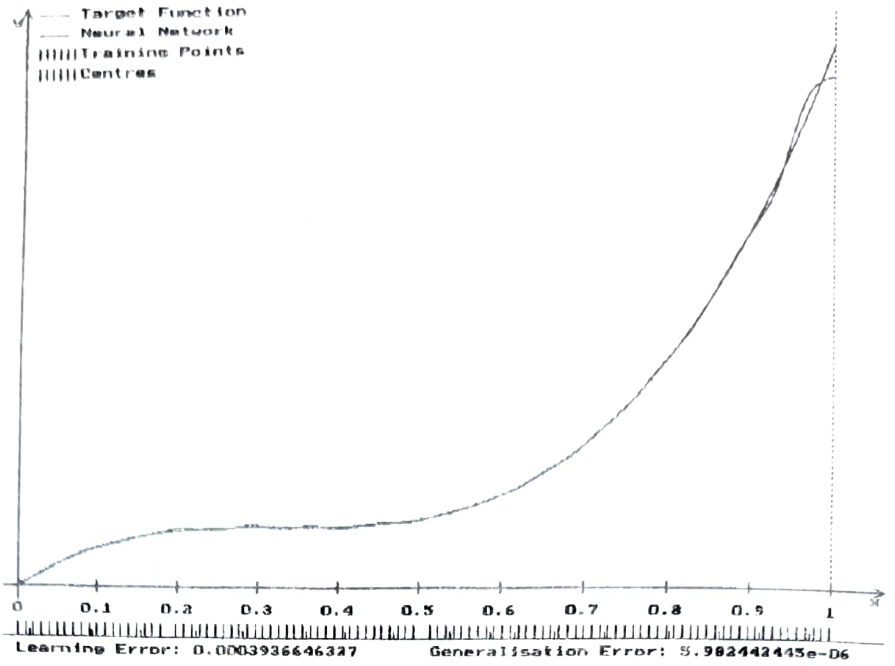

FIGURE 1. Approximation of the target function (24) by an RBF neural network using a random passive supervised learning algorithm: $N = 100$, 10000 epochs, 25 centres

epochs	random passive	uniform passive	active
1.000	$E_l = 0.00111933647$	$E_l = 0.00538671535$	$E_l = 0.005939686434$
	$E_g = 2.00543792e\text{-}5$	$E_g = 9.25674175e\text{-}5$	$E_g = 0.000305306076$
5.000	$E_l = 0.00042799210$	$E_l = 6.77417526e\text{-}6$	$E_l = 8.411126178e\text{-}5$
	$E_g = 6.62271543e\text{-}6$	$E_g = 1.19316687e\text{-}6$	$E_g = 1.167369815e\text{-}5$
10.000	$E_l = 0.00039366463$	$E_l = 5.59375032e\text{-}5$	$E_l = 5.386507373e\text{-}5$
	$E_g = 5.98244244244$	$E_g = 1.01227192e\text{-}5$	$E_g = 5.824087429e\text{-}7$

TABLE 1. Results of the learning process to learn the target function with N = 100 training data samples, 25 centres using random passive, uniform passive and active supervised learning

Analysing the learning performances we will take in consideration not only the learning error E_l but also the generalisation error E_g that correspond to the error generated by the neural networks in points that does not belong to the training set. This generalisation error is the real measure of comparing the performances of different approximation schemes.

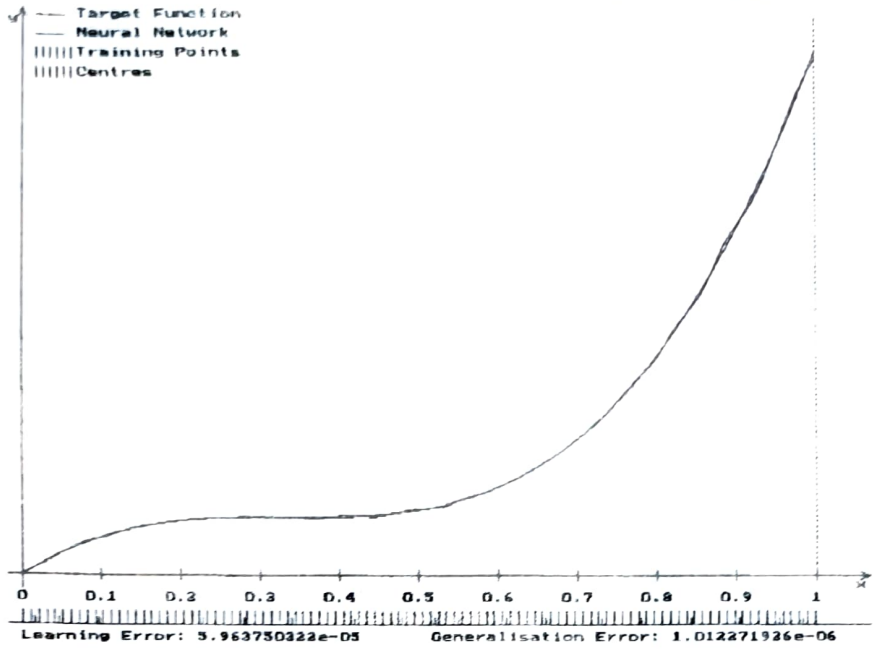

FIGURE 2. Approximation of the target function (24) by an RBF neural network using a uniform passive supervised learning algorithm: $N = 100$, 10000 epochs, 25 centres.

References

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

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

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

[4] Enăchescu, C., (1995) *Learning the Neural Networks from the Approximation Theory Perspective. Intelligent Computer Communication ICC'95 Proceedings*, 184-187, Technical University of Cluj-Napoca, Romania.

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

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

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

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

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

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

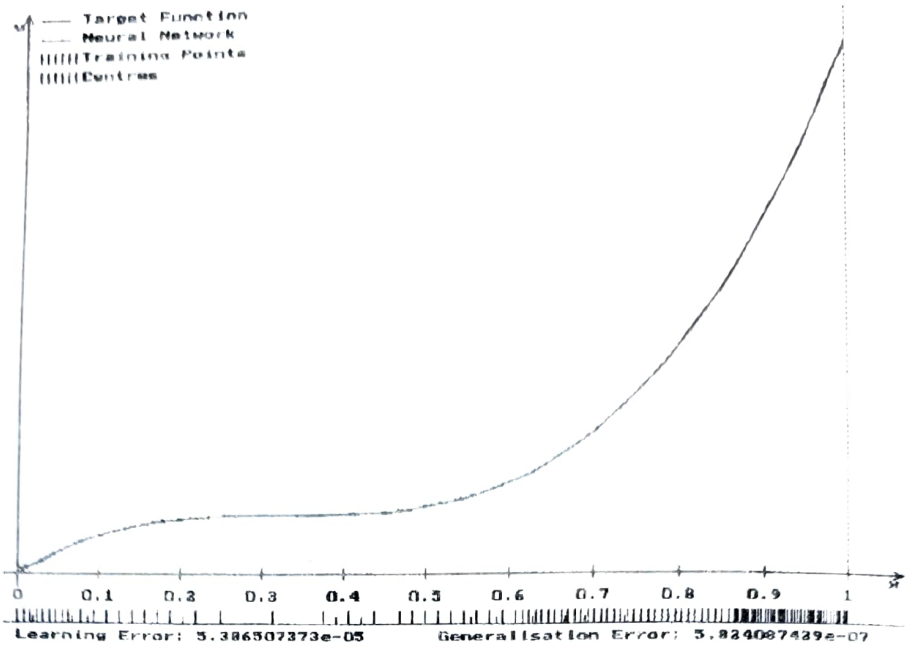

FIGURE 3. Approximation of the target function (24) by an RBF neural network using an active supervised learning algorithm: $N = 100$, 10000 epochs, 25 centres.

[11] Niyogi, P.(1995), Active Learning by Sequential Optimal Recovery. *A.I. Memo No. 1514*, C.B.C.L. Paper No. **113**, M.I.T, Massachusetts.

[12] Petri, A. J. (1991). A Nonlinear Network Model for continuos learning. *Neurocomputing* 3, 157-176.

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

[14] Valiant, L.G (1984), A theory of learnable. *Communication of ACM* **27** (11), 1134-1142.

"PETRU MAIOR" UNIVERSITY OF TÂRGU-MUREŞ, ROMANIA
E-mail address: ecalin@uttgm.ro