NEURAL NETWORKS AND THE APPROXIMATION THEORY

Călin ENĂCHESCU, Professor Ph.D.,
"Petru Maior" University of Tîrgu Mureş

Abstract: 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 of the neural networks 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. The main feature of the neural computing is the learning capability. Learning from examples means being able to infer the functional dependence between input and output spaces X and Z, given the knowledge of the set of examples T. It means that, after we have "learned" N examples, when a new input variable x is presented, we need to be able to estimate, according to some criterion that we will specify, a corresponding value of z. From this point of view learning is equivalent to a function approximation.

Keywords: Neural networks, approximation, learning, RBF neural networks

Introduction

In several papers [5], [6] we have analyzed the learning capabilities of neural networks, arguing that the task of learning from examples can be considered in many cases to be equivalent to multivariate function approximation, that is, to the problem of approximating a smooth function from sparse data, called training set.

Our approach is based on the recognition that the ill-posed problem of function approximation from sparse data must be constrained by assuming an appropriate prior on the class of approximating functions.

In this context, we have to be more precise in establishing the meaning of "ill-posed problem of function approximation from sparse data". We will use the definition given in [20]:

Definition 1: The problem of approximating a function f: X $\rightarrow$ Y is well-posed if the following conditions are satisfied:

(C1) Existence: For any $x \in$ X, there exists $y \in$ Y with $y = f(x)$;

(C2) Uniqueness: For any $x, y \in$ X, $f(x) = f(y)$ if and only if $x = y$;

(C3) Continuity: function f is continuos.

Definition 2: The problem of approximating a function f: X $\rightarrow$ Y is ill-posed if the one of the conditions (C1), (C2), (C3) is not satisfied.

From this point of view, the learning process of a neural network, or the approximation of a smooth function using a set of examples is ill-posed. Usually the training set doesn't contain enough information; therefore the condition of uniqueness (C2) is not satisfied. In many cases, the training set contains noisy data that implies that the condition of continuity (C3) set is not satisfied.

In order to transform the ill-posed problem of approximating a function from sparse examples into a well-posed problem, we need to take into consideration some a priori hypothesis about the function to be approximated. What is the weakest a priori hypothesis that can be considered without affecting the general frame of function approximation? The learning process (function approximation) is efficient if we obtain good generalization properties. But the generalization properties are a result of a certain level of redundancy, more precisely we can say that generalization properties are a result of the property that small changes of the input parameters results in small changes of the output parameters. This property is usually called *smoothness*.

Concluding, we can say that the learning process of a neural network is equivalent to the approximation of a smooth function from examples (the training set) [12].

Neural networks as a solution of regularization techniques

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 [20] [21], identifies the approximating function as the minimizer 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 stabilizer.

In the Bayesian interpretation [8] 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 $T = \{(x_i, z_i) | i = 1,2,\ldots,N\}$ has been obtained by random sampling of a function *f*, belonging to some space of functions X defined on $\mathbf{R}^n$, 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 contained in *T*. This problem is clearly ill-posed, 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, 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.

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 [18]:

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

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 (1), the solutions of the minimization of the functional (1) all have the same form [13].

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) [11].

The function that minimizes the functional (1) has the following form [18]:

$$f(\mathrm{x}) = \sum_i w_i G(\mathrm{x}; \mathrm{x}_i) + p(\mathrm{x}) \tag{2}$$

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

We make the following notation: $\{\psi_\alpha\}_{\alpha=1}^{k}$ 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:

$$f(\mathrm{x}) = \sum_{i=1}^{N} w_i G(\mathrm{x}; \mathrm{x}_i) + \sum_{\alpha=1}^{k} d_\alpha \psi_\alpha(\mathrm{x}) \tag{3}$$

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 (1), we can consider the following practical solution:

$$f(\mathrm{x}) \approx \sum_i w_i G(\mathrm{x}; \mathrm{x}_i) \tag{4}$$

The solution (4) of the variational problem (1) has a simple interpretation in terms of a neural network with one layer of hidden units, of Multylayer Perceptron type [10], represented in Figure 1. Let's analyze the architecture of the neural network that corresponds to the solution function (4):

1. *The architecture of the neural network corresponds to a neural network of the Multylayer Perceptron type with one hidden layer:*
   - The input layer contains n input neurons, n representing the dimensionality of the input space $\mathbf{x}_i = \left(x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)}\right) \in \mathbf{R}^n$.
   - The hidden layer having a number of hidden neurons equal to the dimension of the training set $T = \left\{\left(\mathbf{x}_i, f(\mathbf{x}_i)\right) \middle| i = 1,2,\ldots,N\right\}$. The activation functions of the hidden neurons are the Green functions $G(\mathrm{x} - \mathrm{x}_k)$ [18]. The dimension of the hidden layer can be reduced using an unsupervised clustering algorithm [8];

- 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 [2];

2. *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 (4);
   - The vector $\mathrm{w} = (w_1, w_2, \ldots, w_N)$ represents the weights between the hidden layer and the output layer.

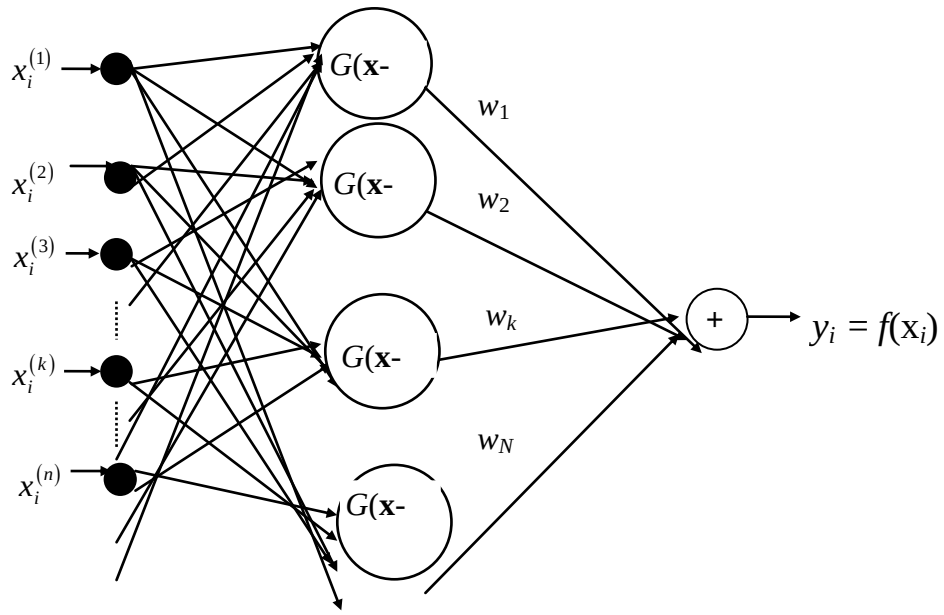

***Fig. 1*: Architecture of a neural network that corresponds to the solution function (4).**

Radial Basis Functions and radial stabilizers

In this section we want to obtain the classes of well known RBF - Radial Basis Function as a particular case of the general solution (3). In order to achieve this goal we have to make an a priori presumption about the smoothness of the function that must be approximated.

Most of the commonly used stabilizers have radial symmetry [3], that is, they satisfy the following equation:

$$\Phi\big[f(\mathbf{x})\big] = \Phi\big[f(R\mathbf{x})\big] \qquad (5)$$

for any rotation matrix R.

This choice reflects the a priori assumption that all the variables have the same relevance, and that there are no privileged directions. Rotation invariant stabilizers correspond clearly to radial basis function $G\big(\|\mathrm{x}\|\big)$ [18]. Much attention has been dedicated to this case, and the corresponding approximation technique is known as Radial Basis Functions [16], [17].

The class of admissible Radial Basis Functions is the class of conditionally positive definite functions of any order, since it has been shown [4], [15] that in this case the functional of equation (2) is a semi-norm, and the associated variational problem is well defined.

All the Radial Basis Functions can therefore be derived in this framework [19]. We will consider explicitly the following example.

Example 1: Gaussian

We will consider a stabilizer of the form [7]:

$$\Phi[f] = \int_{\mathbf{R}^n} d\mathbf{s}\, e^{\frac{\|\mathbf{s}\|^2}{\beta}} \left|\tilde{f}(\mathbf{s})\right|^2 \tag{6}$$

where β is a fixed positive parameter. We obtain as result of the variational problem the following function:

$$\tilde{G}(\mathbf{s}) = e^{-\frac{\|\mathbf{s}\|^2}{\beta}} \tag{7}$$

The Gaussian function is positive definite, and it is well known from the theory of reproducing kernels that positive definite functions can be used to define norms of the type (2). Since $\Phi[f]$ is a norm, its null space contains only the zero element, and the additional null space terms of equation (3) are not needed.

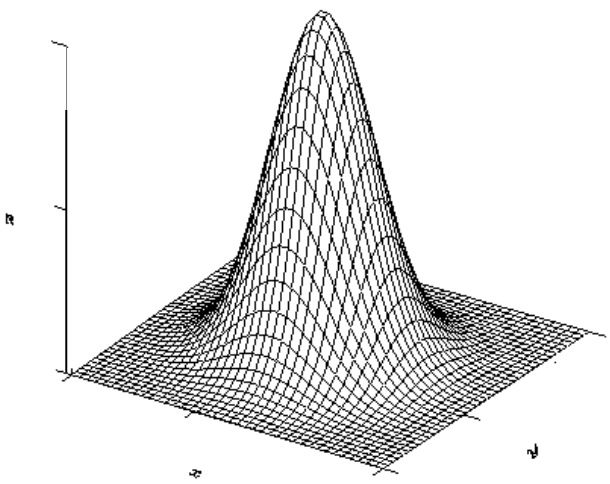

Fig. 2.: **Graphic of RBF** $G(\mathrm{x}) = e^{-\|\mathrm{x}\|^2}$.

Here we give a list of other functions that can be used as basis functions in the Radial Basis Functions technique, and that are therefore associated with the minimization of some functional.

$$G(\mathbf{x}) = \sqrt{\|\mathbf{x}\|^2 + c^2} \text{ - multi-quadratic function} \tag{8}$$

$$G(\mathbf{x}) = \frac{1}{\sqrt{\|\mathbf{x}\|^2 + c^2}} \text{ - multi-quadratic inverse function} \tag{9}$$

$$G(\mathbf{x}) = \|\mathbf{x}\|^{2n} \ln\|\mathbf{x}\| \text{ - spline} \quad (10)$$

$$G(\mathbf{x}) = \|\mathbf{x}\|^{2n+1} \text{ - spline} \quad (11)$$

$$G(\mathbf{x}) = e^{-\|\mathbf{x}\|_{L_1}} \text{ - Gaussian} \quad (12)$$

RBF Neural Networks Topology

Following the previously presented results RBF is a particular type of feed-forward neural network of the one presented in Figure 1, with an input layer (made up of source nodes: sensory units), a single hidden layer and an output layer [9].

The specific architecture of the RBF neural network is presented in Figure 3 [4].

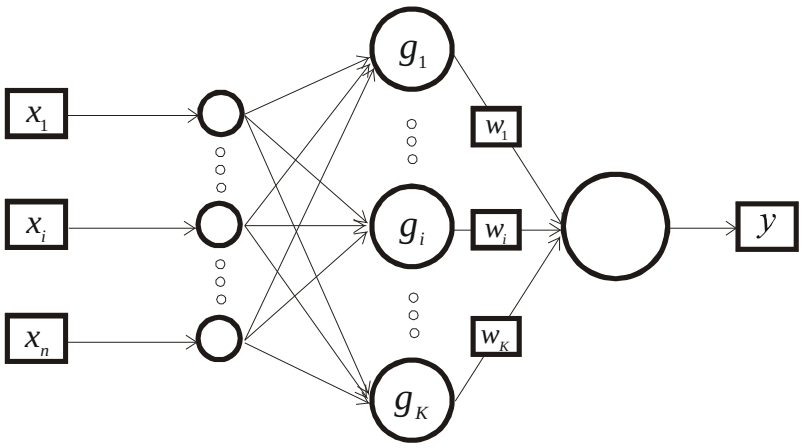

***Fig. 3:* RBF neural network topology.**

If $\mathbf{x} = (x_1,...,x_n)$ is the input vector, $g()$ is the Radial Basis Function and $\mathbf{c}_i$ is the centre parameter for the function corresponding to neuron *i*, then the output created by the network will be:

$$y = \sum_{i=1}^{K} w_i g_i(\mathbf{x}) = \sum_{i=1}^{K} w_i g\left(\|\mathbf{x} - \mathbf{c}_i\|\right) \quad (14)$$

Generally, the Gaussian function (12) is used:

$$g_i(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2\sigma_i^2}} \quad (15)$$

where σ_i is the scale parameter for the function corresponding to neuron *i*.

There are some methods to select the parameters ($\mathbf{c}_i$, σ_i) of the activation function. If few training points are present, then all of them could be used as centre parameter. In this case the number of the processor units in the hidden layer is equal with the number of training points. If the number of training points is high, a single neuron for a group of similar training

points can be considered. These groups of similar training points can be identified using clustering methods [6].

Leaning Strategies for RBF Neural Networks

The RBF Neural Network may be trained with a supervised learning algorithm. A descendent gradient-based algorithm can be considered [14]. The aim is to calculate the synaptic weights w_i, $i = 1,2,\ldots,K$ of the network.

Let $T = \{(\mathbf{x}_i, z_i) | \mathbf{x}_i \in \mathbf{R}^n, z_i \in \mathbf{R}, i = 1,2,\ldots,N\}$ be the set of the training samples. A clustering algorithm can be is used on the points of the set T. The cluster centres c_i, $i = 1,...,K$ are considered (in this way the number of the neurons in the hidden layer is K).

Parameters $\sigma_i \in \mathbf{R}$, $i = 1,...,K$ can be determined corresponding to the diameter of clusters. This step is not executed when K is equal with N ($K = N$), because in this case $\mathbf{c}_i = \mathbf{x}_i$, $i = 1,...,N$ (every training point is a cluster centre too and the value of the width parameters is $\sigma_i = 1/N$).

If the Gaussian function is used as activation function, then at the l^{th} step the global learning error is

$$E_l = \frac{1}{N} \cdot \sum_{i=1}^{N} (z_i - y_i)^2 \tag{15}$$

where

$$y_i = \sum_{j=1}^{K} w_j \cdot e^{-\frac{(\mathbf{x}_i - c_j)^2}{2\sigma_j^2}}, \; i = 1,...,N \tag{16}$$

Using the gradient descendent rule, we have:

$$\Delta w_i = -\eta \cdot \frac{\partial E}{\partial w_i}, \; i = 1,...,K \tag{17}$$

where η is the *learning rate* and E is the *global learning error*.

Weights updating is based on the following correction rule:

$$w_i = w_i + \Delta w_i, \; i = 1,...,K \tag{18}$$

When the learning process is finished, M points, which are not from the training set T, are randomly generated. The corresponding *generalization error* is defined by the expression [5]:

$$E_g = \frac{1}{M} \cdot \sum_{i=1}^{M} (z_i - y_i)^2 \tag{19}$$

Approximation and Interpolation With RBF Neural Networks

The interpolation problem, in its strict sense, may be stated as follows: given a set of N different points $\{x_i \in \mathbf{R}^p \mid i = 1,...,N\}$ and a corresponding set of N real numbers $\{d_i \in \mathbf{R} \mid i = 1,...,N\}$, find a function $F : \mathbf{R}^p \to \mathbf{R}$ that satisfies the interpolation condition [2], [6]:

$$F(x_i) = d_i, i = 1,...,N \tag{20}$$

The RBF technique consists of choosing a function F that has the following form [10]:

$$F(x) = \sum_{i=1}^{N} w_i g(\|x - x_i\|) \tag{21}$$

where

$$\{g(\|x - x_i\|) \mid i = 1,...,N\} \tag{22}$$

is a set of N arbitrary radial basis functions. The known data points $x_i \in \mathbf{R}^p, i = 1,...,N$ are taken to be the centres of the radial basis functions.

A RBF network is considered, with a single processor unit in the output layer, and N processor units in the hidden layer, where $\{g(\|x - x_i\|) \mid i = 1,...,N\}$ is the set of the activation functions for the hidden processor units. The interpolation problem is reduced to the determination of weights (learning process) [3].

From this point of view, the neural network represents a map from the p-dimensional input space to the single-dimensional output space, written as:

$$s : \mathbf{R}^p \to \mathbf{R} \tag{23}$$

The map s could be considered as a hyper-surface $\Gamma \subset \mathbf{R}^{p+1}$. The surface Γ is a multidimensional plot of the output as a function of the input. In a practical situation, the surface Γ is unknown and the training data are usually affected by noise. Accordingly, the training phase and generalization phase of the learning process may be respectively viewed as follows [1], [4]:

- The *training phase* constitutes the optimization of a fitting procedure for the surface Γ, based on known data points presented to the network in the form of input-output examples.
- The *generalization phase* is synonymous with the interpolation between the data points, with the interpolation being performed along the constrained surface generated by the fitting procedure as the optimal approximation to the true surface Γ.

Conclusions

The paper presents a general model of the application of neural networks in the domain of function approximation and interpolation, viewed as a learning process. The synthesis is based on two theoretical aspects, namely, the approximation properties of the

neural networks and the regularization theory. The results presented in the paper confirm the fundamental role of the theoretical models for the application of neural networks for function approximation and interpolation.

We have investigated the neural networks based approximation methods. The approximation of a function is equivalent with the problem of learning for neural networks. 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. Our aim was to improve the learning performances of neural networks, using some additional information from the training data set.

References

[1] Broomhead D.S., Lowe D., (1988), Multivariable functional interpolation and adaptive networks, Complex Systems 2, 321-355.

[2] Bugmann, G., *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.

[3] Duchon, J., *Spline minimising rotation-invariant semi-norms in Sobolev spaces*. In Zeller, K., editors. Constructive Theory of functions of several variables, Lecture Notes in Mathematics, 571. Springer-Verlag, Berlin, 1977.

[4] Dyn, N., Interpolation and approximation by radial and related functions. In: Chui, C.K., Schumaker, L.L., Ward, D.J., editors, Approximation Theory, VI, 211-234, Academic Press, New-York, 1991.

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

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

[7] 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.

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

[9] 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.

[10] Enăchescu, C. (1998), The Theoretical fundamentals of Neural Computing, Casa Cărţii de Ştiinţă, Cluj-Napoca. (in Romanian).

[11] Girosi, F., Jones, M., Poggio, T., *Priors, Stabilisers and Basis Functions: from reguralization to radial, tensor and additive splines*. M.I.T, A.I. Memo No. 1430, 1993.

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

[13] Haykin, S., *Neural Networks. A Comprehensive Foundation*. IEEE Press, MacMillian, 1994.
[14] Hertz, J., Krogh, A., Palmer, R.G. *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Co., 1992.
[15] Madich, W.R., Nelson, S.A., *Multivariate interpolation and conditionally positive definite functions*. II. Mathematics of Computations, 54 (189): 211-230, 1990.
[16] Micchelli, C.A., *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*. Constr. Approx., Vol. 2, 11-22, 1986.
[17] Moody J., Darken C., (1989), Fast learning in networks of locally tuned processing units, Neural Computation, 1, 281-294.
[18] Poggio, T., Girosi, F., *Networks for Approximation and Learning*. Proceedings of the IEEE, Vol. 78, No. 9, Sept. 1990.
[19] Powell M.J.D., (1988), Radial basis function approximations to polynomials, Numerical Analysis 1987 Proceedings, 233-241.
[20] Tichonov, A.N., Arsenin, V.A., *Solutions of Ill-posed Problems*. Washington, DC: W.H. Winston, 1977.
[21] Tikhonov, A.N., *Solution of incorrectly formulated problems and regularization method*. Soviet Math. Dokl., Vol. 4, 1035-1038, 1963.