Vol. **128** (2015) *ACTA PHYSICA POLONICA A* No. 2-B

Special issue of the International Conference on Computational and Experimental Science and Engineering (ICCESEN 2014)

Comparison of Second Order Algorithms for Function Approximation with Neural Networks

E. Boutalbi[a,b,*], L. Ait Gougam[b] and F. Mekideche-Chafa[b]

[a]Laboratory of Particle Physics and Statistical Physics, Ecole Normale, Supérieure BP 92 Vieux Kouba, Algeria

[b]Laboratory of Theoretical Physics, Faculty of Sciences-Physics, USTHB, B.P. 32, El-Alia, Algeria

The Neural networks are massively parallel, distributed processing systems representing a new computational technology built on the analogy to the human information processing system. They are usually considered as naturally parallel computing models. The combination of wavelets with neural networks can hopefully remedy each other's weaknesses, resulting in wavelet based neural network capable of approximating any function with arbitrary precision. A wavelet based neural network is a nonlinear regression structure that represents nonlinear mappings as the superposition of dilated and translated versions of a function, which is found both in the space and frequency domains. The desired task is usually obtained by a learning procedure which consists in adjusting the "synaptic weights". For this purpose, many learning algorithms have been proposed to update these weights. The convergence for these learning algorithms is a crucial criterion for neural networks to be useful in different applications. In this paper, we use different training algorithms for feed forward wavelet networks used for function approximation. The training is based on the minimization of the least-square cost function. The minimization is performed by iterative first and second order gradient-based methods. We make use of the Levenberg-Marquardt algorithm to train the architecture of the chosen network and, then, the training procedure starts with a simple gradient method which is followed by a BFGS (Broyden, Fletcher, Glodfarb et Shanno) algorithm. The conjugate gradient method is then used. The performances of the different algorithms are then compared. It is found that the advantage of the last training algorithm, namely, conjugate gradient method, over many of the other optimization algorithms is its relative simplicity, efficiency and quick convergence.

DOI: 10.12693/APhysPolA.128.B-271

PACS: 02.30.Mv, 07.05.Mh

1. Introduction

Artificial neural network is based on using computer network system to simulate the intelligent computing of biological neural networks, to process the information by means of simulating brain network processing, memory information processing through the nonlinear, self-adapted information processing system that is composed by a large number of processing units linked together [1]. Function approximation from a set of input-output pairs has numerous scientific and engineering applications. Multilayer feed forward neural networks have been proposed as a tool for nonlinear function approximation [2, 3]. Parametric models represented by such networks are highly nonlinear. Inspired by the biological nervous system, ANN operates on the principle of largely interconnected simple elements operating as a network function. In doing so, no previous knowledge is assumed, but data, records, measurements, observations are considered. ANN research stands on the fact of learning from data to mimic the biological capability of linear and nonlinear problem solving. An important theoretical issue for a class of neural networks is its approximation capability. There have been many papers related to this topic.

In this paper, we use different training algorithms for feed forward wavelet networks used for function approximation. The training is based on the minimization of the least-square cost function. The minimization is performed by iterative first and second order gradient-based methods. We make use of the Levenberg-Marquardt algorithm to train the architecture of the chosen network and, then, the training procedure starts with a simple gradient method which is followed by a BFGS (Broyden, Fletcher, Glodfarb et Shanno) algorithm. The conjugate gradient method is then used. The performances of the different algorithms are then compared.

2. Network architecture and training

Neural networks are a convenient means of representation because they are known to be universal approximators that can learn data [4]. The desired task is usually obtained by a learning procedure which consists in adjusting the "synaptic weights". For this purpose, many learning algorithms have been proposed to update these weights. The convergence for these learning algorithms is a crucial criterion for neural networks to be useful in different applications. In fact, considerable effort has gone into developing techniques for accelerating the convergence of the training algorithms.

In the present contribution, we use wavelets stemming from the continuous wavelet transform. This approach uses a set of formal neurons (wavelets) with various translation parameters b and scale ones λ, hence a space of elementary responses translated by b and scaled with λ. The function to be approximated is then expanded in this set [5]. The integral form of the expansion is then reduced

*corresponding author; e-mail: emboutalbi@yafoo.fr to a discrete sum where the development coefficients are the output synaptic weights and are the unknowns of the problem.

3. Numerical implementation

We now proceed with the presentation of our numerical results with a target task given by

$$F(x) = \sin(x) + (1/x^2 + 1)\exp(-x^2). \quad (1)$$

Let us consider a three-layered network consisting of an input X, a hidden layer of elementary responses and a linear output neuron. The neural elementary units receive the same input X. Each unit returns an output f which depends on two parameters: the translation parameter b and the scale one λ. Output synaptic weights $\omega(b,\lambda)$ linearly regroup these elementary outputs into a global output $F(X)$. We have then the expansion [6]

$$F(X) = \int \mathrm{d}b\,\mathrm{d}\lambda\,\omega(b,\lambda) f((X-b)/\lambda). \quad (2)$$

Hence, discretizing Eq. (2) with N units and neglecting the translation parameter, we obtain the following approximation function

$$F_{\mathrm{app}}(X) = \sum_{i=1}^{N} \omega(\lambda_i) f((X)/\lambda_i). \quad (3)$$

3.1. Model and architecture

To define the "best" $\mathrm{F_{app}}$, one has to minimize the square norm of the error

$$\varepsilon = F - F_{\mathrm{app}}. \quad (4)$$

First, in terms of the output weights ω_i and second, of the scale parameters. In terms of the ω_i, this consists in solving the equation

$$\frac{\partial}{\partial \omega_i}\left(\left\langle F - F_{\mathrm{app}} \middle| F - F_{\mathrm{app}}\right\rangle\right) = 0. \quad (5)$$

Using Eq. (3) and adopting short notation, we obtain

$$\frac{\partial}{\partial \omega_i}\left(\langle F|F\rangle - 2\sum_{j=1}^{N}\omega_j \langle f_j|F\rangle + \sum_{j,k=1}^{N}\omega_j \langle f_j|f_k\rangle \omega_k\right)$$
$$= 0, \quad i = 1,\ldots,N \quad (6)$$

the solution of which is given by

$$\omega_i = \sum_{j=1}^{N} (\hat{g}^{-1})_{ij} \langle f_j|F\rangle, \quad i = 1,\ldots,N, \quad (7)$$

where $\hat{g}$ is the matrix with elements

$$g_{jk} = \langle f_j|f_k\rangle. \quad (8)$$

Let now minimize the mean square norm of the error (MSE) in terms of scale parameters. We have thus

$$\frac{\partial}{\partial \lambda_i}\left(\left\langle F - F_{\mathrm{app}} \middle| F - F_{\mathrm{app}}\right\rangle\right) = 0$$
$$\frac{\partial \varepsilon}{\partial \lambda_i} = \frac{2\omega_i}{\lambda_j^2}\left\langle X f'(X/\lambda_j) \middle| F - F_{\mathrm{app}}\right\rangle = 0,$$
$$j = 1,\ldots,N. \quad (9)$$

We make use, first, of the Levenberg-Marquardt algorithm to train the architecture of the chosen network and, then the conjugate gradient method. Numerical results are presented in Figs. 1 and 2.

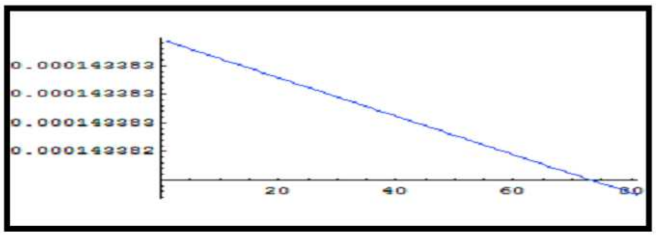

Fig. 1. Plot of the mean square norm of the error (MSE) versus the number of iterations, with $N = 8$, using gradient conjugate algorithm.

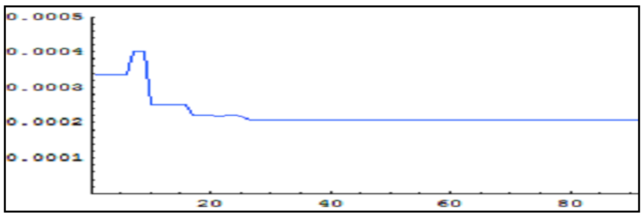

Fig. 2. Plot of the mean square norm of the error (MSE) versus the number of iterations, with $N = 8$, using the Levenberg-Marquardt algorithm.

4. Conclusion

Neural networks have been recently seen as attractive tools for developing efficient solutions. We have investigated some properties of neural nets made of "artificial neurons" used for function approximation. Attention has been devoted to the role of training algorithms in searching the best approximation. In fact, by varying the dimension N of the elementary task basis, we noticed that the latter may affect the performance of the neural network as well as the best approximation. As the number of neurons increases, the MSE decreases and performs damped oscillations, thereby showing a net improvement of the function approximation.

On the other hand, It is found that last training algorithm, namely, conjugate gradient method, is the more appropriate algorithm. Its advantage over many of the other optimization algorithms is its relative simplicity, efficiency and quick convergence.

References

[1] J.J. Hopfield, D.W. Tank, *Science* **233**, 625 (1986).

[2] Q. Zhang, A. Benveniste, *IEEE Trans. Neural Networks* **3**, 889 (1992).

[3] J. Zhang, G.G. Walter, Y. Miao, W.N.W. Lee, *IEEE Trans. Signal Processing* **43**, 1485 (1995).

[4] K. Hornik, M. Stinchcombe, H. White, *Neural Networks* **2**, 359 (1989).

[5] L. Ait Gougam, M. Tribeche, F. Mekideche, *Neural Networks* **21**, 1311 (2008).

[6] B. Giraud, A. Touzeau, *Phys. Rev. E* **65**, 016109 (2001).