DRO
Deakin University's Research Repository
DEAKIN UNIVERSITY AUSTRALIA Worldly

This is the published version:

Nguyen, Thi T. 2010, Smoothing supervised learning of neural networks for function approximation, *in KSE 2010 : Proceedings of the Knowledge and Systems Engineering 2010 international conference*, IEEE, Piscataway, N.J., pp. 104-109.

Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30056337

©2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Copyright: 2010, IEEE

dro.deakin.edu.au
Deakin University CRICOS Provider Code: 00113B

Smoothing Supervised Learning of Neural Networks for Function Approximation

Thi T. Nguyen
Department of Econometrics and Business Statistics
Monash University, Australia
thi.nguyen@buseco.monash.edu.au

Abstract

Two popular hazards in supervised learning of neural networks are local minima and overfitting. Application of the momentum technique dealing with the local optima has proved efficient but it is vulnerable to overfitting. In contrast, deployment of the early stopping technique might overcome the overfitting phenomena but it sometimes terminates into the local minima. This paper proposes a hybrid approach, which is a combination of two processing neurons: momentum and early stopping, to tackle these hazards, aiming at improving the performance of neural networks in terms of both accuracy and processing time in function approximation. Experimental results conducted on various kinds of non-linear functions have demonstrated that the proposed approach is dominant compared with conventional learning approaches.

1. Introduction

A vast number of applications for artificial neural networks have been found in various fields [1], including pattern recognition and regression (also known as function approximation), identification, classification, speech, vision, and control systems. Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling is a striking application of neural networks. To be capable in performing complex functions, a neural network needs to be trained using the supervised learning technique. It is however there are two major hazards that occur during neural network training called overfitting and local minima.

The overfitting happens when the errors calculated on the training data are very small values, but the errors are large for new data presented to the network. In other words, the network has not learned to generalize to new situations though it has memorized the training data. To deal with this problem, one possible solution is to use a simple network which will not have enough power to overfit the data. For a specific application, it is however not easy to determine how complex a network should be. Thus there are two widely used ways for improving generalization of the network: regularization and early stopping [2].

The operation of the gradient descent algorithm [3] is analogous to dropping a ball on the error surface, and letting it roll down till it is fixed at the lowest point (global minimum). However, if the error surface is not flat, having many valleys, then because the ball is dropped randomly it is only able to go to the nearest valley. The valley that the ball traps into may not be the globally lowest point, and is called a local minimum. Generally, local minima are the points on the error surface where small adjustment of weights in any direction cannot decrease errors. One of the popular techniques to escape the local minima is the momentum method [4]. With momentum, which acts like a lowpass filter, a network can ignore small features in the error surface.

Recent researchers have shown many efforts in overcoming overfitting and local minima. Caruana *et al.* utilized excess capacity backpropagation networks, which conventionally generalize poorly, with the early stopping to enhance generalization capability of networks [5]. A new perspective in dealing with overfitting from the geometrical interpretation of multilayer perceptron has been suggested by Ding and Xiang [6]. A repeated two-phase model (back-propagation and gradient ascent) has been proposed by Tang *et al.* to escape from the local minima [7]. Li and Xu divided the process of training into a certain number of phases using phase evaluation backpropagation [8].

Once the above approaches have dealt with overfitting and local minima separately, an overall and simultaneous solution for both these hazards is still a must. This paper proposes a combination between two prevalent techniques: early stopping and momentum for smoothing the learning process with the aim of increasing the performance of neural networks in function approximation. The approach is tested with the feedforward neural network using the backpropagation training.

978-0-7695-4213-3/10 $25.00 © 2010 IEEE
DOI 10.1109/KSE.2010.15

IEEE computer society

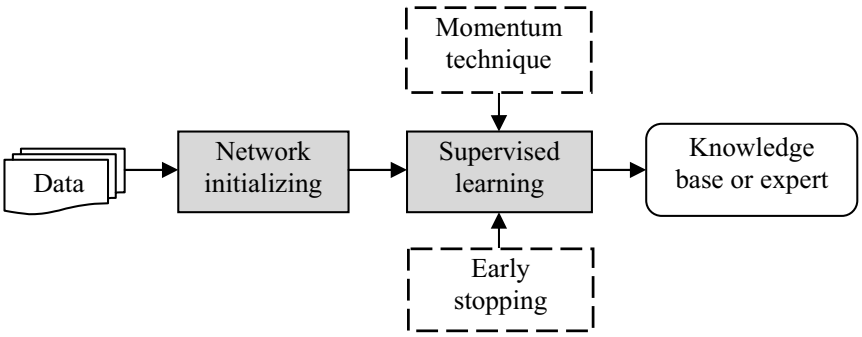


Figure 1. Integration of the momentum and early stopping techniques

2. Feedforward neural networks and hybrid supervised learning

2.1 Feedforward neural networks background

The feedforward neural network [9] was the simplest type of artificial neural network. With this network, the information moves forwardly from the input layer through the hidden layers and to the output layer without cycles or loops. The number of hidden layers is varied depending on the complexity of the problem being solved.

2.2 Hybrid supervised learning

The proposed approach is outlined in Figure 1 that shows the simultaneous integration of the momentum and early stopping techniques to the conventional supervised learning of neural networks.

2.2.1 Conventional supervised learning

The backpropagation learning approach was utilized to tune the parameters relying upon the aim of minimizing the squared error function:

$$e(x) = \frac{1}{2}\left[f(x) - F(x)\right]^2$$

where $f(x)$ is the observed (real) value and $F(x)$ is the value computed from the ANNs and is the value at the output layer. The parameters ξ are updated using the Levenberg-Marquardt algorithm [2] based on the Jacobian matrix that contains first derivatives of the network errors:

$$\xi(t+1) = \xi(t) - \left[J^T J + \mu I\right]^{-1} J^T e$$

with J as the Jacobian matrix, I as the identity matrix and μ as the learning rate.

2.2.2 Momentum technique

The momentum technique is integrated in the parameter tuning process to increase the speed of convergence by simply adding a fraction of the previous weight update to the current one. The learning formula with momentum is as follows:

$$\xi(t+1) = \xi(t) - \left[J^T J + \mu I\right]^{-1} J^T e + \varepsilon.\Delta\xi(t)$$

where $0 < \varepsilon < 1$ is the momentum coefficient.

2.2.3 Early stopping

The training data are divided into two subsets. The first subset is used for updating the network weights whereas the second subset is for validating the occurrence of overfitting. Both training and validation errors normally decrease during the initial phase of training. However, the validation error will increase when the network begins to overfit the data (Figure 2).

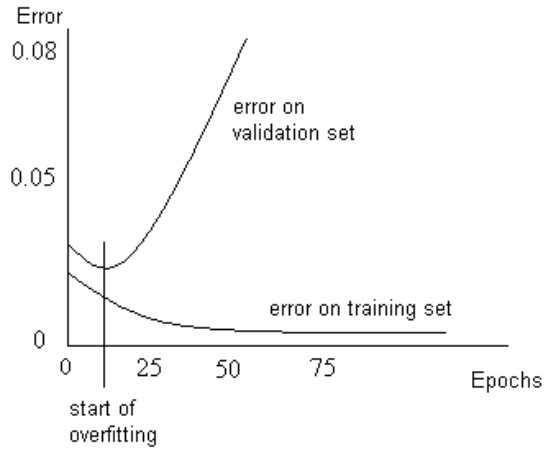


Figure 2. Collating training and validation errors for early stopping

The training is stopped when the validation error increases for a specified number of iterations, and the weights and biases at the minimum of the validation error are returned [2].

3. Experiments and results

The experiments are performed on a variety of function types: 1-D (Dimension), 2-D and 3-D, and the

results are assessed in terms of the mean squared error (MSE) of the function approximation. Below are three sample test functions used as approximands. The variables x, y, z are all investigated in [-1, 1].

$$f(x) = 10\left(e^{-5|x|} + e^{-3|x-0.8|/10} + e^{-10|x+0.6|}\right)$$

$$g(x,y) = 8\sin\left(10x^2 + 5x + 1\right)\times$$

$$\times 2\left(e^{-\left(\frac{y-0.1}{0.25}\right)^2} - 0.8e^{-\left(\frac{y+0.75}{0.15}\right)^2} - 0.4e^{-\left(\frac{y-0.8}{0.1}\right)^2}\right)$$

$$h(x,y,z) = 0.1\left(e^{-\frac{|x|}{0.2}} + e^{-\frac{|x-0.8|}{0.3}} + e^{-\frac{|x+0.6|}{0.1}}\right)\times$$

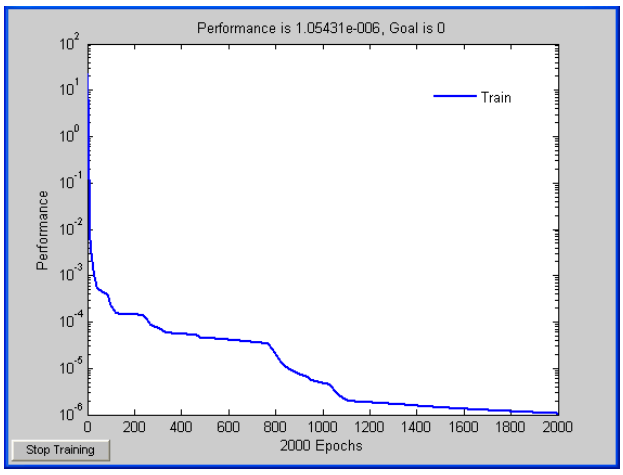
$$\times\left(\tan^3(1.5y) + 10\tan^2(y) - 20\tan(0.7y)\right)\times$$

$$\times\left(\arccos^3(z) - \arccos^2(-z) - \arccos(-z)\right)$$

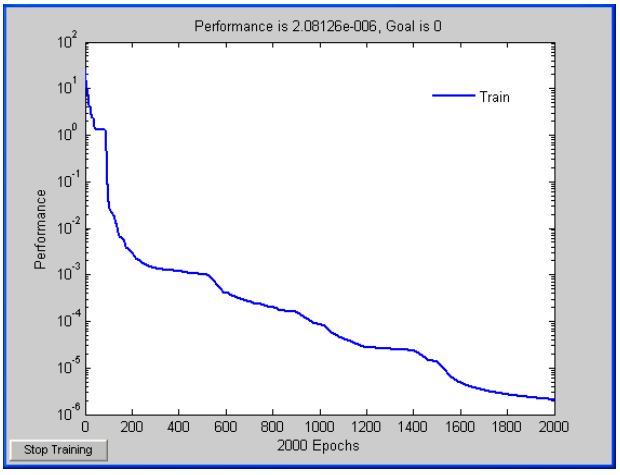
The data samples for experiments are collected by the process of spreading points uniformly in the input space where the number of points greatly exceeds the number wanted. Then randomly select points with probability equal to wanted number/spread number. The split of input data into training, validation and testing dataset is also performed randomly. Corresponding to three function types, experiments are divided into 1-D, 2-D and 3-D. Each type of experiment uses the same network configuration in order for comparisons to be established. The 3-layer network configurations for 1-D, 2-D and 3-D experiments respectively are 1-50-1, 2-75-1, and 3-100-1. Results of experiments are tabulated in Table 1, 2 and 3 in which MSEs on testing samples and numbers of training epochs are used as criteria to compare respectively the accuracy and processing time between models.

Table 1. Resulting table of 1-D experiments (the network configuration: 1-50-1)

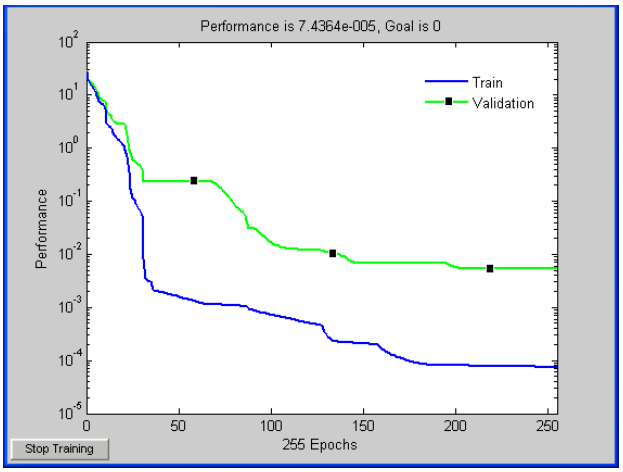
		Conventional Supervised Learning (SL)	SL + momentum	SL + early stopping	SL + momentum + early stopping
Learning sample number		238	238	182	182
Evaluating sample number		0	0	56	56
Testing sample number		53			
Learning rate		0.01			
Momentum coefficient		-	0.9	-	0.9
Epoch number		2000	2000	255	442
Starting MSE on	training samples	23.2086	25.7937	27.4463	25.0564
	evaluating samples	-	-	29.0796	34.6397
Stopping MSE on	training samples	1.0543e-006	2.0813e-006	8.0069e-005	2.1173e-005
	evaluating samples	-	-	0.0155	0.0099
MSE on testing samples		0.0459	0.0276	0.0317	0.0161
Illustrated figures		Fig. 3a	Fig. 3b	Fig. 3c	Fig. 3d



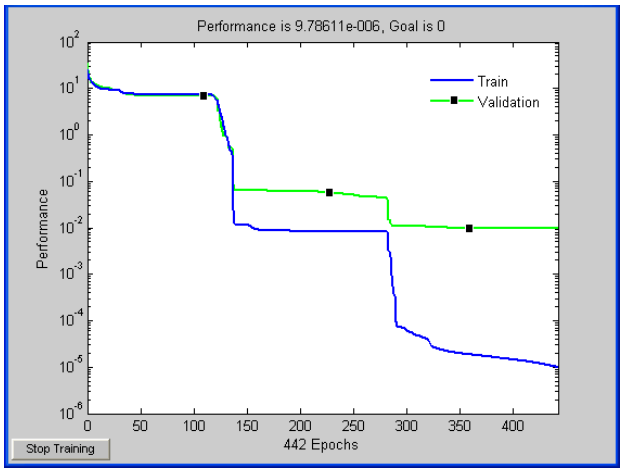
(a)



(b)



(c)

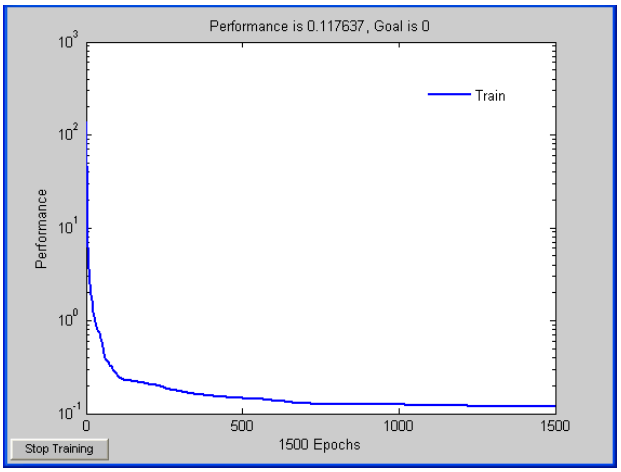


(d)

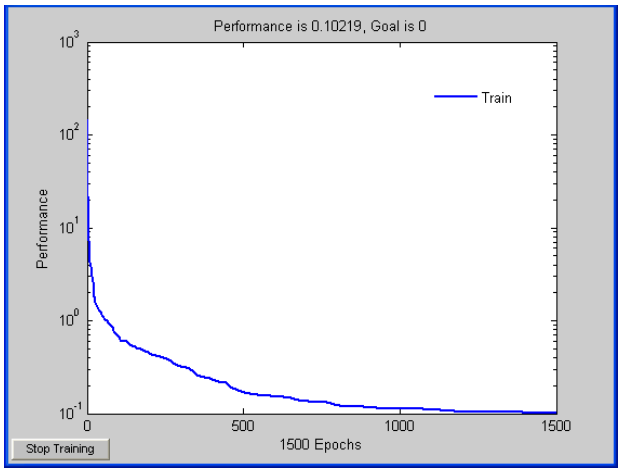
Figure 3. MSE on training and validation data sets in the 1-D case

Table 2. Resulting table of 2-D experiments (the network configuration: 2-75-1)

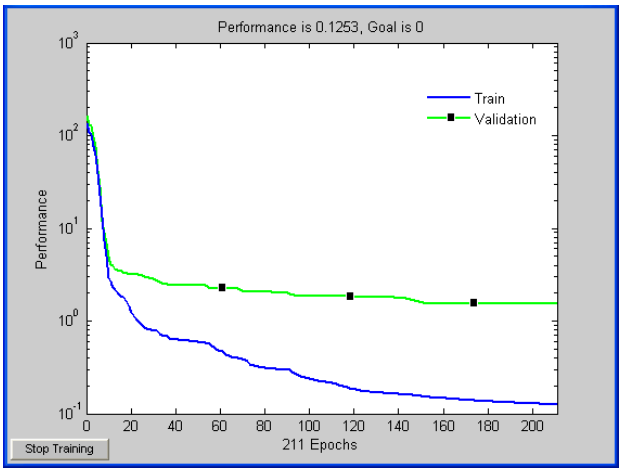
		Conventional Supervised Learning (SL)	SL + momentum	SL + early stopping	SL + momentum + early stopping
Learning sample number		752	752	558	558
Evaluating sample number		0	0	194	194
Testing sample number		173			
Learning rate		0.01			
Momentum coefficient		-	0.9	-	0.9
Epoch number		1500	1500	211	267
Starting MSE on	training samples	139.0608	146.4266	134.9765	134.9403
	evaluating samples	-	-	161.8126	133.1697
Stopping MSE on	training samples	0.1176	0.1022	0.1467	0.1287
	evaluating samples	-	-	1.5320	1.5477
MSE on testing samples		0.7640	0.6985	1.4507	0.6181
Illustrated figures		Fig. 4a	Fig. 4b	Fig. 4c	Fig. 4d



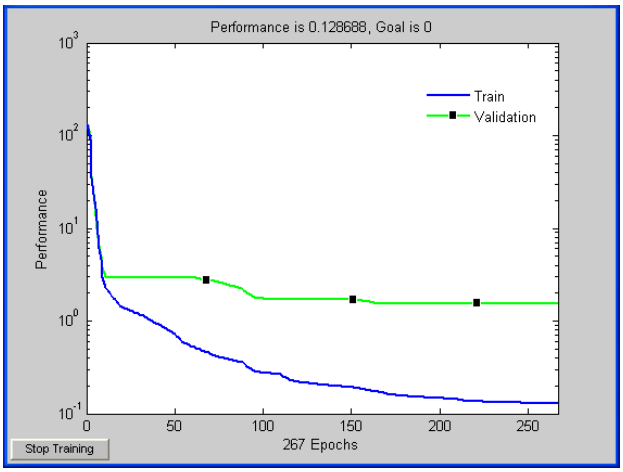
(a)



(b)



(c)

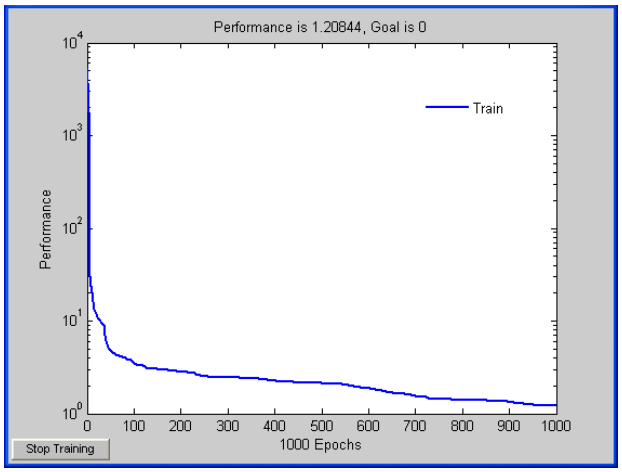


(d)

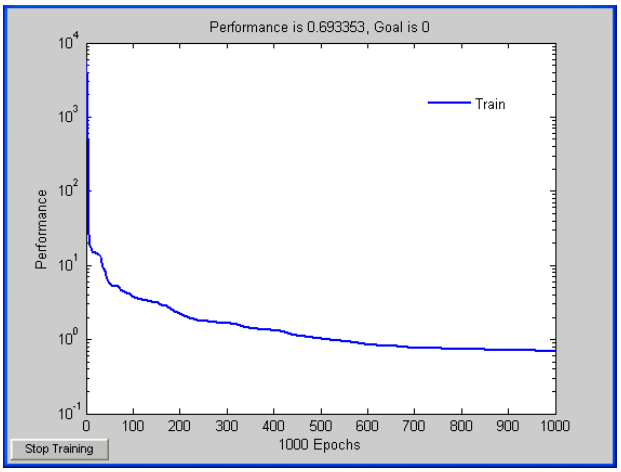
Figure 4. MSE on training and validation data sets in the 2-D case

Table 3. Resulting table of 3-D experiments (the network configuration: 3-100-1)

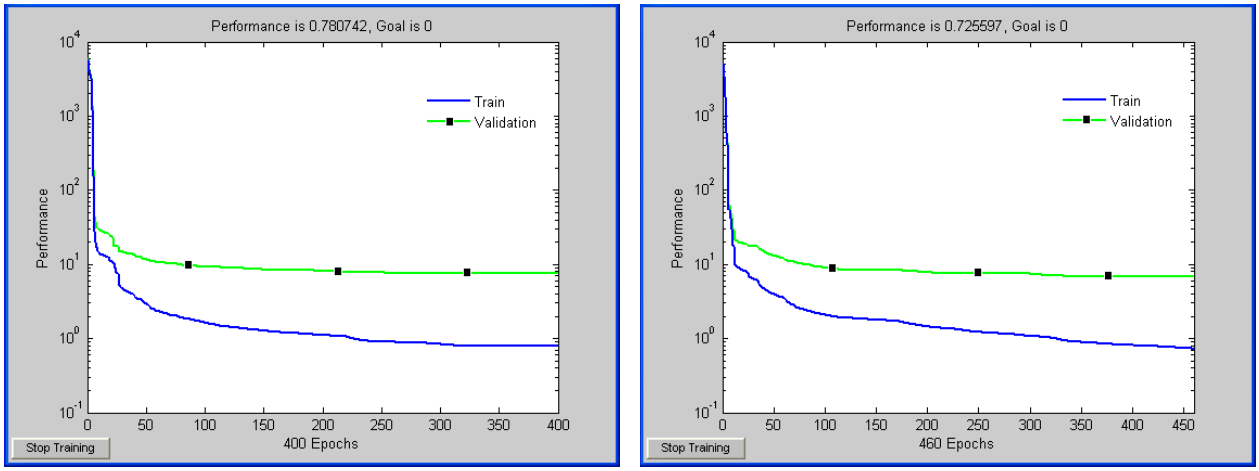
		Conventional Supervised Learning (SL)	SL + momentum	SL + early stopping	SL + momentum + early stopping
Learning sample number		6220	6220	4683	4683
Evaluating sample number		0	0	1537	1537
Testing sample number		1571			
Learning rate		0.01			
Momentum coefficient		-	0.9	-	0.9
Epoch number		1000	1000	400	460
Starting MSE on	training samples	5.2635e+003	5.8022e+003	5.7884e+003	5.2702e+003
	evaluating samples	-	-	5.8444e+003	5.2432e+003
Stopping MSE on	training samples	1.2084	0.6934	0.7807	0.7982
	evaluating samples	-	-	7.5379	6.7845
MSE on testing samples		4.4243	4.1243	4.8095	3.2908
Illustrated figures		Fig. 5a	Fig. 5b	Fig. 5c	Fig. 5d



(a)



(b)



(c) (d)

Figure 5. MSE on training and validation data sets in the 3-D case

Experimental results for all three cases (1-D, 2-D, and 3-D) demonstrated that the supervised learning integrated with momentum results in the accuracy approximately equal to that of the proposed approach. It however takes more epochs (more time) to train the networks than the cases having the early stopping integrated. The early stopping usually stops the learning process early but MSE in these cases is still high unless the momentum also is integrated. It is obvious in the above results that the simultaneous combination of both momentum and early stopping lead to the efficiency in terms of both time processing and accuracy.

4. Conclusions

By using the combined model between early stopping and momentum techniques with supervised learning, the feedforward neural networks become smoother and more stable in the training process and thus result in less time and better accuracy in non-linear function approximation capability. The proposed technique can be applied to other types of neural networks in particular or machine learning techniques in general where the supervised learning is utilized. The accuracy obtained herein, though not very significantly improved compared to the conventional approaches, but definitely has an importance in some applications where accuracy and time are very crucial factors.

References

[1] Rabunal, J.R. and Dorrado J. (2006) *Artificial Neural Networks in Real-life Applications*, Idea Group Publishing, 375 pages.

[2] Demuth, H.B., Beale, M.H. and Hagan, M.T. (2009) *Neural Network Toolbox™ 6: User's Guide*, The MathWorks, Inc.

[3] Snyman, J.A. (2005) *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms,* Springer Publishing, ISBN 0-387-24348-8, 257 pages.

[4] Qian, N. (1999) On the momentum term in gradient descent learning algorithms, *Neural networks*, vol. 12, pp. 145-151.

[5] Caruana, R., Lawrence, S. and Giles, C.L. (2001) Overfitting in neural networks: backpropagation, conjugate gradient, and early stopping, *Advance in Neural Information Processing Systems*, vol. 13, pp. 402-408.

[6] Ding, S.Q., and Xiang, C. (2003) Overfitting problem: a new perspective from the geometrical interpretation of MLP, pp 50 - 57. In Abraham, A., Köppen, M. and Franke, K. (eds.), *Design and application of hybrid intelligent systems*, IOS Press, Amsterdam, The Netherlands, 1133 pages.

[7] Tang, Z., Wang, X.G., Tamura, H. and Ishii, M. (2003) An algorithm of supervised learning for multilayer neural networks, *Neural Computation*, vol. 15, no. 5, pp. 1125-1142.

[8] Li, Z. and Xu, L. (2006) Research on Overcoming the Local Optimum of BPNN, *The Sixth World Congress on Intelligent Control and Automation (WCICA 2006)*, Dalian, China, vol. 1, pp. 2681-2685.

[9] Fine, T.L. (1999) *Feedforward Neural Network Methodology*, New York, Springer-Verlag, ISBN 0-387-98745-2, 340 pages.