J Glob Optim (2011) 51:271–284
DOI 10.1007/s10898-010-9597-6

An improved training algorithm for feedforward neural network learning based on terminal attractors

Xinghuo Yu · Bin Wang · Batsukh Batbayar · Liuping Wang · Zhihong Man

Received: 13 August 2010 / Accepted: 19 August 2010 / Published online: 2 September 2010

Abstract In this paper, an improved training algorithm based on the terminal attractor concept for feedforward neural network learning is proposed. A condition to avoid the singularity problem is proposed. The effectiveness of the proposed algorithm is evaluated by various simulation results for a function approximation problem and a stock market index prediction problem. It is shown that the terminal attractor based training algorithm performs consistently in comparison with other existing training algorithms.

Keywords Feedforward neural network · Terminal attractor · Back-propagation · Training · Optimization

X. Yu (✉)
Platform Technologies Research Institute, RMIT University, Melbourne, VIC 3001, Australia
e-mail: x.yu@rmit.edu.au

X. Yu
School of Automation, Southeast University, Nanjing, China

B. Wang · L. Wang
School of Electrical and Computer Engineering, RMIT University, Melbourne, VIC 3001, Australia
e-mail: b.wang@rmit.edu.au

L. Wang
e-mail: liuping.wang@rmit.edu.au

B. Batbayar
School of Information Technology, National University of Mongolia, Ulaanbaatar, Mongolia
e-mail: bbatsukh@yahoo.com

Z. Man
Faculty of Engineering and Industrial Sciences, Swinburne University of Technology,
Melbourne, VIC 3122, Australia
e-mail: zman@swin.edu.au

1 Introduction

Feedforward neural networks (FNNs) have been widely used for various learning tasks, for example, in data prediction, multimedia processing, high accuracy optical character recognition, dynamical modelling, medical diagnosis and finance [1–4]. One of the basic training algorithms for FNNs, the gradient descent based back-propagation (GDBP) algorithm [2] has been widely used to determine the weights in the paradigm of supervised learning in multi-layer FNNs. A common characteristic in this class of algorithms is that, because of the fixed learning rate, it incurs an asymptotic convergence. It means that the closer to the desired weights, the slower the convergence speed of the weight updates, making fast convergence with high precision difficult. One major drawback of the GDBP algorithm is its slow convergence and it can easily be trapped into local minima. To address this problem, many variations of the basic GDBP algorithm have been proposed in the last few decades, for example, an on-line GDBP learning algorithm for time-varying inputs [5], an adaptive learning algorithm with reduced complexity [6], a fast learning algorithm based on the gradient descent of neuron space [7], the Levenberg-Marquardt back-propagation (LMBP) algorithm [8], a general GDBP learning algorithm [9], and the improved fast GDBP based algorithms [10,11].

The terminal attractor based back-propagation algorithm (TABP) is another fast back-propagation (BP) training algorithm [12], which makes use of the terminal attractor (TA) concept to enhance convergence speed. The key concept in the TABP training algorithm is the introduction of an 'equivalent' varying learning rate in the weight update law [13] to improve the convergence speed when the actual weights are close to the ideal weights of the FNN. In this paper, we propose an improved training algorithm based on a fast TA concept. We will first give a detailed convergence analysis. We will then show that our new algorithm enables a fast convergence both at a distance from and in a close range to the ideal weights. A condition to guarantee avoidance of the singularity problem is derived. Simulation studies are presented to show the effective training performance of our new algorithm in comparison with several typical existing algorithms and an application to stock market index prediction.

2 The improved training algorithm based on the fast TA concept

Before we proceed, we denote the inputs, weights, desired outputs and actual outputs of the FNN as

$$x(t) = (x_1, x_2, \ldots, x_n)^\top \in R^n \tag{1}$$

$$w(t) = (w_1, w_2, \ldots, w_l)^\top \in R^l \tag{2}$$

$$y_d(t) = (y_{d1}, y_{d2}, \ldots, y_{dm})^\top \in R^m \tag{3}$$

$$y(t) = (y_1, y_2, \ldots, y_m)^\top \in R^m \tag{4}$$

where $x(t)$ is the input vector, $w(t)$ is the weight vector (including weights for input, output and hidden layers), $y_d(t)$ is the desired output vector and $y(t)$ is the output vector of the FNN. The error at any instant is represented as

$$\begin{aligned} e(t) &= \frac{1}{2}(y(t) - y_d(t))^\top (y(t) - y_d(t)) \\ &= \frac{1}{2}\|y(t) - y_d(t)\|^2 \end{aligned} \tag{5}$$

Note that here the input is of a general type, and it can be discrete, continuous and time varying. The weight vector, $w(t)$, represents weights for a perceptron (single layer FNN) as well as a multi-layer FNN.

We now develop the criterion for evaluating the performance of the FNN learning. Since the inputs can be time varying, a time window should be used to evaluate the training efficiency [9], that is

$$J = \frac{1}{\tau} \int_{t-\tau}^{t} e(\theta) d\theta \tag{6}$$

where τ is the length of the time window. The formulation (6) is particularly useful for on-line continuous time learning as it considers the evolution of learning in an average sense within a prescribed time window. However, for discrete data sets, since the evaluation of errors can only be done at "discrete moments", (6) can be rewritten as

$$J_k = \lim_{\tau \to 0} \frac{1}{\tau} \int_{t_k-\tau}^{t_k} e(\theta) d\theta = e(k) \tag{7}$$

which becomes the usual form for training FNNs with discrete data sets.

To enable the FNN to learn from given example data, the following GDBP algorithm is commonly used [2]

$$\frac{dw}{dt} = -\zeta \nabla_w J \tag{8}$$

where $J = J(w)$ is the error function, $\nabla_w J$ is the gradient vector for weights, and ζ is a constant learning rate. A general type of BP algorithms was proposed in [9], which is

$$\frac{dw}{dt} = -\left(\mu I + \sigma \frac{\partial^2 J}{\partial w \partial w^\top}\right)^{-1} \frac{\frac{\partial J}{\partial w^\top}}{\|\frac{\partial J}{\partial w}\|^2} \left(\mu \frac{\partial J}{\partial t} + \sigma \frac{\partial J}{\partial w} \frac{\partial^2 J}{\partial t \partial w^\top} + \varsigma \|\frac{\partial J}{\partial w}\|^2 + \eta J\right) \tag{9}$$

where I is an identity matrix of proper dimension, and $\mu, \sigma, \varsigma, \eta$ are constants. This formula can interpret many existing GDBP based algorithms. For example, the conventional BP algorithm can be obtained by setting $\partial J/\partial t = 0, \sigma = 0$ and $\eta = 0$, which becomes

$$\frac{dw}{dt} = -\mu^{-1} \varsigma \frac{\partial J}{\partial w^\top} = -\zeta \nabla_w J, \quad \zeta = -\mu^{-1} \varsigma \tag{10}$$

If we choose ζ as

$$\zeta = \frac{\Omega(J)}{\|\nabla_w J\|^2} \tag{11}$$

where $\Omega(J)$ is a non-negative continuous function of J, then, the dynamics of the error function becomes

$$\frac{dJ}{dt} = (\nabla_w J)^\top \frac{dw}{dt} = -\Omega(J) \tag{12}$$

which makes the error function continuously decreasing to zero.

The GDBP algorithm is known to be slow in convergence and can be easily trapped into local minima. To improve the learning performance, a TABP training algorithm was proposed in [13–15]. Replacing $\Omega(J)$ with ηJ^ρ $(0 < \rho < 1)$ in (8) leads to

$$\frac{dw}{dt} = -\frac{\eta J^{\rho}}{\|\nabla_w J\|^2}\nabla_w J \tag{13}$$

and

$$\frac{dJ}{dt} = -\eta J^{\rho} \tag{14}$$

where η a constant learning rate. With this learning algorithm, J will reach zero in a finite time $t_r = \frac{1}{\eta(1-\rho)} J^{1-\rho}(0)$. It was shown that the effectiveness of the TABP algorithm lies in the introduction of an equivalent time-varying learning rate which increases exponentially when the actual weights are close to the ideal weights. However, when the initial weights are far away from the ideal weights, the convergence is shown to be slower than the conventional GDBP algorithms.

The so-called Fast Terminal Sliding Mode (FTSM) was recently introduced in [16], which is described as below,

$$s = \dot{z} + \alpha z + \beta z^{\rho} = 0 \tag{15}$$

where $\alpha, \beta > 0, \rho = q/p, 0 < \rho < 1$, with $q, p(q < p)$ being positive odd integers. The dynamics (15) reaches $z = 0$ in finite time. When z is far away from the origin, the approximate dynamics becomes $\dot{z} = -\alpha z$. When z is close to the equilibrium, the dynamics (15) is dominated by the conventional terminal attractor $\dot{z} = -\beta z^{\rho}$ [12]. The exact time to reach the equilibrium 0 with initial state of $z(0)$ can be solved from (15) as

$$t_s = \frac{p}{\alpha(p-q)}\left(\ln(\alpha z(0)^{(p-q)/p} + \beta) - \ln\beta\right) \tag{16}$$

Now we combine the general BP algorithm with the FTSM (15) to derive the Fast TABP (FTABP) training algorithm, which is

$$\frac{dw}{dt} = -\gamma\nabla_w J - \frac{\eta J^{\rho}}{\|\nabla_w J\|^2}\nabla_w J \tag{17}$$

Then the dynamics of the error function becomes

$$\begin{aligned}\frac{dJ}{dt} &= (\nabla_w J)^{\top}\frac{dw}{dt} \\ &= -(\nabla_w J)^{\top}(\gamma\nabla_w J) - (\nabla_w J)^{\top}\frac{\eta J^{\rho}}{\|\nabla_w J\|^2}(\nabla_w J) \\ &= -\gamma\|\nabla_w J\|^2 - \eta J^{\rho} \le -\eta J^{\rho}\end{aligned} \tag{18}$$

It means the FTABP algorithm enables the weights to converge in a time which is much less than $t_r = \frac{1}{\eta(1-\rho)} J^{(1-\rho)}(0)$. The first part of the new algorithm (17) dominates when the weights are far away from the ideal weights. The second part with TA dominates when the weights are close to the ideal ones. As a result of this combination, the FTABP algorithm will be much faster than the conventional GDBP and GABP algorithms for FNNs, taking the advantage of both the GDBP and the TABP based algorithms.

The "singularity" problem also exists in the FTABP algorithm, similar to that in the TABP algorithm [15]. The term that causes the singularity is the second term in (17), denoted as

$$T_{dw}^2 = \frac{\eta J^{\rho}}{\|\nabla_w J\|^2}\nabla_w J \tag{19}$$

Hence

$$\|T_{dw}^2\| = \frac{\eta J^\rho}{\|\nabla_w J\|} \tag{20}$$

When $\nabla_w J \to 0$, the term $\|T_{dw}^2\|$ tends to infinity but J tends to a nonzero constant due to local minima, hence does the weight variation rate dw/dt, resulting in a disruptive change to the weight w. This behavior will affect its dynamical characteristics near the global minimum. To design a strategy to avoid this occurrence, we need to study the dynamic characteristics of the FTABP algorithm in the neighborhood of the global minimum. First, the error function is represented by its Taylor series expansion at w^*, which corresponds to the global minimum, as

$$J(w) = J(w^*) + G(w^*)\triangle w + \frac{1}{2}\triangle w^\top H(w^*)\triangle w + \cdots \tag{21}$$

where $\triangle w = w - w^*$, $G(w^*) = \nabla_w J^\top|_{w=w^*}$, $H(w^*) = \nabla_w^2 J|_{w=w^*}$. If $\|\triangle w\|$ is very small, then the higher order part of (21) can be neglected and the only important terms of interest are the first and second order terms. If w^* is the global minimum, which means $J(w^*) = 0$ and $G(w^*) = 0$, then (21) becomes

$$\triangle J(w) = J(w) = \frac{1}{2}\triangle w^\top H(w^*)\triangle w \tag{22}$$

where $\triangle J(w) = J(w) - J(w^*)$. From (22), it can be concluded that

$$\nabla_w J = \frac{1}{2}H(w^*)\triangle w \tag{23}$$

Replacing $\Omega(J)$ with ηJ^ρ $(0 < \rho < 1)$ and substituting (22) and (23) into (20) yield

$$T_{dw}^2 = \frac{2\eta((1/2)\triangle w^\top H(w^*)\triangle w)^\rho}{\|H(w^*)\triangle w\|} \tag{24}$$

Because w^* is the global minimum, $H(w^*)$ must be positive semi-definite [8]. To better understand (24), the following transformation is performed for $H(w^*)$[8]

$$H(w^*) = B\Lambda B^\top \tag{25}$$

where matrix Λ is a diagonal matrix with all eigenvalues of $H(w^*)$ on the diagonal and B is an orthogonal matrix. Note that since B is orthogonal, $B^{-1} = B^\top$. Now let $\triangle w = Bc$ where c is a n-dimensional vector. (24) can be rewritten as

$$T_{dw}^2 = \frac{2\eta\left((1/2)c^\top B^\top B\Lambda B^\top Bc\right)^\rho}{\left((B\Lambda B^\top Bc)^\top (B\Lambda B^\top Bc)\right)^{1/2}} = \frac{1}{2^{\rho-1}}\frac{\eta\left(\sum_{i=1}^n \lambda_i c_i^2\right)^\rho}{\left(\sum_{i=1}^n \lambda_i^2 c_i^2\right)^{1/2}} \tag{26}$$

where $\lambda_i (i = 1, 2, \ldots, n)$, are the eigenvalues of $H(w^*)$ and n (which may be less than the rank of matrix $H(w^*)$) is the total number of the nonzero eigenvalues of $H(w^*)$. The following relation is then easily derived from (26),

$$\|T_{dw}^2\| \le \eta\frac{1}{2^{\rho-1}}\frac{\lambda_{\max}^\rho}{\lambda_{\min}}\left(\sum_{i=1}^n c_i^2\right)^{\rho-\frac{1}{2}} \tag{27}$$

where $\lambda_{\max}$ is the largest eigenvalue among $\lambda_i (i = 1, 2, \ldots, n)$, and $\lambda_{\min}$ is the smallest nonzero eigenvalue. Since

$$\|c\| = \|B^{-1}\triangle w\| \le \|B\|\|\triangle w\| = \|\triangle w\| \tag{28}$$

(27) can be rewritten as

$$\|T_{dw}^2\| \leq \frac{\eta}{2^{\rho-1}} \frac{\lambda_{\max}^{\rho}}{\lambda_{\min}} \|\triangle w\|^{2\rho-1} \tag{29}$$

One can see

$$\rho = \frac{1}{2} \tag{30}$$

is a critical value. When $\rho > 1/2$, $\lim_{w \to w^*} \|T_{dw}^2\| = 0$, which means $\|T_{dw}^2\|$ will asymptotically decrease to zero when w approaches the global minimum w^*, hence $\lim_{w \to w^*} \|\frac{dw}{dt}\| = 0$. However, if $\rho < 1/2$, then the "singularity" might happen, that is, $\|T_{dw}^2\|$ may go to infinity, resulting in a disruptive change to $\frac{dw}{dt}$ which causes a significant change to the weights w. This phenomenon will be demonstrated in the next section.

3 Simulation studies

In this section the proposed FTABP algorithm is illustrated by application to two system identification problems, the function approximation and stock market index forecasting. In order to check the performance of the FTABP, it is compared with nine popular fast training methods, among which are the gradient descent with adaptive momentum and step sizes (GDX), the BFGS Quasi-Newton (BFG), the resilient backpropagation (RP), the scaled conjugated gradient (SCG), the conjugate gradient with Powell/Beale restarts (CGB), the Fletcher-Powell conjugate gradient (CGF), the Polak-Ribiére conjugate gradient (CGP), the One Step Secant (OSS), and the celebrated Levenberg-Marquardt (LM) [17]. All the simulations share the following conditions:

- The network topology and neural functions. In all cases, the logistic function is used for hidden neurons, while for output neurons the linear function is used.
- Initial step size is 0.01.
- The training data set is normalized.

A number of simulations are performed using a different set of initial weights for each one. The initial set is the same for all algorithms for each simulation and is obtained by the most common initialization method which uses uniformly distributed random numbers and arrays produced. Finally, statistical tests are carried on in order to compare the performances of different training algorithms. The model behavior and performance are evaluated in terms of the Mean Square Error (MSE).

3.1 Function approximation

The aim here is to achieve a mapping for a function with two variables which is $y = \sin(x_1^2) + \sin(x_2^2)$ (see Fig. 1). The training data set is used by different training algorithms to derive a set of weights of a FNN, which has a single hidden layer with 5 nodes (neurons), such that the FNN can give a good approximation of the function. 100 samples are employed for the learning process. In order to obtain the average MSE curves during the training process, 100 simulations are done for each learning algorithm, with the stopping time when the MSE reaches 10^{-2} and 10^{-3} respectively. The maximum of the iteration number is 1,000. First, the function is approximated by different learning algorithms. In FTABP training algorithm (17), the parameters are set as $\gamma = \eta = 0.5$. We first test the singularity condition. Figures 2

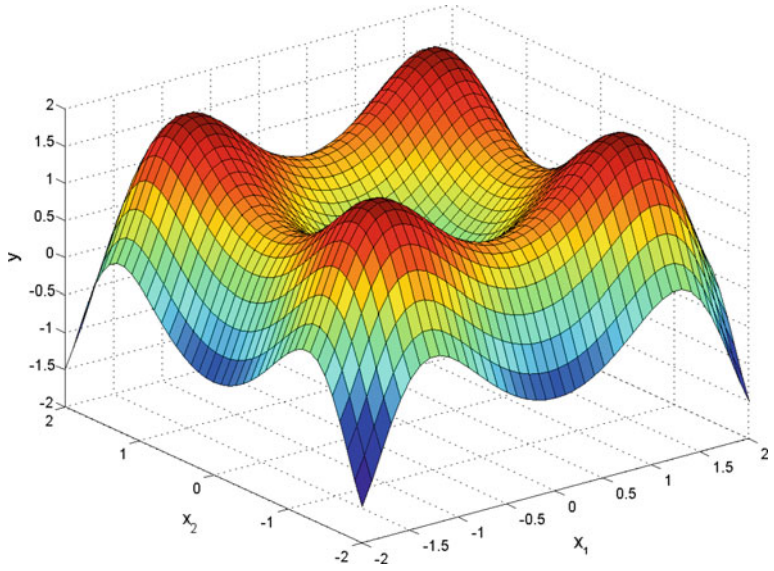

Fig. 1 The function to be approximated, $y = \sin(x_1^2) + \sin(x_2^2)$

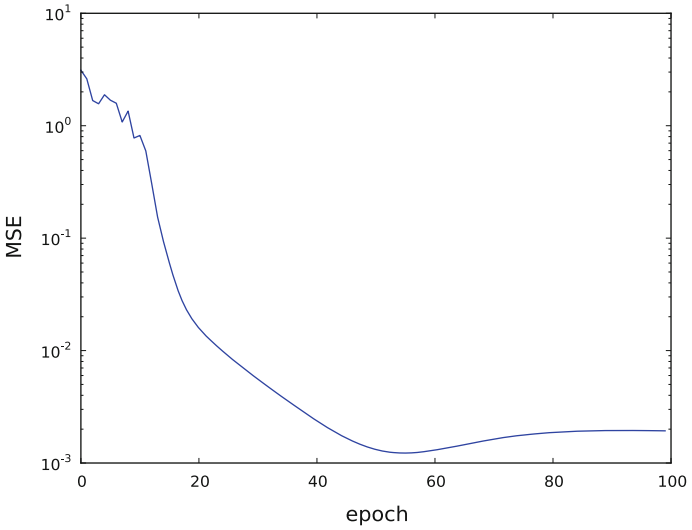

Fig. 2 Performance of FTABP training algorithm when $\rho = 0.6$

shows the average MSE curve of the FTABP over the 100 simulations with $\rho = 0.6$. Figure 3 shows that the singularity phenomenon occurs when ρ is chosen as $\rho = 0.4$. This shows the effect of the critical value of $\rho = 0.5$ to the FNN training. Figures 4 and 5 show the the average MSE curves of the FTABP, traditional and fast training algorithms for FNNs over the 100 simulations. From these figures, one can see FTABP outperforms the traditional training algorithms including the gradient descent algorithm (GD) and the gradient descent algorithm with momentum (GDM). Comparative measures between FTABP and the fast training

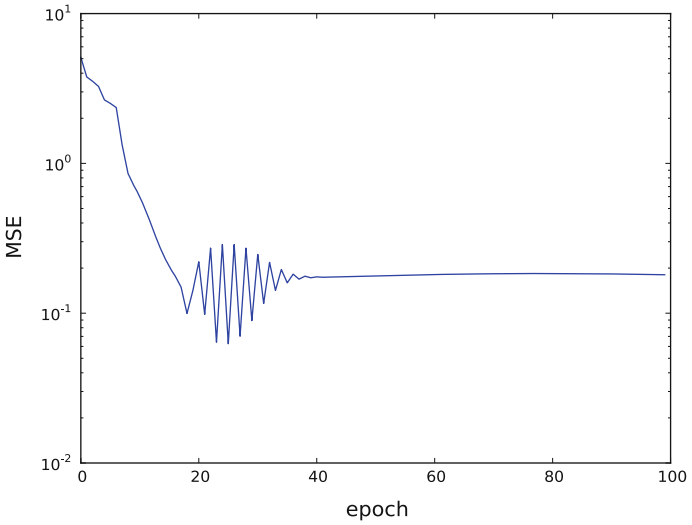

Fig. 3 Singularity appears when $\rho = 0.4$

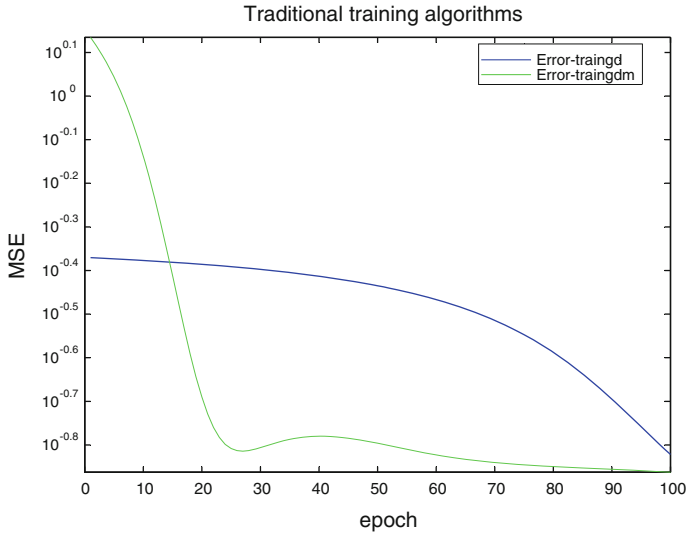

Fig. 4 Performances of traditonal training algorithms

algorithms are collected in Tables 1 and 2. These measures are the minimum, maximum, mean and the standard deviation of the epochs for each algorithm when it reaches the target. Figures 6 and 7 show the box-and-whisker plots for each learning algorithm. In these two graphics the box corresponds to the interquartile range, the bar inside the box represents the median, the whiskers extend to the farthest points that are not outliers, and outliers are represented by the plus sign.

From these figures and tables, one can see FTABP achieves a rather satisfying performance, reaching the target within one iteration in its best action. One feature making FTABP stand out is the consistency of epochs to reach the target. This feature can be observed by

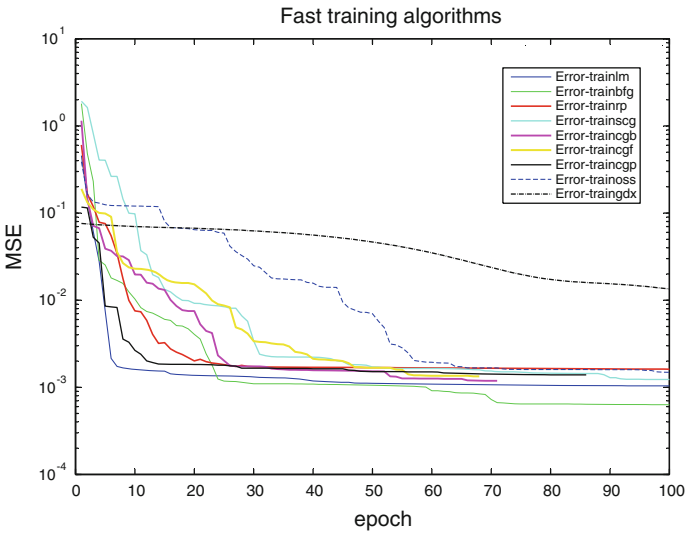

Fig. 5 Performances of the fast training algorithms

Table 1 Comparative measures on different learning algorithms with target as 10^{-2} for 2-dimensional function

Training algorithms	Min.	Max.	Mean	Std.
FTABP	1	321	36	49
LM	2	47	8	8
BFG	2	1,000	23	100
RP	7	111	18	15
SCG	2	1,000	34	99
CGB	2	39	10	7
CGF	2	102	19	16
CGP	2	58	12	11
OSS	2	70	16	10
GDX	69	1,000	177	124

Table 2 Comparative measures on different learning algorithms with target as 10^{-3} for 2-dimensional function

Training algorithms	Min.	Max.	Mean	Std.
FTABP	1	682	329	148
LM	2	447	99	111
BFG	2	624	188	165
RP	269	1,000	946	119
SCG	2	1,000	536	366
CGB	2	140	33	27
CGF	2	195	57	49
CGP	2	1,000	63	103
OSS	2	1,000	598	360
GDX	1,000	1,000	1,000	0

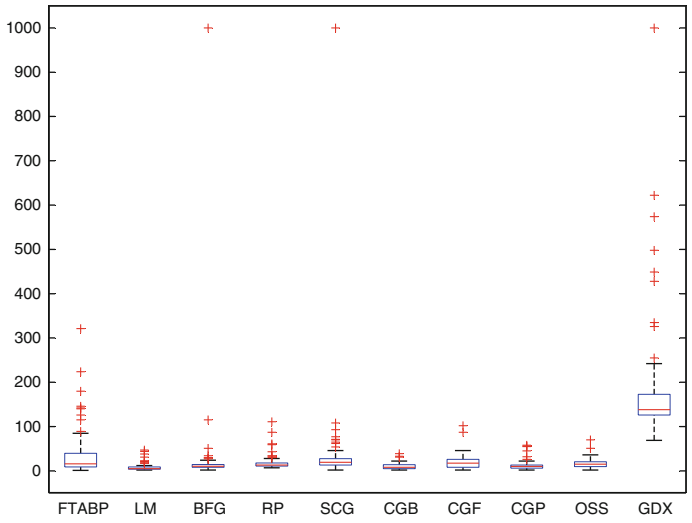

Fig. 6 The *boxplots* of TABP and other fast training algorithms with the target as 10^{-2} for the 2-dimensional function

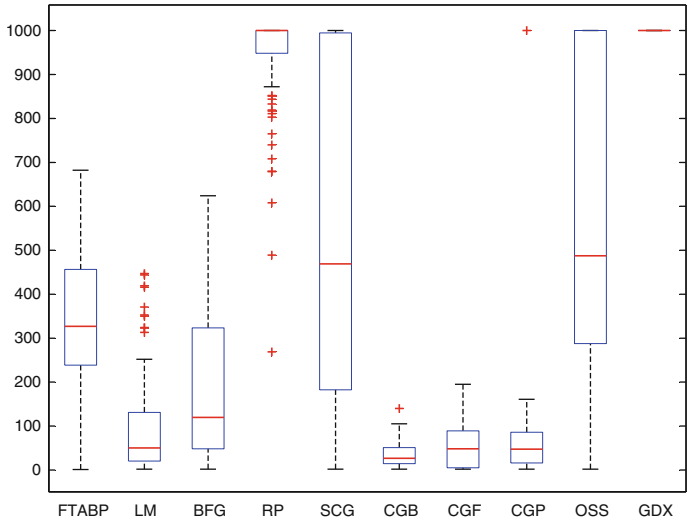

Fig. 7 The *boxplots* of TABP and other fast training algorithms with the target as 10^{-3} for the 2-dimensional function

the numbers of maximum epochs and standard deviations in Tables 1 and 2 and the box-and-whisker plots in Figs. 6 and 7. While the GDX algorithm has the worst performance in both tasks, the LM and CGB algorithms shine in the task to reach the target 10^{-2}. The latter worked out better than others in the second task. However, if LM and CGB are combined with the fast TA concept, it will produce better results than any algorithm alone. This will be investigated in future research.

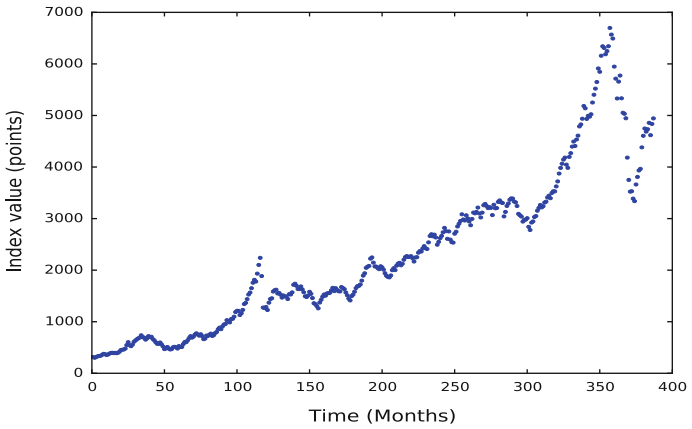

Fig. 8 ASX index monthly data from Jan. 1978 to Apr. 2010

3.2 ASX index time series prediction

Neural networks have made a significant impact on financial forecasting technologies [18,19]. Comparing with conventional statistic tools the main advantage of neural networks are that they are able to learn to recognize patterns in the data set, hence they are flexible in a changing environments and they can build models when more conventional approaches fail. In this simulation, the FTABP will be applied for the forecasting task. The data set used for forecasting is the time series corresponding to the ASX (Australian Stock Market) index values from Jan. 1978 to Apr. 2010 which has been plotted in Fig. 8. The goal of the network in this case is to predict the index for a given day based on the previous index. A 1-5-1 topology (1 input, 5 hidden neurons and 1 output neuron) is used in the simulation, $\gamma = \eta = 0.5$, and $\rho = 0.6$. 388 samples are employed for the learning process. To assemble training, testing and validation sets, the random resampling approach is employed to divide the dataset to multiple subsets for training, testing and validation due to the limited size of available data. In order to render the data suitable as input to the neural network model, the data are normalized in range of (0, 1), which is the range of the logistic function.

The FNN forecaster can be described as follows:

$$y = x(t) = f(x(t-k), x(t-2k), \ldots, x(t-mk)) \tag{31}$$

where k is the time lag and m is the number of inputs. However, the selection of the input variables and lag structures is challenging when using FNNs in forecasting [20]. One good method is using autocorrelation functions to select the input data and lag structures to model the FNN [21]. In a time series, given observations, $x_1, x_2, \ldots, x_i, \ldots, x_N$ at times $1, 2, \ldots, i, \ldots, N$, the lag k autocorrelation function is defined as [22]

$$r_k = \frac{\sum_{i=1}^{N-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{32}$$

where $\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$ is the mean. r_k is called the autocorrelation coefficient at lag k whose value ranges between -1 and 1. The autocorrelation coefficient r_k can be plotted against the lag k to develop a correlogram which gives a visual look at a range of correlation coefficients

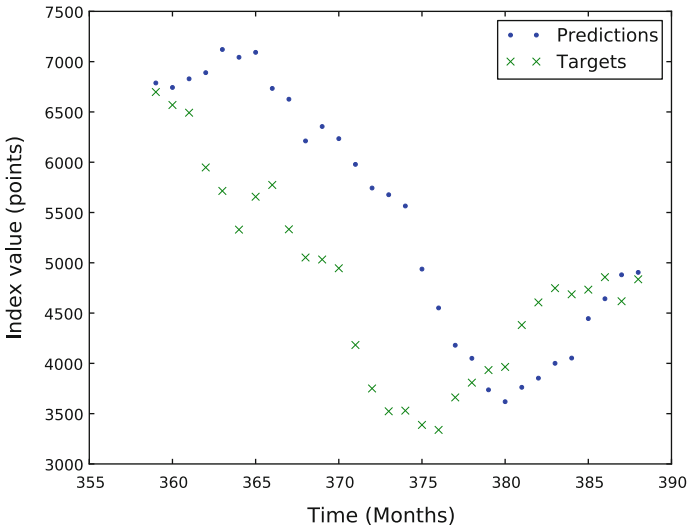

Fig. 9 Short term prediction of the last 30 months

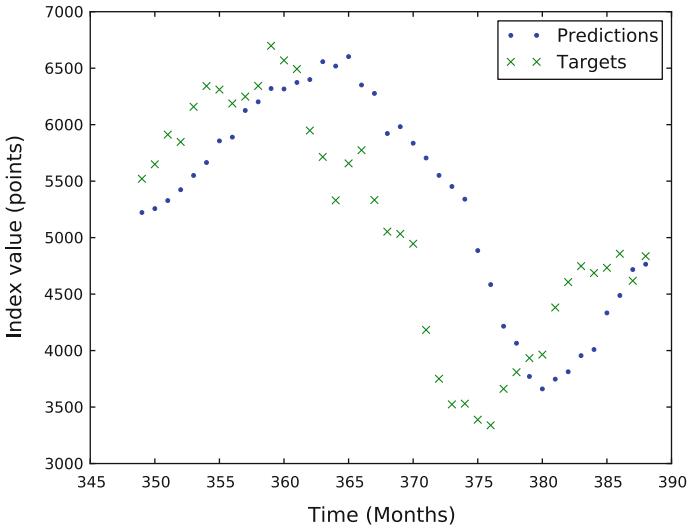

Fig. 10 Short term prediction of the last 40 months

at relevant time lags so that significant values may be seen. In the simulation, the time lag is chosen as $k = 2$ and the number of inputs $m = 4$. After the training session, the short term prediction of the last 30, 40 and 50 steps are illustrated in Figs. 9, 10 and 11 respectively. In each simulation, the maximum iteration number is 1,000, the stopping time is when when the MSE on the training set reaches 10^{-3}. One can observe that the FNN performs very well in predicting future trends.

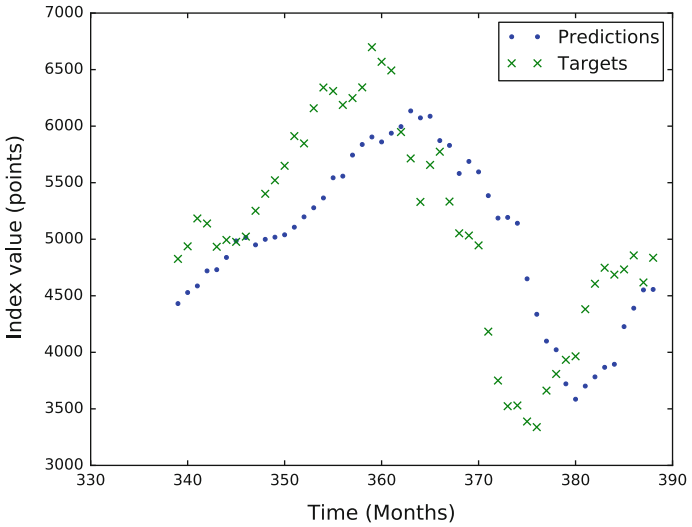

Fig. 11 Short term prediction of the last 50 months

4 Conclusion

In this paper, an improved BP training algorithm combining with fast TA for FNN learning has been proposed. A condition to guarantee the avoidance of the singularity problem has been derived. The effectiveness and efficiency of the proposed algorithm have been shown and demonstrated by a number of empirical simulations. Further work will be carried on in the direction of combining the FTABP algorithm with other efficient learning methods such as LM algorithm to further improve their performance.

References

1. Mehra, P., Wah, B.W.: Artificial Neural Networks: Concepts and Theory. IEEE Computer Society Press, California (1992)
2. Zurada, J.M.: Introduction to Artificial Neural Systems. West Publishing, St. Paul, MN (1992)
3. Pardalos, P.M., Resende, M.G.C. (eds.): Handbook of Applied Optimization. Oxford University Press, New York (2002)
4. Pardalos, P.M., Romeijn, E. (ed.): Handbook of Global Optimization, Heuristic Approaches, vol. 2, Kluwer Academic Publishers (2002)
5. Zhao, Y.: On-line neural network learning algorithm with exponential convergence rate. Electronic Lett. **32**(15), 1381–1382 (1996)
6. Zhou, G., Si, J.: Advanced neural network training algorithm with reduced complexity based on Jacobian deficiency. IEEE Trans. Neural Netw. **9**(3), 448–453 (1998)
7. Parisi, R., Di Claudio, E.D., Orlandi, G., Rao, B.D.: A generalized learning paradigm exploiting the structure of feedforward neural networks. IEEE Trans. Neural Netw. **7**(6), 1450–1459 (1996)
8. Hagan, M.T., Menhaj, M.B.: Training feedforward neural networks with the Marquardt algorithm. IEEE Trans. Neural Netw. **5**, 989–993 (1994)
9. Yu, X., Efe, O., Kaynak, O.: A general backpropagation algorithm for feedforward neural networks learning. IEEE Trans. Neural Netw. **13**, 251–259 (2002)
10. Abid, S., Fnaiech, F., Najim, M.: A fast feedforward training algorithm using a modified form of the standard backpropagation algorithm. IEEE Trans. Neural Netw. **12**(2), 424–430 (2001)
11. Kathirvalavakumar, T., Tangavel, P.: A modified backpropagation training algorithm for feedforward neural networks. Neural Process. Lett. **23**, 11–119 (2006)

12. Zak, M.: Terminal attractors in neural networks. Neural Networks **2**, 259–274 (1989)
13. Wang, S., Hsu, C.H.: Terminal attractor learning algorithms for backpropagation neural networks. In: Proceedings of the International Joint Conference on Neural Networks Singapore, pp. 193–189. IEEE Press (1991)
14. Bianchi, B., Fanelli, S., Gori, M., Maggini, M.: Terminal attractor algorithms: a critical analysis. Neurocomputing **15**, 3–13 (1997)
15. Jiang, M., Yu, X.: Terminal attractor based backpropagation learning for feedforward neural networks. In: Proceedings of the 2001 IEEE International Symposium on Circuits and Systems (ISCAS 2001), pp. 711–714 (2001)
16. Yu, X., Man, Z.: Fast terminal sliding-mode control design for nonlinear dynamical systems. IEEE Trans. Circuits Syst. I **49**(2), 261–264 (2002)
17. Demuth, H.B.,Beale, M., Hagan, M.T.:Neural Network Toolbox User's Guide. The MathWorks Inc (2010)
18. Azoff, E.M.: Neural Network Time Series Forecasting of Financial Markets. John Wiley & Sons, London (1994)
19. Nenortaite, J., Simutis, R.: Development and evaluation of decision-making model for stock markets. J. Glob. Optim. **36**, 1–19 (2006)
20. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks: the state of the art. Int. J. Forecast. **14**, 35–62 (1998)
21. Lachtermacher, G., Fuller, J.D.: Backpropagation in time-series forecasting. J. Forecast. **14**, 381–393 (1995)
22. Chatfield, C. : The Analysis of Time Series: An Introduction. Chapman and Hall, London (2004)