Improving the Quickprop algorithm

Chi-Chung Cheung, *Senior Member*, *IEEE,* Sin-Chun Ng, *Senior Member, IEEE,* Andrew K Lui, *Member IEEE*

***Abstract*— Backpropagation (BP) algorithm is the most popular supervised learning algorithm that is extensively applied in training feed-forward neural networks. Many BP modifications have been proposed to increase the convergence rate of the standard BP algorithm, and Quickprop is one the most popular fast learning algorithms. The convergence rate of Quickprop is very fast; however, it is easily trapped into a local minimum and thus it cannot converge to the global minimum. This paper proposes a new fast learning algorithm modified from Quickprop. By addressing the drawbacks of the Quickprop algorithm, the new algorithm has a systematic approach to improve the convergence rate and the global convergence capability of Quickprop. Our performance investigation shows that the proposed algorithm always converges with a faster learning rate compared with Quickprop. The improvement in the global convergence capability is especially large. In one learning problem (application), the global convergence capability increased from 4% to 100%.**

I. Introduction

Backpropagation (BP) algorithm [1] is the most popular supervised learning algorithm that is extensively applied in training feed-forward neural networks. BP is successful because it is simple and its computational complexity is low. The learning rate of BP is slow, however, and it is easy to be trapped in a local minimum, especially for some non-linearly separable problems such as XOR problem [2, 3]. Some modifications have been proposed to address these drawbacks and enhance the performance of the standard BP algorithm, and most of those modifications have focused on handling the "flat spot" problem to speed up the learning process [12 – 22].

The main reason to have the "flat spot" problem is due to the premature saturation in the derivative of the activation function [4 – 7]. If a learning process is trapped into a "flat spot" or a local minimum, the weight update of the learning process will become very slow or even unchanged. Different fast learning algorithms use different way to solve this problem. Quickprop [12] predicts the location of a minimum and adjusts the weights accordingly to converge to the minimum by considering successive values of the gradient of the error surface. RPROP [13] is a variable step size algorithm that updates the weight to adapt the learning process. SARPROP [14], which is modified from RPROP, employs Simulated Annealing (SA) [15, 16] to enhance the global convergence capability of RPROP. There are other modifications of RPROP to enhance its performance [17, 18]. The Levenberg-Marquardt algorithm (LM) [19] uses both the gradient descent method and the Gauss-Newton algorithm (GN) [20] to speed up the learning process. All these fast learning algorithms focus on how to increase the convergence rate of BP. However, their performance is limited by the local minimum problem, which means the learning process will be sometimes trapped in a local minimum and thus it cannot converge to the global minimum.

MGFPROP (Backpropagation with Magnified Gradient Function) was recently proposed in [21]. It magnifies the gradient functions of the activation functions to improve the convergence rate and the global convergence capability. Moreover, it does not violate the gradient-descent property of BP. In MGFPROP, the magnified gradient can effectively escape from a "flat spot". However, its performance is limited by the error overshooting problem (i.e., the step size of a fast learning algorithm is too large so that it overshoots the system error and thus it cannot smoothly converge to the global minimum) and the effect of the local minimum problem is reduced by cannot be totally eliminated.

The Enhanced Two-Phase Method (E2P) is proposed in [22] to solve the local minimum problem and the error overshooting problem. E2P can effectively identify them when one of these two problems exists, and hence assign suitable fast learning algorithms to speed up the learning process with a better global convergence capability. However, E2P still cannot completely eliminate the effect of these two problems. In most cases, when the first fast learning algorithm cannot escape from a local minimum or overshoot the system error, the second algorithm can solve it (i.e., after switching to the second fast learning algorithm, the learning process can converge to the global minimum). In some cases, however, one of these two problems occurs but neither fast learning algorithm in E2P can solve it, therefore the learning process still cannot converge properly. A systematic approach is proposed in [23] to assign different fast learning algorithms to the learning process so that it can converge to the global minimum, but the choices to select different fast learning algorithms are highly dependent on the nature of its application.

This paper studies the performance of Quickprop, one of the most popular fast learning algorithms. The drawbacks of this fast learning algorithm are discussed and some modifications are made to overcome those drawbacks. In our

This work was supported in part by Project 09/1.3 from "The OUHK Research and Development Fund" and in part by Project A-PC0K from "The Hong Kong Polytechnic University Competitive Research Grants for Newly Recruited Junior Academic Staff 2007-2008".

Chi-Chung Cheung is with Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong, China (e-mail: encccl@polyu.edu.hk).

Sin-Chun Ng and Andrew Lui are with the School of Science and Technology, The Open University of Hong Kong.

simulation experiments, the Quickprop algorithm with such modifications always converges with a faster learning rate in three popular complicated applications, whereas the original Quickprop algorithm gives very poor global convergence capabilities in these applications.

The paper is organized as follows. Section II describes the basic operations of the Quickprop algorithm. Section III discusses the drawbacks of the Quickprop algorithm and proposes some modifications to overcome the drawbacks. Section IV compares its performance with Quickprop and Quickprop with MGFPROP in three different applications. Finally, conclusions and future work are shown in Section V.

II. QUICKPROP

The Quickprop algorithm was developed in [12] to improve the convergence of BP. Instead of using the gradient function of BP and the first order derivative of the overall error to update the current weights, Quickprop uses successive values of the gradient of the error surface in the weight space to estimate the location of a minimum. It then changes the weights to move directly towards this minimum. The main assumptions of the Quickprop algorithm are that the error surface is concave and locally quadratic (see Fig. 1).

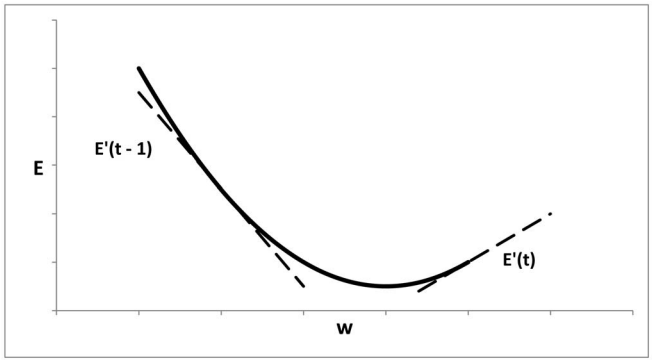

Fig. 1. The assumptions of the Quickprop algorithm

Based on these assumptions, the weight update rule of the algorithm is:

$$\Delta\omega_{ij}(t) = \frac{E'(t)}{E'(t-1) - E'(t)}\Delta\omega_{ij}(t-1), \qquad (1)$$

where $\Delta\omega_{ij}(t)$ is the change of the weight connecting the i^{th} and j^{th} nodes in a neural network at the t^{th} iteration, $E(t)$ is the system error at t^{th} iteration, and $E'(t) = \partial E(t)/\partial\omega_{ij}(t)$. In Equation (1), the Quickprop algorithm will take an infinite step and the network will behave chaotically when the difference between $E'(t-1)$ and $E'(t)$ is very small. To avoid it happening, a parameter called the maximum growth factor, ϕ, is introduced to limit the growth of a step size, i.e.,

$$\Delta\omega_{ij}(t) = \max\left(\phi, \frac{E'(t)}{E'(t-1) - E'(t)}\right)\Delta\omega_{ij}(t-1). \qquad (2)$$

The value of ϕ can significantly affect the performance of Quickprop: if it is too small, the step to converge to the global minimum is bounded by $\phi\Delta\omega_{ij}(t-1)$, which may also be too small; if it is too large, the network may become unstable and fail to converge. The default value of ϕ in [12] is 1.75 and it works well in different applications.

III. IMPROVING THE QUICKPROP ALGORITHM

Two main drawbacks exist in the Quickprop algorithm: the limitation of the gradient of the error surface, and the local minimum problem.

In the Quickprop algorithm, the original weight update rule of BP will still be followed and added to the value computed by Equation (2):

$$\Delta\omega_{ij}(t) = \mu E'(t) + \alpha\Delta\omega_{ij}(t-1) \qquad (3)$$

where μ is the learning rate and α is the momentum. Thus, the gradient of the error surface is still highly related to the standard BP algorithm. Hence, one possible way to increase the convergence rate of the Quickprop algorithm is to increase the value of the gradient of the error surface. Fig. 2 below can be used to illustrate this point. In this figure, there are two learning curves with different gradient values. At $(t-1)^{th}$ and t^{th} iterations, the magnitudes of the gradients of the error surface in learning curve B are greater than those in learning curve A (i.e., $\left|E'_B(t-1)\right| > \left|E'_A(t-1)\right|$ and $\left|E'_B(t)\right| > \left|E'_A(t)\right|$). When the Quickprop algorithm is applied and they both reach their minima, the minimum of learning curve B is lower than in learning curve A, which shows that the system error of a learning curve in the Quickprop algorithm can be further reduced by increasing the gradient of the error surface. Some fast learning algorithms that increase the gradient of the error surface were described in Section I. To speed up the convergence process, it is possible to combine the Quickprop algorithm with a fast learning algorithm which can increase the gradient of the error surface. To achieve this, it is essential to ensure that the operation of this fast learning algorithm does not disturb the basic operations of the original Quickprop algorithm.

Among all these kinds of fast learning algorithms, MGFPROP [21] satisfies the above requirements. MGFPROP magnifies the gradient functions of the activation functions of the standard BP algorithm without violating its gradient-descent property. The factor $o(1-o)$ is magnified by using a power factor $1/S$ where o is the output signal and S is a positive real number greater than or equal to 1 (i.e., $[o(1-o)]^{1/S}$). In [21], it was proved that the rate of change of the system error with respect to time for MGFPROP is always greater than that for BP, i.e.,

$$\left|\left(\frac{\partial E}{\partial t}\right)_{MGFPROP}\right| > \left|\left(\frac{\partial E}{\partial t}\right)_{BP}\right|. \qquad (4)$$

This means that the system error can be further reduced by using the Quickprop algorithm with MGFPROP. MGFPROP magnifies the gradient that is not related to any operations in the Quickprop algorithm, and hence they can coexist in the learning process without disturbing each other. The next section shows that the performance of the Quickprop

algorithm with MGFPROP is better than the Quickprop algorithm alone in terms of the convergence rate and the global convergence capability.

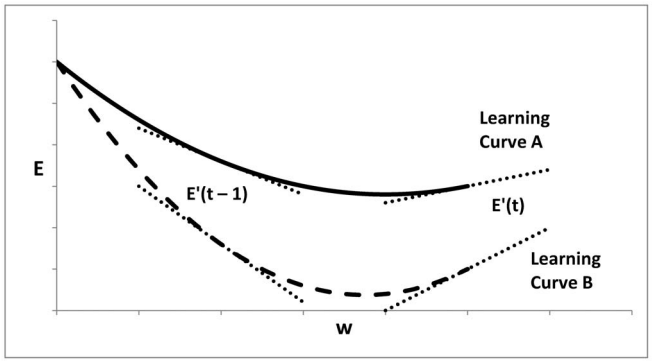

Fig. 2. The use of MGFPROP in the Quickprop algorithm

The second drawback is the local minimum problem. Quickprop always reaches the first minimum when it is found, whether it is a local minimum or the global minimum. That explains why it is easy to be trapped into a local minimum and its global convergence capability is very poor, especially when there are many local minima in a learning problem (application).

Figs. 3 and 4 show the learning process of Quickprop in the 5-bit Counting and the Wine applications (these applications will be described later in Section III). Each learning curve shows one typical run (out of the 100 runs) in our experiment. The 5-bit Counting application has a number of local minima and thus fast learning algorithms are easily trapped into those local minima and are slow to proceed further. In Fig. 3, Quickprop was very fast at the beginning and very close to the error threshold at around 400 iterations. Then it is trapped into a local minimum and finally cannot converge. The Wine application is a very complicated application such that it is difficult to converge quickly when using BP. In Fig. 4, Quickprop gets trapped into a local minimum at the 20th epoch and proceeds very slowly afterwards.

One possible way to solve the local minimum problem is to systematically search minima one by one until the global minimum is found. Consider the learning curve in Fig. 5. When the learning curves reaches X, there is a minimum on the left hand side. In this case, Quickprop always goes to such minimum because this minimum is located. However, there may be other minima on the right hand side and it does not know which one is the global minimum. The simplest way to handle it is to go to the minimum on the left hand side first. If it is not the global minimum, it starts again from the point X and goes to the right hand side to search for other minima. This procedure will be repeated until the global minimum is found. The idea of the searching is simple but its implementation is complicated. Many issues have to be considered in the implementation, e.g., the procedure to force the learning process to go to other minima when the minimum found before is a local minimum, some actions must be taken if the learning process is going back to the previous local minimum. All these issues will be discussed later in this section.

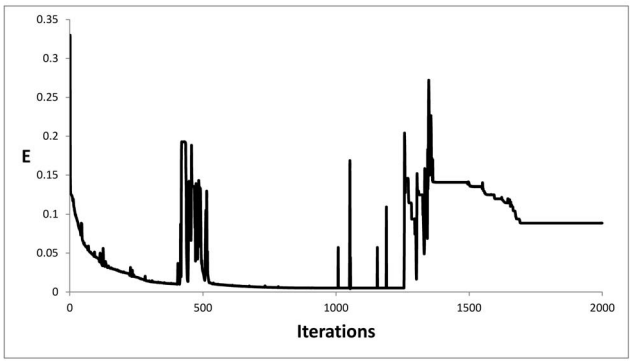

Fig. 3. The performance of Quickprop in the 5-bit counting problem.

Fig. 4. The performance of Quickprop in the wine application

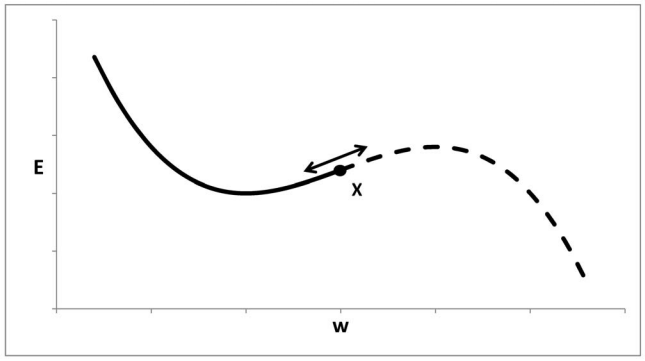

Fig. 5. The choice to reach a minimum

Based on the above discussion, a new algorithm called Better Quickprop (B-Quickprop) is proposed and listed in Fig. 6. Here MGFPROP is applied in the whole learning process with the Quickprop algorithm to increase the gradient of the error surface and hence increase the convergence rate. Step 2 is to store the current weights so that they can be restored later if the learning is trapped into a local minimum. In Step 3, if the change of the system error (i.e., ΔE) is very small, it is claimed that the learning curve is trapped into a local minimum. Note that the change of the system error is equal to the mean of the changes of the system error in the last 10 runs. Thus, a sudden change of the system error will not confuse the identification (i.e., it will not misidentify the existence of a local minimum). The equation in Step 3 is used to force the

learning curve to go to another minimum. If a stored weight is greater than its current value (i.e., the value of the weight when it is trapped into a local minimum), the stored weight is increased by the equation, and vice versa. The parameters Δ and λ are the constants to develop the expanding distance (i.e., the factor which times stored weights. It is used to change the value of a weight so that the difference between the updated weight and its current weight is larger than that between the original stored weight and its current weight.). The value of n is also related to the expanding distance. If the number of times that the local minimum has been visited increases, the expanding distance should also increase and it is hoped that this time the expanding distance is large enough to escape from the local minimum. The value of n is limited by N_{max} because the learning curve will diverge if n is too large and also the value of weights is too large. The procedure in Step 3 is to escape from the local minimum, but it may not work and the learning curve may still be going back to the a local minimum again. The objective of Step 4 is to take actions when this happens. In Step 4, the current weights are checked again because it is necessary to make sure that all of them do not keep going back to the previous local minimum. If it happens, all weights are updated to go further away from the local minimum.

IV. NUMERICAL RESULTS

This section reports a number of experiments that were conducted on three popular complicated applications — the 5-bit Counting, the Breast Cancer, and the Wine applications (data sets from the UCI Machine Learning Repository [23]) — to investigate the performance of Quickprop, Quickprop with MGFPROP and B-Quickprop. Brief descriptions of these three applications are shown in Table I where μ and α are the learning rate and the momentum of BP for these three applications respectively.

Note that M, H, and I represent the number of output, hidden, and input nodes respectively. The input and the target patterns for these three applications are binary (i.e., consist of 0s and 1s only). All fast learning algorithms in this paper were terminated when the mean square error E reached the error threshold 0.001 within 100,000 epochs, where

$$E = \sum_{p=1}^{P} \sum_{m=1}^{M} [t_{pm} - o_{pm}]^2 / PM . \qquad (5)$$

Here P is the total number of the input patterns. The initial weights were drawn randomly between -0.3 and 0.3. Each application was performed 100 times with 100 different sets of initial weights. The algorithm is implemented in a C program and the program is executed under Windows XP environment.

In the B-Quickprop algorithm, the values of T, Δ, N_{max}, λ, and d were set to -10^{-8}, 0.1, 5, 2, and 10 respectively. Note that these parameters are robustness for different learning applications. To apply MGFPROP into the Quickprop algorithm, it is considered to apply in the output layer only because it is found that the effect is better than that applied in both layers. The value of S in MGFPROP was set to 2, which is the default value in [21].

1. At the beginning, use MGFPROP and Quickprop.
2. When the first time Quickprop is used, record the values of all current weights $\omega^{(S)}_{ij}$.
3. Calculate the change of the system error, ΔE.
If $T \le \Delta E \le 0$, check the number of times that it has been visited (say n). Then apply the following equations to change the current weights ($\omega^{(L)}_{ij}$ is the current weight and ω_{ij} is the updated weight):
If ($\omega^{(S)}_{ij} > \omega^{(L)}_{ij}$) $k_{ij} \leftarrow \Delta$ else $k_{ij} \leftarrow -\Delta$
If ($n > N_{max}$) $n \leftarrow N_{max}$
$\omega_{ij} \leftarrow \omega^{(S)}_{ij} [1 + k_{ij} 2^{n/\lambda}]$
(T = a small threshold)
(ΔE = the change of the system error)
(Δ = the expanding constant)
(N_{max} = the maximum value of n)
(λ = the expanding step size)
(n is equal to 1 if this is the first time trapped into such local minimum)
4. After d iterations, the current weights are checked again. If the changes of all weights show that the learning process will be trapped into the same local minimum again, apply the following equations to change the current weights; otherwise, go to Step 2.
$n \leftarrow n + 1$
If ($n > N_{max}$) $n \leftarrow N_{max}$
$\omega_{ij} \leftarrow \omega^{(S)}_{ij} [1 + k_{ij} 2^{n/\lambda}]$

Fig. 6. The B-Quickprop algorithm

The performance comparisons among different fast learning algorithms in three different applications are shown in Table II. Note that the values shown in the second to the fourth column are the average number of epochs to converge over 100 runs, which is inversely proportional to the convergence rate. The values inside the bracket are the percentages of global convergence, which count the percentages of runs over 100 different runs that successfully converged to the system error of less than the error threshold. For example, the average number of epochs to converge and the percentage of global convergence of Quickprop in 5-bit Counting application are 479.90 and 63% respectively.

Table II shows that the convergence rate of Quickprop in the three applications was quite fast but its global convergence capability was poor. In the Breast Cancer application, the global convergence capability of Quickprop was extremely poor because there were many local minima in this application. By introducing MGFPROP into the Quickprop algorithm, the performance of the learning process was improved significantly in terms of the convergence rate and the global convergence capability. The improvement in the global convergence capability was very significant, especially in the Breast Cancer application, because MGFPROP not only increases the convergence rate by increasing the gradient of the error surface, it also solved the "flat spot" problem at the same time. On the other hand, the difference of the convergence rate between Quickprop and Quickprop with MGFPROP was not very significant — in the Breast Cancer application, the convergence rate of Quickprop with MGFPROP was only slightly greater than Quickprop alone. This is usually because the cases that cannot converge in Quickprop may spend more epochs to converge in Quickprop with MGFPROP, and thus the average number of epochs to converge increases in general. The difference in convergence rates between them will be large if only the cases that can converge into both algorithms are considered.

To compare the performance of Quickprop and B-Quickprop, the most significant difference was the global convergence capability — all cases in the three applications converged using B-Quickprop. In the Breast Cancer application, the percentage of global convergence increased from 4% to 100%. On the other hand, the convergence rate of B-Quickprop is greater than Quickprop, which can be easily identified in the 5-bit Counting problem. In the Breast Cancer and the Wine applications, however, the B-Quickprop convergence rate is slower than Quickprop because the cases that usually cannot converge in Quickprop may spend more epochs to converge in B-Quickprop, thereby increasing the average number of epochs to converge in general.

V. Conclusions and Future Work

This paper investigated the performance of the Quickprop algorithm. It showed that the Quickprop algorithm has two major drawbacks: the gradient of the error surface is small and the learning curve is easily trapped into a local minimum. A new algorithm called Better Quickprop (B-Quickprop) was proposed to overcome these two drawbacks: MGFPROP is introduced into the Quickprop algorithm to increase the gradient of the error surface and hence speed up the convergence process; and a new local minimum solving algorithm is developed so that the learning process can search minima one by one until the global minimum is found. The simulation results found that B-Quickprop always significantly outperforms the original Quickprop algorithm in terms of the convergence rate and the global convergence capability, and the global convergence capability in B-Quickprop has especially been improved very significantly. In the Breast Cancer application, the global convergence capability improved from 4% to 100%.

In the future, the effect of the parameters (i.e., T, Δ, N_{max}, λ, and d) on the performance of B-Quickprop will be studied. Additionally, B-Quickprop will be applied in other complicated applications to investigate its performance and compare it with the Quickprop algorithm.

TABLE I
PROBLEM DESCRIPTIONS

Application	Description	Network Architecture $I-H-M$	μ α
5-bit Counting	Count the number of 1s from the 5 input units [6].	5 – 12 – 6	0.1 0.7
Breast Cancer	These breast cancer databases were obtained from the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg. The databases reflect this chronological grouping of the data [23].	9 – 20 – 1	0.005 0.03
Wine	These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.	13 – 10 – 3	10^{-7} 0.1

TABLE II
PERFORMANCE COMPARISONS

Fast Learning Algorithm	5-bit Counting	Breast Cancer	Wine
Quickprop	479.90 (63%)	2366.00 (4%)	845.66 (38%)
Quickprop with MGFPROP	274.18 (98%)	2938.45 (66%)	672.03 (74%)
B-Quickprop	281.82 (100%)	6311.23 (100%)	2613.40 (100%)

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1. MIT Press, Cambridge, Mass, 1986.

[2] E. K. Blum, and L. K. Li, "Approximation theory and feedforward networks", *Neural Networks*, vol. 4, pp. 511 – 515, 1991.

[3] M. Gori, and A. Tesi, "On the problem of local minima in back-propagation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76 – 86, 1992.

[4] Y. Lee, S. H. Oh, and M. W. Kim, "An Analysis of Premature Saturation in Back Propagation Learning", *Neural Networks*, vol. 6, pp. 719 – 728, 1993.

[5] F. Stager, and M. Agarwal, "Three methods to speed up the training of feedforward and feedback perceptrons", *Neural Networks*, vol. 10, no. 8, pp. 1435 – 1443, 1997.

[6] A. Van Ooyen, and B. Nienhuis, “Improving the convergence of the back-propagation algorithm”, *Neural Networks*, vol. 5, pp. 465 – 471, 1992.

[7] J. E. Vitela, and J. Reifman, “Premature Saturation in Backpropagation Networks: Mechanism and Necessary Conditions”, *Neural Networks*, Vol. 10, no. 4, pp. 721 – 735, 1997.

[8] M. Ahmad and F. M. A. Salam, "Supervised learning using the cauchy energy function", *International Conference on Fuzzy Logic and Neural Networks*, 1992.

[9] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation", *Neural Networks*, Vol. 1, pp. 295 – 307, 1988.

[10] C. C. Yu and B. D. Liu, “A Backpropagation Algorithm with Adaptive Learning Rate and Momentum Coefficient”, *Proceedings of IJCNN’02*, vol. 2, pp. 1218 – 1223, 2002.

[11] X. H. Yu, G. A. Chen and S. X. Cheng, "Acceleration of backpropagation learning using optimised learning rate and momentum", *Electronics Letters*, Vol. 29, No. 14, pp. 1288-1289, 8th July 1993.

[12] S. E. Fahlman, “Fast learning variations on back-propagation: An empirical study”, *Proceedings of the 1988 Connectionist Models Summer School* (Pittsburgh, 1988), D. Touretzky, G. Hinton, and T. Sejnowski, eds., pp. 38 – 51, Morgan Kaufmann, San Mateo, California, 1989.

[13] M. Riedmiller, and H. Braun, “A direct adaptive method for faster back-propagation learning: The RPROP Algorithm”, *Proceedings of International Conference on Neural Networks*, vol. 1, pp. 586 – 591, 1993.

[14] N. K. Treadgold, and T. D. Gedeon, “Simulated Annealing and Weight Decay in Adaptive Learning: The SARPROP Algorithm”, *IEEE Trans. on Neural Networks*, vol. 9, no. 4, pp. 662 – 668, July 1998.

[15] S. Kirkpatrick, “Optimization by simulated annealing: Quantitative studies”, *Journal of Statistical Physics*, vol. 34, pp. 975 – 986, 1984.

[16] S. Kirkpatrick, C. D. Gilatt, and M. P. Vecchi, “Optimization by simulated annealing”, *Science*, vol. 220, pp. 671 – 680, 1983.

[17] C. Igel and M. Hüsken, “Empirical evaluation of the improved Rprop learning algorithms”, *Neurocomputing*, (50), pp. 105 – 123, 2003.

[18] A. D. Anastasiadis, G. D. Magoulas, and M. N. Vrahatis, “An Efficient Improvement of the Rprop Algorithm”, *Proceedings of the First International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR-03)*, 2003.

[19] Y. Bard, *Non-Linear Parameter Estimation*, New York: Academic Press, 1974.

[20] S. Haykin, “Single-layer perceptrons”, *Neural Networks: A Comprehensive Foundation*, 2nd ed., London: Prentice Hall, 1999, Ch.3, pp. 117 – 155.

[21] S. C. Ng, Chi-Chung Cheung and S. H. Leung, "Magnified Gradient Function with Deterministic Weight Evolution in Adaptive Learning", page 1411 – 1423, *IEEE Transactions in Neural Networks*, vol. 15, No. 6, November 2004.

[22] Chi-Chung Cheung, S. C. Ng, A. K. Lui, and Sean Shensheng, “Enhanced Two-Phase Method in Fast Learning Algorithms”, *Proceedings of IJCNN 2010*, Barcelona, Spain, July 2010.

[23] Chi-Chung Cheung, S. C. Ng, A. K. Lui, and Sean Shensheng, “A Fast Learning Algorithm with Promising Convergence Capability”, *Proceedings of IJCNN2011*, San Jose, California, USA, July, 2011.

[24] A. Frank and A. Asuncion, ‘‘UCI machine learning repository,’’ University of California, Irvine, School of Information and Computer Sciences, 2010. [Online].
Available: http://archive.ics.uci.edu/ml/