A Systematic Algorithm to Escape from Local Minima in Training Feed-Forward Neural Networks

Chi-Chung Cheung, Sean Shensheng Xu, and Sin-Chun Ng

***Abstract*— A learning process is easily trapped into a local minimum when training multi-layer feed-forward neural networks. An algorithm called Wrong Output Modification (WOM) was proposed to help a learning process escape from local minima, but WOM still cannot totally solve the local minimum problem. Moreover, there is no performance analysis to show that the learning has a higher probability of converging to a global solution by using this algorithm. Additionally, the generalization performance of this algorithm was not investigated when the early stopping method of training is applied. Based on these limitations of WOM, we propose a new algorithm to ensure the learning can escape from local minima, and its performance is analyzed. We also test the generalization performance of this new algorithm when the early stopping method of training is applied.**

I. Introduction

The Backpropagation (BP) algorithm [1] is the most popular training algorithm and it is extensively applied in training multi-layered feed-forward neural networks. BP is popular because it is simple and its computational complexity is low. However, its convergence rate (the rate to converge to a global solution) is slow and it is easily trapped into local minima, especially for non-linearly separable problems such as the exclusive OR (XOR) learning problem [2, 3]. When the learning is trapped into a local minimum, it cannot escape afterwards to reach a global minimum (solution). This is known as the local minimum problem.

Many learning algorithms based on BP have been proposed to improve the performance of BP in terms of the convergence rate (i.e., speed up the learning process); however, they seldom directly address the local minimum problem [8 – 17].

Recently, [29] proposed a systematic approach called Wrong Output Modification (WOM) to help the learning escape from a local minimum if the learning is trapped by the local minimum. [30] enhances this algorithm by modifying the procedure to escape from a local minimum more effectively and introduces a fast checking procedure to quickly identify whether the learning is moving back to previous local minima. Through the performance investigation in [30], the enhanced version of WOM can effectively escape from local minima. However, the learning may still be trapped occasionally into a local minimum from which it cannot escape using WOM. Moreover, there is no performance analysis to show that WOM can increase the probability of converging to a global solution. Finally the early stopping method of training is a very popular method to overcome the overfitting problem but the performance of WOM with this method was not investigated.

Based on the above limitations, we propose a new algorithm called WOM with Re-attempts (WOM-R). When the learning cannot escape from a local minimum using WOM, a new set of initial weights is generated and the learning follows the new set of initial weights to converge to a global solution. The performance investigation shows it is a very effective way to escape from such a local minimum. According to the performance investigation, the learning uses re-attempt once at most, and it is good enough to escape from all local minima for different learning algorithms in different learning problems. We also studied the performance analysis of WOM and WOM-R, which showed that a learning algorithm with WOM-R can finally converge to a global solution. Finally, our proposed algorithm WOM-R was applied in the early stopping method of training and our performance investigation showed that it can enhance the generalization performance of some learning algorithms in some learning problems.

This paper is organized as follows. Section II briefly describes the WOM algorithm and presents its performance analysis. Section III shows the proposed algorithm called WOM-R with its performance analysis. Section IV compares the performance of our proposed algorithm with WOM and other popular learning algorithms in terms of the convergence rate and the global convergence capability, and the generalization performance of WOM-R is investigated when the early stopping method of training is used. Section V draws the conclusion.

II. WOM

The WOM algorithm (Wrong Output Modification) [30] is shown in Fig. 1. The procedure *Initialization* is to set up parameters and select a learning algorithm (e.g., BP) to process the learning. The learning starts after initialization. If it is trapped by a local minimum, the procedure *Status-Checking* is executed to check the current status and the procedure *Esacpe* is executed to escape from this local minimum. The procedure *Fast-Checking* is processed to avoid the learning going back to previous local minima (not only the previous local minimum but all local minima that were visited before), if any. They will be repeatedly executed until the learning converges to a global solution. Note that the learning

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

Sin-Chun Ng is with the School of Science and Technology, The Open University of Hong Kong, Hong Kong, China.

is said to converge to a global solution if the system error is less than or equal to an error threshold. To checks whether the learning is trapped by a local minimum, WOM monitors the change of the system error and the system error. The learning is declared to be trapped by a local minimum if the change of the system error is very close to zero but the system error is larger than the error threshold.

```
Process the procedure Initialization
Randomly generate a set of initial weights.
Start the learning.
Repeat
  // Forward operation
  Compute the outputs by using the neuron weights.
  If the learning is trapped by a local minimum, then
     // Check the current status
     Process the procedure Status-Checking
     // Require to escape from such local minimum
     If (Escape = true), then
        // Escape from the local minimum
        Process the procedure Escape.
     Endif
  Endif
  // Require to check wehther it goes back to previous
  // local minimum
  If (Checking = true), then
     Process the procedure Fast-Checking.
  Compute the neuron weights by using the outputs.
  Endif
Until the learning converges to a global solution
```

Fig. 1. WOM algorithm.

The procedure *Status-Checking* is shown in Fig. 2. It is used to determine the parameters Γ and α. Γ is a threshold to determine which outputs are declared as wrong outputs. At the beginning, Γ is small to avoid high instability. When the learning cannot escape from a local minimum, it will decrease to declare more outputs as wrong outputs. The minimum value of Γ is 0.5. α is to determine how close that a wrong output is moved to its corresponding desired output. At the beginning, it is small to avoid high instability. When the learning cannot escape from a local minimum, it will descrease so that a wrong output can move closer to its corresponding desired output. The minimum value of α is a very small real value.

The procedure *Escape* shown in Fig. 3 is used to modify wrong output values when the learning is trapped by a local minimum. It is expected that, after modification, the new output value can push the learning to esacpe from the local minimum and converge to a global solution.

The procedure *Fast-Checking* is shown in Fig. 4. When the learning executes the procedure *Escape* to escape from a local minimum, it needs to monitor the system error and change of system error to identify whether it is going back to a previous local minimum. It usually takes many iterations to identify this and the convergence rate may become too slow. This procedure can speed up the checking by monitoring the change of neuron weights. If most neuron weights are approaching a local minimum, the probability that the learning will be trapped by this local minimum is very high. Through the performance investigation, Ψ is sufficiently large if it is set to $0.8L$ for all learning algorithms in all learning problems. Note that WOM can be implemented in any learning algorithms because the operations of WOM are independent of the operations of a learning algorithm.

```
Procedure Status-Checking
Begin
  If it is a new local minimum, then
     Use default parameter setting.
     Escape = true.
     Checking = true;
  Else
     If the local minimum was visited before, then
        α ← α – ε_k. // Move closer to desired outputs
        Escape = true.
        Checking = true.
     Else
        If α is too small but it still cannot esacpe, then
           Γ ← Γ – ε_Γ. // Involve more outputs
           Escape = true.
           Checking = true.
        Endif
     Endif
  Endif
End
```

Fig. 2. Procedure *Status-Checking*.

```
Procedure Escape
Begin
  For each output values,
     Δ = Wrong output value – Desired output value
     If (Δ > Γ)
        Output = α if desired output value is 0
        Output = 1 – α if desired output value is 1
  Endfor
End
```

Fig. 3. Procedure *Escape*.

```
Procedure Fast-Checking
Begin
  After each β iterations,
     For k = 1, 2, …, K, // number of local minima
        Count ← 0;
        For l = 1, 2, …, L, // number of neural weights
           If this weight approaches to k^th local minimum
              Count = Count + 1;
           Endif
        Endfor
        If (Count ≥ Ψ), process the procedure Escape.
        Endif
     Endfor
End
```

Fig. 4. Procedure *Fast-Checking*.

The performance study of WOM is shown here. For simplicity, we investigated the probability of converging to a global solution when the learning escapes from a local minimum. The notations and definitions of symbols used in the performance study are listed below:

- ω_i : i th neuron weight
- $\omega_i(k)$: i th neuron weight at k th iteration
- $R_i = \{\omega_i : \max(\omega_i) \geq \omega_i \geq \min(\omega_i)\}$: the valid range of i th neuron weight, where max(x) and min(x) are the maximum and minimum acceptable values of x respectively. If x is out of this range, the learning is declared to diverge.
- $\underline{\omega} = (\omega_1, \omega_2, \ldots, \omega_L) \in \underline{R}$: a vector of all neuron weights in a neural network and $\underline{R}$ is the set of all valid vectors ($\underline{R} = \{\underline{\omega} : \omega_i \in R_i, \forall i\}$).
- $\underline{\omega}(k) = (\omega_1(k), \omega_2(k), \ldots, \omega_L(k)) \in \underline{R}$: a vector of all neuron weights in a neural network at k th iteration
- $R_{i,j,m}^{(S)} = \{\omega_i : \max(\omega_{i,j,m}^{(S)}) \geq \omega_i \geq \min(\omega_{i,j,m}^{(S)})\}$: the range of i th neuron weight in the m th region of the j th solution (a set of neuron weights is declared to be a solution if its MSE (Mean Square Error) can reduce to be less than or equal to the error threshold within the maximum number of iterations (epochs) allowed) where $\max(\omega_{i,j,m}^{(S)})$ and $\min(\omega_{i,j,m}^{(S)})$ are the maximum and minimum values of the range
- $\underline{R}_{j,m}^{(S)} = \{\underline{\omega} : \omega_i \in R_{i,j,m}^{(S)}, \forall i\}$: the set of vectors of neuron weights in m th region of the j th solution
- $\underline{R}_j^{(S)} = \bigcup_m \underline{R}_{j,m}^{(S)}$: the set of vectors of neuron weights in the j th solution.
- $\underline{R}^{(S)} = \bigcup_j \underline{R}_j^{(S)}$: the set of vectors of neuron weights which can converge to a solution.
- $\|x\|$ is the norm of x such that

$$\|R_i\| = \max(\omega_i) - \min(\omega_i) \tag{1}$$

Thus we have

$$\|\underline{R}\| = \prod_i \|R_i\|, \tag{2}$$

$$\left\|R_{i,j,m}^{(S)}\right\| = \max(\omega_{i,j,m}^{(S)}) - \min(\omega_{i,j,m}^{(S)}), \tag{3}$$

$$\left\|\underline{R}_{j,m}^{(S)}\right\| = \prod_i \left\|R_{i,j,m}^{(S)}\right\|, \tag{4}$$

$$\left\|\underline{R}_j^{(S)}\right\| = \sum_m \left\|\underline{R}_{j,m}^{(S)}\right\| \tag{5}$$

and $$\left\|\underline{R}^{(S)}\right\| = \sum_j \left\|\underline{R}_j^{(S)}\right\|. \tag{6}$$

Let $p^{(S)}$ be the probability of converging to a global solution within the maximum number of iterations allowed where

$$p^{(S)} = \frac{\left\|\underline{R}^{(S)}\right\|}{\|\underline{R}\|}. \tag{7}$$

When the learning is trapped by a local minimum at the K th iteration (before converging to a solution), we have

$$\underline{\omega}(k) \in \underline{R} - \underline{R}^{(S)} \text{ for } k = 1, 2, \ldots, K. \tag{8}$$

When WOM is applied, $\underline{\omega}(K+1) \neq \underline{\omega}(K)$. Some $\underline{\omega}^*(k)$ exist such that $\underline{\omega}^*(k) \in \underline{R} - \underline{R}^{(S)}$ for k = 1, 2, …, K but $\underline{\omega}^*(K+1) \in \underline{R}^{(S)}$. Thus, $\underline{\omega}^*(k) \in \underline{R}^{(S,WOM)}$ for all k where $\underline{R}^{(S,WOM)}$ is $\underline{R}^{(S)}$ when WOM is applied in learning. Moreover, $R_{i,j,m}^{(S,WOM)}$, $\underline{R}_{j,m}^{(S,WOM)}$ and $\underline{R}_j^{(S,WOM)}$ are $R_{i,j,m}^{(S)}$, $\underline{R}_{j,m}^{(S)}$ and $\underline{R}_j^{(S)}$ respectively when WOM is applied in learning. Therefore, there exist some m* and j* such that

$$\left\|R_{i,j^*,m^*}^{(S,WOM)}\right\| > \left\|R_{i,j^*,m^*}^{(S)}\right\|. \tag{9}$$

Thus we have

$$\begin{aligned} \left\|\underline{R}_{j^*,m^*}^{(S,WOM)}\right\| > \left\|\underline{R}_{j^*,m^*}^{(S)}\right\| \quad &\Rightarrow \quad \left\|\underline{R}_{j^*}^{(S,WOM)}\right\| > \left\|\underline{R}_{j^*}^{(S)}\right\| \\ &\Rightarrow \quad \left\|\underline{R}^{(S,WOM)}\right\| > \left\|\underline{R}^{(S)}\right\| \\ &\Rightarrow \quad p^{(S,WOM)} > p^{(S)} \end{aligned} \tag{10}$$

where $p^{(S,WOM)}$ is the probability of converging to a global solution when WOM is applied. Through the above performance study, this probability is greater than the probability of convergeing to a global solution in the learning without WOM.

III. WOM-R

The description and the performance study of WOM in Section II shows that the probability of a learning algorithm converging to a global solution increases when WOM is applied. However, it still cannot guarantee that the learning can converge to a global solution. In our performance investigation, the learning occasionally cannot escape from a local minimum when all wrong outputs are modified and moved sufficiently close to their corresponding target outputs. One possible reason is that the global solution is too far away from the local minimum, so it is not good enough to just modify wrong outputs but it is necessary to change the value of each neuron weight. This reason will be justified later in Section IV.

In this case, a new procedure is required to make the change. The procedure is called WOM with Re-attempts (WOM-R) and the algorithm is described in Fig. 5.

```
Procedure Status-Checking-With-Re-attempts
Begin
  If it is a new local minimum, then
    Use default parameter setting.
    Escape = true.
    Checking = true.
  Else
    If the local minimum was visited before, then
      α ← α – εk. // Move closer to desired outputs
      Escape = true.
      Checking = true.
    Else
      If α is too small but it still cannot esacpe, then
        Γ ← Γ – εΓ. // Involve more outputs
        Escape = true.
        Checking = true;
      Else // Whatever the learning cannot escape
        Randomly re-generate a new set of initial
        weights but the regions in which the learning
        is trapped by the local minima which were
        visited before are excluded.
        (The step in WOM algorithm "Compute the
        neuron weights by using the outputs." can be
        skipped once when it happens.)
        Escape = false.
        Checking = false.
      Endif
    Endif
  Endif
End
```

Fig. 5. Procedure *Status-Checking-With-Re-attempts.*

The procedure *Status-Checking* in WOM is replaced by the procedure *Status-Checking-With-Re-attempts*. In this new procedure, when all wrong outputs are modified and they are very close to their corresponding target outputs but the learning is still trapped by a local minimum, it is declared that modifying wrong outputs only is not good enough to escape from the local minimum. Instead, all neuron weights are modified. One possible way is to randomly re-generate all weights. However, that may lead to the learning moving to such a local minimum again since the attractive region of the local minimum may be very large. To eliminate this possibility effectively, all neuron weights are randomly re-generated but the regions in which the learning is trapped by local minima that were visited before are excluded. Later, in Section IV, our performance investigation will show that at most this re-generation needs to be executed once only, and then the learning can converge to a global solution. It shows that WOM-R is an efficient way to escape from a local minimum when the original WOM cannot escape from it.

The performance study of WOM-R is shown here. The notations and definitions of symbols used in the performance study are listed below:

- $\omega_i(k,n)$: i^{th} neuron weight at k^{th} iteration after n^{th} retries
- $k^{(n)}$: the number of iterations that the learning is trapped by a local minimum after n^{th} retries
- $R_i^{(L)}(n) = \{\omega_i : MAX \geq \omega_i \geq MIN\}$: the range of i^{th} neuron weight such that the learning is trapped by a local minimum after n^{th} retries where $MAX = \max(\omega_i(k^{(n)},n), \omega_i(0,n))$ and $MIN = \min(\omega_i(k^{(n)},n), \omega_i(0,n))$.
- $R_i(n+1) = R_i(n) - R_i^{(L)}(n)$: the valid range of i^{th} neuron weight after n^{th} retries
- $\underline{R}(n) = \{\underline{\omega} : \omega_i \in R_i(n), \forall i\}$

Thus we have $\|R_i(n+1)\| = \|R_i(n)\| - \|R_i^{(L)}(n)\| < \|R_i(n)\|$ and $\|\underline{R}(n+1)\| < \|\underline{R}(n)\|$. Therefore,

$$p^{(S,WOM)}(n+1) = \frac{\|\underline{R}^{(S,WOM)}\|}{\|\underline{R}(n+1)\|} > \frac{\|\underline{R}^{(S,WOM)}\|}{\|\underline{R}(n)\|} = p^{(S,WOM)}(n) \quad (11)$$

and

$$p^{(S,WOM-R)} = 1 - \prod_n (1 - p^{(S,WOM)}(n)) > 1 - (1 - p^{(S,WOM)})^n \to 1 \quad (12)$$

when $n \to \infty$. Note that $p^{(S,WOM)}(n)$ is the probability of converging to a global solution after n retires and $p^{(S,WOM)}(0) = p^{(S,WOM)}$. The performance analysis shows the probability is close to 1 when n is large although, in our performance investigation, it is good enough to re-generate a new set of initial weights at most once.

IV. Numerical Results

This section reports a number of experiments that were conducted using different benchmark learning problems to investigate the performance of different popular learning algorithms with and without WOM-R. All data sets of learning problems can be found in the UCI Machine Learning Repository [25]. Brief descriptions of these learning problems are shown in Tables I and II where μ and α are the learning rate (step size) and the momentum of BP for those learning problems respectively. Table II shows the number of data patterns used for training, validation and testing when the early stopping method of training is applied.

TABLE I
PROBLEM DESCRIPTIONS I

Learning Problem	Description	Network Architecture (N, K ,M)	Parameter Setting (μ, α)
Iris	The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.	(4, 15, 3)	(0.02, 0.05)
Wine	These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from 3 different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.	(13, 10, 3)	(10^{-8}, 0.1)
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.	(9, 20, 1)	(0.005,0.03)

TABLE II
PROBLEM DESCRIPTIONS II

Learning Problem	Training	Validation	Testing
Iris	100	10	40
Wine	118	20	40
Breast Cancer	99	200	400

Note that N, K, and M represent the number of input, hidden, and output nodes respectively. The network architectures are obtained from other research papers or by trial-by-error. The input and the target patterns for these learning problems consist of 0s and 1s only. All learning algorithms in this paper were terminated when the mean square error E reached the error threshold 0.001 within 1,000,000 epochs.

Four popular learning algorithms (Backpropagation (BP) [1], MGFPROP (MGF) [13], Quickprop (QP) [4] and RPROP (RP) [5]) were applied to the above learning problems to investigate the performance of WOM-R in terms of the convergence rate (the number of epochs) and the global convergence capability (the probability of converging to a global solution within 100 runs). A set of initial weights is randomly generated between -0.3 and 0.3. Finally, all simulations are implemented in C and executed in identical personal computers with the same hardware and software environments.

Table III shows the performance of different learning algorithms with and without WOM-R in different learning problems. The values shown outside and inside the brackets are the convergence rate (i.e., the average number of iterations to converge to a global solution) and the global convergence capability (i.e., the percentage of runs in which the learning converges to a global solution) respectively. For example, Backpropagation (BP) without WOM-R in the Iris problem spends 559,337 iterations on average to converge to a global solution and it can converge to a global solution in 14 out of 100 runs. Note that there are no data for Quickprop in the Iris problem because the probability of converging to a global solution is too small so that the learning cannot meet its terminating condition in all 100 runs. Moreover, there are no data for both BP and MGFPROP in the Wine problem because their learning rates are extremely small when the learning is close to converging to a global solution and thus they never meet the terminating condition in all 100 runs.

Table III, except for the above special cases, shows that all learning algorithms can converge to a global solution in all runs when WOM-R is applied. In [30], Quickprop and RPROP can converge to a global solution in all runs without re-generating a new set of initial weights because it was found that they can finally converge to a global solution if it keeps trying to escape from local minima. However, other learning algorithms like BP and MGFPROP cannot converge to a global solution without re-generating a new set of initial weights, which was the motivation why a new procedure was required to escape from a local minimum when the original WOM cannot do it.

TABLE III
PERFORMANCE COMPARISONS

Learning Algorithms	Iris	Wine	Breast Cancer
BP	559,337 (14%)	null (0%)	130,513 (47%)
BP + WOM-R	129,891 (100%)	null (0%)	12,477 (100%)
MGF	39,159 (100%)	null (0%)	7,829 (99%)
MGF + WOM-R	44,210 (100%)	null (0%)	8,619 (100%)
QP	null (0%)	846 (38%)	2,366 (4%)
QP + WOM-R	132,699 (100%)	7,378 (100%)	3,221 (100%)
RP	4,046 (100%)	1.453 (60%)	2,151 (2%)
RP + WOM-R	24,274 (100%)	1,966 (100%)	1,742 (100%)

Table IV shows the the difference between local minima and global solutions. $D(j)$ is the normalized mean difference between the j^{th} local minimum and the global solution that the learning finally converges to. The definition of $D(j)$ is

$$D(j) = \frac{1}{L}\sum_{i=1}^{L}\left|\frac{\omega_i(j,LM) - \omega_i(GM)}{\omega_i(GM)}\right| \quad (13)$$

Note that L is the total number of neurons. Moreover,

$\omega_i(j,LM)$ is the i th neuron weight when the learning is trapped by j th local minimum, and $\omega_i(GM)$ is the i th neuron weight when the learning finally converges to a global minimum (solution). The number inside the bracket is the sequence to identify local minima. A number with * represents that the procedure to re-generate a new set of initial weights is executed when the learning is trapped by such local minimum. For example, in case 3, the first local minimum is found and the normalized mean difference between this local minimum and the final global solution is 89.38. After escaping from the first local minimum, the learning is trapped into the second local minimum and this time the procedure to re-generate a new set of initial weight is executed. After that, the learning is trapped into the third local minimum. Finally, the learning escapes from the third local minimum and converges to a global solution. In case 1 and 2, the original WOM can escape from all local minima because the differences are very small (less than 1). In cases 3, 4 and 5, the procedure to re-generate a new set of initial weights is required because the differences are large (greater than 7.95 and up to 261.31). This means, when the local minima are very far away from its global solution, it becomes necessary to execute this new procedure.

TABLE IV
DIFFERENCE BETWEEN LOCAL MINIMA AND GLOBAL SOLUTIONS

Case	Learning algorithm	Learning Problem	$D(j)$
1	Quickprop	Wine	$D(1) = 0.21$
2	RPROP	Breast Cancer	$D(1) = 0.97$ $D(2) = 0.43$
3	Quickprop	Wine	$D(1) = 89.38$ $D(2) = 261.31$* $D(3) = 0.02$
4	RPROP	Wine	$D(1) = 7.95$ $D(2) = 8.77$ $D(3) = 9.34$ $D(4) = 8.81$*
5	Quickprop	Iris	$D(1) = 26.31$ $D(2) = 26.26$* $D(3) = 1.91$ $D(4) = 0.47$ $D(5) = 0.084$

Finally the generalization performance of WOM-R was investigated when the early stopping method of training is applied. This method is very popular because it generally solves the overfitting problem (see Fig. 6) [31]. The early-stopping point is used to get the set of neuron weights for generalization .

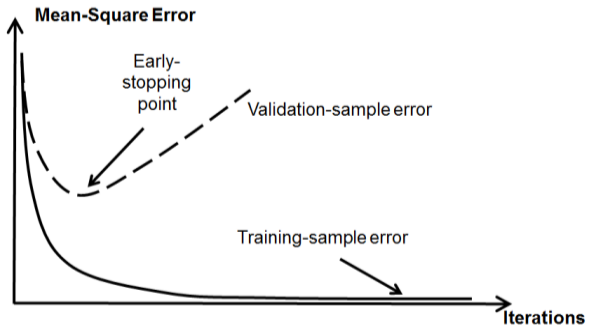

Fig. 6 Early stopping method of training [31]

Table V shows the the generalization performance of different learning algorithms in different learning problems. For each benchmark learning problem, 10,000 runs were executed and the result is the mean mis-classification rate of a learning algorithm with and without WOM-R. A learning algorithm with WOM-R sometimes has better generalization performance. It happens only when the learning is trapped by a local minimum before it reaches the minimum of the curve of the validation-sample error. If that is the case, WOM-R can esacpe from a local minimum and then the learning can reach the minimum of the curve and therefore obtain a better generalization performance.

TABLE V
GENERALIZATION PERFORMANCE

Learning Problem	QP	QP + WOM-R	RP	RP + WOM-R
Iris	0.04795	0.04778	0.04718	0.04613
Wine	0.1414	0.1173	0.1075	0.1003
Breast Cancer	0.04333	0.04334	0.04227	0.04222

V. CONCLUSIONS

This paper presented a new algorithm called Wrong Output Modification with Re-attempts (WOM-R) in training multi-layered feed-forward neural networks. It systemically modify the outputs and the neuron weights to escape from local minima, if any. When all wrong outputs are declared and modified but the learning still cannot escape from a local minimum, WOM-R randomly re-generates all neuron weights but the regions in which the learning is trapped by local minima that were visited before are excluded. Moreover, it showed the performance study of this algorithm, and it investigated the performance of some popular learning algorithms with WOM-R in some brenchmark learning problems. Learning with WOM-R always converged to a global solution in our performance investigation and sometimes produced better generalization performance.

ACKNOWLEDGEMENT

The authors would like to thank Dr. Rex G. Sharman for his proof-reading. This work was supported in part by Project A-PC0K from "The Hong Kong Polytechnic University Competitive Research Grants for Newly Recruited Junior Academic Staff 2007-2008" and in part by The Open University of Hong Kong Research Grant (Project No. 09/1.3 and 2012/1.1).

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] 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.

[9] 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.

[10] 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.

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

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

[13] C. Igel and M. Hüsken, "Empirical evaluation of the improved Rprop learning algorithms", *Neurocomputing*, vol. 50, pp. 105 – 123, 2003.

[14] 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.

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

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

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

[18] 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.

[19] Chi-Chung Cheung, S. C. Ng, A. K. Lui, and Sean Shensheng, "A Fast Learning Algorithm with Promising Convergence Capability", *Proceedings of IJCNN 2011*, San Jose, US, August 2011.

[20] L. Behera, "On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks", *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1116 – 1125, September 2006.

[21] X. Yu, O. Efe, and O. Kaynak, "A General Backpropagation Algorithm for Feedforward Neural Networks Learning", *IEEE Transactions on Neural Networking*, vol. 13, no. 1, pp. 251 – 254, January 2002.

[22] M. T. Hagan and M. B. Menhaj, "Training feedforward neural networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, pp. 989 – 993, November, 1994.

[23] A. Petrowski, "Performance analysis of a pipelined backpropagation parallel algorithm", *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 970 – 981, November 1993.

[24] S. L. Hung and H. Adeli, "A Parallel Genetic/Neural Network Learning Algorithm for MIMD Shared Memory Machines" *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 900 – 909, November 1994.
A. Asuncion and D. J. Newman, "UCI machine learning repository," University of California, Irvine, School of Information and Computer Sciences, 2007. [Online]. Available: http://archive.ics.uci.edu/ml/

[25] G. Huang, Q. Zhu and C. Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, no. 70, pp. 489 – 501, 2006.

[26] J. Luo, C. Vong and P. Wong, "Sparse Bayesian Extreme Learning Machine for Multi-classification", IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 4, pp. 836 – 843, April 2014.

[27] S. C. Ng, Chi-Chung Cheung, A. K. Lui and H. Tse, "Addressing the Local Minima Problem by Output Monitoring and Modification Algorithms", *Advance in Neural Networks – ISNN 2012*, *Lecture Notes in Computer Science*, vol. 7367, pp. 206 – 216, 2012.

[28] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, June 1999.

[29] Chi-Chung Cheung, Sin-Chun Ng, A. K Lui and Sean Shensheng Xu, "Solving the Local Minimum and Flat-Spot Problem by Modifying Wrong Outputs for Feed-Forward Neural Networks", in *Proceedings of IJCNN2013*, pp. 1463 – 1469, Dallas, Texas, USA, August 2013.

[30] Chi-Chung Cheung, Sin Chun Ng, Andrew K. Lui, Sean Shensheng Xu, "Further enhancements in WOM algorithm to solve the local minimum and flat-spot problem in feed-forward neural networks", *in Proceedings of IJCNN2014*, pp. 1225-1230, Bejing, China, July 2014.

[31] S. Haykin, *Neural networks and learning machines*, 3rd edition, Pearson, 2009.