Improving generalization capability of neural networks based on simulated annealing

CONFERENCE PAPER · JANUARY 2007

DOI: 10.1109/CEC.2007.4424918 · Source: DBLP

4 AUTHORS, INCLUDING:

Jong-Seok Lee
Yonsei University
74 PUBLICATIONS **534** CITATIONS
SEE PROFILE

Sun Young Lee
Korea Advanced Institute of Science and Tec…
3 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Improving Generalization Capability of Neural Networks based on Simulated Annealing

Yeejin Lee, Jong-Seok Lee, Sun-Young Lee, and Cheol Hoon Park

Abstract— **This paper presents a single-objective and a multiobjective stochastic optimization algorithms for global training of neural networks based on simulated annealing. The algorithms overcome the limitation of local optimization by the conventional gradient-based training methods and perform global optimization of the weights of the neural networks. Especially, the multiobjective training algorithm is designed to enhance generalization capability of the trained networks by minimizing the training error and the dynamic range of the network weights simultaneously. For fast convergence and good solution quality of the algorithms, we suggest the hybrid simulated annealing algorithm with the gradient-based local optimization method. Experimental results show that the performance of the trained networks by the proposed methods is better than that by the gradient-based local training algorithm and, moreover, the generalization capability of the networks is significantly improved by preventing overfitting phenomena.**

I. Introduction

Multilayer perceptrons (MLPs) having sigmoidal hidden neurons are a popular class of artificial neural networks in many application areas such as signal processing, optimization, pattern recognition, control and identification [1]. The universal approximation theorem for the MLP states that, if the number of hidden units is sufficiently large, it can approximate arbitrarily well any functional continuous mapping [2]. When using an MLP for a specific problem, we need to train the network parameters (i.e., interconnection weights and biases) with training data. Training of MLPs is usually performed in a supervised manner; the network is trained so that the difference between the network outputs and the training targets for the training inputs is minimized. We usually use gradient descent algorithms which minimize the sum of the errors for all training data, e.g., the error backpropagation or the second-order algorithms such as the quasi-Newton and the Levenberg-Marquardt algorithms. Especially, the Levenberg-Marquardt (LM) algorithm has shown to be quite fast and show good performance compared to other algorithms [3].

It is a limitation of such gradient-based algorithms that they only perform local optimization in the parameter space and, thus, tend to be trapped in local optima depending on the initial parameter values. Therefore, the performance of the trained network may not be sufficiently good and vary a lot with the initial values of the parameters (which is called 'randomness' [4]). As a solution to this problem, stochastic optimization methods are presented in this paper. Stochastic algorithms perform global optimization with search agents which are mutated by random generating functions, explore the entire search space and converge to the global optimum gradually [5]. Especially, it has been shown that hybridization of stochastic algorithms and local optimization techniques shows good performance in comparison with gradient-based algorithms [6].

We first propose the hybrid greedy simulated annealing (HGSA) algorithm as one of stochastic optimization methods to globally optimize the weights of MLPs. The proposed algorithm is based on the simulated annealing (SA) algorithm which performs global optimization by an iterative procedure of generation, evaluation and selection of solutions with a controlled annealing schedule [7]. It has been shown that SA can solve difficult problems of wide areas successfully [8]. In addition, its global convergence property can be shown mathematically [9], [10].

The proposed HGSA algorithm differs from the conventional SA in two ways: First, we utilize as a local optimization technique by using a gradient-based training algorithm to obtain good solutions with improved convergence speed. Combining SA with a local optimization technique has been shown to be promising for successful global optimization [11]. Second, we use the greedy selection method for choosing the solution of the next iteration between the current and the newly generated solution. while the conventional SA usually adopts the Metropolis selection rule which allows to accept the newly generated solution having a worse cost than the current solution with a nonzero probability [7]. Since the Metropolis rule concerns the cost difference of the two candidate solutions, we should consider the scaling factor of the cost function for an efficient search process. On the other hand, the greedy selection rule only observes the superiority or the inferiority of a solution to the other and, thus, has an advantage that such a scaling problem does not arise. Greedy SA is also shown to have global convergence property [12].

An important aspect of trained neural networks is their performance for unseen data, called generalization capability. Even though a network performs well for the training data, it may fail to produce satisfactorily the desirable target for an input datum which has not been used for training (such a phenomenon is called overfitting). A factor causing overfitting for a chosen network structure is the dynamic range of the weights in the network [13]. A network having too large weight values may show overfitting by making the corresponding neurons' outputs very sharp for the part of generalization data inputs. Therefore, it is desirable

The authors are with the School of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Korea (phone: +82-42-869-5453; fax: +82-42-869-8053; email: lee.yeejin@gmail.com, jslee@nnmi.kaist.ac.kr, lsy1116@kaist.ac.kr, chpark@kaist.ac.kr).

1-4244-1340-0/07$25.00 ©2007 IEEE

to keep the network weights as small as possible during training of the network to prevent overfitting phenomena and obtain the final network having good generalization performance. Thus, we propose a multiobjective optimization method, called multiobjective hybrid greedy simulated annealing (MOHGSA), which is based on the HGSA algorithm described above. MOHGSA minimizes the training error and the sum of the squared weight values simultaneously and finds the set of the Pareto optimal solutions for the two objectives. Among the final solutions we can choose a solution which shows a sufficiently small training error and has small weights in order to obtain the final network showing good generalization performance.

In the following section we present the proposed HGSA algorithm for global optimization of neural networks. Section III describes the proposed MOHGSA algorithm for improving generalization capability of neural networks. Experimental results are given in Section IV and, finally, conclusion is made in Section V.

II. Hybrid Greedy Simulated Annealing

SA is one of the powerful and robust stochastic optimization algorithms [8]. An attractive merit of SA is its mathematical global convergence property. The proposed HGSA method is based on the fast SA (FSA) which is a variant of SA and uses the Cauchy function for generating new solutions and the reciprocal annealing schedule for fast convergence [14]. Since convergence of the algorithm to the global optimum is guaranteed mathematically, it can overcome the locality of the gradient-based training algorithms. A detailed description of the algorithm and its convergence proof are given in the following subsections.

A. Algorithm

The procedure of HGSA is described as follows:

1) Initialization: Initialize the n-dimensional solution vector, i.e., the collection of the weights and the biases of the network. Set the initial temperature T_0 to a sufficiently large value.
2) Generation: Generate a new solution vector y_k from the current solution x_k by
$$y_k = x_k + \Delta x_k, \tag{1}$$
where k is the iteration index and T_k the temperature at iteration k. The amount of change of the solution, Δx_k, is determined by the multi-dimensional Cauchy probability distribution function (pdf), which is given by [15]
$$p(\Delta x_k, T_k) = \frac{a_n T_k}{(||\Delta x_k||^2 + T_k^2)^{(n+1)/2}}, \tag{2}$$
where a_n is the normalizing constant.
3) Local optimization: Apply a few iterations of the LM algorithm to y_k to produce z_k.
4) Evaluation: Evaluate the cost (training error) of z_k, $f(z_k)$.
5) Selection: Select the solution of the next iteration among x_k and z_k by the greedy selection rule: If $f(x_k) > f(z_k)$, $x_{k+1} = z_k$. Otherwise, $x_{k+1} = x_k$.
6) Annealing: Decrease the temperature by the following reciprocal annealing schedule:
$$T_k = T_0/k. \tag{3}$$
7) Termination: If some termination conditions are satisfied, stop. Otherwise, go to 2).

A new solution is generated from the current one by the combination of the Cauchy generating function and the local optimization. The local optimization step using the LM algorithm ensures $f(z_k) \leq f(y_k)$ and helps the algorithm converge fast.

We set the maximum number of iterations for the termination condition in our experiments.

B. Convergence

It is an advantage of the above HGSA algorithm that its global convergence is supported mathematically. The following theorem proves that the cost sequence in HGSA converges to the global optimum in probability [11].

Theorem 1: For $\epsilon > 0$, let
$$\Psi_\epsilon = \{x \in \Psi | f(x) < f^* + \epsilon\}, \tag{4}$$
$$\Psi'_\epsilon = \{x \in \Psi | f(\phi(x)) < f^* + \epsilon \text{ and } f(x) \geq f^* + \epsilon\}, \tag{5}$$
where f^* is the global minimum cost, Ψ the entire feasible space in $\mathbb{R}^n$ and $\phi(\cdot)$ the local search operation guarantying $f(x) \geq f(\phi(x))$ in which the equality is satisfied when x is the local minimum. Assume that $\zeta(\Psi_\epsilon) > 0$ and $\zeta(\Psi'_\epsilon) > 0$ for every $\epsilon > 0$, where $\zeta(\cdot)$ is the Lebesgue measure on $\mathbb{R}^n$.

For any initial solution $x_0 \in \Psi$, the cost sequence $\{f(x_k), k \geq 0\}$ of HGSA converges in probability to the global minimum f^*.

Proof: To prove the theorem, it is sufficient to show that, for any $\epsilon > 0$ and initial solution $x_0 \in \Psi \setminus \Psi_\epsilon$, the following is satisfied:
$$\lim_{k\to\infty} P(f_k - f^* \geq \epsilon) = 0. \tag{6}$$
We can show that
$$P(f_k - f^* \geq \epsilon) \leq \prod_{j=0}^{k-1} \max_{x_j \in \Psi \setminus \Psi_\epsilon} P(x_{j+1} \in \Psi \setminus \Psi_\epsilon | x_j). \tag{7}$$
Therefore,
$$\lim_{k\to\infty} P(f_k - f^* \geq \epsilon) \leq \exp\left[-\sum_{j=0}^{\infty} \min_{x_j \in \Psi \setminus \Psi_\epsilon} P(x_{j+1} \in \Psi_\epsilon | x_j) \right]. \tag{8}$$
Here, we have
$$\min_{x_j \in \Psi \setminus \Psi_\epsilon} P(x_{j+1} \in \Psi_\epsilon | x_j) = \min\left\{ \min_{x_j \in \Psi \setminus \Psi_\epsilon} P(y_j \in \Psi_\epsilon | x_j), \min_{x_j \in \Psi \setminus \Psi_\epsilon} P(y_j \in \Psi'_\epsilon | x_j) \right\}. \tag{9}$$

We can write each term in the above equation as

$$\min_{x_j \in \Psi \setminus \Psi_\epsilon} P(y_j \in \Psi_\epsilon | x_j) = \int_{\Psi_\epsilon} g(y - x_j, T_j) dy \geq \zeta(\Psi_\epsilon) \min_{x_j, y \in \Psi} g(y - x_j, T_j) \tag{10}$$

and, similarly,

$$\min_{x_j \in \Psi \setminus \Psi_\epsilon} P(y_j \in \Psi'_\epsilon | x_j) \geq \zeta(\Psi'_\epsilon) \min_{x_j, y \in \Psi} g(y - x_j, T_j). \tag{11}$$

where $g(\cdot, T_j)$ is the temperature-dependent generation probability density function used to generate the random vectors.

Let

$$r_{(i)} = \max_{x,y \in \Psi} |x_{(i)} - y_{(i)}|, \quad 1 \leq i \leq n, \tag{12}$$

where $x_{(i)}$ and $y_{(i)}$ are the i-th components of the n-dimensional vector x and y in Ψ, respectively. From (2) and (3), we obtain that

$$\min_{x,y \in \Psi} g(y - x, T_j) = \min_{x,y \in \Psi} \frac{T_j}{(||y - x||^2 + T_j^2)^{(n+1)/2}} \geq \frac{1}{j} \cdot M, \tag{13}$$

where we set

$$M = T_0 / (\sum_i r_{(i)}^2 + T_0^2)^{(n+1)/2}. \tag{14}$$

Hence,

$$\min_{x_j \in \Psi \setminus \Psi_\epsilon} P(x_{j+1} \in \Psi_\epsilon | x_j) \geq \min \{\zeta(\Psi_\epsilon),\ \zeta(\Psi'_\epsilon)\} M/j, \tag{15}$$

which yields

$$\lim_{k \to \infty} P(f_k - f^* \geq \epsilon) \leq \exp \left[- \sum_{j=0}^{\infty} \min \{\zeta(\Psi_\epsilon), \zeta(\Psi'_\epsilon)\} \cdot M/j \right] = 0. \tag{16}$$

■

III. Multiobjective Hybrid Greedy Simulated Annealing

The generalization performance of a trained network for unseen data is the most important property of the network. While the HGSA algorithm can be used for global optimization of a network, it may not guarantee good generalization capability of the network.

In this section, we propose a multiobjective optimization algorithm, MOHGSA, which improves the generalization performance of neural networks as well as minimizes their training errors. The training objectives, the procedure and the decision making rule of the algorithm are presented below.

A. Training Objectives

When the structure of a network is specified, an important factor affecting its generalization performance is the dynamic range of the network's weights. Large weight values may cause the overfitted mappings which approximate the training data with a good accuracy but show large curvatures for the regions between the training data. In [13], it has been shown that, even for networks of relatively small sizes, large dynamic ranges of their weights can produce overfitting. Based on this observation, regularization methods such as the weight decay [16] have been proposed as a way of preventing weights from growing during training by adding penalty terms related to the weight values. However, such methods perform only the gradient-based local search. Moreover, while the regularization parameter balancing the training error and the penalty term significantly affects the networks' training and performance, it is not easy to determine its appropriate value.

The proposed MOHGSA algorithm solves these two problems by global multiobjective optimization. The two objectives (minimizing training error and weight values) are separately optimized at the same time by utilizing the stochastic global optimization ability of HGSA.

The goal of MOHGSA is to minimize the following two objective functions:

$$f_1(x) = \sum (\text{error})^2$$
$$f_2(x) = \sum (\text{weight})^2, \tag{17}$$

where x is the set of the weights of a network. The summation of each equation is done over all training data and all weights in the network, respectively.

The solution of (17) is not unique but there are many non-dominated solutions. We say that a solution x dominates the other solution y if all objectives of x are not worse than those of y and at least one objective of x is better than that of y. If x does not dominate y and vice versa, they are said to be non-dominated by each other. The optimal solutions for a multiobjective problem are called the Pareto optimal solutions. A Pareto optimal solution is not dominated by any other feasible solution. The following subsection presents the MOHGSA algorithm to find the Pareto optimal solutions of (17) [17].

B. Algorithm

The MOHGSA algorithm is based on the previously developed HGSA algorithm; the ways of generation of offsprings and selection of solutions and the annealing schedule are the same in both algorithms. However, MOHGSA uses a population of search agents to find the multiple Pareto optimal solutions of (17). A detailed description of the algorithm is given below.

1) Initialization: Randomly generate the initial population of P n-dimensional solution vectors. Each solution defines the set of weight values of a network. Set the initial temperature T_0 to a sufficiently large value.

2) Generation: For each parent solution, generate an offspring by (1). Again, the Cauchy pdf in (2) is used for generation of offsprings.
3) Local optimization: Apply a few iterations of the LM algorithm to each offspring.
4) Evaluation: Calculate the two objective values of each offspring by (17). The costs of the parents and the offsprings are determined by the Pareto-based costs [17], [18]. The Pareto-based cost of a solution u is defined by

$$C(u) = \sum_{p=1}^{P} dom(x_k^p, u) + \sum_{p=1}^{P} dom(z_k^p, u), \quad (18)$$

where x_k^p and z_k^p are the p-th parent and offspring at iteration k, respectively, and $dom(\cdot,\cdot)$ is defined by

$$dom(y, x) = \begin{cases} 1 & \text{if } y \text{ dominates } x, \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

5) Selection: Select the population of the next iteration based on the calculated Pareto-based costs by the greedy selection rule, i.e., choose P solutions of smaller cost values than the others among the P parents and the P offsprings.
6) Annealing: Decrease the temperature by the reciprocal annealing schedule given by (3).
7) Termination: If some termination conditions are satisfied, stop. Otherwise, go to 2).

C. Decision Making

Once obtaining the non-dominated optimal solutions by the algorithm described in the previous subsection, we should choose one solution for obtaining the final network. We use the following rules for this final decision making:

1) We choose the solutions which show sufficiently small training error among the final non-dominated solutions. An upper bound of the training error is used to choose such solutions.
2) Among the chosen solutions, we select the one having the smallest value of the second objective f_2.

Using an upper bound of the training error is to ensure that the network is sufficiently trained with the training data and does not show underfitting. The solutions satisfying the first condition have different values for the second objective function, i.e., the sum of the squared weight values. Since the final goal of our method is to obtain a network with good generalization capability, we select the solution having the smallest f_2. By these steps, we can obtain the final network showing good generalization capability without underfitting and overfitting.

IV. Experimental Results

In this section, we demonstrate the performance of the proposed HGSA and MOHGSA algorithms for function approximation problems. We compare the performance of the MLPs trained by the LM, HGSA and MOHGSA algorithms for each problem.

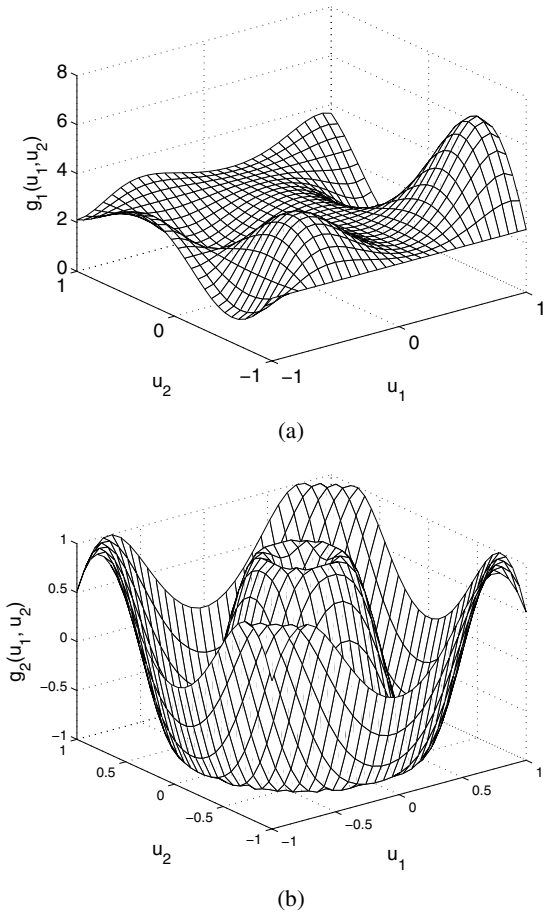

Fig. 1. Three-dimensional perspective plots of the test functions: (a) g_1 and (b) g_2.

A. Test Problems

We employ two function approximation problems, which are popularly used for comparing performance of neural networks [4], [19]. They are given by the following equations and shown in Fig. 1.

- Complicated interaction function:

$$g_1(u_1, u_2) = 1.9(1.35 + e^{0.5(u_1+1)} \sin(13(0.5u_1 - 0.1)^2) \cdot e^{-0.5(u_2+1)} \sin(3.5u_2 + 3.5)) \quad (20)$$

- Harmonic function:

$$g_2(u_1, u_2) = \sin(2\pi\sqrt{u_1^2 + u_2^2}) \quad (21)$$

We use 121 randomly generated points and four corner points as the set of the training data. For generalization test, we generate 10,000 test data from the 100×100 uniform grid in the input domain.

B. Setup

We perform experiments with two-layer MLPs having 22 and 30 hidden neurons for g_1 and g_2, respectively. The result does not vary with the number of hidden neurons in any significant amount.

For the LM algorithm, the initial weights of an MLP are randomly generated within the range of $[-0.05, 0.05]$. The maximum number of LM iterations is set to 20,000, which is considered to be sufficient to reach local optima. We repeat the experiments for LM 100 times with different initial random seeds.

In our HGSA and MOHGSA, we use 30 iterations of the LM algorithm for the local optimization. The maximum number of iterations for HGSA and MOHGSA is set to 10,000.

In the HGSA algorithm, we set the bound of the solutions to $[-10, 10]$ because boundedness of search spaces is necessary for the convergence property shown in Section II-B. When new solutions are generated from the current ones by the Cauchy generating function, we use the normalized solution values within the bound $[-1, 1]$. We limit the new solutions in this bound after generating by the Cauchy function, and allow solutions to be out of the bound after local optimization. The initial temperature T_0 was set to 10 in HGSA. In MOHGSA, the population size is set to 100 and $T_0 = 200$ is used.

The training and test errors of networks are measured in terms of the root-mean-squared error (RMSE).

C. Results

First, we examine how the network errors and the weight values change during training of the network by the LM algorithm. Fig. 2 shows the evolutions of the errors for the training and the test data and the sum of the squared weights when MLPs are trained by the LM algorithm. For g_1, most cases of 100 trials were ended without overfitting, as shown in Fig. 2(a), but a few cases showed overfitting as in Fig. 2(b). For g_2, all trials showed overfitting and a typical example is shown in Fig. 2(c). In Fig. 2(a), we see that the training and the test errors gradually become small and converge after about 1000 iterations. However, when overfitting occurs, while the training RMSE consistently decreases by LM, the test RMSE decreases only up to a certain point and increases when the training continues. Consequently, although the training RMSE at the end of the training is smaller than 10^{-2}, the test RMSE is larger than 2.0 and 0.7 for each function, respectively. In these cases, the weight values grow excessively, which causes the bad generalization performance of the networks for the test data. These observations justify the necessity of the proposed MOHGSA algorithm for preventing overfitting and producing networks with good generalization capability.

Fig. 3 shows an example of MLPs showing overfitting phenomena for each test function when the MLPs are trained by LM. We can observe undesirable sharp peaks which are caused by large weights and produce poor generalization performance of the networks.

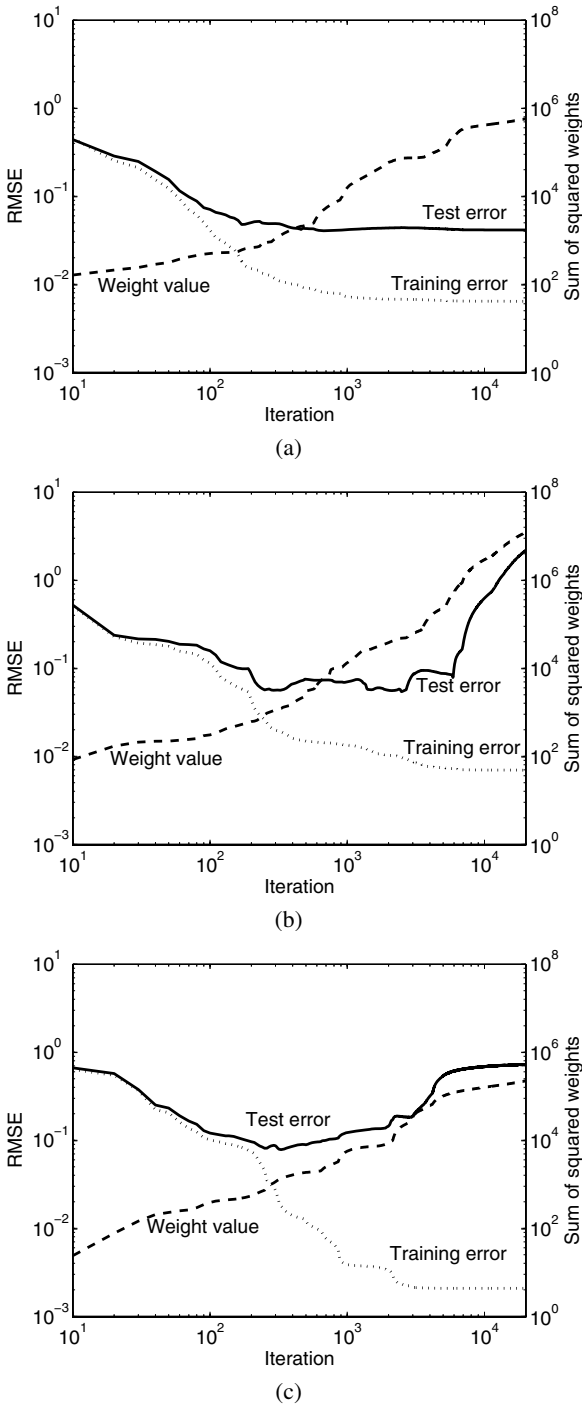

Fig. 2. Evolution of the training and the test errors and the sum of squared weight values with respect to the iteration of the LM training. (a) A typical case without overfitting for g_1. (b) A case with overfitting for g_1. (c) A typical case with overfitting for g_2.

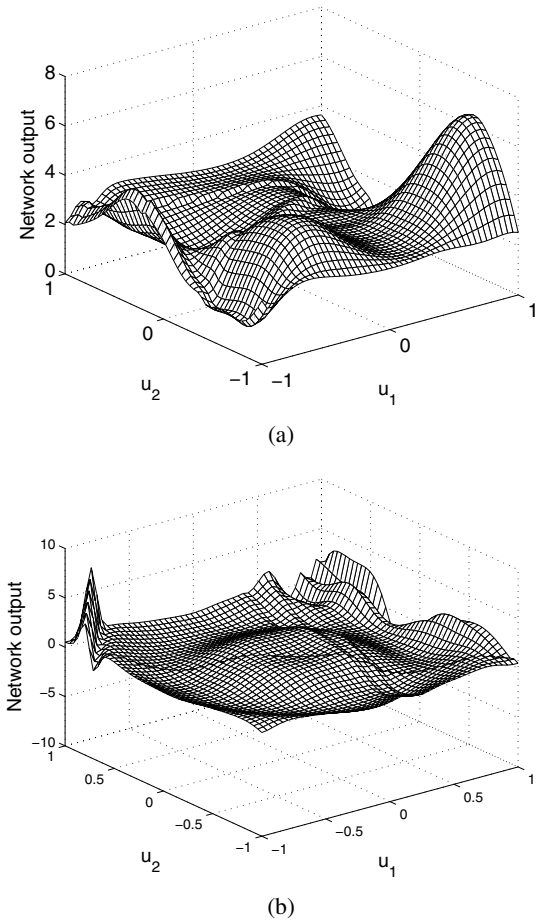

Fig. 3. Examples of networks showing overfitting for (a) g_1 and (b) g_2.

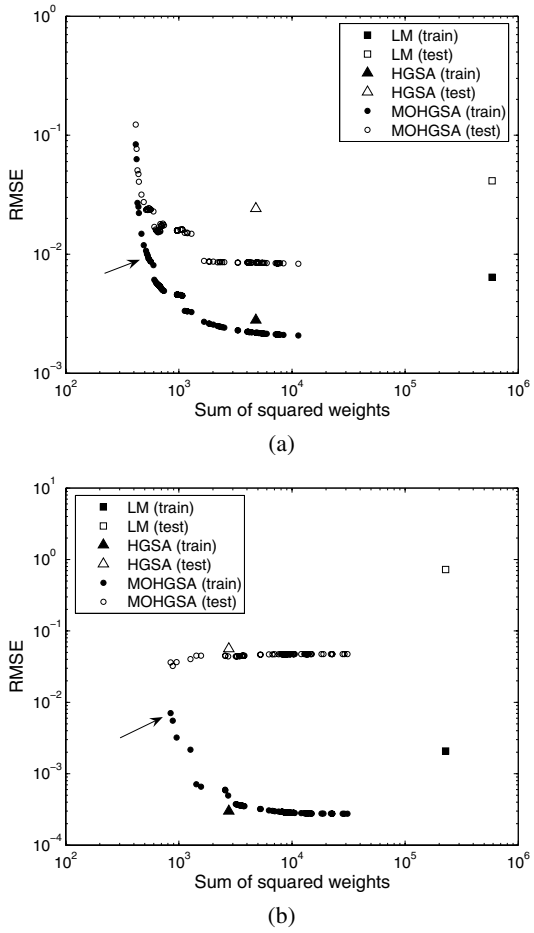

Fig. 4. Final solutions by LM, HGSA and MOHGSA for (a) g_1 and (b) g_2.

Next, we compare the performance of the conventional and the proposed methods in terms of the weight values and the errors for the training and the test data. Fig. 4 shows the final solutions by LM, HGSA and MOHGSA. For the case of LM, the typical cases shown in Figs. 2(a) and 2(c) are shown. In the case of MOHGSA, all 100 solutions composed the final set of the non-dominated solutions for both functions. We observe that the solutions by LM are dominated by those by HGSA and MOHGSA, which shows the global optimization capabilities of HGSA and MOHGSA. The MOHGSA solutions indicated by arrows are the chosen ones for the final networks by the decision-making rule with the upper bound of the training RMSE of 0.01.

Finally, Table I compares the RMSE for the generalization data when the networks are trained by each of LM, HGSA and MOHGSA. The results for LM show the best, the worst and the mean values for 100 repeated experiments. For the case of MOHGSA, we use the decision rule described in Section III-C to choose the final network among the obtained non-dominated solutions; among the non-dominated solutions whose training RMSE values are less than 0.01, we choose the leftmost solution, i.e., the one with the smallest sum of the squared weight values (indicated by arrows in Fig. 4), and then the chosen solution was evaluated with the generalization test data. From the results in the table, we see that MOHGSA produces the networks with the best generalization performance among the three methods for both test functions, which demonstrates effectiveness of MOHGSA for improving generalization capability of networks. While the LM algorithm sometimes shows overfitting, the generalization performance of the networks trained by MOHGSA is much better than that by LM on average. Since the HGSA algorithm does not consider the generalization performance of the networks, it shows poorer generalization performance than MOHGSA.

TABLE I

GENERALIZATION RMSE OF MLPS TRAINED BY LM, HGSA AND MOHGSA. FOR THE CASE OF LM, THE BEST, THE WORST AND THE MEAN VALUES FOR 100 TRIALS ARE SHOWN. FOR THE CASE OF MOHGSA, THE SHOWN VALUES ARE OBTAINED BY THE NETWORKS CHOSEN FROM THE OBTAINED NON-DOMINATED SOLUTIONS BY THE DECISION RULE OF THE ALGORITHM.

Function	LM	HGSA	MOHGSA
g_1	0.0127 (best) 0.0652 (mean) 2.1968 (worst)	0.0242	0.0235
g_2	0.0350 (best) 0.3681 (mean) 24.973 (worst)	0.0563	0.0362

V. CONCLUSION

This paper has proposed the global training algorithms of MLPs based on SA. First, we proposed the HGSA algorithm performing global optimization of MLPs' weights to overcome the limitation of the conventional gradient-based training methods. We proved its global convergence mathematically. Second, we proposed the MOHGSA algorithm for improving the generalization capability of the trained networks. MOHGSA is based on HGSA and performs global optimization of the networks with the two goals: optimization of the networks' weights and good generalization capability of the networks. Through function approximation problems, we showed that the proposed MOHGSA algorithm is effective in producing networks with good generalization capability.

Applying the proposed methods to various problems and parallel implementation of MOHGSA with multiple computers and investigating the relationship of the network errors and the bound of the solution space are currently in progress.

ACKNOWLEDGMENT

This work was supported by the Brain Korea 21 Project in 2007.

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice-Hall, Inc., 1999.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press, Inc., 1995.

[3] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 989–993, Nov. 1994.

[4] J.-S. Lee, H. Lee, J.-Y. Kim, D. Nam, and C. H. Park, "Self-organizing neural networks using construction and pruning," *IEICE Trans. Information and Systems*, vol. E87-D, no. 11, pp. 2489–2498, Nov. 2004.

[5] J. C. Spall, *Introduction to Stochastic Search and Optimization*. Hoboken, NJ: John Wiley & Sons, 2003.

[6] H. A. Abbass, "A memetic Pareto evolutionary approach to artificial neural networks," in *Proc. Australian Joint Conf. Artificial Intelligence*, Adelaide, Australia, Dec. 2001, pp. 1–12.

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

[8] P. J. M. V. Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1987.

[9] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Advances in Applied Probability*, vol. 18, pp. 747–771, 1986.

[10] R. L. Yang, "Convergence of the simulated annealing algorithm for continuous global optimization," *Journal of Optimization Theory and Applications*, vol. 104, no. 3, pp. 691–716, Mar. 2000.

[11] J.-S. Lee and C. H. Park, "Training hidden Markov models by hybrid simulated annealing for visual speech recognition," in *Proc. IEEE Int. Conf. Systems, Man, Cybernetics*, Taipei, Taiwan, Oct. 2006, pp. 198–202.

[12] C.-Y. Lee, S.-Y. Lee, J.-S. Lee, S. Lee, and C. H. Park, "Fast simulated annealing with greedy selection," in *The Institute of Electronics Engineers of Korea*, 2007 (submitted).

[13] S. Park and C. H. Park, "Network complexity and generalization," in *Proc. Int. Joint Conf. Neural Networks*, vol. 1, Houston, TX, June 1997, pp. 298–301.

[14] H. H. Szu and R. L. Hartley, "Fast simulated annealing," *Physics Letters A*, vol. 122, no. 3–4, pp. 157–162, June 1987.

[15] D. Nam, J.-S. Lee, and C. H. Park, "n-dimensional cauchy neighbor generation for the fast simulated annealing," *IEICE Trans. Inf. Syst.*, vol. E87-D, no. 11, pp. 2499–2502, Nov. 2004.

[16] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Procoessing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. San Mateo, CA: Morgan Kauffmann Publishers, 1995, pp. 950–957.

[17] D. Nam and C. H. Park, "Pareto-based cost simulated annealing for multiobjective optimization," in *Proc. Asia-Pacific Conf. Simulated Evolution and Learning*, vol. 2, Singapore, Nov. 2002, pp. 522–526.

[18] J.-S. Lee and C. H. Park, "Discriminative training of hidden Markov models by multiobjective optimization for visual speech recognition," in *Proc. Int. Joint Conf. Neural Networks*, Montreal, Canada, July 2005, pp. 2053–2058.

[19] J.-N. Hwang, S.-R. Lay, M. Maechler, R. D. Martin, and J. Schimert, "Regression modeling in backpropagation and projection pursuit learning," *IEEE Trans. Neural Networks*, vol. 5, no. 3, pp. 342–353, 1994.