Global Learning of Neural Networks by Using Hybrid Optimization Algorithm

Yong-Hyun Cho Seong-Jun Hong

School of Computer and Information Comm. Eng., Catholic Univ. of Daegu, 330, Kumrakri, Hayangup, Kyungsan, Kyungbuk, 712-702, Korea(South)

Abstract

This paper proposes a global learning of neural networks by hybrid optimization algorithm. The hybrid algorithm combines a stochastic approximation with a gradient descent. The stochastic approximation is first applied for estimating an approximation point inclined toward a global escaping from a local minimum, and then the backpropagation(BP) algorithm is applied for high-speed convergence as gradient descent. The proposed method has been applied to 8-bit parity check and 6-bit symmetry check problems, respectively. The experimental results show that the proposed method has superior convergence performances to the conventional method that is BP algorithm with randomized initial weights setting.

Keywords: Neural networks, Global learning, Stochastic approximation, Gradient descent, Backpropagation algorithm

1. Introduction

Neural networks (NNs), due to its massive parallelism, have been rigorously studied as an alternative to the conventional numerical approaches. Over the past several years, NNs have been increasingly applied to the identification, control of nonlinear systems, and pattern recognitions [1]-[2]. A basic model structure for static nonlinearities is the multilayer feedforward NN, in which learning, i.e. estimation of weights, involves the minimization of an output error criterion using backpropagation (BP) [3]-[6].

Although the BP algorithm can be generalized for more complex NN structures (e.g. recurrent NNs, hybrid physical-neural models), which are useful in modeling dynamic nonlinear systems, the resulting algorithms are usually more complicated to implement and more computationally demanding. Hence, it is appealing to develop a more straightforward numerical procedure for computing the gradient of the output error criterion [5]-[6].

Most of the researches have been concentrated on the improvement of either the convergence speed or the convergence to the global minimum in consideration of the weights parameter of the energy function, cooling scheduling, the learning parameters, or the number of hidden units, etc[3]-[7]. But there are few that consider the initial weights setting to accelerate the speed of convergence. And there are few that try to solve both of the global convergence and its speed simultaneously. If the initial synapse weights in the NNs are set close to the global minimum, then the learning procedure will be expected to converge quickly. Consequently the global minimum will be found correctly with high speed.

This paper proposes an efficient method for improving the convergence performances of the NN by applying a global optimization method. The global optimization method is a hybrid of a stochastic approximation [8] and a gradient descent method. The approximation value inclined toward a global escaping from a local minimum is estimated first by applying the stochastic approximation, and then the gradient descent of BP algorithm is applied for high-speed convergence. The proposed algorithm has been applied to the 10 training patterns of 8-bit parity check and 6-bit symmetry check problems, respectively. We demonstrate the convergence performances of the proposed algorithm compared with the conventional method that is BP algorithm with randomized initial synapse weights setting.

2. Estimation of initial value by stochastic approximation

We consider the following problem of global unconstrained optimization: minimize the multiextremal function $f(\mathrm{x}) \in \boldsymbol{R}^{\mathrm{i}}$, $\mathrm{x} \in \boldsymbol{R}^{\mathrm{n}}$, i.e.

$$\min_{x \in R^n} f(x) \qquad (1)$$

A multiextremal function can be represented as a superposition of uniextremal function(i.e., having just one minimum) and other multiextremal function that add some noise to the uniextremal function. The objective of smoothing can be visualized as filtering

out the noise and performing minimization on the smoothed uniextremal function, in order to reach the global minimum. In general, since the minimum of the smoothed uniextremal function does not coincide with the global function minimum, a sequence of minimization runs is required to zero in the neighborhood of the global minimum [8]. The smoothing process is performed by averaging $f(x)$ over some region of the parameter space $\boldsymbol{R}^n$ using a proper weighting (or smoothing) function $\hat{h}(x)$.

Let us introduce a vector of random perturbations $\eta \in \boldsymbol{R}^n$, and add η to x. The convolution function $\tilde{f}(x, \beta)$ is created as follows [8].

$$\tilde{f}(x,\beta) = \int_{R^n} \hat{h}(\eta,\beta) f(x-\eta) d\eta \qquad (2)$$

Hence:

$$\tilde{f}(x,\beta) = E_\eta[f(x-\eta)] \qquad (3)$$

Where $\tilde{f}(x, \beta)$ is the smoothed approximation to the original function $f(x)$, and the kernel function $\hat{h}(\eta, \beta)$ is the probability density function(pdf) used to sample η. β controls the dispersion of $\hat{h}(\eta, \beta)$, i.e. the degree of $f(x)$, smoothing [8].

Note that $\tilde{f}(x, \beta)$ can be regarded as an averaged version of $f(x)$, weighted by $\hat{h}(\eta, \beta)$. $E_\eta[f(x- \eta)]$ is the expectation with respect to the random variable η. Therefore an unbiased estimator $\tilde{f}(x, \beta)$ is the average:

$$\tilde{f}(x,\beta) = \frac{1}{N}\sum_{i=1} E_n[f(x-\eta^i)] \qquad (4)$$

Where η is sampled with the pdf $\hat{h}(\eta, \beta)$.

The kernel function $\hat{h}(\eta, \beta)$ should have the following properties [8]:

- $\hat{h}(\eta, \beta) = (1/ \beta^n)\, h(\eta/\beta)$ is piecewise differentiable with respect to η. (5)
- $\lim_{\beta\to 0} \hat{h}(\eta, \beta) = \delta(\eta)$ (Dirac delta function)
- $\hat{h}(\eta, \beta)$ is a pdf.

Under above the conditions, $\lim_{\beta\to 0} \tilde{f}(x, \beta) = \int_{\boldsymbol{R}^n} \delta(\eta)\, f(x- \eta)\, d\eta = f(x - 0) = f(x)$. Several pdfs fulfill above conditions, such as the Gaussian, uniform, and Cauchy pdfs.

Smoothing is able to eliminate the local minima of $\tilde{f}(x, \beta)$, if β is sufficiently large. If $\beta \to 0$, then $\tilde{f}(x, \beta) \to f(x)$. This should actually happen at the end of optimization to provide convergence to the true function minimum [8]. Formally, the optimization problem can be written as:

$$\min_{x\in R^n} \tilde{f}(x,\beta) \qquad (6)$$

with $\beta \to 0$ as $x \to x^*$. Where x^* is the global minimum of the original function $f(x)$. One class of methods to solve the modified problem Eq. (6) – to be called large sample(LS) stochastic methods - can be characterized as follows: for each new point x, a large number of points sampled with the pdf $\hat{h}(\eta, \beta)$ (Eq. (5)) is used to estimate $\tilde{f}(x, \beta)$ and its gradient $\nabla_x \tilde{f}(x, \beta)$. The number of samples used should be sufficiently large to give small errors of the relevant estimators. Optimization and averaging are separated in this process. This is very inefficient [8].

Optimization and the averaging can be combined into one iterative process, leading to much more efficient small-sample (SS) methods of stochastic programming. A large class of SS methods, called stochastic approximation, is applied to stochastic function minimization or maximization [8]. Their basic principle of operation is that only a small number of samples are used in each iteration to find the necessary estimators, but all the information is being averaged over many steps.

In function minimization, SA methods create stochastic equivalent to the gradient methods of nonlinear programming. The advanced algorithms are proposed to estimate the gradient $\nabla_x \tilde{f}(x, \beta)$. As the algorithm progresses, $\beta \to 0$, reducing the smoothing degree of the $f(x)$ function, and providing convergence to the true minimum. The SA algorithm implements a well-defined approximation to the conjugate gradient [1]. The value x based on the smoothed function $\tilde{f}(x, \beta)$ is updated, as following,

$$\begin{aligned}
&\xi_k = \nabla_x \tilde{f}(x_k, \beta)\\
&S_k = \mathrm{STEP} / \|\xi_k\|\\
&d_k = (1-\rho_k)\, d_{k-1} + \rho_k\, \xi_k \quad (0 \le \rho_k \le 1)\\
&\rho_k = (1-\rho_{k-1}) / (1+ \rho_{k-1} - R)\\
&x_{k+1} = x_k - S_k\, d_k
\end{aligned} \qquad (7)$$

where k(1,2,…,MAXITER) is the number of iterations, ξ is the gradient, S is a step size, d is the search direction, ρ is the gradient averaging coefficient of $\tilde{f}(x, \beta)$, and R($0<R<1$) is a constant controlling the rate of change of ρ_k.

Therefore, we can find the global minimum of original function $f(x)$ by iteratively performing one cycle of the SA optimization as $\beta \to 0$. This is called the stochastic approximation with smoothing (SAS) [8].

Fig.1 is the flow chart of SA algorithm. In this Fig. 1, each new value x is performed in the direction $S_k d_k$, where d_k is a convex combination of the previous direction d_{k-1} and a new gradient ξ_k. Especially R is responsible for the rate of change of ρ_k, that is, it modifies the search direction d_k and provides a suitable amount of inertia of gradient direction.

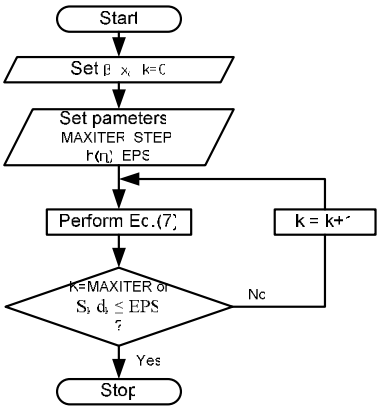

Fig. 1: Flow chart of stochastic approximation (SA).

Fig. 2 is the flow chart of SAS algorithm that repeatedly performs the SA algorithm according to a sequence: $\{\beta_0, \beta_1, ...\} \rightarrow 0$. We can get the global minimum by using the SAS algorithm based on specific sequences $\{\beta\}$ and {NMAX}. It turned out that the final solutions were not very sensitive to a specific choice of theses sequences based on rough heuristic criteria such as: low problem dimensionality requires a smaller number of function evaluations, β should be large at the beginning of optimization (to determine the approximate position of the global minimum), and small at the end of optimization (for precision) [8].

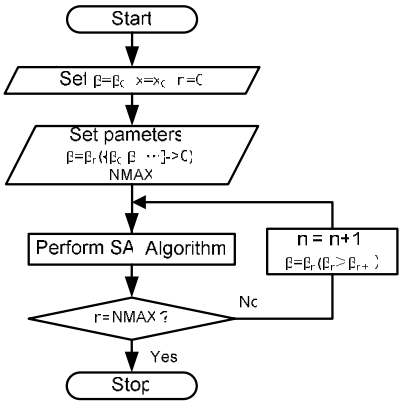

Fig. 2: Flow chart of stochastic approximation with smoothing (SAS).

We consider the function $f(x) = x^4 - 16x^2 + 5x$ as an example [8]. This function is continuous and differentiable, and it has two distinct minima as shown in Fig.3. The smoothed function $\tilde{f}(x, \beta)$ is plotted to different values of $\beta \rightarrow 0$({5, 4, 3, 2, 1, 0.001, 0.0}) and MAXITER=100 for uniform pdf. We can show that minimize the smoothed functional $\tilde{f}(x, \beta)$ with $\beta \rightarrow 0$ as $x \rightarrow x^*$. As shown in Fig. 3, the $\tilde{f}(x, \beta)$ has a uniextremal function having one minimum x_I from β =5 to β=3. That is, smoothing is able to eliminate the local minima of $\tilde{f}(x, \beta)$, if β is sufficiently large. If $\beta \rightarrow 0$, then $\tilde{f}(x, \beta) = f(x)$. Above all, we can find out that the minimum x_I of uniextremal function leads toward the global minimum x^* of $f(x)$.

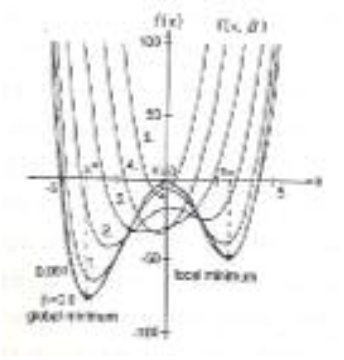

Fig. 3: Smoothed function $\tilde{f}(x, \beta)$ to different β values.

On the other hand, the simulated annealing is often explained in terms of the energy that particle has at any given temperature [7]. A similar explanation can be given to the smoothed approximation approach discussed. Perturbing x can be viewed as adding some random energy to a particle which x represents. The larger the β, the larger the energy (i.e., the larger the temperature in the simulated annealing), and the broader the range of x changes. Reducing β for the smoothed approximation approach corresponds to temperature reduction in the simulated annealing.

Although the global minimum can be found by repeatedly applying the SA according to a sequence: $\{\beta_0, \beta_1, ...\} \rightarrow 0$, there are a few problems as follows: a specific sequences and a parameters should be determined heuristically in each iterations, and, due to its stochastic process, its convergence speed is rather slower lower than that of the deterministic algorithm and sometimes results in approximate solution.

For this reason, SAS is the stochastic algorithm as simulated annealing. The stochastic algorithms guarantees that converge to the global minimum, but their convergence speed are lower than that of the deterministic algorithms. In order to solve the limitation of convergence speed, we present a new optimization method that combines advantages of stochastic algorithm and deterministic algorithms. That is, we propose a hybrid method of SA algorithm and gradient descent algorithm. SA algorithm is previously applied to estimate an initial value leading to the global minimum, and the gradient descent algorithm as deterministic algorithm is also applied for high-speed convergence. In Fig.3, if we utilize the minimum x_I as an initial value of gradient descent algorithm, the

global minimum of original function can be quickly and correctly looked for rather than that find by repeatedly applying SA according to a sequence {β}.

Fig.4 is the flow chart of proposed method. If the other minima exist between x_f (minimum of original function by using the gradient descent algorithm) and global minimum x^*, x^* can be find out by repeatedly applying the proposed method.

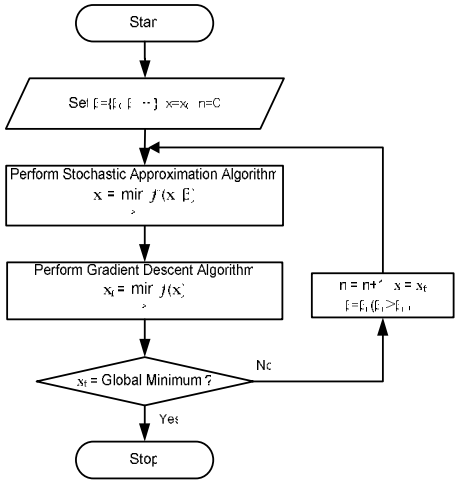

Fig. 4: Flow chart of proposed method.

3. Training of neural network by the proposed method

The basic idea in this paper is that, in applying the SA, if we choose a large β initially, we can get a uniextremal smoothed function $f^{\sim}(x, \beta)$, the minimum value x_I of which can approximately point out the hill side value of the global minimum well. In optimization of functions and learning of NNs, Minimization of the function $f(x)$ to variable x are much the same as the minimization of total error function E(**W**) to synapse weights **W**.

Accordingly, we apply the proposed method to learn the multilayer neural network. BP algorithm is also used in learning as a gradient descent algorithm. SA algorithm is previously applied to estimate an initial weights leading to the global minimum, and then the BP algorithm is also applied for high-speed convergence. The multilayer neural network will be quickly learned and clearly guaranteed that converges to a global minimum in weight space if we go about it like this.

The proposed algorithm to improve the learning performances of the multilayer neural network by applying the SA and the BP algorithm can be detailed as follows:

Step 1: Define the total error function E(**W**) and the square error function $E_p(\mathbf{W})$ of each of the p-th input patterns.

$$E(\mathbf{W}) = \sum_p E_p(\mathbf{W})$$
$$E_p(\mathbf{W}) = (1/2) \sum_i (x_{ip}(\mathbf{W}) - d_{ip})^2 \quad (8)$$

Where $x_{ip}(\mathbf{W})$ and d_{ip} are the physical value and the desired value of i-th output neuron over p-th input pattern.

Step 2: Calculate the smoothed gradient $\nabla_w E_p^{\sim}(\mathbf{W}, \beta)$ over the $E_p(\mathbf{W})$.

$$E_p^{\sim}(\mathbf{W}, \beta) = (1/2)[E_p^{\sim}(\mathbf{W}+\beta\eta) + E_p^{\sim}(\mathbf{W}-\beta\eta)]$$
$$= (1/4)\sum_i [2d_{ip}^2 - 2d_{ip}(x_{ip}(\mathbf{W}+\beta\eta) + x_{ip}(\mathbf{W}-\beta\eta)) + x_{ip}^2(\mathbf{W}+\beta\eta) + x_{ip}^2(\mathbf{W}-\beta\eta)] \quad (9)$$

$$\nabla_w E_p^{\sim}(\mathbf{W}, \beta) = (1/4) \sum_i [-2d_{ip}(\nabla_w x_{ip}(\mathbf{W}+\beta\eta) + \nabla_w x_{ip}(\mathbf{W}-\beta\eta)) + \nabla_w x_{ip}^2(\mathbf{W}+\beta\eta) + \nabla_w x_{ip}^2(\mathbf{W}-\beta\eta)] \quad (10)$$

i) For the output layer,

$$\nabla_w E_p^{\sim}(\mathbf{W},\beta) = (1/2)\sum_i [(f(\sum_j (W_{ij}+\beta\eta)\, y_{jp}(\mathbf{W})) - d_{ip})\, f(\sum_j (W_{ij}+\beta\eta)\, y_{jp}(\mathbf{W}))\,(1 - f(\sum_j (W_{ij}+\beta\eta)\, y_{jp}(\mathbf{W}))) + (f(\sum_j (W_{ij}-\beta\eta)\, y_{jp}(\mathbf{W})) - d_{ip})\, f(\sum_j (W_{ij}-\beta\eta)\, y_{jp}(\mathbf{W}))\,(1 - f(\sum_j (W_{ij}-\beta\eta) y_{jp}(\mathbf{W})))] y_{jp}(\mathbf{W}) \quad (11)$$

ii) For the hidden layer,

$$\nabla_w E_p^{\sim}(\mathbf{W}, \beta) = (1/2) \sum_i [(f(\sum_j W_{ij}\, f(\sum_m (W_{jm} - \beta\eta) o_{mp})) - d_{ip})\, f(\sum_j W_{ij}\, f(\sum_m (W_{jm} - \beta\eta)\, o_{mp}))\,(1 - f(\sum_j W_{ij}\, f(\sum_m (W_{jm} - \beta\eta) o_{mp}))) \sum_j W_{ij} f(\sum_m (W_{jm}+\beta\eta) o_{mp}) + (f(\sum_j W_{ij}\, f(\sum_m (W_{jm}-\beta\eta)\, o_{mp})) - d_{ip})\, f(\sum_j W_{ij}\, f\sum_m (W_{jm}- \beta\eta)\, o_{mp}))\,(1 - f(\sum_j W_{ij}\, f(\sum_m (W_{jm}-\beta\eta)\, o_{mp}))) \sum_j W_{ij}\, f(\sum_m (W_{jm}-\beta\eta)\, o_{mp})]\, o_{mp}\,(1 - f(\sum_m (W_{jm}+\beta\eta) o_{mp})) \quad (12)$$

Where, o_{mp} is the output from the m-th neuron of (s-2)-th layer corresponding to the p-th input pattern. The i, j, and m are the numbers of the s-, (s-1)-, and (s-2)-th layer, respectively.

Step 3: Set the randomized initial synapse weights **Wo**.

Step 4: Estimate the synapse weights by performing SA with a large β according to the gradient $\nabla_w E_p^{\sim}(\mathbf{W}, \beta)$.

Step 5: Perform the conventional BP algorithm using the synapse weights estimated in Step 4.

Step 6: If the total error function E(**W**) by the step 5 is less than the tolerance limit EPV, then stop. Otherwise go to step 4.

4. Experiments and discussions

The proposed method has been applied to the 10 patterns of 8-bit parity check and 6-bit symmetry check problems for evaluating the convergence performances (rate and speed).

The multilayer NN is a 3-layered, feedforward network that is fully interconnected by layers. The range of the initial synapse weights are randomly chosen as [-0.5 ~ +0.5] among layers. The stopping rule is used in each experiment so as to terminate the calculation if all the weights do not change any more or the total error E(W) becomes less than the tolerance limit EPV. The dispersion control parameter β_0=3.0 and the smoothing function h(η) with uniform pdf are chosen, respectively.

The experimental results for each example are shown in tables of 1, 2, and 3, where N_{bp}, and N_{SA} are the number of iterations of BP algorithm and SA algorithm, respectively. E_t is the final error value in termination. t_{bp}, and t_{pm} are the CPU time in [sec] of BP algorithm and proposed algorithm. In table 2, $\tilde{x}$ and σ are mean and standard deviation, respectively.

4.1. Parity check

Parity check refers to the use of parity bits to check that data has been transmitted accurately. The parity bit is added to every data unit (8 bits in this experiment) that is transmitted. The parity bit for each unit is set so that all bytes have an odd number of set bits. If the number of set bits is odd, it sets the parity bit to 1. A tolerance limit EPV for terminating the algorithm sets up 0.0001.

The input and hidden layers are constructed 8 neurons; the output layer is constructed 1 neuron. Fig. 5 shows 10 training patterns and NN's architecture, respectively.

Training patterns	Desired outputs
00000000	1
01000000	0
00000100	0
01010000	1
00001010	1
01010000	1
00010100	1
01010100	0
00101010	0
11111110	0

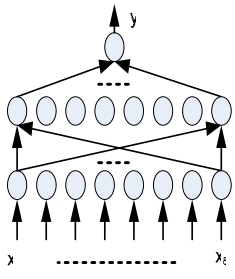

Fig. 5: Training patterns and NN's architecture of parity check.

Table 1 shows the experimental results for the different learning parameters (learning rate α, moment τ). Each row represents the mean of results that only satisfy the stopping rule with the 10 different initial synapse weights. The learning parameters have an influence on the convergence speed of BP algorithm, and the proposed algorithm is influenced by the learning parameters because it also includes the BP algorithm for learning NN. In case of learning rate α=0.5 and moment τ=0.9, the convergence speed is higher than that of another learning parameters. The convergence speed (time) of proposed algorithm is averagely about 53.9 times higher than that of BP algorithm with randomized initial weights setting. We can also know that one cycle of SA algorithm takes more time than BP algorithm. Compared with BP algorithm of the deterministic method, SA algorithm is by reason of stochastic method. But the SA algorithm is executed by little iteration in the proposed algorithm.

α, τ	BP Algorithm			Proposed Algorithm		
	N_{BP}	E_t	t_{BP}	N_{SA}, N_{BP}	E_t	t_{pm}
1.0, 0.0	10424.3	0.0001	4.67	3, 170.5	0.0001	0.08
0.8, 0.8	2475.6	0.0001	1.08	2, 51.8	0.0001	0.03
0.3, 0.7	10128.2	0.0001	6.4	1, 175.7	0.0001	0.11
0.5, 0.9	2050.4	0.0001	0.75	2, 37.3	0.0001	0.02

Table 1: Results of parity check for 10 patterns of 8-bit.

Table 2 represents mean, standard deviation, and convergence ratio of the experimental results for the 100 trials in case of α=0.5 and τ=0.9. Especially, table 2 shows the results that satisfy the stopping rule. N, t, and P_r are the number of iterations, the CPU time, and the convergence ratio. As seen, the convergence rate by the proposed algorithm is about 1.5 times and its convergence speed (time) is about 15.5 times higher than that of the BP algorithm, respectively. The experimental results show that the convergence performances of the proposed algorithm are superior to that of the BP algorithm with randomized initial weights setting. The standard deviation of proposed algorithm is about 1.5 lower than that of the BP algorithm in the CPU time. It means that the proposed algorithm is affected less by the initial weights setting than the BP algorithm.

N, t	BP Algorithm		Proposed Algorithm	
	$\tilde{x}$	σ	$\tilde{x}$	σ
N	806.6	84.2	42.6	35.3
t	0.31	0.03	0.02	0.02
P_r(%)	68		100	

Table 2: Results for 100 trials in α=0.5 and τ=0.9.

4.2. Symmetry check

This task is a problem that checks the symmetry between (n/2) bits and the other half (n/2) bits. In this task, a string of six bits (0 or 1) is judged for mirror symmetry. An output of 1 indicates that the first three digits, when reversed, match the last three digits. An output of 0 indicates that they do not match. A tolerance limit EPV also sets up 0.0001.

The input and hidden layers are constructed 6 neurons; the output layer is constructed 1 neuron. Fig. 6 shows 10 training patterns of 6-bit and NN's architecture, respectively.

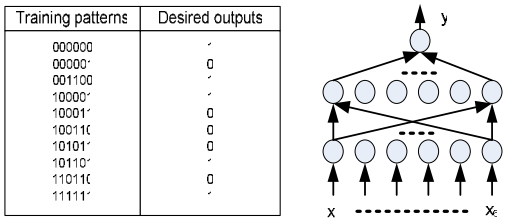

Fig. 6. Training patterns and NN's architecture of symmetry check.

Table 3 shows also the experimental results for the different learning parameters. Each row also represents the mean of the results that satisfy the stopping rule with the 2 different initial synapse weights. The learning parameters have an influence on the convergence speed of BP algorithm and the proposed algorithm, respectively. In case of learning rate α=0.5 and moment τ=0.9, the convergence speed is higher than that of another learning parameters. The convergence speed (time) of proposed algorithm is averagely about 12.1 times higher than that of BP algorithm with randomized initial weights setting. We can also know that one cycle of SA algorithm takes more time than BP algorithm as parity check result.

α, τ	BP Algorithm			Proposed Algorithm		
	N_{BF}	E_t	t_{BF}	N_{SA}, N_{BF}	E_t	t_{Pm}
0.3, 0.7	3730.2	0.0001	1.04	5, 425.2	0.0001	0.12
0.8, 0.8	2143.8	0.0001	0.62	5, 133.3	0.0001	0.04
0.5, 0.9	1994.1	0.0001	0.54	5, 101.7	0.0001	0.03
0.6, 0.7	3416.5	0.0001	0.95	5, 242.9	0.0001	0.07

Table 3. Results of symmetry check for 10 patterns of 6-bit.

Consequently, the convergence rate and convergence speed by the proposed method is higher than that of BP algorithm with randomized initial weights setting. Especially, the proposed method is affected less by the initial weights setting.

5. Conclusions

This paper presents an efficient method for improving the performances of the neural network by global optimization. The method is a hybrid of a stochastic approximation and a gradient descent method. The approximation point inclined toward a global escaping from a local minimum is estimated first by applying the stochastic approximation, and then the BP algorithm is applied for high-speed convergence.

The proposed method is applied to the 10 training patterns of 8-bit parity check and 6-bit symmetry check problems, respectively. The experimental results show that the proposed method has superior convergence performance (rate and speed) to the conventional method that is BP algorithm with randomized initial synapse weights setting.

Our future research is to solve on a large scale learning problems by using neural networks of the proposed method.

References

[1] D. P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation Numerical Method*, Prentice-Hall, London, pp. 1-50, 1989.

[2] A. Cichock and R. Unbehaun, *Neural Networks for Optimization and Signal Processing*, John-Wiley & Sons, New York, 1993.

[3] N. Baba, A New Approach for Finding the Global Minimum of Error Function of Neural Networks. *IEEE Trans. on Neural Networks*, 2: 367-373, 1989.

[4] S. Cho and T.W.S Chow, Training Multilayer Neural Networks Using Fast Global Learning Algorithm - Least-Squares and Penalized Optimization Methods. *Neurocomputing*, 25; 115-131, 1999.

[5] R. Moddemeijer**,** A Fast Heuristic Global Learning Algorithm for Multilayer Neural Networks**.** *Neural Processing Letters*, 9: 177-187, 1999.

[6] B. Samanta, S. Bandopadhyay, and R. Ganguli, Comparative Evaluation of Neural Network Learning Algorithms for Ore Grade Estimation. *Mathematical Geology*, 38: 175-197, 2006.

[7] H. Szu and R. Hartley, Fast Simulated Annealing. *Physics Letters A*, 122: 157-162, 1987.

[8] M.A. Styblinski and T.S. Tang, Experiments in Nonconvex Optimization: Stochastic Approximation with Function Smoothing and Simulated Annealing. *IEEE Trans. on Neural Networks*, 3: 467-483, 1990.