Weight Perturbation for Efficient Learning of Neural Networks

Weight Perturbation for Efficient Learning of Neural Networks

Weight Perturbation for Efficient Learning of Neural Networks

Samkeun Kim, Changwoo Min, and Myungwon Kim

Abstract

The Error Back-Propagation(EBP) algorithm is widely applied to train a multi-layer perceptron, which is a neural network model frequently employed in order to solve complex problems including pattern recognition and adaptive control. However, it suffers from two major problems: local minima and network structure design.

This paper presents a modified error back-propagation algorithm which have the capability to solve the local minima problem and the network structure design in a unified and efficient way. Our algorithm is basically the same as the conventional EBP algorithm except application of stochastic perturbation in order to escape a local minimum. In our algorithm when a local minimum is detected weights associated with hidden units are probabilistically reinitialized and the normal EBP learning is continued with the new set of weights. Addition of a new hidden unit also can be viewed as a special case of stochastic perturbation, i.e., reinitializing all-zero weights of a virtually existing unit. The results of our experiments with several benchmark test problems, the parity problem and the two-spirals problem, including "credit-screening" data, a practical problem of credit card approval, demonstrate that our algorithm is very efficient.

Keywords : Error Back-Propagation, Local Minima, Stochastic Perturbation, Multi-Layer Perceptron, Network Structure Design

I. Introduction

The Error Back-Propagation(EBP) algorithm[1,2] is widely applied to train a multi-layer perceptron, which is a neural network model frequently employed to solve complex problems such as pattern recognition and adaptive control, etc. However, it mainly suffers from two problems. The EBP is basically a gradient descent method, which may get stuck in a local minimum, leading to failure in finding the globally optimal solution. In addition, it is difficult to systematically determine the network structure which is suitable for a given problem[1]. It is usually the case to determine the numbers of hidden layers and hidden units by trial and error.

An *evolutionary* approach using Genetic Algorithm in order to solve the above problems has been attempted by Leung, et al.[3]. Genetic Algorithm(GA) furnishes a global search technique based on simulating the evolutionary behaviour of biological systems. However, using GA does not always insure fast learning. Especially in many practical problems with large scale data, it is inappropriate for applying GA. A constructive approach has been proposed by Fahlman and Lebiere[4]. In their approach learning starts with a network with one hidden unit, and add a new hidden unit whenever the network is trapped in a local minimum. However, this approach suffers from difficulties in determining appropriate parameter values and it is often the case that fails to converge to a global solution in the event that initial hidden units are trained inappropriately.

In this paper, we propose the EBP/SP(Error Back-Propagation/Stochastic Perturbation), a new algorithm to efficiently train a multi-layer perceptron. Our algorithm utilizes stochastic perturbation in weight space to effectively escape from local minima in multi-layer perceptron learning. We probabilistically reinitialize weights associated with hidden units when the EBP learning gets stuck to a local minimum. Addition of new hidden units also can be viewed as a special case of stochastic perturbation. Using stochastic perturbation we have the capability to solve the local minima problem and the network structure design in a unified and efficient way.

Manuscript received February 20, 1998; accepted July 3, 1998.

Samkeun Kim is with the Dept. of Computer Engineering, Ansung National University, 67 Ansung 3-dong, Ansung-city, Kyunggi 456-749, Korea.

Changwoo Min is with the Korea Software Decelopment Institute IBM Korea, Inc., Hanjun Building, 25-11 Yoido-dong, Yeongdeungpo-ku, Seoul 150-010, Korea.

Myung Won Kim is with the School of Computing, 1-1 Sangdo 5-dong, Dongjak-ku, Seoul 156-035, Korea.

II. The EBP/SP Learning Algorithm

1. Hidden Units in a Multi-Layer Perceptron

Consider the role of hidden units as *feature detectors* in a multi-layer perceptron [2]. Hidden units identify *features* hidden in the input data and it is important to adjust hidden units to identify appropriate features to solve a given problem using a multi-layer perceptron.

The basic idea underlying stochastic perturbation is that we reinitialize weights of those hidden units which are evaluated to be insignificant in the network. Hidden units of which associated weights to reinitialize are selected with the probability depending on their significance in the network. The mean square error E is defined as follows[1]:

$$E=\frac{1}{P}\sum_{p}^{P}E_p=\frac{1}{PN_o}\sum_{p}^{P}\sum_{k}^{N_o}(T_{pk}-O_{pk})^2, \tag{1}$$

where P = the number of training patterns,

N_o = the number of output units,

T_{pk} = the desired output value of the k-th output unit for pattern p

O_{pk} = the actual output value of the k-th output unit for pattern p.

For a sufficiently trained network the strength of unit i can be defined as

$$S(i)=E'(i)-E, \tag{2}$$

where $E'(i)$ be the mean square error evaluated by the network with unit i eliminated[5]. $S(i)$ can, at this point in time, be interpreted as a measure of the significance of unit i in which the unit plays in the network. In the event that $S(i)$ is near to 0, it means that hidden unit i plays a nominal role, while $S(i)$ is near to 1, it means that the hidden unit plays an important role in the network.

2. Stochastic Perturbation

The basic idea underlying our algorithm is to perturb stochastically weights associated with hidden units in the weight space and add stochastically new hidden units when the EBP learning gets stuck to a local minimum. Fig. 1 illustrates the EBP/SP algorithm.

The EBP learning phase in Figure 1 is the same as the conventional EBP learning[1]. In the figure, step (1) checks the termination condition. In the event that the error E in eq. (1) becomes smaller than the error tolerance E_{tol}, learning terminates. The following step (2) detects a local minimum trapped during learning by the EBP. The EBP is basically a gradient descent method, thus it may get stuck in a local minimum, leading to failure in finding the globally optimal solution. We detect a local minimum as follows[4]: First, we compute the error gradient δE by $|E(t)-E(t-\tau)|/\tau$, where τ is a time lag(in the number of epochs) for estimating δE, and t is the current epoch. If $\delta E<\lambda$ and $E>E_\theta$ ($\geq E_{tol}$), then we determine that the EBP learning is trapped in a local minimum, where λ is a *patience* parameter and E_θ the threshold of error, both given by the user. Here, E_θ is used to enhance performance by prohibiting unnecessary perturbation when E becomes close enough to E_{tol}, as used in Leung, et al.[3]. For our algorithm, λ is the parameter that governs the determination of local minima. That is, if λ is as large as infinite, our algorithm manifests the behavior of simulated annealing[6,7], since it will be determined to be local minima at any step. In contrast to this, zero-value of λ corresponds to the EBP learning algorithm. Also, if the error threshold E_θ is enough as large as initial system error in eqn (1), our algorithm corresponds to the EBP learning algorithm. Just as for λ, zero-value of E_θ ($E_{tol}=0$) corresponds to our EBP/SP algorithm.

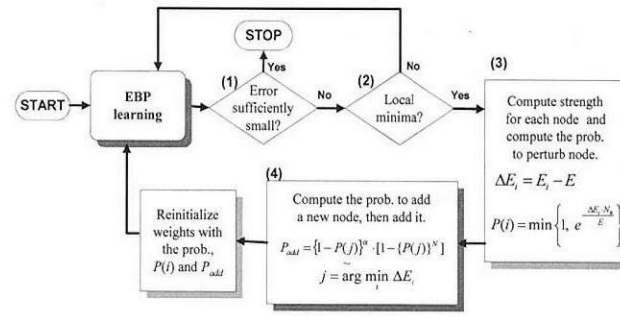


Fig. 1. The control flow diagram of the EBP/SP algorithm

Next, if the network is evaluated to be trapped in a local minimum at time t, step (3) is applied to escape the local minimum. Consider stochastic perturbation of weights associated with hidden units. The probability that each hidden unit to be perturbed is determined by:

$$P(i)=\min\left\{1, e^{-\frac{\Delta E_i \cdot N_h}{E}}\right\}, \tag{3}$$

where ΔE_i = $E_i - E$

E_i = the error evaluated by the network with unit i,

E = the residual error of the network stuck to a local minimum,

N_h = the number of hidden units.

Here, one might notice the similarity between eq. (3) and the Metropolis criterion in simulated annealing. ΔE_i corresponds to the energy change $\Delta\varepsilon$ while E corresponds to temperature T in simulated annealing. At the present time eqn (3) allows a *coarse* search in weight space for large

residual error while it performs a *fine* search for small residual error. We note that the strength $S(i)$ of unit i approximates ΔE_i provided that the weights are reinitialized to be sufficiently small. In eqn (3), we note that the larger the strength is, the smaller the probability is perturbed. Also, in the event that the residual error is large, then a considerable number of hidden units are perturbed, however, if it is small, then a small number of hidden units are perturbed. When E is large and $S(i)$ is small, our algorithm makes all the more global search in the weight space.

At the present time we will take into consideration adding new hidden units as a special case of stochastic perturbation. The probability $P(i)$ obtained by eqn (3) is also used to add new hidden units. First, consider the probability to add a new hidden unit. We first find a hidden unit j of the minimum strength. The probability to add a new hidden unit can be approximated to the probability that the hidden unit j of the minimum strength fails to be perturbed, $1-P(i)$.

Next, take into consideration a sequence of trials with probability $1-P(i)$ of success. This sequence is observed until the first success occurs and we let K denote the number of failures prior to this first success. Thus, the number of failures, K, follows a *geometric distribution*[8]. Here we also need to take the number of failures into consideration in determining the probability that a new hidden unit is added. At this point in time we have the probability to add a new hidden unit at the N-th trial after N-1 successive failures is given by

$$\begin{aligned} P_{add} &= P(K<N) \\ &= \sum_{K=0}^{N-1}\{P(j)\}^K\{1-P(j)\} \\ &= 1-\{P(j)\}^N. \end{aligned} \tag{4}$$

Note that the larger N is, the larger the probability to add a new hidden unit. However, we would like to slightly modify eqn (4) to control the resistance to adding a new hidden unit and we have:

$$P_{add}=\{1-P(j)\}^{\alpha}[1-\{P(j)\}^N], \tag{5}$$

where α is the *resistance* parameter controlling the difficulty with which a new hidden unit is added. If addition of a new hidden unit is determined by eqn (5), then a new hidden unit is added by reinitializing its associated all-zero weights by small random numbers near zero.

3. Time Complexity

The time complexity of our algorithm is approximately equal to that of the conventional EBP algorithm. Noting that the EBP/SP is approximately the same as the EBP except it takes an additional time for computing ΔE_i for each hidden unit and computing ΔE_i for a unit only takes about half of the time of a epoch, we have that the EBP/SP takes $n+rN_h/2$ (in the number of epochs) where n and r represent the number of epochs and the number of stochastic perturbations, respectively. However, it is usually the case that r is sufficiently small compared to n, the EBP/SP takes a nominal amount of additional time compared to the conventional EBP.

III. Simulation Results and Discussions

We experimented our algorithm with the parity problem and the two-spirals problem, including the "credit-screening" problem, a practical problem of credit card approval. For fast training we adopt a common practice of using only the relatively linear portion of the sigmoid function. In all our experiments, we set two desired output values to 0.1 and 0.9, respectively, instead of 0 and 1. For comparison, simulations were performed using three different algorithms; the EBP, the EBP#, and the EBP/SP. The EBP# corresponds to the case of EBP training of the network with the same number of hidden units as the average number of hidden units resulted from the EBP/SP. Also "Ng" in the Tables 1 and 2 represents simulations by Ng, et. al.[3]. The results presented were obtained by the parameter values recommended by the authors of the corresponding algorithm[3]. It can thus be expected that the parameter values were chosen almost optimally. In "Ng", the initial weights are drawn at random from uniform distribution between -0.3 and 0.3. And the mutation probability(P_m), offset range(D), and the number of offsprings($POPSIZE$) are set as 0.8, 0.5 and 5, respectively.

1. The Parity Problem

Table 1. Results of 4-bit, 5-bit, and 6-bit parity problems using 3 different algorithms.

The N-bit Parity problem		No. of tests	Avg. no of hidden units	Avg. no. of epochs	Average performance (%)
4-bit	EBP	5	3	81097	97.5
	EBP#	5	4	74139	96.3
	EBP/SP†	5	4	22424	100 0
	Ng	5	4	-	98.75
5-bit	EBP	5	3	100000	96.9
	EBP#	5	5	100000	96.9
	EBP/SP†	5	4.7	36370	100.0
	Ng	5	5	-	99.38
6-bit	EBP	5	4	100000	94.4
	EBP#	5	6	58074	99.4
	EBP/SP†	5	6 2	40973	100.0
	Ng	5	6	-	99.69

†Commences with 2 hidden units.

We have experimented the N-bit parity problem with N ranging from 4 to 6. The learning rate and the momentum rate were 0.1 and 0.8, respectively. When the residual error decreased to less than 0.002, we determine that learning converged. Whether or not the network converged, the learning procedure terminated after 100,000 epochs for 4-bit, 5-bit, and 6-bit parity problems. For each problem we performed several tests with different sets of initial weights and obtained an average performance as shown in Table 1.

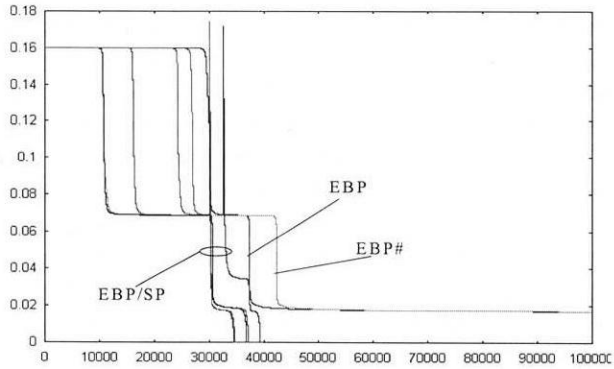


Fig. 2. Representative results for the 5-bit parity problem.

In all cases, by the EBP/SP the network was trained to classify patterns perfectly. Fig. 2 illustrates how effectively our algorithm escapes from a local minimum compared to other algorithms. However, we observed that the EBP/SP did not well escape from local minima in *premature saturation*, which often occurs at an early stage of learning[9]. We perform not yet fully understand the reason for this behaviour.

2. *The Two-Spirals Problem*

We applied our algorithm to a considerably large network for the two-spirals problem, which is an extremely hard problem for algorithms of the conventional error back-propagation family to solve. The network has two continuous-valued inputs and a single output. The training set consists of 194 X-Y value pairs, a half of which are to produce the output 0 while the other half the output 1. These training points are arranged in two interlocking spirals that go around the origin three times.

We commenced the EBP/SP algorithm with 20 hidden units in the network, since rather a large number of hidden units would be needed to solve the problem. The learning rate and the momentum rate were 0.1 and 0.0, respectively. In this case we set the error tolerance to 0.002, and whether or not the network converged, the learning procedure terminated after 200,000 epochs. Fig. 3 demonstrates how effectively our algorithm escape from a local minimum compared to other algorithms.

Table 2. Results of the two-spirals problem using three different algorithms.

The two-spirals problem	No. of tests	Avg. no. of hidden units	Avg. no. of iterated epochs	Average performance (%)
EBP	5	20	200000	87.1
EBP#	5	26	200000	98.0
EBP/SP	5	26	105988	100.0
Ng	5	26	-	93.41

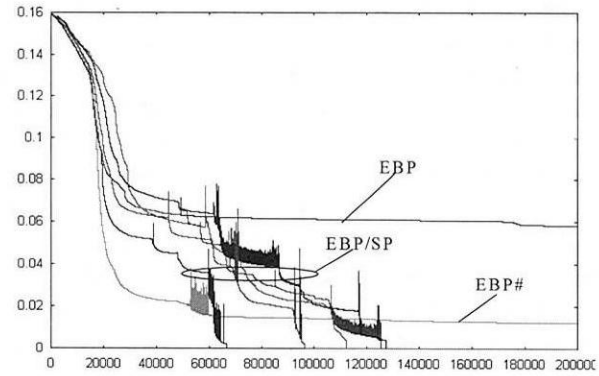


Fig. 3. Representative results for the two-spirals problem.

Also, in eqn (5), note that the larger α is, the simpler the network structure we get. All calculations in Table 2 were performed using the resistance parameter of 2. From additional simulations with smaller(or larger) α's, we obtained larger(or smaller) network structures. However, the network trained with a large α slows down convergence. Fig. 4 illustrates the convergence rate and the complexity of resulting network, depending on the magnitude of α.

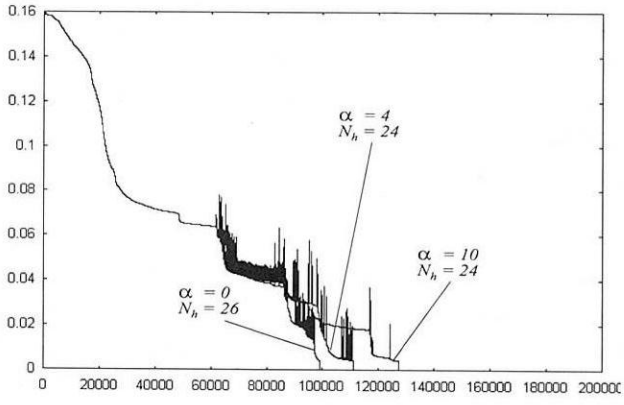


Fig. 4. The convergence rate and the complexity of resulting network, depending on the magnitude of α.

3. The "credit-screening" Problem

In training a network, the minimum error may produce an overtrained network, which may not be desirable. We investigated improving the generalization capability of a multi-layer perceptron using our algorithm by introducing an appropriate measure for generalization capability into the error function[2,10,11]. Here we utilize the weight decay term[2] into the error function.

In an additional experiment with the "credit-screening" data, a practical problem of credit card approval, our algorithm outperformed the conventional EBP algorithm. The class distribution of this data consists of 307 cases(44.5%) granted and 383 cases denied(55.5%). There are 15 input attributes of very different kinds. For 7 of them, there are a few missing values(in 37(=5%) of the records). All these misses are explicitly encoded(using 51 instead of 43 inputs).

Table 3 demonstrates the results that six different algorithms are applied, where utilize 345 training and 175 test examples. In all simulations, the learning rate is 0.1, and the momentum term 0.8. When the residual error E decreased to less than 0.002, the network was considered to have converged. Whether or not the network converged, the learning procedure terminated after 10,000 epochs. Table 3 describes the statistics of runs for each learning method.

Table 3. Results of the "credit-screening" problem using 3 different simulations without weight decay term and 3 different simulations with weight decay term.

The two-spirals problem	Avg. no. of hidden units	Average performance(%)			
		Train		Test	
		Avg.	Std. dev.	Avg.	Std. dev.
EBP	2	95.59	0.130	83 02	0.635
EBP with WD*	2	95.24	0.440	84 07	0.884
EBP#	3	97.45	0.629	81.74	1.058
EBP# with WD*	3	95.77	1.941	84.07	1.400
EBP/SP†	4.4	97.68	2.418	79 42	2.893
EBP/SP† with WD*	3	96.75	1.185	85 35	0.865

*Means that includes the weight decay term.
†Commences with 2 hidden units.

Due to the fact that the weight decay term was included into the error function, the classification capability for the training data was deteriorated. In contrast to this, the classification performance for the test data was improved. Thus, it is our understanding that we may improve the generalization capability of a multi-layer perceptron using our algorithm by introducing an appropriate measure for generalization capability(for example, the weight decay term) into the error function.

IV. Conclusions

In this paper, we propose a new learning algorithm, which solves the local minima problem and determines systematically the network structure appropriate for a given problem. Our algorithm uses stochastic perturbation in weight space to escape effectively from local minima in multi-layer perceptron learning. Particularly, we generalize stochastic perturbation to encompass addition of new units, which allows us to deal with it in a unified way. Stochastic perturbation reinitializes weights associated with hidden units to escape from a local minimum when the EBP learning gets stuck to it. Addition of new hidden units also can be viewed as a special case of stochastic perturbation. We demonstrated that using stochastic perturbation we can solve the local minima problem and the network structure design in a unified and efficient way. As a result our algorithm significantly improves the convergence speed of the network. In our algorithm, addition of new units can be omitted simply to train a fixed size network. In this case our algorithm also outperforms the conventional EBP.

In training a network, the minimum error may produce an overtrained network, which may not be desirable. We demonstrated that, by introducing an appropriate measure for generalization capability into the error function, we can improve the generalization capability of a multi-layer perceptron using our algorithm.

Further investigations include extending the algorithm so that it is applicable to learning of other neural network models, and theoretically proving its convergence.

Acknowledgement

This research has been supported by Korea Research Foundation

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning, "Internal Representation by Error Propagation," in *Parallel Distributed Processing*, J. A. Feldman, P. J. Hayes, & D. E. Rumelhart(Eds.), pp.318-362, 1986.

[2] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," in *Backpropagation - Theory, Architectures, and Applications*, Y. Chauvin & D. E. Rumelhart(Eds.), pp.1-34, 1995.

[3] S. C. Ng, S. H. Leung and A. Luk, "Evolution of Connection Weights Combined with Local Search for

Multi-layered Neural Networks," IEEE, pp.726-731, 1996.

[4] S. E. Fahlman, and C. Lebiere, "The Cascade-Correlation Learning Architecture.," Dept. of Computer Science, Carnegie Mellon University, *Technical Reports*, CMU-CS-90-100, 1991.

[5] David J. Montana and Lawrence Davis, "Training Feedforward Neural Networks Using Genetic Algorithms," IJCAI-89, pp. 762-767, 1989.

[6] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science 220, pp.671-680, 1983.

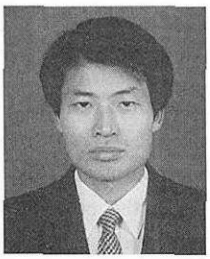
[7] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Macmillan College Publishing Co., Inc., pp.187-189, 1994.

[8] R. V. Hogg and E. A. Tanis, "Probability and Statistical Inference," Macmillan Publishing Co., Inc., 1977.

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

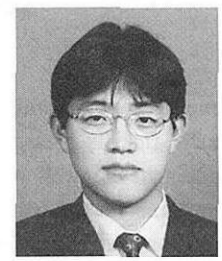
[10] C. E. Rasmussen, "Generalization in Neural Networks," M.S. thesis, Dept. of Mathematical Modeling, Technical University of Denmark, 1993.

[11] L. Prechelt, "Adaptive Parameter Pruning in Neural Networks," TR-95-009, International Computer Science Institute, 1995.



Samkeun Kim received the B. S. degree in computer science and statistics from Busan National University in 1985, and the M. S. and Ph. D. degrees in computer science from Soongsil University, Seoul, Korea in 1988 and 1998, respectively. He is currently an Assistant Professor in the Department of Computer Engineering in Ansung National University. His current interests are in the areas of neural networks, genetic algorithms, global optimization.

Myung Won Kim graduated from Seoul National University, Seoul Korea with B.S. in Applied Mathematics 1972. He received an M.S. and Ph.D. in Computer Science from University of Massachusetts at Amherst, U.S.A. in 1981 and University of Texas at Austin, U.S.A. in 1986, respectively. He worked with AT&T Bell Laboratories at Naperville, Illinois, U.S.A. as Member of Technical Staff from 1985 to 1987. He superivised a neural network research group at Electronics and Telecommunications Research Institute in Korea from 1987 to 1994. He was also President of Korean Neural Network Research Group during 1990 - 1992. He is currently an associate professor of School of Computing, College of Information Science, Soongsil University, Korea. He is also Vice President of Korean Society for Brain Science, and is Director of Information and Media Technology Institute at Soongsil University. His research interests include neural network, fuzzy system, genetic algorithm, and pattern recognition.



Changwoo Min received the B.S. and M.S. degrees in computer science from Soongsil University, Seoul, Korea in 1996 and 1998, repectively. He is currently a researcher in IBM Korea. His research interests are neural networks, genetic algorithms, artificial life, data mining and natural language processing.