Technical Note

An Empirical Study of Generalization of Backpropagation Using Speed-up Techniques

JOSE LUIS PEREZ,† YUKINORI SUZUKI† and ICHIRO SUGIOKA†

Flat spots are known to be one cause for slow training in backpropagation (BP). One proposed technique to eliminate flat spots is the sigmoid prime offset. However, we have found that the sigmoid prime offset can greatly deteriorate generalization of the resulting network. Therefore, we propose techniques not only to eliminate flat spots, but also to retain the generalization ability to be as high as standard BP. Simulations show that the proposed techniques are effective for both speed-up and generalization.

1. Introduction

The backpropagation (BP) algorithm is very popular for training neural networks. Despite its popularity, one of the problems with BP is that it may be slow to train the network, e.g. many presentations of the training patterns (epochs) may be needed to complete training. One cause of slow training is the magnitude of the components of the gradient vector[1]. The derivative of the weight is small in magnitude in the area where the error surface is fairly flat along a weight dimension. In this area, the value of the weight is adjusted by a small amount and many steps may be required to achieve a significant reduction in error. This is a "flat spot." To reduce the effect of flat spots, many speed-up techniques have been proposed[2]. These include the well-known momentum technique, variable learning rates, determining the optimal descent direction, calculating initial weight values, and selecting a proper training set. However, since most of these techniques are complicated and/or increase the need for memory as the network becomes larger, utilization of these techniques is not simple nor practical for use in many real-world applications. Fahlman has proposed the sigmoid prime offset to eliminate flat spots[3]. Fahlman's technique to eliminate flat spots is easy to implement and very effective. However, as we will illustrate later, Fahlman's technique can decrease the generalization ability of the resulting network when his technique is used for weights connected to hidden nodes. Generalization ability is one of the most important abilities of neural networks. Networks with a low generalization ability cannot reliably recognize patterns which were not used during training, even if the patterns are very similar. Therefore, we will propose two techniques which not only eliminate flat spots, but also keep the generalization ability as high as standard BP. The simulation shows that the proposed techniques are effective for both speed-up and generalization.

† Department of Computer Science and Systems Engineering, Muroran Institute of Technology

2. Flat Spot Problem

A feed-forward neural network has a layered structure where each layer consists of nodes. A node will receive inputs from previous layers' nodes through connections or from an external source and use this information to compute its output. BP is an algorithm that can be used to train this type of neural network by modifying the weights to minimize an error function. In BP, for every weight a gradient is calculated for every input pattern, and this gradient dictates how each weight should be changed to minimize the error function.

Rumelhart, et al.[4] show that the weight update is carried out as

$$\Delta w_{ji}(n) = -\eta \sum_p \frac{\partial E_p}{\partial w_{ji}} \tag{1}$$

where E_p is the error for a particular pattern, w_{ji} is the weight between node j and node i, n is the nth epoch, and η is the learning rate. Rumelhart, et al. also show that the weight update according to Eq. (1) is carried out as

$$\Delta w_{ji} = -\eta o_j \delta_j \tag{2}$$

where o_j is the actual output for node j, and δ_j is the error term which is backpropagated. If we use a sigmoid function, δ_j is simply calculated for weights to output nodes as

$$\delta_j = o_j(1 - o_j)(t_j - o_j) \tag{3}$$

where t_j is the target output for node j.

In BP, the error term δ_j contains the term $o_j(1 - o_j)$ which is called the sigmoid prime function. The sigmoid prime function has a

value of zero where o_j is zero or one. It can be easily seen that for outputs which give a value of nearly 1.0 or 0.0, only a tiny fraction of the error will be passed back. This is regardless of whether the actual output should be zero or one. When outputs which represent a nearly maximum possible error are multiplied by the sigmoid prime function, δ_j becomes very small, leading to very small weight updates. This is the cause of flat spots.

3. Techniques to Improve Training Speed with High Generalization

Fahlman's technique to eliminate flat spots is called the sigmoid prime offset and it is done by simply adding a constant 0.1 to the sigmoid prime function. This technique is simple, but effective. Using this technique, the sigmoid prime function never falls below 0.1, and it is quite effective for speed-up training. Perekh, et al. reported that the sigmoid prime offset is the most successful among flat spot elimination techniques[5]. However, as we shall show in Section 5, we have found that the sigmoid prime offset greatly deteriorates generalization when the sigmoid prime offset is used to train weights which lead to hidden nodes.

We propose two techniques which improve training speed but do not have a detrimental effect on generalization. One technique is for weights which connect to the output nodes, and the other is for weights which connect to hidden nodes.

The first technique is a modification of the sigmoid prime function and is used only for weights which connect to output nodes. The sigmoid prime function is clamped at its maximum for output nodes with an error greater than 0.5. This technique is used in conjunction with the sigmoid prime offset as shown in **Fig. 1** and is calculated as

$$\begin{aligned} &\text{sigmoid prime function} = o_j(1-o_j)+0.1 \\ &\qquad \text{when } |t_j - o_j| < 0.5, \\ &\text{sigmoid prime function} = 0.35 \\ &\qquad \text{when } |t_j - o_j| > 0.5. \end{aligned} \tag{4}$$

Figure 1(a) shows the modified sigmoid prime function for target values of zero, while Fig. 1(b) shows the modified sigmoid prime function for target values of one. Using this modification, the sigmoid prime function stays at its maximum of 0.35 whenever an error becomes larger than 0.5. This is in contrast to the original sigmoid prime function which returns a relatively small value for large errors (much greater than 0.5). This modification does not approach zero for large errors, therefore it eliminates flat spots.

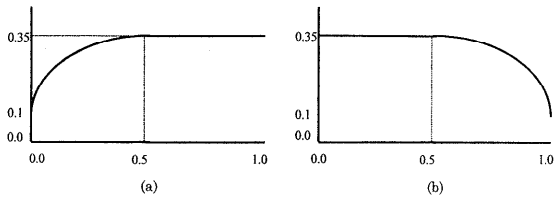

Fig. 1 Modified sigmoid prime function using clamping and 0.1 offset. (a) is for target values of zero, (b) is for target values of one.

Since we have seen that a constant sigmoid prime offset for weights to hidden nodes deteriorates generalization, we investigated a number of techniques which dynamically change the offset. The second technique we propose is a dynamic sigmoid prime offset which showed the best results and it is used only for weights which connect to the hidden nodes. The dynamic offset was added to the sigmoid prime function and was changed in proportion to the number of patterns correctly identified during training. The dynamic offset speeds up training of BP, with no apparent deterioration of the resulting generalization ability. The dynamic offset was calculated as

$$\text{offset} = A\left(1 - e^{-B(M-m)}\right) \tag{5}$$

where A and B are constants, M is the total number of patterns, and m is the number of correctly identified patterns. In the case of all patterns being incorrectly classified, the maximum offset, which is limited by A, is added to the sigmoid prime function. This offset remains close to A until more than half patterns are correctly identified. When nearly all patterns are correctly classified, the offset decreases rapidly. Finally, the minimum offset, which is limited by B, is reached when all patterns are correctly classified. This type of dynamic offset showed better results than a lincarly or an exponentially decreasing offset.

4. Simulations

We used the soybean problem in PROBEN1[6] which is a problem to recognize 19 different soybean diseases. We used this problem because it has been used often in machine learning literature, it is an example of real world data, and it is available for other researchers to use for easy comparison with their proposals. There is a total of 683 patterns. We used three sets of data for training and testing the network. The

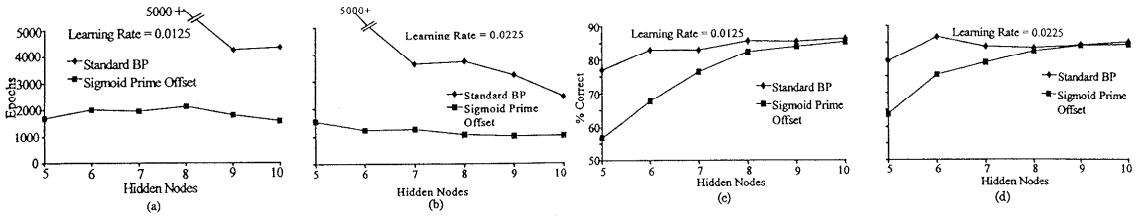

Fig. 2 Comparison of standard BP and BP using the sigmoid prime offset. (a) and (b) show training speed, (c) and (d) show the generalization ability.

first set of data is called the training set and consists of 342 patterns. It was used to change the weights of the network. The second set of data is called the validation set and it consists of 171 patterns. It was used to determine when to stop training. The third set of data is called the test set and it consists of 170 patterns. It was used once after training to give the actual generalization results. Training speed was evaluated by the number of epochs needed to train the network. Generalization was evaluated by the rate of correct recognition of the test set.

For the dynamic offset shown in Eq. (5), A was chosen to be 0.1, which was the maximum offset. B was chosen to be 0.000029239, this value was calculated so that when all patterns are correctly identified, the offset was very small (here, the minimum offset is less than 0.001).

We employed two layered neural networks. This means one layer of variable weights between the input layer and hidden layer, and one layer of variable weights between the hidden layer and output layer. There were 82 input nodes and 19 Boolean output nodes. The number of hidden nodes were varied from five to ten. We used the "threshold and margin" criterion in which any output which is 0.4 or below is considered a zero, and any output which is 0.6 or above is considered a one. Before training, all weights were set to be random numbers between -0.5 and 0.5, and each trial used different random weights. To avoid overtraining, training was stopped when there was no reduction in the validation set error for one hundred epochs. Weights which produced the lowest error for the validation set were used for testing the generalization ability of the test set.

5. Results and Discussion

We made simulations according to the techniques described in Section 4. The learning rate was changed from 0.0125 to 0.04 and the results in which the learning rate was 0.0125 and 0.0225 are shown here since the results of other simulations were similar (the value 0.04 was close to the largest practical learning rate for this problem).

The top two panels (a and b) in **Fig. 2** show simulation results in which training speed of standard BP is compared with the training speed of BP with the sigmoid prime offset. Training with the sigmoid prime offset was much faster than standard BP. These panels show that the sigmoid prime offset is effective for speed-up training.

The lower two panels (c and d) in Fig. 2 show simulation results in which the generalization ability of standard BP is compared with the generalization ability of BP with the sigmoid prime offset. These panels show that generalization was greatly deteriorated by the sigmoid prime offset. This deterioration is especially notable in smaller network size. Where the number of hidden nodes is five and the learning rate is 0.0125, the generalization ability of the network trained with the sigmoid prime offset is deteriorated to about 55% recognition, while standard BP attains about 75% recognition. The deterioration of networks trained with the sigmoid prime offset becomes smaller as the network becomes larger. This is expected as generalization (usually) increases as the network becomes larger, and there are redundant nodes in the hidden layer. Since generalization is one of the most important abilities for practical applications, the sigmoid prime offset is not effective for those applications which require high generalization.

Our speculation is that speed-up techniques that are used for weights to the output nodes do not deteriorate generalization. The teacher signal is given to the output nodes and the weights from the hidden nodes to the output nodes are updated in proportion to the error. The weight updates for one output node do not affect the output of other output nodes. Because of this, large weight updates by speed-up techniques for weights to the output nodes do not deteriorate generalization. Conversely, the input signal from the input to hidden nodes will go through weights, then the signal will go to

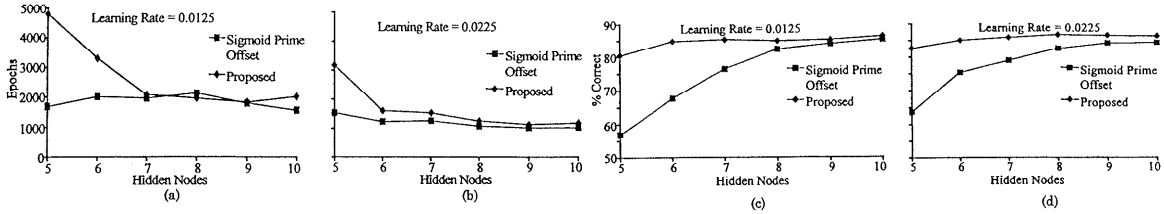

Fig. 3 Comparison of BP using the sigmoid prime offset and BP using the proposed techniques. (a) and (b) show training speed, (c) and (d) show the generalization ability.

all output nodes. When one weight to a hidden node changes, all output nodes are affected by that change. One output node may decrease in error, while another may increase. BP makes adjustments for this to decrease total error. Therefore, we speculate that large updates in weights to hidden nodes affects generalization. Since speed-up techniques make large (larger than standard BP) weight updates, error is decreased quickly, but a point where good generalization is attained may not be found. Because of this, when using the constant sigmoid prime offset for weights to the hidden nodes, the resulting generalization is less than standard BP.

The top two panels (a and b) in **Fig. 3** show the training speed of the proposed techniques compared with the sigmoid prime offset. The training speed of the sigmoid prime offset was faster than the training speed of the proposed techniques when the number of hidden nodes was five and six with a learning rate of 0.0125, however the proposed techniques were faster than standard BP with equivalent (or better) generalization ability. Except for these, the training speed of the proposed techniques is practically the same as the training speed of the sigmoid prime offset. These results show that the proposed techniques are effective to improve the training speed of BP.

The lower two panels (c and d) in Fig. 3 show the generalization ability of the proposed techniques compared with the generalization of the sigmoid prime offset. The simulation results show that the generalization of the networks which utilized the proposed techniques is much greater than the generalization of the networks which utilized the sigmoid prime offset. For example, when the number of hidden nodes is five and the learning rate is 0.0125, the resulting generalization for the proposed techniques is about 80% recognition while the resulting generalization of the sigmoid prime offset is around 55% recognition. This also shows that the generalization of the proposed techniques is slightly better than the generalization of standard BP.

6. Conclusion

In conclusion, the proposed techniques are simple but the simulations show that the techniques are effective in reducing training time and do not deteriorate generalization. In the proposed techniques, the weights to the output nodes are updated by large amounts by the clamping technique, and its main contribution is for speed-up training. The weights to the hidden nodes are updated using the dynamic offset, and simulations show that these speed-up techniques do not deteriorate generalization. Therefore, these techniques are practical for real-world applications.

References

1) Jacobs, R.: Increased Rates of Convergence Through Learning Rate Adaptation, *Neural Networks*, Vol.1, pp.295–307 (1988).
2) Takagi, H.: Neural Networks, Part II: Introduction for Beginners, *Journal of Japan Society for Fuzzy Theory and Systems*, Vol.4, No.4, pp.664–675 (1992).
3) Fahlman, S.E.: An Empirical Study of Learning Speed in Backpropagation Networks, Technical Report, CMU, CMU-CS-88-162 (1988).
4) Rumelhart, D.E., Hinton, G.E. and Williams, R.J.: Learning Representation by Back-propagating Errors, *Nature*, Vol.323, No.9, pp.553–536 (1986).
5) Parekh, R., Balakrishnan, K. and Honavar, V.: An Empirical Comparison of Flat-Spot Elimination Techniques in Back-propagation Networks, *Proc. Simmtec/WWW'92*, pp.463–468 (1992).
6) Prechelt, L.: PROBEN1 – A Set of Neural Network Benchmark Problems and Benchmarking Rules, Technical Report of University of Karlsrule 21/92 (1994).

(Received April 9, 1997)
(Accepted July 1, 1997)