

AN EMPIRICAL COMPARISON OF FLAT-SPOT ELIMINATION TECHNIQUES IN BACK-PROPAGATION NETWORKS

Rajesh Parekh, Karthik Balakrishnan, & Vasant Honavar
Department of Computer Science
Iowa State University.
Ames, Iowa 50011

ABSTRACT

Back-Propagation (BP) [Rumelhart et al, 1986] is a popular algorithm employed for training multi-layer connectionist learning systems with non-linear activation function (sigmoid). However, BP is plagued by excruciatingly slow convergence for many applications, and this drawback has been partly attributed to the *Flat-Spots* problem. Literature defines flat-spots as regions where the derivative of the sigmoid activation function approaches zero, and in these regions the weight changes become negligible, despite the presence of considerable classification error. Thus learning slows down dramatically. Several researchers have addressed this problem posed by flat-spots [Fahlman, 1988, Balakrishnan & Honavar, 1992]. In this paper we present a new way of dealing with the flat-spots in the output layer. This new method uses a Perceptron-like weight-modification strategy to complement BP in the output layer. We also report an empirical evaluation of the comparative performances of these techniques on some data-sets that have been used extensively in benchmarking inductive learning algorithms.

1 Introduction

Neural network research has received a tremendous boost with the development of several multi-layer learning algorithms. Amongst them the *Generalized Delta Rule* (or more commonly BP) is probably the most popular one in use today. BP suffers from extremely slow learning for complex problems. Flat-Spots have long been known to be responsible for the slowness of BP [Fahlman, 1988]. This paper systematically compares the performance of three methods for dealing with flat-spots :

1. Sigmoid-prime offset [Fahlman, 1988]
2. Scaled Linear Approximation of sigmoid [Balakrishnan & Honavar, 1992]
3. Perceptron-based strategy that is proposed in this paper. This technique calculates the weight-change required to get out of flat-spots by *inverting the sigmoid* and making the weight-changes using a variation of the perceptron rule [Rosenblatt, 1958], namely the *fractional correction method* [Nilsson, 1965].

2 Back-Propagation - an Overview

Back Propagation involves minimization of an error function which is accomplished by performing gradient descent search on the error surface. With the conventional *mean squared error* function, and with an additional term to *dampen* oscillations, the BP weight-update rule can be shown to be:

$$\Delta w_{ji}(n+1) = \eta o_i \delta_j + \alpha \Delta w_{ji}(n) \quad (1)$$

where Δw_{ji} is the change in the weight connecting unit i to unit j , η is the learning rate (a small positive constant), o_i is the output of unit i , α is the momentum term and the n indicates the epoch of the pattern presentation sequence. δ_j is the error associated with unit j , commonly known as the *instantaneous gradient*, and is calculated as

$$\delta_j = \begin{cases} o'_j(t_j - o_j) & \text{if unit } j \text{ is an output unit} \\ o'_j \sum_k w_{kj} \delta_k & \text{if unit } j \text{ is a hidden unit} \end{cases} \quad (2)$$

In this equation o_j' stands for the derivative of the output (also called *sigmoid prime*) and t_j is the target output of unit j . Assuming a *Sigmoid Activation* function for the units, the derivative evaluates to

$$o_j' = o_j(1 - o_j) \quad (3)$$

2.1 Flat Spots

The derivative of the Sigmoid, given by Equation 3, is seen to approach zero as the output of the corresponding unit approaches a 0 or a 1. For unit outputs sufficiently close to 0 or 1, the derivative is almost zero, and these regions are referred to as *flat-spots* in the literature. Since the derivative value is used in calculating δ_j (as given by Equation 2), the weight-update equation implies negligible weight-changes in the presence of flat-spots. This slows down learning as units stuck in the wrong state (and hence in the flat-spot) take an inordinately long time to get out of the flat-spot, thereby slowing down the learning process.

3 Methods for Handling Flat-Spots

3.1 The Sigmoid Prime Offset

[Fahlman, 1988] proposed that the flat-spots problem could be solved simply by preventing the derivative of the sigmoid function from going to zero; this is achieved by adding an offset of 0.1 to it. This simple change has the effect of making the value of the sigmoid prime function, given in equation 3 to lie between 0.1 and 0.35 rather than 0.0 and 0.25. With this modification there are no flat-spots as the sigmoid prime function never goes to zero. Experiments show that this small offset results in a two-to-ten fold speedup on some benchmark problems [Fahlman, 1988].

3.2 A Scaled Linear Approximation of Sigmoid for Error Calculation

[Balakrishnan & Honavar, 1992] proposed the use of a different error function to tackle flat-spots in the output layer. The modification is based on representing the error function as

the *mean sum-squared error* over the *inputs* of the output layer rather than over the outputs as is conventionally done in BP. This results in the following error function

$$E = \frac{1}{2} \sum_p \sum_i (net_target_i^p - net_obtained_i^p)^2 \quad (4)$$

with $net_target_i^p$ and $net_obtained_i^p$ representing the desired and obtained values at the *input* of output unit i , for the p th pattern. Using a straight line approximation of the sigmoid they derived the following equation for the error δ_j

$$\delta_j = \begin{cases} \frac{(t_j - o_j)}{o_j} & \text{if unit } j \text{ is an output unit} \\ o_j' \sum_k w_{kj} \delta_k & \text{if unit } j \text{ is a hidden unit} \end{cases} \quad (5)$$

The derivative of the sigmoid function now appears in the denominator for output layer units. There are no flat-spots, in the conventional sense, however this leads to the problem of the *vanishing denominator* (for a solution to this problem the reader is referred to [Balakrishnan & Honavar, 1992]). Note that with this formulation the weight-changes occur whenever necessary and are not swamped by the presence of flat-spots. This technique is yet to be extended to multi-layer networks but preliminary studies have shown substantial speedups on some benchmark problems.

3.3 Sigmoid Inversion : Using Perceptron based strategy to counter Flat-Spots

In this technique we define an output unit to be trapped in a potential flat-spot if its output is greater than $1 - \epsilon$ while the target output is a 0 or if its output is less than ϵ and the target output happens to be a 1, ϵ being a small positive constant (0.1 in our experiments). An output-layer unit is said to be in the *safe region* if its output lies between ϵ and $1 - \epsilon$. By inverting the sigmoid activation function we determine the *safe range* of the net input of an output unit j denoted, (net_ip_j) , as

$$-\ln \frac{1 - \epsilon}{\epsilon} < net_ip_j < \ln \frac{1 - \epsilon}{\epsilon} \quad (6)$$

Now,

$$net_ip_j = \sum_i w_{ji} \cdot y_i \quad (7)$$

where w_{ji} is the weight of the link between an output unit j and a unit i in the hidden layer immediately below the output layer and y_i is the output of unit i . The proposed technique works exactly like BP unless a flat-spot is encountered. If an output-layer unit is in a flat-spot, the desired net input of that unit is determined by inverting the sigmoid to find out the minimum change required to pull the network out of the flat-spot and into the safe-region. This change is then incorporated into the system by modifying the weights according to the *Fractional Correction rule* for perceptrons [Nilsson, 1965]. According to this rule the new weights which pull the system out of a flat-spot and into the safe-region, are given by

$$W_{new} = W_{old} \pm c \cdot Y \quad (8)$$

where W stands for the weight vector of connection strengths between an output unit and units of a hidden layer immediately below it, Y represents the output vector of units in the hidden layer under consideration, and c is a constant, computed by solving

$$\begin{aligned} (W_{old} + c \cdot Y) \cdot Y &> -\ln \frac{1-\epsilon}{\epsilon} \quad \text{if } o_j \text{ is } < \epsilon \\ (W_{old} - c \cdot Y) \cdot Y &< \ln \frac{1-\epsilon}{\epsilon} \quad \text{if } o_j \text{ is } > 1 - \epsilon \end{aligned} \quad (9)$$

This weight-change in the output layer moves the network to a different point in the weight space and error-backpropagation proceeds from that point. But before standard BP can take over, the network outputs are recalculated because the δ_j values of the units in the output layer are no longer valid following the perceptron-based weight update.

4 Experimental Methodology

Performance evaluation of any learning algorithm should involve a good mix of benchmark problems, a fair combination of parameter settings, and well-defined convergence criteria.

4.1 Benchmarks

The choice of suitable benchmark problems is a much debated issue. The fact that algorithms may be able to exploit problem specific properties to give superior performance prohibits any single data set from emerging as a

consensus choice. We have selected a mix of toy and real-world problems as basis of our comparison. Each of these uses a single hidden layer. Table 1 lists the benchmarks and their corresponding network architectures.

Each data-set(except 838 and 10510) was

Table 1: Network Architectures

Problem domain	Input units	Hidden units	Output units
10510	10	5	10
838	8	3	8
Iris	22	4	3
Soybean	208	23	17

further divided into a *training set* consisting of two-thirds randomly selected patterns from the data-set, and a *test set* containing the remaining one-third of the patterns. Ten such divisions were made for each data-set.

4.2 Parameter Settings

To find an optimal choice of learning parameters one has to exhaustively search a 3-dimensional space defined by the learning rate η , the momentum term α , and the range of random initial weights r (i.e. the initial weights vary from $-r$ to $+r$). We have selected $r=1$ for simulations with BP, Sigmoid-prime and Perceptron-based techniques, however simulations with Linear-sigmoid approximation and a variation of Sigmoid-prime used a value of $r=0.5$ to maintain consistency with earlier reported results. Simulations were conducted for η varying from 0.25 to 1.5 and α in the range of 0.0 to 1.0. A few trials were carried out with η and α assuming values outside these ranges to confirm the widely accepted belief that the learning time escalates by several hundred epochs outside the above ranges.

4.3 Convergence Criterion

Several convergence criteria are commonly used in inductive learning algorithms. Since BP endeavors to minimize the output error, one scheme allows the learning to continue till the error falls below a pre-specified error threshold. Choice of an appropriate threshold is crucial to the performance of the algorithm. A very low threshold results in a dramatically increased learning time. In all our runs we have used a standard criterion of allowing the

learning to continue till 99% of the training patterns are correctly classified. This leads to another question. How is a pattern certified as being correctly classified? We have adopted a scheme of finding the unit having the maximum obtained output and finding its corresponding target output. If this target output happens to be a 1 the pattern is said to be correctly classified. This scheme, though applicable to the benchmarks considered in this paper, will not work for data-sets with target outputs having multiple 1's.

4.4 Simulation Runs

A set of 10 trials with random initial weights was run for each parameter setting for the 10510 and 838 problems. For the Iris and Soybean data-sets one trial was run for each parameter setting but on ten divisions of each data-set. In each trial the learning was terminated either when the convergence criterion was satisfied or when number of epochs reached a certain limit called *MAX-EPOCHS* (500 for soybean and 5000 for the other data sets). If *MAX-EPOCHS* was reached the run was treated as a failure and the results were excluded from the statistical analysis. Following the runs, statistical analysis was conducted to compute the averages of the number of epochs and classification accuracy. The sigmoid-prime offset technique can be used to handle flat-spots in the hidden layers also, something that cannot be done with the sigmoid linear-approximation or with the perceptron-based technique that we have proposed in this paper. To enable a fair comparison and to also bring out the importance of handling flat-spots in the hidden layers, we implemented a variation of the sigmoid-prime offset method which handles only output-layer flat-spots and this is another method figuring in our studies.

5 Simulation Results

Simulation runs were conducted for values of η varying from 0.25 to 1.5 in steps of 0.25. For each value of η , trials were run for $\alpha = 0.0, 0.25, 0.5, 0.8, 0.9, \& 1.0$. One interesting observation was a trough-like curve for each setting of η . The floors of the U-shaped valleys were generally obtained for α in the range of 0.25 to 0.9 (except for the Soybean dataset)

and $\alpha = 0.0$ and $\alpha = 1.0$ were the high points of the curves as can be seen in a 3-d plot included. We report the best results for each technique on each data-set. These results are reported in Table 2. The best results are not always informative, what is more important is how the algorithm performs on the average. After a careful analysis of the detailed results for each method we picked up a parameter setting which gave good results (not necessarily the best) for the two toy problems and another such parameter setting for the two real-world data-sets and these results are summarized in Tables 3 and 4. All the tables report the Average number of epochs with standard deviation, the percentage of trials that converged and in the case of Iris and Soybean) the classification accuracy of the test patterns. The speedup of each technique with respect to BP, is also shown. We calculate the speedup of a method m as $((\text{avg epochs using BP}) - (\text{avg epochs using } m))$ expressed as a percentage of the avg epochs using m . The following is the legend for interpreting the tables:

- BP(SP) - refers to the Sigmoid-prime technique
- BP(P) - refers to the Perceptron-based modification proposed
- BP(LA) - refers to the sigmoid Linear Approximation technique
- BP(OSP) - refers to the Sigmoid-prime applied only to the output-layer units

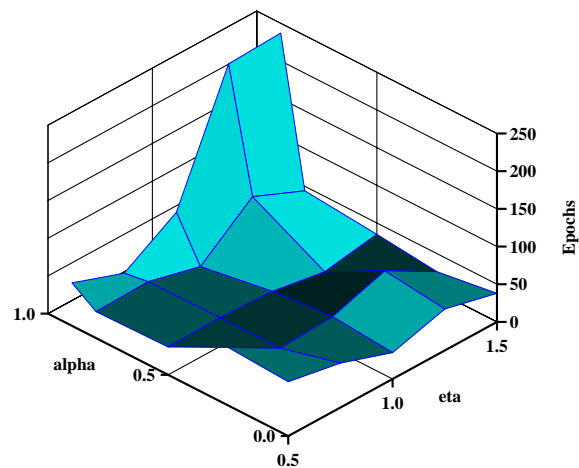


Figure 1: Parameter Vs Performance plot for Iris

Table 2: The Best Results for all Techniques on all data-sets

Problem Domain	Technique	η	α	Training Epochs	% Trials Converged	% Classification Accuracy	% Speedup
838	BP	1.25	0.8	43 ± 12.21	100	-	0
	BP(SP)	1.25	0.8	24 ± 5.95	100	-	79.2
	BP(LA)	0.25	0.9	32.6 ± 17.7	100	-	40.0
	BP(OSP)	1.5	0.8	21.8 ± 4.73	100	-	97.2
	BP(P)	1.25	0.9	69 ± 35.88	100	-	-37.7
10510	BP	1.0	0.8	39 ± 7.22	100	-	0
	BP(SP)	1.25	0.8	16 ± 3.63	100	-	143.8
	BP(LA)	0.25	0.9	10.8 ± 1.72	100	-	261.1
	BP(OSP)	1.5	0.8	19 ± 5.51	100	-	105.3
	BP(P)	0.75	0.9	21 ± 3.95	100	-	85.7
Iris	BP	0.5	0.5	16 ± 5.76	100	94	0
	BP(SP)	0.5	0.9	13 ± 6.24	100	90	23.0
	BP(LA)	0.75	0.0	15.3 ± 5.5	100	93.4	4.6
	BP(OSP)	1.0	0.5	17.1 ± 6.8	100	94	-6.4
	BP(P)	1.0	0.8	19 ± 11.62	100	94.8	-15.8
Soybean	BP	1.25	0.0	12 ± 3.51	100	93.10	0
	BP(SP)	1.25	0.0	7 ± 1.10	100	92.3	71.42
	BP(LA)	0.25	0.25	25.3 ± 2.32	100	90.6	-52.6
	BP(OSP)	0.5	0.25	25.3 ± 3.19	100	92.06	-52.6
	BP(P)	1.0	0.25	9 ± 1.52	100	92.1	33.3

6 Conclusions

The results in the above tables clearly reflect the slow learning speed of BP. Our simulations showed that each of the methods outperformed BP by a significant margin mainly in terms of the number of epochs for learning. Fahlman's modification of adding an offset was found to be the most successful among the flat-spot handling techniques considered. The technique proposed by Balakrishnan & Honavar, BP(LA) is computationally inexpensive but its performance did not quite match up to the BP(SP) technique. A major drawback of this technique, when compared to BP(SP), is that it is limited to handling output-layer flat-spots only. If a suitable way to extend this method to hidden layers is devised a considerable improvement in speed is foreseen. In fact, BP(OSP), which applies the sigmoid-prime offset only to output-layer units, was found to perform worse than BP(LA). This suggests that the reason BP(SP) works so well could very well be due to its ability to handle hidden layer flat-spots. The perceptron-based modification, BP(P),

did not perform quite as expected, on the average, working worse than BP(SP) and BP(LA) but better than BP. This might be due to the fact that this method differs considerably from the general gradient descent. The perceptron learning rule involves a big step which may result in a correct classification of the given pattern but may disturb the weights so much that considerable learning effort may be wasted. However, it must be pointed out that these results are very preliminary in nature and may be improved upon by finetuning the technique. All the techniques, except BP(SP) were found to suffer from delayed learning because of the presence of flat-spots in the hidden layers. Since flat-spots are a major cause for the slowness of BP, extensions of these techniques for handling flat-spots in the hidden layers are worth exploring to speedup the convergence of BP on large real-world problems.

References

- [Balakrishnan & Honavar, 1992]
Balakrishnan, K., & Honavar, V.G.

Table 3: Best Average-Case results for 838 and 10510 Encoder/Decoder

Technique	η	α	Problem Domain	Training Epochs	% Trials Converged	Speedup
BP	1.25	0.8	838	43 ± 12.21	100	0
			10510	39 ± 11.55	100	0
BP(SP)	1.25	0.8	838	24 ± 5.95	100	79.2
			10510	16 ± 3.63	100	143.8
BP(LA)	0.25	0.9	838	32.6 ± 17.72	100	40.0
			10510	10.8 ± 1.72	100	261.1
BP(OSP)	1.5	0.8	838	21.8 ± 4.73	100	97.2
			10510	19 ± 5.51	100	105.3
BP(P)	1.0	0.9	838	73 ± 54.91	100	-41.1
			10510	27 ± 10.77	100	44.4

Table 4: Best Average-Case results for Iris and Soybean data-sets

Technique	η	α	Problem Domain	Training Epochs	% Trials Converged	% Classification Accuracy	% Speedup
BP	1.0	0.5	Iris	16 ± 13.74	90	92.4	0
			Soybean	16 ± 5.23	100	93.1	0
BP(SP)	1.0	0.5	Iris	19 ± 9.97	100	91.4	-15.8
			Soybean	8 ± 1.22	100	92.5	100
BP(LA)	0.75	0.0	Iris	15.33 ± 5.5	100	93.4	4.6
			Soybean	30.9 ± 4.46	100	90.6	-48.2
BP(OSP)	1.25	0.0	Iris	26.5 ± 23.43	100	94.8	-39.6
			Soybean	27.8 ± 3.19	100	93	-42.4
BP(P)	1.25	0.5	Iris	23 ± 22.56	100	92.4	-30.4
			Soybean	13 ± 2.3	100	91.8	23.1

Improving Convergence of Back Propagation by Handling Flat-spots in the Output Layer. In *Proceedings of the Second International Conference on Artificial Neural Networks*, Brighton, U.K., 1992.

[Fahlman, 1988] Fahlman, S.E. Faster-learning variations of back propagation : An empirical study. In D. Touretzky, G.E. Hinton, and T.J. Sejnowski (editors), *Proceedings of the 1988 Connectionist Models Summer School*, San Mateo, CA : Morgan Kaufmann Publishers, pp 38-51, 1988.

[Nilsson, 1965] Nilsson, Nils J. Learning Machines – Foundations of trainable pattern classification systems. *New York, McGraw-Hill*. 1965.

[Rosenblatt, 1958] Rosenblatt, F. The Perceptron : a probabilistic model for information storage and organization in the brain. *Psychological Review* 65, pp 386-408, 1958.

[Rumelhart et al, 1986] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing, Vol. I*, Rumelhart D.E., & McClelland, J.L. (editors), MIT Press, Cambridge, MA, 1986.