IMPROVING BACKPROPAGATION VIA AN EFFICIENT COMBINATION OF A SATURATION SUPPRESSION METHOD AND MOMENTUM TERM

Payman Moallem∗, Seyed Arvin Ayoughi∗

Abstract: The gradient descent backpropagation (BP) algorithm that is widely used for training MLP neural networks can retard convergence due to certain features of the error surface like the local minimum and the flat spot. Common promoting methods, such as applying momentum term and using dynamic adaptation of learning rates, can enhance the performance of BP. However, saturation state of hidden layer neurons, which is the cause of some flat spots on the error surface, persists through such accelerating methods. In this paper, we propose a grading technique to gradually level off the potential flat spots into a sloping surface in a look-ahead mode; and thereby progressively renew saturated hidden neurons. We introduce symptoms indicating saturation state of hidden nodes. In order to suppress the saturation, we added a modifying term to the error function only when saturation is detected. In normal conditions, the improvement made to the learning process is adding a momentum term to the weight correction formula. We have recorded remarkable improvements in a selection of experiments.

Key words: Backpropagation, error surface, flat spot, local minima, momentum term, neuron’s saturation

Received: November 21, 2008
Revised and accepted: January 12, 2010

1. Introduction

The gradient descent learning procedure by error backpropagation (BP) has been the most competent multilayer neural network (MLP) training method since its initial formulation. In the error backpropagation that is an effective method to calculate the gradient of the MLP error function, the error signals (which can preserve a transformation of the weight space in summation) propagate backwards from the output nodes to the inner nodes [1–3].

∗Payman Moallem – Corresponding author, Seyed Arvin Ayoughi
Department of Electrical Engineering, Faculty of Engineering, University of Isfahan, Rezjarjirib Avenue, Isfahan, 81746, Iran, Corresponding author e-mail: p.moallem@eng.ui.ac.ir
The error surface, as a plot of the error function against the weights, is a fine illustration of the backpropagation algorithm. The conventional error surface upon which the BP algorithm searches for an acceptable minimum is an ingeniously engineered and often a very hilly surface. When the MLP reaches a solution region, it has learned what is expected. The topography of the error surface, especially in complex problems, contains numbers of local minima, flat spots, and saddle points which can distort network’s learning, in some cases BP may never converge. The saturation of the hidden layer neuron has been found to be a major cause of local minima in MLP trained by BP [4, 5].

As a persistent problem in backpropagation algorithm, saturation of hidden neurons can form vast regions on the error surface that are nearly flat. On these flat spots, gradient assumes a very low value. As a result, before the output units have approximated to any desired signals, weight changes begin to drop to negligible amounts. Under such conditions, the algorithm is said to be trapped in local minima [5, 4], and in some cases, the network can no longer learn [6]. This drawback has frequently been documented by different researchers [7, 6, 8].

We will briefly review basic characteristics of the BP algorithm in the following section, and then investigate improvements on BP; including the modifying term for the standard error function which can suppress untimely saturation of hidden neurons and the momentum term. After that, we introduce symptoms of saturation state of hidden neurons and then our proposed training method. Finally, the implementation results for some artificial and real problems are investigated. The results show higher convergence rates in our proposed learning method.

2. Backpropagation Algorithm

A generic representation of an overall performance of a BBP (Batch BP) learning system is a cumulative measure of the output error. The standard error function, _Sum Squared Errors (SSE)_ formulated in the batch format is given by equation 1:

\[
E_S = \frac{1}{2} \sum_{j=1}^{P} \sum_{i=1}^{S} (d_i - o_i)^2,
\]

where \(i\) indexes neurons in the output layer, \(j\) indexes a certain training pattern, \(o_i\) indicates the value obtained from a certain output neuron, and \(d_i\) indicates a desired output value corresponding to \(o_i\). \(P\) and \(S\) index the number of patterns and output neurons, respectively.

The encapsulation of desired and obtained output vectors into an error function has proven a highly practical mathematical model for the MLP learning behavior. It reduces the task of network training (optimizing the weight space) to a process of minimizing the error function. BP uses recursively differentiable functions to normalize the activation threshold of the network neurons. In this paper, we used the _Logistic Sigmoid_ (LogSig) as the activation function described in the equation 2.
Moallem P., Ayoughi S. A.: Improving backpropagation via an efficient...

\[ f(x) = \frac{1}{1 + e^{-x}} \]
\[ f'(x) = f(x) \cdot (1 - f(x)) \]  \hspace{1cm} (2)

Being nonlinear and recursively differentiable, LogSig activation functions guarantee non-linearity throughout the learning process. This property promotes the network to a universal function approximator. Therefore, backpropagation algorithm is founded on the manipulation of the \textit{delta rule} (equation 3), which requires the weight correction term be proportional to the \textit{gradient of the error} (sensitivity of the error to change in the weights).

\[ \Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}, \]  \hspace{1cm} (3)

where \( E \) is the neural network error and \( \eta \) is a positive value included to control the stepwise adjustment of the learning process. We can present the formula for the weight updates in \( k \text{th} \) epoch as:

\[ w_{ji}^{k+1} = w_{ji}^k + \Delta w_{ji} = w_{ji}^k - \eta \frac{\partial E^k}{\partial w_{ji}} \]  \hspace{1cm} (4)

This schema further involves a \textit{forward pass} which computes ‘functional activation signal’ by feed forwarding propagation of input pattern signals through the network. And a \textit{backward pass}, which performs gradient descent operations to pinpoint weights calling for credit or blame by propagating the ‘error signal’ backwards through network, starting at output neurons.

Initializing all weights within a small range of values can prevent retardation of the learning process. It is a privilege of the BP algorithm that can afford random initial values. Thus, we can restart from a convenient location on the error surface (plot of \( E \) against weights) to arrive at the global minimum and complete the learning process.

Derivative of LogSig function reaches its maximum at \( x=0 \), and is symmetric about this point falling to zero as approaching its extremes on the sides. This implies the weight changes will reliably be greatest when a neuron receives an approximate mid-range value, i.e. more information is passed through the layer being updated. Also, sliding down to either side, the gradient will be very small, where the neurons are saturated and minute (if any) weight changes occur. This can be ideal provided that the MLP system has reached the global minimum on the error surface, where overall optimization has been realized.

The schema sketched above is a mathematical model known as the \textit{generalized delta rule}, generally considered the backbone of the standard backpropagation algorithm. This learning model tunes the weights of the MLP iteratively at a pace that corresponds to the contours of the error surface charted by derivational properties of the LogSig. A persistent problem in this scheme is that when some neurons perpetually operate in their saturation region, the network may remain on a flat plateau of the error surface for an uncertain number of epochs. Under such conditions the algorithm is trapped in some local minima \([5-7]\), and it may never reach a global minimum or even near an acceptable solution.
3. Improving Performance of BP

Despite all its efficiency, BP algorithm can prevent convergence through untimely saturation of hidden neurons. Three techniques currently used to deal with saturation problem of the MLP networks include hardware measures:

- Utilizing extra neurons in hidden layer;
- Adding extra hidden layer to convergence is guaranteed;
- And adding one hidden neuron at a time to grade the error surface [9].

These techniques are most efficient when the network is not required to comply with hardware constraints.

There are also soft methods [4, 5, 8]; some of them include renewing saturated units by adding a modifying term to the standard error function to offset the error surface. For these methods, in many cases, the fact remains that an extensive alteration in the topography of the standard error surface can obscure the correlation between the dynamics of the problem at hand and its mathematical model [10]. It is important to note that a saturated hidden neuron can provide us with a valuable evidence of its saturation severity. Henceforth, we detect the saturation severity of the network by monitoring the saturation level of the hidden neurons. Therefore, in this research we have proposed a dynamic monitoring of some symptoms indicating saturation state of hidden neurons to applying the modifying term only when it is called for.

In order to improve BP’s performance common accelerating and stabilizing techniques are adding a momentum term to the weight correction formula and dynamic adaptation of learning rates. Successive signs of weight corrections when identical can be interpreted as steps in the right direction toward convergence, and represent undesirable aberrations when different; in which case a curb for the second step helps in smoothing the process, thus speeding the gradient search. The formula which makes this promotion possible is the momentum term we shall introduce in Section 3.2.

3.1 Modifying term for the standard error function

On a plateau, we need to renew the search path by modifying term, $E_B$ presented below [4]:

$$E_B = \frac{1}{2} \sum_{j=1}^{P} \left[ \sum_{i=1}^{S_L} (d_i - o_i)^2 \times \sum_{i=1}^{S_{L-1}} (y_i - 0.5)^2 \right],$$

(5)

where $L$ and $L - 1$ index the output layer and the hidden layer respectively, $S_L$ indicates the total neurons in layer $l$, $y_i$’s indicate the output values from the hidden layer neurons, for LogSig as activation function 0.5 is $y_i$’s midrange.

The term $E_B$ is to suppress untimely saturation of hidden neurons and is formulated in batch format. This term is meant to incorporate the behavior of the hidden layer neurons in the gradient descent operations, in such a way that the network harmoniously gropes for the global minimum. Thus, the modified error
function \((E_M)\) is the sum of standard function (equation 1) and the modifying term (equation 5).

\[
E_M = E_S + E_B
\]  

(6)

The idea is to inhibit the saturation of hidden layer neurons when \(E_S\) is large and the network is nowhere near the global minimum; ‘the output does not approximate to the desired signal’. \(E_B\) is a product of \(SSE\) and \(\sum_{i=1}^{s_{i-1}} (y_i - 0.5)^2\) for \(j\)th pattern, therefore, besides inhibiting the saturation state, when \(SSE\) approaches zero, this term will become small enough to yield convergence.

Adding this modifying term to standard error function leads to a modifying term for the weight correction:

\[
\Delta w_{ji} = -\eta_S \frac{\partial E_S}{\partial w_{ji}} - \eta_B \frac{\partial E_B}{\partial w_{ji}},
\]  

(7)

where \(\eta_S\) and \(\eta_B\) are the learning rates along the gradient of \(E_S\) and \(E_B\) respectively. Here, \(\frac{\partial E_S}{\partial w_{ji}}\) is the gradient of the standard error function, while \(\frac{\partial E_B}{\partial w_{ji}}\) ought to be computed as follows. For hidden layer parameters:

\[
\frac{\partial E_B^P}{\partial w_{ji}^{L-1}} = \frac{\partial E_S^P}{\partial w_{ji}^{L-1}} \sum_{j=1}^{s_{L-1}} (y_{pj} - 0.5)^2 + \sum_{j=1}^{s_L} (d_{pj} - y_{pj})^2 (y_{pj} - 0.5) \times \frac{\partial y_{pj}}{\partial w_{ji}^{L-1}}
\]  

(8)

For output layer parameters:

\[
\frac{\partial E_B^P}{\partial w_{ji}^L} = \frac{\partial E_S^P}{\partial w_{ji}^L} \sum_{j=1}^{s_{L-1}} (y_{pj} - 0.5)^2,
\]  

(9)

where

\[
\frac{\partial y_{pj}}{\partial w_{ji}^L} = \begin{cases} 
1 & \text{for biases} \\
 f'(.) & \text{for weights}
\end{cases}
\]  

(10)

In general, neural networks are application sensitive and the exactly same network can perform very differently from case to case. Saturation of hidden layer neurons can occur on certain occasions under limited conditions during the training process. The added term \((E_B)\) to the standard error function being repeated in all iteration could mean excessive computational burden in many applications. Furthermore, a comprehensive survey of the new topography of the error surface (considering the deficiencies of the benchmark today) is pended to the performance of the ANN in various applications. We suspect that a sweeping alteration to the entire error surface will inevitably cause adverse outcomes. To illustrate the effect of applying the modifying term in all epochs, we have plotted the standard and the modified error surfaces of the well-known historical MLP problem, 2-2-1 XOR network, against two biases of the hidden neurons in Fig. 1.

As illustrated in Fig. 1, there is a global minimum on the modified error surface. We have obtained this point by optimizing the modified function. But, in this region, standard error function assumes an unacceptably large value. This figure provides an example of cases in which minimizing the modified error function does
not minimize network’s error. It shows that adding the modifying term to the standard error function of backpropagation may lead to new undesirable features, e.g. new local/global minima, saddle points etc.

We believe, when saturation state of hidden layer neurons is not so severe to impede (hamper) learning, this modification to the error function is redundant, if not a potential source of insidious errors. Therefore, to enhance the effect of the error function, in this paper, we propose a dynamic monitoring criterion to alternate between the modifying term and the momentum term in the weight correction formula.

3.2 The momentum term

Momentum term prevents search aberrations by monitoring two consecutive gradient steps in order to curb or promote the second. It functions as a low-pass filter in that shallow craters are bypassed in the search path. The momentum term is normally a fraction of the previous weight correction:

$$\Delta w_{ji}^k = -\eta \frac{\partial E_k}{\partial w_{ji}} + \alpha \Delta w_{ji}^{k-1},$$  \hspace{1cm} (11)$$

where $\Delta w_{ji}^{k-1}$ is the previous weight correction, and $\alpha$ is the momentum constant. This normalized term carries on valuable information readily available throughout the search trend to maintain the gradient descent’s inertia.
In other words momentum term has two significant effects on the training process [11]:

- **It accelerates** the gradient descent in smooth downhill regions. Two consecutive weight corrections of the same signs indicate a downhill search trend. In which case, the momentum term becomes large in the long run, elongated steps ensure faster convergence.

- **It stabilizes** the search process when it oscillates. Two consecutive weight corrections of the different signs indicate an oscillation and the momentum term becomes smaller and it has stabilizing effects on learning.

The momentum constant $\alpha$ scales the magnitude weight correction interdependence, for which we have employed a dynamic self-adaptation procedure.

### 4. Symptoms Indicating Saturation State

We have suggested two mathematical symptoms through which saturation state of hidden neurons can be detected.

The first is when the norm of gradient matrix assumes a small value. When the search path enters a flat region of the error surface, gradient of the standard error function versus weights will drop to some small values. Therefore, a reduction in the absolute values of elements of the gradient matrix may be considered a symptom of saturation state of hidden neurons. This can be detected when the norm of the gradient matrix compared to a predetermined parameter, which we have called **Norm of Gradient’s Threshold** (NOG Thr).

The second symptom is when the measure of hidden neurons’ saturation exceeds a certain maximum limit. We have called it **Normalized Saturation Criterion** (NSC) and calculated by the formula below:

\[
NSC = \frac{1}{0.25 \cdot P \cdot S_{L-1}} \sum_{j=1}^{P} \sum_{i=1}^{S_{L-1}} (y_i - 0.5)^2,
\]

where $P$ is the number of patterns, $S_{L-1}$ is the number of hidden neurons of the MLP, $y_i$'s are hidden neurons' outputs and 0.5 is the midrange of hidden neurons' outputs, which is determined by the LogSig activation function.

Since neuron output is kept within the range of 0 to 1 by Logsig, NSC will be in the range of [0,1]. A small value for NSC indicates normal operation of the hidden neurons; while a maximum value of NSC (i.e. 1) means hidden units are saturated. We have found out that outside a certain value of NSC (i.e. NSC Thr) entails saturated behavior of hidden units.

### 5. Alternating Modifying Term for Weight Correction

When a number of neurons perpetually operate in their saturation region, the network remains retarded on a flat plateau of the error surface for an uncertain
number of iterations; where the modifying term helps in renewing the saturated neurons. In this region, the gradient is so low that sign difference of two successive weight correction terms has no effect on the gradient search. Therefore, here we recommend the application of the modified error function. In downhill directions or when the gradient shows significant oscillations the momentum term rises to a considerable amount, the modifying term is of no function; the momentum term can accelerate or stabilize the search.

To alternate between the modifying and the momentum terms as promoting methods we invariably monitor gradient norm and NSC (equation 12) in each epoch. When a saturation state is detected, we apply the modifying term to prevent flat spot formation on the error surface, otherwise a momentum term is applied. The weight correction is carried out by the formula below:

\[
\text{if } (\text{NSC} > \text{NSC, Thr AND } ||\nabla E_S|| < \text{NOG, Thr}) \text{ then } \Delta w_{ji}^k = -\eta_S \frac{\partial E^k}{\partial w_{ji}} - \eta_B \frac{\partial E^k}{\partial w_{ji}},
\]

\[
\text{else } \Delta w_{ji}^k = -\eta_S \frac{\partial E^k}{\partial w_{ji}} + \alpha \Delta w_{ji}^{k-1}.
\]

(13)

We have also used the dynamic self-adaptation method [13] for the learning rate along the gradient of standard error function \((\eta_S)\), and also for momentum constant \((\alpha)\) or, to maintain adaptive step sizes in conformation with especial contours of the error surface in each epoch. We have also suggested an adaptive step size scheme for the learning rate along the gradient of the modifying term \((\eta_B)\), presented in section 6.2.

6. Learning Rates

The learning rate \(\eta\) in the BP algorithm scales the speed of convergence while preventing fluctuations of the weight vector towards a minimum error value. When \(\eta\) is fine tuned to a right value, the delta rule is also tuned up. If \(\eta\) is too small, convergence takes too long. And if it is too large, the weight vector fluctuates around a minimum.

6.1 Learning rate along the gradient of standard error function \((\eta_S)\) and momentum constant \((\alpha)\)

In the \(k^{th}\) epoch of the backpropagation algorithm, each parameter is updated by the standard delta-rule:

\[
w_{ji}^{k+1} = w_{ji}^k - \eta \frac{\partial E^k}{\partial w_{ji}},
\]

(14)

where \(W\) is a weight matrix of neural network parameter, and \(\eta\) is the learning rate. A constant value for \(\eta\) may be too large in one epoch, but it may be too small in the next, depending on the contours of the error surface. A dynamic learning rate \((\eta)\) assuming an optimum local value yields harmonious progression of the learning process. A large value for \(\eta\) can make the weight vector fluctuate around a minimum. While for too small values convergence can be retarded. This has brought about a whole genre of self-adaptation of learning rate schemata.
The dynamic self-adaptation method for learning rates [7] can enhance the
dynamic properties of the standard algorithm. This method first assigns an initial
value to the learning rate (e.g. $\eta_{init} = 1$), then in each epoch, $\eta_1$ and $\eta_2$ are
computed by:

$$\eta_1 = \eta / \xi$$

(15)

$$\eta_2 = \eta \cdot \xi,$$

(16)

where $\eta$ is the learning rate in the previous epoch, and $\xi$ is greater than 1.0.
Salomon et al. [7] showed that any $\xi > 1.0$ can normally work well, but $\xi = 1.839$
is an optimum value for the error surface with elliptical contours. Now, either of
these rates ($\eta_1$ or $\eta_2$) which yield a smaller SSE is chosen as the rate for that epoch.
We have employed this method for adapting $\eta_S$ and $\alpha$.

6.2 Learning rate along the gradient of modifying term
In the $k^{th}$ epoch, if a saturation state is detected and it is necessary to apply the
modifying term, we have chosen NSC (equation 12) of that epoch as the learning
rate along the gradient of modifying term:

$$\eta_B = NSC.$$ (17)

The idea is that: as the saturation of hidden neurons becomes severer, NSC becomes
larger. Therefore, the modifying term has more influence on weight changes to
overcome saturation. Thus, the effect of the modifying term on weight changes will
be proportional to the level of hidden neurons' saturation.

7. Compared Algorithms
In this section, we present the compared learning algorithms including: our sug-
gested method, DS $\eta - \alpha$ and PTGVLR.

7.1 Dynamic self-adaptation of learning rate ($\eta$) and
momentum constant ($\alpha$)
DS $\eta - \alpha$ method [12] makes use of the momentum term to improve performance
of standard backpropagation algorithm. In the $k^{th}$ epoch of the method weight
correction is obtained by:

$$\Delta w_{ji}^k = -\eta_S \frac{\partial E_S}{\partial w_{ji}} + \alpha \Delta w_{ji}^{k-1},$$

(18)

where $\Delta w_{ji}^{k-1}$ is the weight correction of the previous iteration and $\alpha$ is the mo-
mentum constant. In this method, learning rate ($\eta$) and momentum constant ($\alpha$)
are adapted using dynamic self-adaptation method [12].
7.2 Parallel tangent gradient with variable learning rates

PTGVLR makes use of the parallel tangent gradient acceleration direction [13] instead of the momentum term. In order to improve performance of BP, the method takes an acceleration step, denoted by $A$, after each gradient step. In the $k$\textsuperscript{th} epoch of PTGVLR, the weight correction is given by:

$$
\Delta w^k_{ji} = -\eta S \frac{\partial E_S}{\partial w_{ji}} + \mu A^k_{ji}
$$

$$
A^k_{ji} = w^{2k-1}_{ji} - w^{2k-4}_{ji},
$$

(19)

where $A^k_{ji}$ is the acceleration direction [13] and $\mu$ is the parallel tangent coefficient. In this method, learning rate ($\eta$) and parallel tangent coefficient ($\mu$) are adapted using dynamic self-adaptation method [12]. The variable learning rates are also bounded by a minimum value, to prevent retardation of algorithm, and a maximum value, to inhibit fluctuation of the weight vector [13].

7.3 The suggested method

Here we summarize our proposed method denoted by AS-DS $\eta - \alpha$ (Anti Saturation DS $\eta - \alpha$).

1 – Initialize weights and biases with random small values. Set the learning rates to their initial values.
2 – If termination condition is not satisfied, do steps 3 to 5.
3 – Compute $NSC$ (equation 12) and norm of the gradient matrix for current weights.
4 – If ($NSC > NSC_{\text{Thr}}$ and $||\nabla E_S|| < NOG_{\text{Thr}}$) then:
   Compute $\eta_S$ using dynamic self-adaptation method [12] and set $\eta_B = NSC$.
   Take a step along the gradient of $E_S$ and along the gradient of $E_B$:

$$
\Delta w_{ji} = -\eta_S \frac{\partial E_S}{\partial w_{ji}} - \eta_B \frac{\partial E_B}{\partial w_{ji}}.
$$

(20)

Else
   Compute $\eta_S$ and $\alpha$ using dynamic self-adaptation method [12].
   Take a step along the gradient of $E_S$ with a momentum term added.

$$
\Delta w_{ji} = -\eta_S \frac{\partial E_S}{\partial w_{ji}} + \alpha \Delta w_{ji}^{\text{previous}}.
$$

(21)

Check termination condition.

8. Simulation Results

In order to evaluate the performance improvements made by our proposed method, AS-DS $\eta - \alpha$ (Anti Saturation DS $\eta - \alpha$), we have carried out a number of binary and real word classifications and compared the results with the PTGVLR (Parallel Tangent Gradient with Variable Learning Rates) [13] and DS $\eta - \alpha$ (Dynamic Self-adaptation of learning rate, $\eta$, and momentum constant, $\alpha$) [12] methods which
respectively make use of the parallel tangents and the momentum term to improve the convergence of the standard BP.

In all simulations, the initial values of the learning rates are set to 1. The initial values for weights and biases are randomly chosen from the range \([-1,1]\). Results of our simulations are summarized in Tabs. I, II, and III. The results were derived from 1000 independent training trials. The presented statistics include percentage of successful trails in 1000 runs (Training success) and average number of the epochs to convergence for successful trails (Mean Epoch), for all compared methods. As the benchmark problem, we apply the proposed algorithm (\(\text{AS-DS} \eta - \alpha\)), PTGVLR and \(\text{DS} \eta - \alpha\) for the parity generators, the modified-XOR, IRIS and Ionosphere data classification problems that will be explained in detail.

In all simulations, the value of \(\text{NOG}_n\_\text{Thr}\) in equation 13 was set to 0.1. This value for \(\text{NOG}_n\_\text{Thr}\) has been obtained through experience, and yields rather good results for almost all problems. In order to demonstrate the effect of \(\text{NSC}_n\_\text{Thr}\) in equation 13 on performance of the proposed method, we have reported the simulation results for two values of \(\text{NSC}_n\_\text{Thr}\), 0.25 and 0.5, in the tables. Choosing \(\text{NSC}_n\_\text{Thr}\) out of the range 0.2 to 0.5 for most of the problems yields better results.

Generalization capability of the trained MLP is a very important factor in the performance of any training algorithm. We assessed the generalization performance of all algorithms for the real world classification problems. After training, each instance of the test set was applied to the network, and its classification error was evaluated. Then, the average of classification errors of the test set, multiplied by 100, (Classification Error) is reported.

### 8.1 Parity generators

The parity generators are common benchmarks \([5, 6, 12, 13]\). Their error surfaces are very hilly and filled with local minima and flat spots, therefore ideal for assessing saturation proof algorithms. For an n-bit parity generator, which has \(2^n\) binary training patterns, we will need an \(n\times n-1\) network. We have tried out 2, 3, and 4-bit parity generators. The 2-bit parity generator is in fact, the well-known historical MLP problem, XOR. The criterion for a successful trial is when the network correctly classifies all patterns with a tolerance of 0.1 to be achieved for every target elements in less than 5000 epochs.

Convergence rate of parity generator problems was considerably improved through our algorithm as against the other two methods. Better results are obtained in the most sophisticated problem, 4-bit parity generator. However, our algorithm requires more epochs to converge to renew the saturated neurons.

In order to demonstrate the effect of \(\text{NSC}_n\_\text{Thr}\) (in equation 13) value on the performance of the proposed method we have carried out some simulations on 4-bit parity generator, with different values of \(\text{NSC}_n\_\text{Thr}\). In these simulations we did not monitor the gradient norm, i.e., the only condition for applying the modifying term is that when NSC becomes greater than \(\text{NSC}_n\_\text{Thr}\). This is the only difference between these simulations and others. Results of these simulations are presented in Figs. 2 and 3.
Parity Generator

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>2-bit Training success (%)</th>
<th>2-bit Mean epoch</th>
<th>3-bit Training success (%)</th>
<th>3-bit Mean epoch</th>
<th>4-bit Training success (%)</th>
<th>4-bit Mean epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS $\eta$-$\alpha$</td>
<td>27.1</td>
<td>33.4</td>
<td>40.0</td>
<td>76.6</td>
<td>9.8</td>
<td>296.1</td>
</tr>
<tr>
<td>PTGVLR</td>
<td>44.2</td>
<td>35.5</td>
<td>44.8</td>
<td>80.0</td>
<td>9.5</td>
<td>340.9</td>
</tr>
<tr>
<td>AS-DS $\eta$-$\alpha$ (NSC, Thr=0.25)</td>
<td>77.3</td>
<td>85.1</td>
<td>90.5</td>
<td>182.2</td>
<td>45.0</td>
<td>618.3</td>
</tr>
<tr>
<td>AS-DS $\eta$-$\alpha$ (NSC, Thr=0.5)</td>
<td>64.4</td>
<td>112.4</td>
<td>77.1</td>
<td>286.5</td>
<td>36.2</td>
<td>959.1</td>
</tr>
</tbody>
</table>

Tab. 1 Comparison of algorithms’ performances for parity generator problems.

Fig. 2 Success rate vs. NSC, Thr for 4-bit parity generator problem.

Fig. 3 Mean epoch vs. NSC, Thr for 4-bit parity generator problem.
Since NSC (equation 12) only assumes positive values, for NSC,\(_{\text{Thr}} = 0\), NSC is always greater than its threshold, which means permanent utilization of the modifying term. Whereas, increasing NSC,\(_{\text{Thr}}\) results in less utilization of\(E_B\) and for NSC,\(_{\text{Thr}} = 1\) the algorithm only uses momentum term throughout learning.

As depicted in Fig. 2, permanent utilization of the modifying term yields poor success rate. Setting NSC,\(_{\text{Thr}}\) in the range of 0.05 to 0.5, we have recorded some noticeable values for success rate that are obtained by spending fewer epochs.

### 8.2 Modified XOR

The conventional XOR problem has 4 training patterns, including inputs (0,0), (1,1), (0,1), and (1,0) with corresponding outputs 0,0,1, and 1. The modified XOR problem has an additional input (0.5,0.5) as the input with 1.0 as the target, which are applied to a 2-2-1 network. This problem is a good benchmark to evaluate the performance of the learning algorithm, since the error surface of modified XOR has only one global minimum and numerous local minima [4, 14]. The criterion for a successful trial is when the network correctly classifies all patterns with a tolerance of 0.1 to be achieved for every target elements in less than 5000 epochs.

In this problem, our method represents considerable improvement in convergence rate as against DS \(\eta-\alpha\) and PTGVLR. However, our algorithm demands more epochs to converge to renew the saturated neurons.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Training success (%)</th>
<th>Mean epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS (\eta-\alpha)</td>
<td>19.2</td>
<td>41.3</td>
</tr>
<tr>
<td>PTGVLR</td>
<td>27.5</td>
<td>43.7</td>
</tr>
<tr>
<td>AS-DS (\eta-\alpha) (NSC,(_{\text{Thr}} = 0.25))</td>
<td>68.1</td>
<td>135.2</td>
</tr>
<tr>
<td>AS-DS (\eta-\alpha) (NSC,(_{\text{Thr}} = 0.5))</td>
<td>61.1</td>
<td>196.4</td>
</tr>
</tbody>
</table>

**Tab. II** *Comparison of algorithms’ performances for Modified-XOR.*

### 8.3 IRIS data

This is perhaps the best-known database to be found in the pattern recognition literature that has been obtained from [15]. The IRIS data set [16] contains 3 classes of 50 instances. Each class refers to a type of iris plant. One class is linearly separable from the other 2; but these two classes are not linearly separable from each other. Total number of instances is 150 and number of attributes is 4. We have used 75 first instances as a training set and the remaining 75 instances as a test set. We used 4-3-3 network to classify data, with 3 neurons in the output layer, one for each class. Every trial in which SSE reduced to a value below 0.05 within a maximum of 5000 epochs was deemed as a successful run. Results of the simulations are presented in Tab. III.
We have recorded better success rates using our method in comparison with DS $\eta$-$\alpha$, at the expense of more epochs to renew the saturated neurons. However, parallel tangent gradient can do well compared with this combination of the saturation proof method and momentum term in this problem. Our method can also yield rather better generalization compared with DS $\eta$-$\alpha$ and PTGVLR.

### 8.4 Ionosphere data

This problem involves classification of radar returns from the ionosphere. The radar data were collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. “Good”, denoted by “0”, radar returns are those showing evidence of some type of structure in the ionosphere. “Bad”, denoted by “1”, returns are those that do not; their signals pass through the ionosphere. The data set was created by Johns Hopkins University [17] and obtained from the databases [15]. There are 34 attributes and total number of instances is 351. We applied this problem to a 34-3-1 network.

Simulation results for this data set are given in Tab. III. Every trial in which SSE reduced to a value below 0.05 within a maximum of 5000 epochs was deemed as a successful run.

Our saturation proof method increases the possibility of convergence by renewing the saturated neurons. Thus, the convergence rate improvement is done at a price of more epochs to renew the search.

To assess generalization capability of the trained MLPs, we have used the 200 first instances for training, which are split almost 50% good and 50% bad, and the 151 remaining instances for testing. For 100 trained networks with fewer classification errors, generalization capability of our method is better than PTGVLR, and is the same as DS $\eta$-$\alpha$. Although, considering all successful runs, classification errors of networks trained by AS-DS $\eta$-$\alpha$ are slightly greater than those of DS $\eta$-$\alpha$ and PTGVLR.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>IRIS Data</th>
<th>Ionosphere Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training success (%)</td>
<td>Mean epoch</td>
</tr>
<tr>
<td></td>
<td>All</td>
<td>Success</td>
</tr>
<tr>
<td>DS $\eta$-$\alpha$</td>
<td>47.2</td>
<td>47.8</td>
</tr>
<tr>
<td>PTGVLR</td>
<td>85.5</td>
<td>41.3</td>
</tr>
<tr>
<td>AS-DS $\eta$-$\alpha$ (NSC, Thr=0.25)</td>
<td>60.4</td>
<td>323.4</td>
</tr>
<tr>
<td>AS-DS $\eta$-$\alpha$ (NSC, Thr=0.5)</td>
<td>58.1</td>
<td>235.1</td>
</tr>
</tbody>
</table>

Tab. III *Comparison of algorithms’ performances for IRIS and Ionosphere data.*
9. Conclusion

Poor convergence is one of the most important challenges in many gradient-based learning algorithms. Saturated hidden layer neurons have a fundamental role in this drawback, which corresponds to flat spots on the error surface.

Three of the most common techniques dealing with the flat spot problem are: utilizing extra neurons or hidden layers so as to guarantee a timely convergence; adding one hidden neuron at a time to grade the error surface more efficiently; and finally applying a modifying term to the standard error function to offset the error surface.

Although, second order algorithms (e.g. LM algorithm) seem to work quite well and result in high success rates [18], the demand for memory to operate with large matrices is their major disadvantage. For the LM algorithm, as the dimensionality of the network increases, the training would need costly hardware due to the exponential growth in the computational complexity [19].

We have introduced a fourth technique which makes use of the standard error function in normal state of affairs and an offset term to gradually level off the anticipated flat spots to a sloping surface, only when an offset is called for [10]. In this paper, we have evaluated the capability of our saturation proof method in combination with the momentum term, which has an accelerating and stabilizing effect on the learning process. Moreover, we have used adaptive learning rates algorithms to tune the learning rates in different conditions.

We ran a number of benchmark problems to verify the expected network performance. In almost all the problems studied, considerable success rates were recorded, though at the expense of more epochs to renew the saturated neurons. There were also cases in which the generalization capability of the MLPs trained by the proposed method was also improved.

References


