IEICE TRANS. FUNDAMENTALS, VOL.E89–A, NO.8 AUGUST 2006

PAPER *Special Section on Papers Selected from the 20th Symposium on Signal Processing*

An Adaptive Penalty-Based Learning Extension for the Backpropagation Family

Boris JANSEN[†a)], ***Student Member and*** **Kenji NAKAYAMA**[†b)], ***Member***

SUMMARY Over the years, many improvements and refinements to the backpropagation learning algorithm have been reported. In this paper, a new adaptive penalty-based learning extension for the backpropagation learning algorithm and its variants is proposed. The new method initially puts pressure on artificial neural networks in order to get all outputs for all training patterns into the correct half of the output range, instead of mainly focusing on minimizing the difference between the target and actual output values. The upper bound of the penalty values is also controlled. The technique is easy to implement and computationally inexpensive. In this study, the new approach is applied to the backpropagation learning algorithm as well as the RPROP learning algorithm. The superiority of the new proposed method is demonstrated though many simulations. By applying the extension, the percentage of successful runs can be greatly increased and the average number of epochs to convergence can be well reduced on various problem instances. The behavior of the penalty values during training is also analyzed and their active role within the learning process is confirmed.

key words: *backpropagation, learning algorithm, convergence, error function, neural networks, generalization*

1. Introduction

Since the introduction of the backpropagation (BP) [1] learning algorithm, it has proved to be efficient in many applications. Presently, this gradient descent method has emerged as one of the most well-known and popular learning algorithms for artificial neural networks (ANNs). However, in various cases its convergence speed often tends to be very slow and it often yields suboptimal solutions.

As a result, much research has been focusing on improving the BP learning algorithm and numerous new algorithms and techniques have been proposed. Many attempts to speed up training and to reduce convergence to local minima have been made in the context of dynamically adjusting the learning rate during training, including learning algorithms such as SAB [2] and SuperSAB [3], Quickprop [4], and RPROP [5], [6].

Other directions that have been studied, include the application of alternative cost functions. Squared-error functions have been replaced by possible better cost functions, such as the cross-entropy measure [7]. Furthermore, error functions have been extended with extra terms to direct the search in the weight space towards specific goals, such as the addition of noise as in simulated annealing [8], [9] or the application of penalties as in weight decay [9], [10].

In this paper, a new adaptive penalty-based extension for various objective functions is proposed. Penalties are applied in order to put pressure on incorrect binary outputs to get them initially into the correct half of the output range. The penalties are dynamically adjusted during training to reflect the difficulty of this task. Here, the new method is applied to standard backpropagation as well as to the effective RPROP learning algorithm. Simulations have been performed on a number of problem instances and the performance of the extended algorithms is compared to their original counterparts.

2. New Adaptive Penalty-Based Learning Extension

2.1 Idea behind New Approach

Consider learning of artificial neural networks with binary target values ±1. Of course, the targets also can be 1 and 0, or values from any other binary defined set. The learning process can be divided into two phases. In the first phase, an ANN is trained so as to move all its outputs for all training patterns to the correct side, that is greater than or less than a certain threshold, which equals zero in this case. In the second phase, the ANN is trained so as to move its outputs located in the correct region towards the actual targets, that is $+1$ or -1. Compared to the second phase, it can be expected that the first phase is relatively complex and time-consuming, because for each single output this process easily affects many other outputs. Furthermore, these two phases are likely to coexist among the different outputs during training, meaning that the ith output has already been located into the correct half of the output range, while the jth output still resides on the wrong side. Therefore, the difficulty of learning differs for each output.

We propose an adaptive penalty-based learning extension. In this method, learning for the outputs located on the wrong side, will be accelerated by applying penalties. In order to make this acceleration more effective, the penalties are increased epoch by epoch, while the outputs reside in the incorrect half of the output range. Furthermore, in order to make the learning process more stable, penalties are gradually decreased after the outputs have been moved to the correct side.

Figures 1 and 2 show two example situations. A cir-

Manuscript received December 8, 2005.
Manuscript revised March 13, 2006.
Final manuscript received April 20, 2006.

[†]The authors are with the Graduate School of Natural Science and Technology, Kanazawa University, Kanazawa-shi, 920-1192 Japan.
a) E-mail: boris@leo.ec.t.kanazawa-u.ac.jp
b) E-mail: nakayama@t.kanazawa-u.ac.jp

DOI: 10.1093/ietfec/e89–a.8.2140

Copyright © 2006 The Institute of Electronics, Information and Communication Engineers

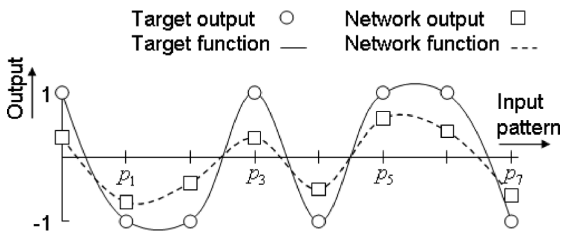


Fig. 1 Network having all its outputs in the correct half of the output range.

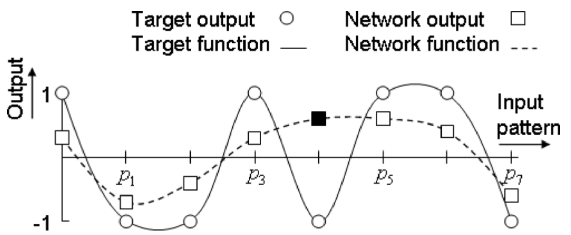


Fig. 2 Network having an output residing in the incorrect half of the output range.

cle is a target output and a square is an actual output of an ANN. The value p_i represents the input pattern number. In Fig. 1, all network outputs are located on the correct side. On the other hand, in Fig. 2, the output for the input pattern p_4 resides in the incorrect half of the output range. Moving this output towards the correct, lower side, will be affected by the outputs for the input patterns p_3 and p_5, which are being moved towards +1. Therefore, it can be expected that it will take a long time to convergence, if ever reached.

In the new proposed method, the correction term for the output of p_4 is amplified by applying an adaptive penalty. The amplification, that is the penalty, is adaptive in the sense that it is being increased every epoch, while the output resides on the wrong, in this case upper, side. As a result, more and more pressure is being put on the ANN in order to move the incorrect output to the right side. After the output enters into the correct lower half of the output range, the penalty is decreased. However, in order to avoid the danger that the output 'makes a big jump back' to the incorrect side, the penalty is gradually decreased epoch by epoch. This way of controlling the penalties can make the learning process more stable.

2.2 Formal Description

In the backpropagation learning algorithm, the errors of output neurons are backpropagated through the network during training. The error signal $e_{i,p}(n)$ of an output neuron i at an epoch n for a training pattern p, can be defined by taking the difference between the target output $t_{i,p}(n)$ and the actual output $o_{i,p}(n)$:

$$e_{i,p}(n) = t_{i,p}(n) - o_{i,p}(n) \tag{1}$$

In the new proposed method, for every output neuron i and every training pattern p, a penalty $z_{i,p}(n)$ is created. The error backpropagated in the new algorithm is given by the following equation:

$$e_{i,p}^{new}(n) = z_{i,p}(n)e_{i,p}(n) \tag{2}$$

whereby the penalties are being updated after each epoch as defined below:

$$z_{i,p}(n+1) = \begin{cases} max(z_{i,p}(n)z^-, 1) & \text{if } o_{i,p}(n) \text{ is at the same side as } t_{i,p}(n) \\ min(z_{i,p}(n)z^+, z^{max}) & \text{otherwise} \end{cases} \tag{3}$$

and $z^- < 1$, $z^+ > 1$ and $z^{max} \gg 1$. The initial penalties $z_{i,p}(0)$ are set to one.

The application of the new proposed method results in the addition of penalties to the backpropagated error signal. The task of these penalties is to put pressure on the network to get all the outputs initially into the correct half of the output range.

The penalties are dynamically adjusted as shown in Eq. (3) in order to reflect the *hardness* of this task, by assuming that the more difficult it is to move a certain output for a certain pattern to the right side, the more often it resides in the incorrect half of the output range. Every epoch an output for a certain pattern resides in the incorrect half, its corresponding penalty is increased in order to put more pressure on the network to move the output to the right side. Once an output reaches its correct half of the output range, its corresponding penalty is gradually decreased and the focus of the network on moving the output to the right side shifts away to outputs for which the corresponding penalties are increasing.

Figure 3 shows a representative curve of a change of a single penalty during training. A penalty is being raised while its corresponding output resides at the wrong side. The change occurs exponentially, because the penalty is multiplied by z^+ every epoch the output resides in the incorrect half. Once an output reaches the correct side, the penalty is gradually decreased by multiplying it with z^-. Finally, it can reach to a minimum of one. The steepness of the upward and the downward curve is controlled by the parameters z^+ and z^-, respectively. Once an output enters its correct half of the output range, it is not guaranteed that the output stays there. Therefore, multiple successive phases of increasing and decreasing a penalty can be expected during training.

From a different point of view, the error surface can be considered dynamic. The true error surface is given by using $z_{i,p}(n) = 1$. In a learning process, the error surface is modified by changing the penalties $z_{i,p}(n)$ so that the neural network, that is its connection weights escape from temporal local minima and move towards the global minimum. As the connection weights approach to the global minimum, the penalties also approach to unity. As a result, the error surface approaches the true one, and then finally the global minimum becomes the true one.

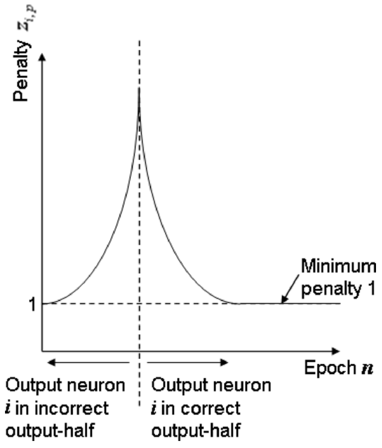


Fig. 3 Representative curve of a change of a single penalty during training.

Dynamic penalties are preferred over static penalties for two reasons. Different states of a neural network require different penalty values. Dynamic penalties are able to adjust to the shape of the error surface during learning, opposite to static penalties which lack this ability. Furthermore, static penalties still have the risk that a neural network moves towards a local minimum, as a result of penalty values not large enough to move away from the local minimum.

Finally, it should be noted that it is not guaranteed that the proposed method will converge to the global minimum. However, from the ability of the penalties to adjust to the error surface and to push networks out of local minima, it can be expected that the rate of convergence to the global minimum is increased.

3. Control of Upper Bound

In the previous section, the upper bound for the penalties is simply defined as $z^{max} \gg 1$. However, it is highly dependent on the distribution of the training data. Pattern classification problems can be categorized into two cases. In one case, the probability density function (pdf) of the training data for all groups are non-overlapping. They are clearly separated. An example is shown in Fig. 4. For instance, the parity check problem, the encoder problem and so on belong to this category. In the other case, the pdf of the training data for several groups are overlapping each other. In this case, perfect classification is impossible. Some data locate in the other group region, and are incorrectly classified. An example is shown in Fig. 5. Many real world problems belong to this case.

In the first case, the proposed penalties are not amplified to very large values. Because, as the learning process makes progress, the boundaries of the neural network can move between the pdf of the training data sets, which are clearly separated. After all training data locate in their own regions, the penalties are decreased following Eq. (3). Therefore, as the learning process converges, the penalties are also decreased. In this case, the static condition for z^{max} given by Eq. (3) is sufficient.

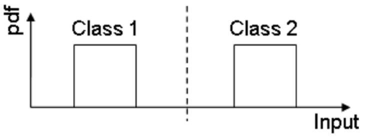


Fig. 4 Two non-overlapping distributions.

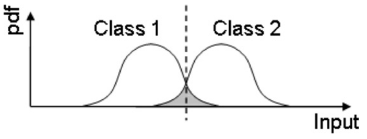


Fig. 5 Two overlapping distributions.

On the other hand, in the second case, some training data cannot locate in the correct region, even though the learning process converges. In this case, the penalties for these training data are always amplified and amplified. As a result, the penalties become very large numbers, which causes unstable behavior.

In order to avoid this problem, an annealing method is proposed for the upper bound z^{max}, which is controlled so as to be gradually decreased, and approach to unity after a specified number of epochs. Defining this number be the maximum number of epochs n^{max}, z^{max} is controlled by:

$$z^{max}(n) = z^{max}2^{-Tn} \tag{4}$$

$$T = \frac{\log_2(z^{max})}{n^{max}} \tag{5}$$

T is a temperature, controlling the speed of lowering the upper bound. After n^{max} epochs, the upper bound becomes $z^{max}(n^{max}) = 1$. The parameter n^{max} is also problem dependent. This is true for other annealing methods as well, such as the Boltzmann machine [11]. However, the proposed method is not a statistical method, and can provide fast convergence.

The proposed annealing method can also be applied to the first case, in which the penalties are gradually decreased by the extension itself as the learning process makes progress.

Furthermore, when the number of the hidden units is not enough, it is inevitable that some training data still remain on the wrong side after learning has converged. The penalties for these training data are continuously amplified, and become very large numbers. The proposed annealing upper bound is also effective to overcome this kind of problem.

4. Generalization Performance

In pattern classification by neural networks, an important point is generalization. The aim of the proposed adaptive penalty-based method is to increase the rate of successful convergences and to accelerate the learning process itself. In this section, the generalization performance is addressed.

As described in Sect. 2.1, the learning process can be divided into two phases. Also described in Sect. 2.2, the

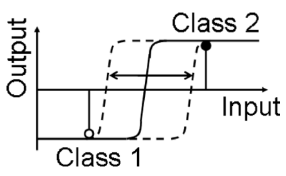


Fig. 6 Boundary has a degree of freedom.

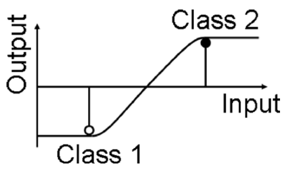


Fig. 7 Boundary is optimized w.r.t. generalization.

proposed method mainly works during the first phase. Furthermore, the proposed method is not a stand-alone learning algorithm, but is rather combined with other stand-alone learning algorithms. This is why it is called an *extension*. In the second phase, since the penalties are gradually decreased until unity, it is mainly the underlying learning algorithm which is active. Therefore, generalization is also dependent on the underlying learning algorithm.

Figure 6 shows an example, where the pdf of the training data sets are completely separated. As shown in this figure, there are many solutions which can provide a well reduced output error. However, in this situation, the proposed penalties are gradually decreased, and do not affect the optimization of the boundary for generalization. In the case of overlapping training data sets, the penalties are also controlled to be gradually decreased until unity by using the annealing upper bound proposed in Sect. 3. Therefore, generalization is also mainly dependent on the underlying learning algorithm.

If we employ the weight decay learning algorithm [10], it is expected that the boundary locates in the center as shown in Fig. 7, where there is no degree of freedom, being optimized w.r.t. generalization. The proposed method also can cooperate with this algorithm.

5. Comparative Study

In order to give an indication of the performance of the new proposed method in terms of convergence speed and success rate, comparisons have been performed between the standard backpropagation [1],[11] and RPROP learning algorithms extended with the new adaptive penalty-based method on one side and their original counterparts on the other side on various problem instances. The RPROP learning algorithm is an improvement of the backpropagation learning algorithm and it has proven its superiority in many cases [5], [6].

5.1 Test Problems

5.1.1 N-Bit Parity Problem

The N-bit parity problem is a generalization of the 'exclusive-or' (XOR) problem. The task is concerned with detecting whether the number of activated input bits is even or odd. In this study, N-bit input strings composed of $\{-1, +1\}$ are considered and the corresponding target output values are defined as -1 and $+1$ for input data consisting of an even, respectively odd number of activated bits. The number of training patterns is equal to 2^N.

The N-bit parity problem is considered as a very hard problem to be solved by neural networks, because a single 'flip' of a bit in the input string requires a complementary classification.

5.1.2 M-N-M Encoder Problem

The task of the M-N-M encoder problem is to learn an auto-association between M different input/output patterns. Each training pattern has one bit turned on, i.e. set to one, while the remaining bits are set to zero. Therefore, the number of training patterns equals M.

The network applied to learn this auto-association is a two-layered M-N-M feed-forward neural network. The complexity of this task resides in the fact that the number of hidden neurons is less than the number of input and output neurons, i.e. $N < M$. Consequently, the hidden neurons perform compression or encoding, while the output neurons perform decompression or decoding. Whenever $N \leq \log_2 M$, the network is being referred to as a 'tight' encoder.

5.1.3 Two Spirals Problem

The task of the two spirals problem is to learn to discriminate between two sets of training points which lie on two distinct spirals in the x-y plane. These spirals coil three times around the origin and around one another. The training data consists of 194 patterns and here, the target values describing the two classes for the two different spirals are within the set $\{-1, 1\}$.

The difficulty of the two spirals problem has been demonstrated in many attempts to solve this problem by applying backpropagation and many of its variants over the years. One modification to the adapted neural networks that has often been applied is the usage of shortcut connections [12]. By using shortcut connections, every neuron is not only connected to all neurons in the last previous layer as is in standard feed-forward neural networks, but a neuron is connected to all neurons in all previous layers. Shortcut connections may ease the training process, because information learned by neurons is directly inserted in all its following neurons.

5.1.4 Thyroid Function Diagnosis

This problem is concerned with diagnosing thyroid diseases. Based on a patient's query and examination data, the functioning of the patient's thyroid has to be classified into one of the following three classes; under function, normal function or over function.

The problem is obtained from the Proben1 [13] database. The data set consists of 3600 training patterns and 1800 test patterns. The target output is encoded in a 3-bit binary string, where only a single bit is active representing the class of the input pattern and the other two bits equal zero. The difficulty of this task lies in the fact that the data set is formed by three highly unbalanced groups, i.e. 2.3%, 92.5% and 5.1%, respectively.

5.2 Simulation Setup

The neural networks used in our simulations have been developed using the Java Object-Oriented Neural Engine (Joone) [14], an open source neural net framework implemented in the Java programming language.

All the adapted neural networks used in our experiments are multilayer feed-forward neural networks. Here, the backpropagation learning algorithm operates in online training mode, i.e. weights are updated on a pattern-by-pattern basis. The connection weights and biases for all networks were randomly initialized within the interval $[-1, 1]$. In the simulations applying the proposed method to the thyroid problem, a dynamic upper bound for the penalties following Eqs. (4) and (5) was used, setting the initial upper bound $z^{max}(0)$ to 10000 and 10 for the backpropagation learning algorithm and the RPROP learning algorithm, respectively. In all other simulations featuring the new proposed method, a constant value of 10000 was used for the maximum penalty z^{max}. Varying parts of the applied network configurations are summarized for each experiment individually together with the simulation results in the tables below. RPROP's parameters set to their default, previously proposed values [5] are omitted from this network configuration summary.

In addition, a constant value of 0.1 was added to the derivative of the logistic and the hyperbolic tangent activation function for all algorithms, to overcome the 'flat spot' problem [4], i.e. the problem where training progresses very slowly, because the derivative of the activation function approaches zero, caused by the fact that the output of a neuron is close to one of its asymptotic output values.

For the parity, encoder and two spirals problem, learning of the binary task was considered complete, if the '40-20-40' criterion, described by Fahlman [4], was fulfilled, i.e. all outputs of output neurons for all training patterns are within the correct upper or lower 40% of its output range. The maximum training time n^{max} for these experiments was set to 20000 epochs.

In simulations involving the thyroid problem, the maximum number of epochs n^{max} was set to 2000. Learning was considered complete once an RMSE of 0.125 and 0.10 was reached for backpropagation and RPROP, respectively

For each problem instance and network configuration, 25 independent runs have been performed. The number of successful runs and the average number of epochs to convergence, neglecting unsuccessful runs, are reported.

Table 1 Simulation results for 6-bit parity problem.

6-Bit Parity			
Algorithm	Epochs	SR	Settings
BP	9879	2/25	η : 0.0005
	7916	2/25	η : 0.001
RPROP	7492	4/25	Δ_{max} : 0.001
BP + Extension	5953	25/25	η : 0.0005, z^- : 0.9, z^+ : 1.05
	5522	24/25	η : 0.001, z^- : 0.8, z^+ : 1.05
	6270	21/25	η : 0.001, z^- : 0.9, z^+ : 1.01
	3436	25/25	η : 0.001, z^- : 0.9, z^+ : 1.05
	6567	22/25	η : 0.001, z^- : 0.9, z^+ : 1.1
	4695	25/25	η : 0.001, z^- : 0.95, z^+ : 1.05
RPROP + Extension	7792	7/25	Δ_{max} : 0.001, z^- : 0.9, z^+ : 1.05
	7516	19/25	Δ_{max} : 0.001, z^- : 0.99, z^+ : 1.05
Network structure :	*6-6-1*		
Activation function :	*hyperbolic tangent*		

5.3 Simulation Results

Tables 1 and 2 show the simulation results for the 6-bit and 8-bit parity problem, respectively. SR stands for success rate, η is the learning rate used in the backpropagation learning algorithm and Δ_{max} is the maximum update-value used in the RPROP learning algorithm.

The low number of success rates for the backpropagation and RPROP learning algorithm indicate the difficulty of this problem. The networks get easily trapped in local minima. However, applying the new proposed method resulted in an increase of the number of successful runs by a magnitude. The new method provides a way to escape from local minima. Moreover, in general the average number of epochs to convergence was also greatly reduced by the new method.

Observing the results in greater detail, we see that the parameter values z^- and z^+ of the new method rather have some influence on the performance. Tuning the parameters carefully can result in a very good performance, but search-

Table 2 Simulation results for 8-bit parity problem.

8-Bit Parity			
Algorithm	Epochs	SR	Settings
BP	7663	2/25	η : 0.0005
	5961	3/25	η : 0.001
RPROP	-	0/25	Δ_{max} : 0.001
BP + Extension	4931	23/25	η : 0.0005, z^- : 0.9 z^+ : 1.05
	2807	20/25	η : 0.001, z^- : 0.9 z^+ : 1.05
RPROP + Extension	10444	14/25	Δ_{max} : 0.001 z^- : 0.99 z^+ : 1.05
Network structure :	*8-8-1*		
Activation function :	*hyperbolic tangent*		

Table 3 Simulation results for 8-2-8 encoder problem.

8-2-8 Encoder			
Algorithm	Epochs	SR	Settings
BP	-	0/25	η : 0.005
RPROP	99	25/25	
BP + Extension	4883	22/25	η : 0.005 z^- :0.9999 z^+ :1.01
RPROP + Extension	94	25/25	z^- : 0.9999 z^+ : 1.01
Network structure :	*8-2-8*		
Activation function :	*logistic*		

Table 4 Simulation results for 32-2-32 encoder problem.

32-2-32 Encoder			
Algorithm	Epochs	SR	Settings
RPROP	3727	25/25	
RPROP + Extension	2985	25/25	z^- : 0.9999 z^+ : 1.01
Network structure :	*32-2-32*		
Activation function :	*logistic*		

Table 5 Simulation results for 48-2-48 encoder problem.

48-2-48 Encoder			
Algorithm	Epochs	SR	Settings
RPROP	13914	14/25	
RPROP + Extension	12170	25/25	z^- : 0.9999 z^+ : 1.01
Network structure :	*48-2-48*		
Activation function :	*logistic*		

ing for an optimal parameter set is usually considered a very time-consuming task. However, less well tuned parameters still result in a performance much better than the learning algorithms without the proposed extension.

Tables 3, 4 and 5 show the results for the 8-2-8, 32-2-32 and 48-2-48 encoder problem, respectively.

It can be easily noticed that the learning algorithms extended with the new approach outperform their original counterparts also for the encoder problem. For a large range of different learning rates η, standard backpropagation was unable to find a solution for the tight encoder problems. However, backpropagation extended with the new method was still able to find a solution for the 8-2-8 encoder in 88%.

Table 6 Simulation results for two spirals problem.

Two Spirals			
Algorithm	Epochs	SR	Settings
BP	14141	7/25	η : 0.0005
	9838	9/25	η : 0.001
RPROP	8964	16/25	Δ_{max} : 0.001
BP + Extension	11650	18/25	η : 0.0005 z^- : 0.99 z^+ : 1.001
	9005	19/25	η : 0.001 z^- : 0.99 z^+ : 1.001
RPROP + Extension	7179	23/25	Δ_{max} : 0.001 z^- : 0.9999 z^+ : 1.001
	9259	25/25	Δ_{max} : 0.001 z^- : 0.99999 z^+ : 1.001
Network structure :	*2-5-5-5-1 + shortcut connections*		
Activation function :	*hyperbolic tangent*		

The RPROP learning algorithm has a much more satisfactory performance, even on complex encoder problems. RPROP easily finds a solution for the 8-2-8 and 32-2-32 encoders, however by applying the new method the average number of epochs to convergence was reduced. For the 48-2-48 encoder problem, RPROP also experienced difficulties and was unable to find a solution in all runs, while by applying the new proposed method in combination with the RPROP learning algorithm, the networks converged to a solution in all runs.

Table 6 shows the simulation results of the two spirals problem. All applied ANNs used shortcut connections.

Again, the learning algorithms extended with the new proposed method are superior to their original counterparts. Although backpropagation as well as the RPROP learning algorithm are able to find solutions, the number of successful runs is greatly increased by applying the new method and in general the average number of epochs to convergence is decreased.

Table 7 shows the simulation results of the thyroid problem. Networks being trained by the backpropagation learning algorithm were halted at an RMSE of 0.125 and networks being trained by the RPROP learning algorithm were halted at an RMSE of 0.10 at which point the test set was applied. The average of the correctly classified percentage of training and test pattens for these converged networks are given by `TrnC` and `TstC`, respectively. `TrnA` and `TstA` give the average percentages of correctly classified training and test patterns for all 25 neural networks, that is, the converged networks and the networks that were halted after the maximum number of epochs.

For a large range of different learning rates η, the standard backpropagation learning algorithm was unable to train the neural networks in such a way that the networks were

Table 7 Simulation results for thyroid problem.

Thyroid				
Algorithm	Epochs	SR	Results	Settings
BP	-	0/25	TrnC : - TstC : - Tr nA : 92.4% TstA : 92.7%	η : 0.0005
RPROP	754	10/25	TrnC : 98.9% TstC : 97.9% Tr nA : 97.9% TstA : 97.0%	
BP + Extension	1045	12/25	TrnC : 98.0% TstC : 97.0% Tr nA : 97.8% TstA : 96.9%	η : 0.0005 z^- : 0.9999 z^+ : 1.05
RPROP + Extension	696	13/25	TrnC : 98.9% TstC : 98.0% Tr nA : 98.7% TstA : 97.8%	z^- : 0.99999 z^+ : 1.005 z^{max} : 10
Network structure :	*21-4-3*			
Activation function :	*logistic*			

able to make any distinction between patterns from different classes. It classified all patterns as the major class, i.e. normal function.

By applying the extension to the backpropagation learning algorithm, the performance is greatly improved. The neural networks were able to classify also a large part of the two minor groups correctly.

The RPROP learning algorithm itself is able to obtain very good results. Still, by applying the proposed method, the number of networks converged to an RMSE below 0.10 was increased, the average number of epochs to convergence was reduced and also the quality of the solutions itself regarding generalization was improved. However, in order to obtain these results, the initial maximum penalty $z^{max}(0)$ had to be reduced to 10.

Finally, we have confirmed that the proposed method can be successfully applied in combination with the weight decay method [10]. Regarding the thyroid problem, there isn't much space left for optimization. However, simulations applying the RPROP learning algorithm extended with the proposed method in combination with the weight decay method resulted in a slightly higher success rate and still outperformed the RPROP learning algorithm without the extension in combination with the weight decay method.

6. Observation of Penalties

In order to learn more about the effects and behavior of the applied penalties, the penalty values during training have been studied. The lower the values of the penalties are during the learning process, the less pressure the learning algorithm puts on the network, and the more the extended learning algorithm resembles its original counterpart. On the other side, the higher the penalty values are, the more pressure the learning algorithm puts on the network to get the outputs into the correct half of the output range, and the more it operates differently from the original learning algorithms.

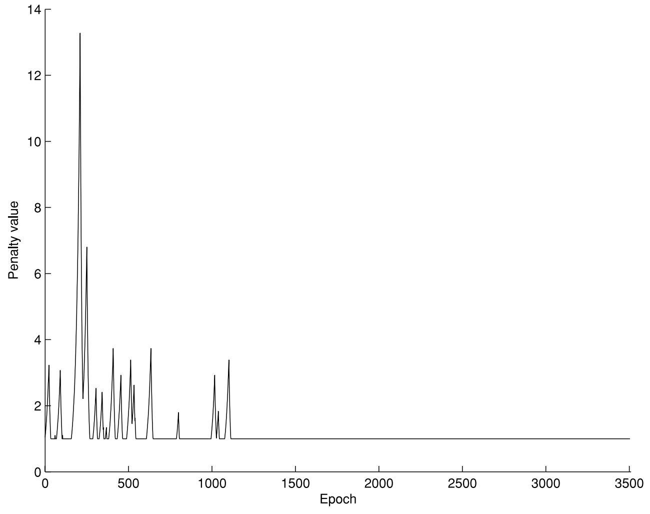


Fig. 8 Representative graph of the change of the values of a penalty from the first group.

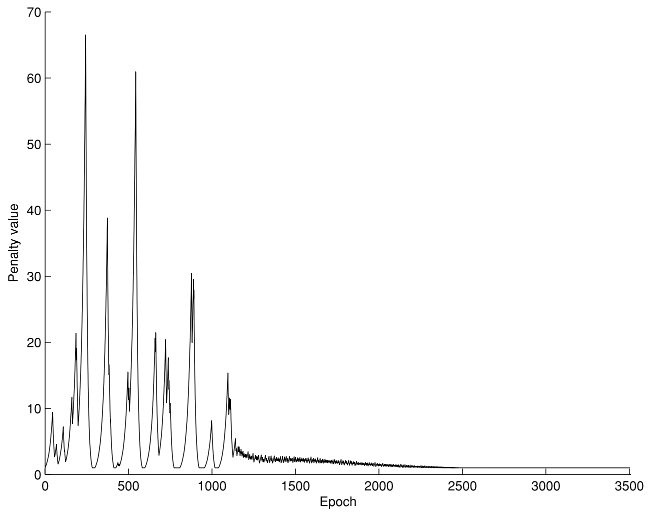


Fig. 9 Representative graph of the change of the values of a penalty from the second group.

Here, a single representative simulation run of a neural network, implementing the backpropagation learning algorithm extended with the new proposed method and having its parameter values η, z^- and z^+ set to 0.001, 0.9 and 1.05 respectively, applied to the 6-bit parity problem, has been further investigated. For this single run, the '40-20-40' criterion was fulfilled at the 3503rd epoch.

By observing the change of the penalty values that takes place during training, two main groups of penalties can be characterized. Most penalties belong to the first group, where the penalties undergo a change only in the beginning of the learning process and their values vary somewhere between 1 and 20. Of course, this range is highly dependent on the parameter values z^- and z^+. An example of a change of a single penalty, representative for the penalties of the first group, is shown in Fig. 8. The second group contains penalties, which values are raised to larger values, somewhere in the range of 1 to 100. Furthermore, these penalties often seem to undergo a change longer than the penalties from the first group. A representative for the penalties of the second group is shown in Fig. 9.

Not only from the results of the performed simulations, also from the observation of the change of penalty values during training, it can be concluded that the penalties in the new proposed method do play an active role in the learning process. Especially, the penalties from the second group heavily pressure the neural network in order to get the outputs in the correct half of the output range.

7. Tuning Extension Parameters

In the new proposed method two important parameter values, namely z^- and z^+, need to be determined. The two parameters have a great influence on the performance of the new method and are problem dependent. How to determine the decremental and incremental penalty values efficiently remains an open problem.

However, from our experience the following remarks can be made. By initializing z^+ with values too close to one, penalties can't be raised to large enough values to become effective resulting in a performance similar to underlying learning algorithm without the extension. On the other side, initializing z^+ with values too large, the danger exists that too much pressure is being put on the network driving its weights into saturation. Furthermore, a larger z^+ seems to require a smaller z^- in order to keep the learning process stable.

The RPROP learning algorithm seems more responsive to the penalties than the backpropagation learning algorithm. As a result, the RPROP algorithm requires values closer to one for z^- and z^+, in comparison to backpropagation. The two main differences between the backpropagation learning algorithm and the RPROP learning algorithm are a static learning rate versus a dynamic learning rate and online training mode versus batch mode. The reason behind these sensitive extension parameters in combination with the RPROP learning algorithm can be analyzed as follows: In the RPROP learning algorithm, the learning rate is adjusted during training. For instance, if the derivative of the backpropagated error doesn't change in direction in successive epochs, the learning rate is increased [5], [6]. Thus, it has similar effect as the proposed method. Therefore, z^- and z^+ cannot be set far from unity. Otherwise, the pressure might become too large, resulting in an unstable learning process.

In case of the parity, encoder and two spirals problem, the penalty upper bound z^{max} has a rather small influence on the performance of the learning algorithm, at least if it is set to a large enough value. However, the RPROP learning algorithm extended with the new method applied to the thyroid problem seems to benefit from a small initial maximum penalty. The reason is the same as analyzed above.

8. Concluding Remarks

A new adaptive penalty-based approach applicable as an extension for squared-error functions in backpropagation and its variants is proposed. The new method initially puts pressure on artificial neural network in an attempt to get all the outputs for all training patterns into the correct half of the output range, instead of mainly focusing on minimizing the difference between the target and actual outputs.

Simulations have been performed and the results have demonstrated the usefulness of the proposed approach. By applying the new algorithm, the rate of successful runs can be greatly increased and the average number of epochs to convergence can be well reduced on various problem instances. The new method is easy to implement and computationally inexpensive.

Furthermore, the observation of the change of the penalty values during training has demonstrated the active role the penalties play within the learning process.

Future research will be directed towards learning tasks consisting of patterns having continuous target output values. We intent to investigate on how to decide appropriate thresholds defining the output halves for real-values output patterns. Furthermore, how to decide appropriate decremental and incremental penalty values, i.e. values for z^- and z^+ will also be a future research project.

Acknowledgement

The authors would like to thank A. Hirano and A. Horita for their useful discussions on the subject and the anonymous reviewers for their comments to help improve the quality of this paper.

References

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol.1, pp.318–362, 1986.

[2] M.R. Devos and G.A. Orban, "Self adaptive backpropagation," Proc. NeuroNimes 1988, pp.469–476, Nimes, France, 1988.

[3] T. Tollenaere, "SuperSAB: Fast adaptive back propagation with good scaling properties," Neural Netw., vol.3, no.5, pp.561–573, 1990.

[4] S.E. Fahlman, "An empirical study of learning speed in backpropagation networks," Tech. Rep. CMU-Cs-88-162, 1988.

[5] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," Proc. IEEE International Conference on Neural Networks, pp.586–591, San Francisco, CA, 1993.

[6] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons—From backpropagation to adaptive learning algorithms," Int. J. Computer Standards and Interfaces, Special Issue on Neural Networks, vol.16, pp.265–278, 1994.

[7] C.M. Bishop, Neural networks for pattern recognition, Clarendon Press, Oxford, 1995.

[8] J. Robert, M. Burton, and G.J. Mpitsos, "Event-dependent control of noise enhances learning in neural networks," Neural Netw., vol.5, no.4, pp.627–637, 1992.

[9] N.K. Treadgold and T.D. Gedeon, “Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm,” IEEE Trans. Neural Netw., vol.9, no.4, pp.662–668, July 1998.

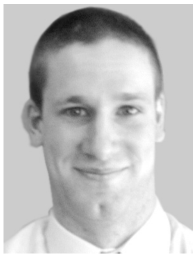
[10] P. Werbos, “Backpropagation: Past and future,” Proc. IEEE International Conference on Neural Networks (ICNN), pp.343–353, 1988.

[11] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, New Jersey, 1994.

[12] K.J. Lang and M.J. Witbrock, “Learning to tell two spirals apart,” Proc. 1988 Connectionist Summer School, pp.52–59, Morgan Kaufman, 1988.

[13] L. Prechelt, “PROBEN1—A set of benchmarks and benchmarking rules for neural network training algorithms,” Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany, Sept. 1994. Anonymous FTP: /pub/papers/techreports/1994/1994-21.ps.Z on ftp.ira.uka.de

[14] P. Marrone, “Java object-oriented neural engine (JOONE),” 2005. http://www.jooneworld.com



Boris Jansen was born in Rotterdam, the Netherlands in 1978. He received a M.Sc. degree in 2000 from Leiden University, Leiden, the Netherlands. He is currently working towards his Ph.D. degree at Kanazawa University, Kanazawa, Japan. His research interests include artificial neural networks, evolutionary algorithms and cryptography. Moreover, he is an active developer of the open source project Joone, a neural net framework implemented in Java.

Kenji Nakayama received the B.E. and Dr. degrees in electronics engineering from Tokyo Institute of Technology (TIT), Tokyo, Japan, in 1971 and 1983, respectively. From 1971 to 1972 he was engaged in research on classical network theory at TIT. He was involved in NEC Corporation from 1972 to 1988, where his research subjects were filter design methodology and digital signal processing. He joined the Department of Electronical and Computer Engineering at Kanazawa University, in Aug. 1988. He is currently a Professor of Graduate School of Natural Science and Technology. He has served as the Director of the International Student Center from April 2003 to March 2005. His current research interests include adaptive signal processing and neural networks. He is a senior member of IEEE and a member of INNS and RISP, Japan.