An Adaptive Penalty-Based Learning Extension for Backpropagation and its Variants

Boris Jansen, *Student Member, IEEE*, and Kenji Nakayama, *Member, IEEE*

***Abstract*— Over the years, many improvements and refinements of 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 technique is easy to implement and computationally inexpensive. In this study, the new approach has been applied to the backpropagation learning algorithm as well as the RPROP learning algorithm and simulations have been performed. The superiority of the new proposed method is demonstrated. By applying the extension, the number of successful runs can be greatly increased and the average number of epochs to convergence can be well reduced on various problem instances. Furthermore, the change of the penalty values during training has been studied and its observation shows the active role the penalties play within the learning process.**

I. 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 Squared-error functions have been replaced by possible better cost functions, such as the cross-entropy measure [7].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 [10], [9].

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.

Boris Jansen is with the Graduate School of Natural Science and Technology, Kanazawa University, Kakuma-machi, Kanazawa, 920-1192, JAPAN (email: boris@leo.ec.t.kanazawa-u.ac.jp).

Kenji Nakayama is with the Graduate School of Natural Science and Technology, Kanazawa University, Kakuma-machi, Kanazawa, 920-1192, JAPAN (email: nakayama@t.kanazawa-u.ac.jp).

II. New Adaptive Penalty-Based Learning Extension

A. Idea behind New Approach

Consider learning of artificial neural networks with binary target values +1 and -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 i^{th} output has already been located into the correct half of the output range, while the j^{th} 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 circle 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

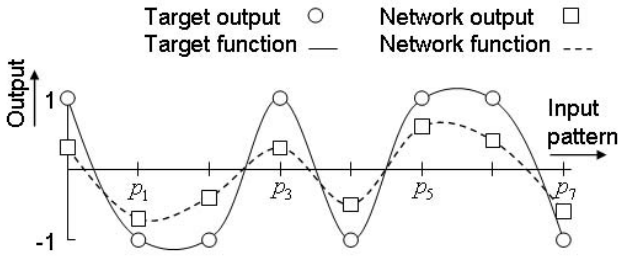

Fig. 1. Network having all its outputs in the correct half of the output range

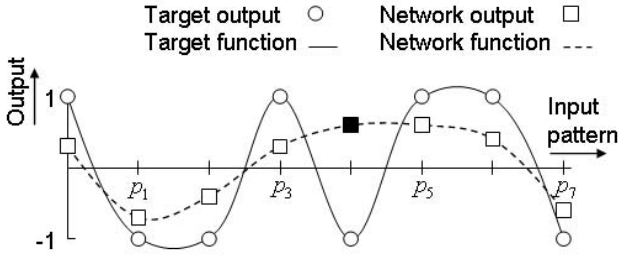

Fig. 2. Network having an output residing in the incorrect half of the output range

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, for example opposite to a sudden reset of the penalty. This way of controlling penalties can make the learning process more stable.

B. 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 output neuron i at epoch n for 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:

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

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 for a certain pattern 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. From a different point of view, the error surface can be considered dynamic.

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 decreased by multiplying it with z^- 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.

The new proposed method implicitly provides a mechanism to escape from local minima. Whenever a network converges towards a local minimum, penalties will be increased for the outputs residing in the incorrect half of the output range. Once the penalties have been raised to large enough values, the network might 'jump' out of or move away from the local minimum.

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

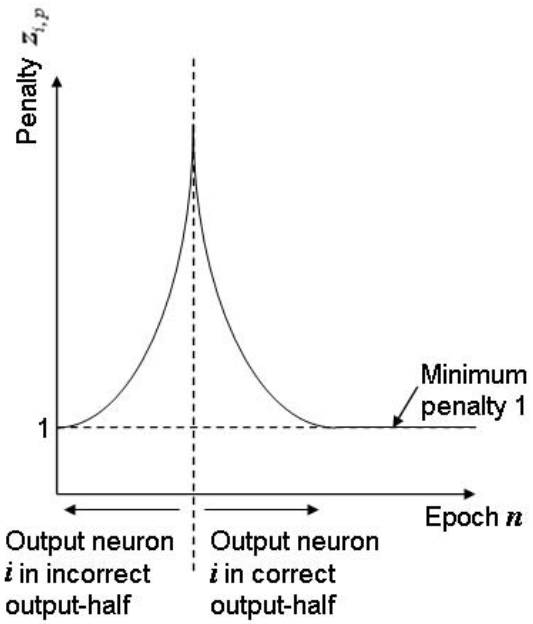

Fig. 3. Representative curve of a change of a single penalty during training

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 a 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 likelihood of convergence to a global minimum is increased. Furthermore, the new method is a true extension, similar to simulated annealing or weight decay, meaning that in theory it can be applied to any binary-output error-based objective function regardless of the underlying learning algorithm, and even in combination with other extensions.

III. 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 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.

A. Test Problems

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.

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.

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 [11]. 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.

B. Simulation Setup

The neural networks used in our simulations have been developed using the Java Object Oriented Neural Engine (Joone)[12], 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 $[-1, 1]$. A constant value of 10000 was used for the maximum penalty z^{max} in all simulations featuring the new proposed method. 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.

Learning of a 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 was set to 20000 epochs for all experiments.

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.

C. Simulation Results

Tables I and II 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.

TABLE I
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*		

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.

TABLE II
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*		

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 searching 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.

In comparison with backpropagation, the RPROP learning algorithm extended with the new proposed approach required a less dynamic error-surface, which is expressed in the fact that the decremental penalty rate z^- was set very close to one in order to obtain satisfactory results. The two main differences between the backpropagation and RPROP learning algorithm are a static learning rate versus a dynamic learning rate and online training mode versus batch mode. The RPROP learning algorithm is an improvement of the backpropagation learning algorithm and it has proven its superiority in many cases [5], [6]. In general, the RPROP learning algorithm converges faster to global or local minima. As a consequence, it can be expected that the RPROP learning algorithm is more sensitive to, that is, responds faster to error-surface changes. Therefore, this might be the reason that the RPROP learning algorithm requires a less aggressive, but more smoothly changing error-surface.

Tables III, IV and V 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

TABLE III
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 IV
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*		

was still able to find a solution for the 8-2-8 encoder in 88%.

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 VI 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.

D. 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.

TABLE V
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*		

TABLE VI
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*		

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 3503^{rd} 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. 4. 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. 5.

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

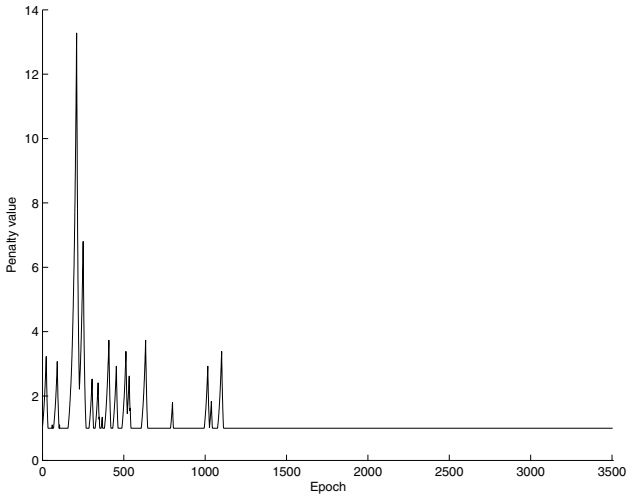

Fig. 4. Representative graph of the change of the values of a penalty from the first group

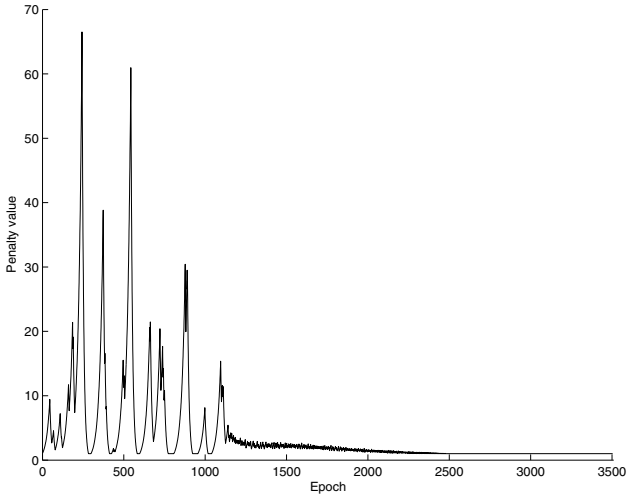

Fig. 5. Representative graph of the change of the values of a penalty from the second group

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.

E. 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. The maximum penalty value z^{max} seems to have a rather small influence on the performance of the learning algorithm, at least if it is set to a large enough value. How to determine the decremental and incremental penalty values efficiently remains an open problem.

IV. 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 order 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 number 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 also 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.

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," in *Proceedings of NeuroNimes 1988*, Nimes, France, 1988.

[3] T. Tollenaere, "SuperSAB: fast adaptive back propagation with good scaling properties," *Neural Networks*, 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," in *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, 1993, pp. 586–591.

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

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

[8] J. Robert M. Burton and G. J. Mpitsos, "Event-dependent control of noise enhances learning in neural networks," *Neural Networks*, 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 Transactions on Neural Networks*, vol. 9, no. 4, pp. 662–668, July 1998.

[10] P. Werbos, "Backpropagation: past and future," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, 1988, pp. 343–353.

[11] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," in *Proc. of the 1988 Connectionist Summer School*. Morgan Kaufman, 1988.

[12] P. Marrone. (2005) Java object oriented neural engine (JOONE). [Online]. Available: http://www.jooneworld.com