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

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

バックプロパゲーション及び類似アルゴリズムに対する適応形ペナルティに基づく学習改善法

Boris Jansen[†]
ヤンセン ボリス

Kenji Nakayama[‡]
中山 謙二

[†] Division of Mathematics and Information Sciences
Graduate School of Natural Science and Technology, Kanazawa University
E-mail: boris@leo.ec.t.kanazawa-u.ac.jp
金沢大学大学院 自然科学研究科 数理情報科学専攻

[‡] Division of Electrical Engineering and Computer Science
Graduate School of Natural Science and Technology, Kanazawa University
E-mail: nakayama@t.kanazawa-u.ac.jp
金沢大学大学院 自然科学研究科 電子情報科学専攻

ABSTRACT

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 in the correct half of the output range, instead of 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 simulation results demonstrate the usefulness and power of the new method.

あらまし

階層形ニューラルネットワークの学習法としてよく用いられるバックプロパゲーション（ＢＰ）アルゴリズムに対して，学習の収束性を改善する多くの方法が提案されている．本稿では，ＢＰアルゴリズムやRPROP法などの類似する学習法を対象として，新しい適応形ペナルティに基づく学習法を提案する．学習で用いられる出力誤差をペナルティにより増減する．ペナルティは出力が目標値と同じ放物面にあれば小さく，そうでなければ大きく制御される．これにより，局所解に陥ることを防ぐことが出来，最適解への収束性を高めることが出来る．多くの例を用いてシミュレーションを行った結果，提案方法の有効性が確認できた．

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 [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 right *ball-park*, i.e. 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 improved algorithms are compared to their original counterparts. The superiority of the new proposed method is demonstrated.

2 Motivation behind Development of New Approach

During our simulations using artificial neural networks in previous research projects, we experienced that once all the binary outputs for all training patterns are in the correct ballpark, the networks converge very fast to a global minimum.

The reason behind this observation might be found in the following. Intuitively, it can be expected that networks will converge faster to a global minimum once all the extremes of the function represented by the network are roughly at the same locations as the extremes of the function to be approximated, compared to networks still having incorrect or missing extremes. Networks having the extremes of the function to be approximated at the correct locations only need to adjust the amplitude of the extremes, while networks having incorrect or missing extremes need to make more radical modifications to their weight vector in order to resolve the incorrect or missing extremes.

Consequently, considering networks having only binary output neurons, once all the outputs for all output neurons and all training patterns are in the correct ballpark, it implies that the function represented by the network has its extremes roughly at the same locations as the function to be approximated[1] and the network only needs to adjust the amplitude of the extremes.

As a result of our observation, we tried to develop an algorithm that initially forces the network to get all the outputs of the training patterns into the correct ballpark, instead of mainly focusing on minimizing the difference between the target and actual outputs as is done in standard squared-error cost functions.

3 New Adaptive Penalty-Based Learning Extension

3.1 Formal Description

In the backpropagation learning algorithm, the errors of the 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 actual output $o_{i,p}(n)$:

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

[1]The real function to be approximated is usually unknown, but is represented by the training data.

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 in the following equation:

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

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

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

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

3.2 Idea behind New Approach

The application of the new 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 in the correct ballpark.

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 get a certain output for a certain pattern in the correct ballpark, the more often it resides in the incorrect ballpark. Every epoch an output for a certain pattern resides in the incorrect ballpark, its corresponding penalty is increased in order to put more pressure on the network to get the output in the correct ballpark. Once an output for a certain penalty reaches its correct ballpark, its corresponding penalty is decreased.

Consequently, the new method implicitly provides a mechanism to escape local minima. Whenever a network converges towards a local minimum, penalties will be increased for the outputs residing in the incorrect ballpark during training. Once the penalties have been raised to large enough values, the network might 'jump' out of the local minimum. From a different point of view, the error surface can be considered dynamic.

4 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 standard backpropagation and the RPROP learning algorithm extended with the new penalty-based method on one side and their original counterparts on the other side on various problem instances.

4.1 Test Problems

4.1.1 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.

In literature [4], the parity problem is often considered as an ill-suited benchmark problem to be solved by ANNs. For most real world problems, it is highly desirable that the applied ANNs generalize well on the training data, i.e. similar input patterns are mapped to similar output activations. However, the parity problem does not inherit this characteristic. A single 'flip' of a bit in the input string requires a complementary classification.

Still, we have included this problem in our experiments, because it is a very hard problem to be solved by ANNs and we believe that the simulation results will demonstrate the power of the new algorithm. In addition, these results in combination with the results reported for the other test problems resembling more real world problems, will provide the reader with a better understanding of the overall performance that can be gained by applying the new proposed method.

4.1.2 M-N-M Encoder

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. 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.

4.1.3 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 in order to be able to solve this problem, is the usage of shortcut connections. 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.

4.2 Simulation Setup

The neural networks used in our simulations have been developed using the Java Object-Oriented Neural Engine (Joone), an open source neural net framework implemented in Java [11].

All the adapted neural networks used in our experiments are multilayer feed-forward neural networks. Here, backpropagation 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 learning algorithm 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 an output of a neuron is close to one of its asymptotic output values.

The 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, neglecting unsuccessful runs, are reported.

4.3 Simulation Results

Tables[2] 1 and 2 show the simulation results for the 6-bit and 8-bit parity problem, respectively. The rows representing experiments of the learning algorithms extended with the new proposed method are indicated with the '+ Extension' text in the algorithm column.

[2] η is the learning rate used in BP, Δ_{max} is the maximum update-value used in the RPROP learning algorithm

Table 1: Simulation Results for 6-Bit Parity Problem

6-Bit Parity			
Algorithm	Epochs	Success	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*			

Table 2: Simulation Results for 8-Bit Parity Problem

8-Bit Parity			
Algorithm	Epochs	Success	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*			

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 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 parameters 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 backpropagation and RPROP 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 changing error-surface.

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

Table 3: Simulation Results for 8-2-8 Encoder Problem

8-2-8 Encoder			
Algorithm	Epochs	Success	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*			

It can be easily noticed that the learning algorithms extended with the new approach outperform their original counterparts also on the encoder problem. For a large range of different learning rates, stan-

Table 4: Simulation Results for 32-2-32 Encoder Problem

32-2-32 Encoder			
Algorithm	Epochs	Success	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	Success	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 6: Simulation Results for Two Spirals Problem

Two Spirals			
Algorithm	Epochs	Success	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*		

dard 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%. 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.

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 to convergence is decreased.

4.4 Extension parameters

It is often said that the design of a neural network is more an art than a science in the sense that many of the numerous factors involved in the design are indeed the results of one's own personal experience [12].

Also 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}, has a rather small influence on the performance of the learning algorithm, at least if it is set to a large enough value.

Unfortunately, how to determine the decremental and incremental penalty values remains an open problem. Choosing incorrect parameter values, especially assigning a value too large to z^+, results in a too dynamic error-surface and the network often drives its weights into saturation. However, by assigning values very close to one to z^- and z^+, $z^1 < 1$ and $z^+ > 1$, the error-surface will change very slowly, but very smoothly. It is our experience that whenever penalties reach the upper bound z^{max} during training, the incremental penalty value z^+ needs to be decreased, or the decremental penalty value z^- needs to be decreased, or both. We intent to research ways to decide the decremental and incremental parameter values in future research.

5 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 ballpark, 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 ap-

proach. 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.

Future research will be directed towards learning tasks consisting of patterns having continuous target output values. We intent to investigate on how to decide an appropriate border defining the ballparks 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," 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 back-propagation 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," Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, pp.586–591, 1993.

[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, Clarendon Press, Oxford, 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," Proceedings of the IEEE International Conference on Neural Networks (ICNN), pp.343–353, 1988.

[11] P. Marrone, "Java object-oriented neural engine (JOONE)," 2005.

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