氏　　名	BORIS JANSEN
学位の種類	博士(工学)
学位記番号	博甲第８５７号
学位授与の日付	平成１８年９月２８日
学位授与の要件	課程博士（学位規則第４条第１項）
学位授与の題目	Neural Networks Empowered by Adaptive Penalty-Based Learning Extension and Its Application to Integer Factorization Problem（ニューラルネットワークにおける適応ペナルティに基づく学習法と素因数分解への応用）
論文審査委員（主査）	中山　謙二（自然科学研究科・教授）
論文審査委員（副主査）	船田　哲男（自然科学研究科・教授），笠原　禎也（総合メディア基盤センター・助教授），堀田　英輔（自然科学研究科・講師），平野　晃宏（自然科学研究科・講師）

Abstract— **Artificial neural networks (ANNs) are a computing paradigm inspired by the way biological nervous systems process information. One of the most famous learning algorithms for ANNs is the back-propagation learning algorithm. Although it has proved its efficiency over the years, its convergence speed often tends to be very slow and it often yields suboptimal solutions. As a result, many improvements to the back-propagation learning algorithm have been reported. In the first part of this dissertation, a new adaptive penalty-based learning extension for gradient descent learning algorithms 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 superiority of the new proposed method is demonstrated. The percentage of successful runs can be greatly increased and the average number of epochs to convergence can be well reduced.**

In the second part of this thesis, ANNs are applied to the integer prime-factorization problem. Nowadays, this problem finds its application often in modern cryptography. A composed number N is applied to the neural networks, and one of its prime factors p is obtained as the output. Previously, ANNs dealing with the data in a decimal format have been proposed. However, accuracy is not sufficient. We have proposed a binary approach. The input N as well as the desired output p are expressed in a binary form. The proposed neural networks are expected to be more stable, i.e. less sensitive to small errors in the network outputs. Simulations have been performed and the results are compared with the results reported in the previous study. The number of required search times for the true prime number can be well reduced. Furthermore, the ANNs have been applied to larger problem instances. Finally, the probability density function of the training patterns is investigated and the need for different data creation or selection techniques is shown.

I. Adaptive Penalty-Based Learning Extension

A. Idea behind New Approach

Consider learning of artificial neural networks with binary target values ± 1. The learning process can be divided into two phases. In the first phase, an ANN is trained so as to move all its outputs to the correct half of the output range. In the second phase, the ANN is trained so as to move its outputs located in the correct region towards the actual targets.

We have proposed an adaptive penalty-based learning extension [1]. In this method, learning for the outputs located on the wrong side, are 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. Furthermore, in order to make the learning process more stable, penalties are gradually decreased after the outputs have been moved to the correct side.

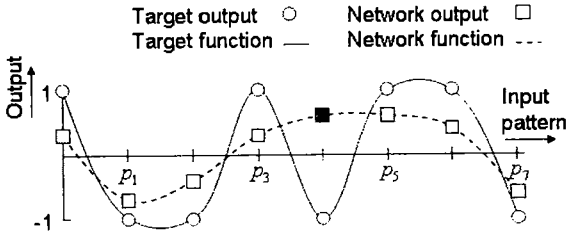

Fig. 1. Network having an output residing in the incorrect half of the output range

Figure 1 shows an example. There is one input pattern p_4, whose output 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 proposed method, the correction term for pattern 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.

B. Formal Description

In the back-propagation learning algorithm, the output errors are back-propagated through the network. 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 new error back-propagated 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 back-propagated error signal. The task of these penalties is to put pressure on the network to get all the outputs initially into the correct half.

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.

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.

C. Comparative Study

In order to give an indication of the performance of the new proposed method, comparisons have been performed between the standard back-propagation 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.

1) N-Bit Parity Problem: is concerned with detecting whether the number of activated input bits is even or odd. The N-bit parity problem is considered as a very hard problem to be solved by ANNs, because a single 'flip' of a bit in the input string requires a complementary classification.

TABLE I
SIMULATION RESULTS FOR 8-BIT PARITY PROBLEM

8-Bit Parity			
Algorithm	Epochs	SR	Settings
BP	7663	2/25	η : 0.0005
RPROP	-	0/25	Δ_{max} : 0.001
BP + Extension	4931	23/25	η : 0.0005, 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*		

2) M-N-M Encoder Problem: is concerned with learning an auto-association of M different patterns, where each pattern has only one active bit. The applied network is a two-layered M-N-M feedforward neural network, having less hidden neurons than the number of input and output neurons. Consequently, the hidden neurons perform compression, while the output neurons perform decompression.

A constant value of 10000 was used for the maximum penalty z^{max}. Learning was considered complete, if the '40-20-40' criterion 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} was set to 20000 epochs. For each problem instance, 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 I shows the simulation results for the 8-bit parity problem. 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 back-propagation 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.

Tables II and III show the results for the 8-2-8 and 48-2-48 encoder problem, respectively. The learning algorithms extended with the new approach outperform their original counterparts also for the encoder problem. Standard back-propagation was unable to find a solution for the encoder problems. However, back-propagation extended with the new method 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. However, 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 II

SIMULATION RESULTS FOR 8-2-8 ENCODER PROBLEM

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

TABLE III

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*		

II. NEURAL NETWORKS APPLIED TO INTEGER FACTORIZATION PROBLEM

A. Problem Description

ANNs are applied [2] in order to factor integers N, which are the product of two odd primes p and q, i.e. $N = p \cdot q$. Throughout the article we will assume $p \leq q$. Here, given N we focus on obtaining the smaller prime p, i.e. we try to approximate the mapping $N \rightarrow p$. It is mentioned for a later reference that it has been proved that N can be factored using any multiple of $\varphi(N) = (p-1)(q-1)$.

B. Neural Network for Factorization

Multilayer feedforward networks with a single hidden layer, adopting a sinusoidal activation function in the hidden layer and a hyperbolic tangent activation function in the output layer have been applied. The networks have been trained by the RPROP learning algorithm. We have experimented with single neural networks (SNNs) as well as multi-model neural networks (MNNs) consisting of three independent ANNs, where the networks do the computation in parallel and the final decision is made by averaging over all outputs. Both the input N and the desired output p were represented in a binary form. During evaluation, any output of an output neuron greater than or equal to zero was considered as an upper bit, while any output less than zero was considered as a lower bit.

Furthermore, the binary output target values have been set to ± 0.7 instead of the asymptotic values ± 1 of the hyperbolic tangent activation functions. This way, the tendency of the networks driving their free parameters to infinity and slowing down the learning process considerably is removed. A constant value was added to the derivative of the hyperbolic tangent activation functions to overcome the so-called 'flat spot' 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. The previously proposed adaptive penalty-based learning extension has also been applied. Finally, besides N, the networks are supplied with extra information in the form of the number of active bits of N. This number is also expressed in a binary form.

TABLE IV

RESULTS FOR IMPROVED NETWORKS TRAINED WITH 66% OF THE DATA SET WITH $N = p \cdot q \leq 10000$

Topology	Epochs	β_0	β_1	β_2	β_3	Data
18-20-7 SNN	5000	67%	80%	91%	97%	Train
		60%	70%	82%	93%	Test
18-20-7 MNN-3	15000	72%	85%	94%	99%	Train
	$3 \cdot 5000$	63%	72%	83%	93%	Test

C. Performance Measure

Two different performance measures are considered. The *binary complete measure*, denoted by β_0, indicates the percentage of the data for which the network produces the exact desired output. The *binary near measure*, denoted by β_i where $i \geq 1$, indicates the percentage of the data for which at most i bits are incorrect in the output.

We felt the need to introduce this second measure. Whenever the network is unable to produce the exact desired output for a certain input, it does not necessary mean that the network output is useless. If the network output contains just a small number of bit errors, the exact target value can still be found within a predetermined number of trial and error procedures. It is very easy to verify if a certain value is the target value, i.e. a factor of N, because a division of N by the number should result in another whole number with no remainder. Therefore, this second measure, which gives an indication of the distance to the exact desired output, provides a better understanding of the real network performance rather than relying on the binary complete measure alone.

D. Simulations

All presented results are averaged over 10 independent runs for each problem instance. Table IV shows the results of networks trained on the problem instance $N < 10000$. Here, 66% of the data set was used for training.

Trying larger problem instances and much smaller training sets, the networks maintain the ability to adapt to the training data and generalize on the test data. This is illustrated in Table V, where neural networks were trained on the problem instance $N < 2^{20} = 1048576$ using only 5% of the data set for training and the remaining part for validation.

E. Comparison between Previous Approach and Current Study

Previously, Meletiou et al. investigated the ability of ANNs to factor integers and reported promising results [3]. Here we address the same problem. However, two major differences

TABLE V

RESULTS FOR IMPROVED NETWORKS TRAINED WITH 5% OF THE DATA SET WITH $N = p \cdot q < 2^{20} = 1048576$

Topology	Epochs	β_0	β_1	β_2	β_3	β_4	Data
25-100-10 SNN	10000	48%	59%	73%	87%	95%	Train
		46%	51%	60%	73%	86%	Test
25-100-10 MNN-3	30000	55%	68%	82%	92%	98%	Train
	$3 \cdot 10000$	50%	55%	64%	76%	88%	Test

TABLE VI

RESULTS REPORTED BY MELETIOU ET AL. FOR NETWORKS TRAINED WITH 66% OF THE DATA SET WITH $N = p \cdot q \leq 10000$

Topology	Epochs	μ_0	$\mu_{\pm 2}$	$\mu_{\pm 5}$	$\mu_{\pm 10}$	$\mu_{\pm 20}$	Data
1-5-5-1	80000	3%	15%	35%	65%	90%	Train
		5%	20%	40%	60%	90%	Test

exist between their study and ours. The differences occur in the approach of solving the integer factorization problem by ANNs, more specifically the differences are in the representation of the data (decimal format) and the function to approximate ($N \rightarrow \phi(N)$).

In Table VI, the results reported by Meletiou et al. are shown for the problem instances $N \leq 10000$, where 66% of the data set was used for training. The measure $\mu_{\pm k}$ indicates the percentage of the data for which the difference between the desired and actual output does not exceed $\pm k$ of the real target. Comparing those results with the results reported in the Table IV, it can be easily noticed that our proposed ANNs outperform the networks proposed in the previous study.

F. Probability Density

Whenever N is restricted by a certain upper bound, i.e. $N < M$ and we consider all possible patterns $N \rightarrow p$ within that limitation, then the lower the value of the smaller prime p the higher the density of patterns. The density of patterns with respect to p for $N \leq 10000$ is shown in Fig. 2.

An observation of the experimental results shows that the networks performed very well on the data patterns having

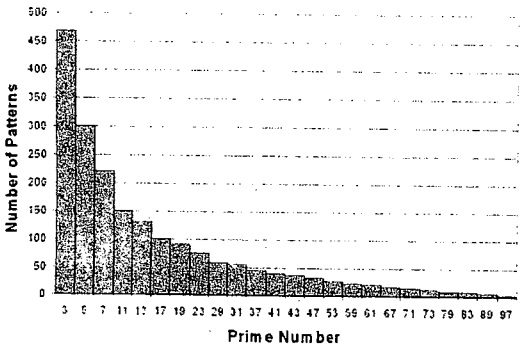

Fig. 2. Pattern density for $N \leq 10000$

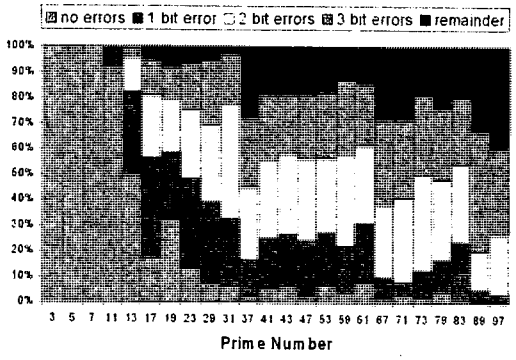

Fig. 3. Percentages of bit errors for $N \leq 10000$

a high density with respect to p and that the network performance gradually decreases as the density of the (training) patterns decreases. This can be seen in Fig. 3, which shows the percentages of bit errors for all patterns with respect to p.

This effect, where the performance of the network is in accordance with the density of the training data, is a natural occurrence. However, in case of the factorization problem related to cryptosystems, it is wishful to obtain a similar or even better performance for N composed of large prime numbers compared to N composed of smaller prime numbers. How to extend the ability of ANNs in order to solve the factorization problem for sparse N composed of larger primes remains an open problem.

III. CONCLUSIONS

In conclusion, a new adaptive penalty-based learning extension has been proposed. The new method initially puts pressure on ANNs in an attempt to get all outputs for all training patterns in the correct half of the output range, instead of mainly focusing on minimizing the difference between the target and actual outputs. By applying the proposed algorithm, the rate of successful runs can be greatly increased and the average number of epochs to convergence can be well reduced.

Furthermore, the ability of ANNs for solving the integer factorization problem has been studied. Multilayer neural networks have been optimized, by using a binary expression of the input and the output data and focusing on p, which is the smaller prime of N, to be obtained as the network output. Simulation results have demonstrated the superiority of the proposed ANNs over previously proposed networks.

REFERENCES

[1] B. Jansen and K. Nakayama, "An adaptive penalty-based learning extension for the backpropagation family," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 89-A, no. 8, August 2006.

[2] B. Jansen and K. Nakayama, "Neural networks following a binary approach applied to the integer prime-factorization problem," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN'05)*. Montréal, Canada: IEEE & INNS, pp. 2577–2582, July-August 2005.

[3] G. C. Meletiou, D. K. Tasoulis, and M. N. Vrahatis, "A first study of the neural network approach in the RSA cryptosystem," in 6^{th} *IASTED International Conference Artificial Intelligence and Soft Computing ASC 2002*, Banff, Canada, pp. 483–488, July 2002.

学位論文審査結果の要旨

平成１８年７月３１日に開催された第１回学位論文審査委員会，及び平成１８年８月２日に行われた口頭発表と第２回学位論文審査委員会で審査した結果，以下の通り判定した。

ニューラルネットワーク（NN）の学習法として誤差逆伝搬法が広く用いられているが，収束が遅いことやローカルミニマムに陥りやすいという問題がある。

本論文では，適応形ペナルティを導入した新しい学習法を提案している。ある入力パターンに対する，ある出力ノードの誤差を適応形ペナルティで増幅・減衰の制御を行うことにより，ローカルミニマムを回避し，学習の高速化を達成している。これはNNの学習において特に重要な部分を入力パターン，出力ノードに対して検出し，それを強調学習する方法であり，新たなアプローチである。また，本提案方法を多くの問題に適用し，シミュレーションによりその有効性を検証している。

次に，本論文では，NNによる素因数分解の方法を研究している。$N=pq$（$p \leqq q$が素因数）をNNに入力し，pを出力する方式とし，さらに，出力のバイナリー表現，出力＝１，０に対するリラックス表現，上記の適応形ペナルティ学習法，汎化能力を高める方法等を適用し，従来法に比べ素因数分解の精度を大幅に向上し，かつ，Nとして大きな数字に適用可能であることを示している。

以上の研究成果は，NNにおける誤差逆伝搬を基礎とする学習ファミリーに対して学習の安定化と高速化を可能とするものであり，学術的及び実用的価値が高く，博士論文に値するものと判定する。