PERGAMON

Neural Networks 13 (2000) 755–764

Neural
Networks

www.elsevier.com/locate/neunet

Contributed article

Adaptive natural gradient learning algorithms for various stochastic models

H. Park[a,*], S.-I. Amari[b], K. Fukumizu[c]

[a]*Department of Computer Science, Yonsei University, Seoul, South Korea*
[b]*Brain Science Institute, RIKEN, Wako, Saitama, Japan*
[c]*Institute of Statistical Mathematics, Tokyo, Japan*

Received 27 April 1999; revised 26 June 2000; accepted 26 June 2000

Abstract

The natural gradient method has an ideal dynamic behavior which resolves the slow learning speed of the standard gradient descent method caused by plateaus. However, it is required to calculate the Fisher information matrix and its inverse, which makes the implementation of the natural gradient almost impossible. To solve this problem, a preliminary study has been proposed concerning an adaptive method of calculating an estimate of the inverse of the Fisher information matrix, which is called the adaptive natural gradient learning method. In this paper, we show that the adaptive natural gradient method can be extended to be applicable to a wide class of stochastic models: regression with an arbitrary noise model and classification with an arbitrary number of classes. We give explicit forms of the adaptive natural gradient for these models. We confirm the practical advantage of the proposed algorithms through computational experiments on benchmark problems. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Feedforward neural network; Gradient descent learning; Plateau problem; Natural gradient learning; Adaptive natural gradient learning

1. Introduction

Feedforward neural networks or multilayer perceptrons have been applied successfully to solve a variety of difficult and diverse problems by using the gradient descent learning method known as the error backpropagation algorithm. However, it is known that the backpropagation method is extremely slow in many cases, which has been a serious obstacle when we use neural networks in real world applications. This slow convergence is caused by the plateau phenomenon which is ubiquitous in the backpropagation type learning, taking a dominant part in the whole learning process (Saad & Solla, 1995).

Although there have been a lot of techniques for accelerating convergence, most of them cannot solve the plateau problems. The adaptive learning rate and the momentum method can provide somewhat improved learning speed (LeCun, Bottou, Orr, & Müller, 1998), but it cannot avoid plateaus because they basically use the same direction as the one used in the standard backpropagation algorithm. The second order methods such as the Newton method, conjugate gradient, and Quasi-Newton method (e.g. BFGS) are alternative approaches which try to find the steepest descent direction to the optimal point at a given position (LeCun et al., 1998). However, since these methods are based on the quadratic approximation, it works well only in a small neighborhood of the optimal point. In addition, most of the second order methods can only be applied to the batch-mode learning, which is not appropriate for a large data set. Recently, Ampazis, Perantonis, and Taylor (1999) proposed a new method which uses eigenvalues of the Hessian matrix of the cost function to avoid or escape from plateaus. However, the computational cost will be very large for large-scale problems.

On the other hand, the concept of natural gradient was proposed to define the steepest direction of a loss function in the parameter space based on information geometry (Amari, 1985; Amari & Nagaoka, 2000). Amari (1998) showed that the natural gradient achieves Fisher efficiency, and Amari (1998) and Yang and Amari (1998) suggested that the natural gradient algorithm has the possibility of avoiding or alleviating plateaus. This possibility has been theoretically confirmed by statistical–mechanical analysis (Rattray, Saad, & Amari, 1998; Rattray & Saad, 2000). Comparing with other second-order on-line methods, the natural gradient method can give a more general learning scheme in the sense that it can be applied for various error functions.

* Corresponding author. Tel.: +82-2-365-4598; fax: +82-2-365-2579.
E-mail address: hypark@csai.yonsei.ac.kr (H. Park).

0893-6080/00/$ - see front matter © 2000 Elsevier Science Ltd. All rights reserved.
PII: S0893-6080(00)00051-4

Nomenclature

$\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}$	Input, output, and parameter vectors of a network, respectively
$p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}), r(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}), q(\boldsymbol{x})$	Probability density function
$l(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta})$	Loss function
$\nabla l, \tilde{\nabla} l, \hat{\nabla} l$	Ordinary gradient, natural gradient, and adaptive natural gradient of loss function, respectively
$G(\boldsymbol{\theta}), \hat{G}$	Fisher information matrix and its estimation
$f_i(\boldsymbol{x},\boldsymbol{\theta})$	A deterministic function for *i*th output node of a network
$F(\boldsymbol{x},\boldsymbol{\theta})$	A vector of $f_i(\boldsymbol{x},\boldsymbol{\theta}),\ i = 1,2,\ldots,L$
L, M	The number of output nodes and the number of parameters of a network
$\eta_t, \alpha_t, \epsilon_t$	User defined parameters in learning

Moreover, it has a theoretically rigorous justification (Rattray & Saad, 2000). Rattray and Saad (1998) also analyzed the dynamics of a number of second order online learning algorithms and the natural gradient learning algorithm and showed the superiority of natural gradient learning method in the transient dynamics of learning. However, there are some problems concerning the implementation of the natural gradient learning method. Firstly, it is required to know the input distribution in order to get the explicit form of the Fisher information matrix which is essential to get the natural gradient. This information is hardly given in practical applications. In addition, even though we know the input distribution, inverse operation on the Fisher information matrix is necessary at each learning step, which is very time consuming when the number of parameters is large.

To solve these problems, Amari, Park, and Fukumizu (2000) proposed an adaptive method of obtaining an estimate of the natural gradient (adaptive natural gradient method) without any information on the input distribution and inverse operation on the Fisher information matrix. They gave an explicit form of the adaptive natural gradient learning method for the single-output stochastic multilayer perceptron model with additive Gaussian noise. Since the natural gradient learning can be applied to various stochastic models, the adaptive natural gradient learning can also be applied to them. The more appropriate model we use for a given problem, the better performance we can expect.

The purpose of this paper is to extend the adaptive natural gradient method to various stochastic models used in practical applications. More specifically, we extend the method to be applicable to more general classes of multi-dimensional regression and classification problems. In the case of regression, we introduce stochastic models having general probability functions other than the Gaussian one. This corresponds to a general loss function other than the squared error. In the case of classification, given an input, the output of the underlying network is required to generate the probability distribution of its classes. We take the Kullback–Leibler error criterion to train the network, and generalize our method to be applicable to such cases. We give a general description of the adaptive natural gradient method for stochastic neural networks, and give its explicit learning algorithms for the typical stochastic models used in practical applications of neural networks. We also conduct computational experiments using the proposed algorithms to show their advantage in practical problems. The results show that the present methods have very fast convergence because they are free of plateau phenomena, showing their practical efficiency.

2. Adaptive natural gradient learning

2.1. Adaptive natural gradient for stochastic neural networks

Before giving various forms of adaptive natural gradient descent learning, let us recapitulate the natural gradient learning algorithm briefly. Natural gradient learning is a kind of stochastic gradient descent learning method. We consider the space of related probability density functions (pdfs) $\{p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}) | \boldsymbol{\theta} \in \Re^M\}$ of stochastic neural networks with input $\boldsymbol{x}$, output $\boldsymbol{y}$ and a parameter vector $\boldsymbol{\theta}$. The goal of learning is to find the optimum $\boldsymbol{\theta}^*$ that maximizes the log likelihood function. The loss function of learning can be defined as

$$l(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}) = -\log p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}) = -\log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})q(\boldsymbol{x}) \tag{1}$$

$$= -\log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) - \log q(\boldsymbol{x}), \tag{2}$$

where $q(\boldsymbol{x})$ is the pdf of the input $\boldsymbol{x}$ and $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ the conditional pdf of $\boldsymbol{y}$ conditioned on $\boldsymbol{x}$. Using this loss function, the ordinary gradient ∇l is calculated and the learning algorithm is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla l(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \eta_t \frac{\partial l(\boldsymbol{x}_t,\boldsymbol{y}_t,\boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}_t}, \tag{3}$$

where η_t is a learning rate.

The natural gradient learning method is based on the fact that the space of $p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})$ is a Riemannian space in which the metric tensor is given by the Fisher information matrix $G(\boldsymbol{\theta})$ defined by

$$G(\boldsymbol{\theta}) = \int\int \frac{\partial \log p}{\partial \boldsymbol{\theta}} \left(\frac{\partial \log p}{\partial \boldsymbol{\theta}}\right)^{\mathrm{T}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})q(\boldsymbol{x})\mathrm{d}\boldsymbol{y}\,\mathrm{d}\boldsymbol{x} \tag{4}$$

$$= E_{\boldsymbol{x}}\left[E_{\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}}\left[\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \left(\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right)^{\mathrm{T}} \right] \right], \quad (5)$$

where $E_x[\cdot]$ and $E_{\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}}[\cdot]$ denote the expectation with respect to $q(\boldsymbol{x})$ and $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$, respectively, and T denotes the transposition. Using this Fisher information matrix, we can obtain the natural gradient $\tilde{\nabla} l$ and its learning algorithm for the stochastic neural networks

$$\tilde{\nabla} l(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta})\nabla l(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta})\frac{\partial l(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \quad (6)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \tilde{\nabla} l(\boldsymbol{\theta}_t). \quad (7)$$

Even though it has been proved that the natural gradient learning algorithm gives a Fisher efficient on-line estimator, there are some problems in the implementation of these algorithms. Firstly, we have to know $q(\boldsymbol{x})$ to get an explicit form of $G(\boldsymbol{\theta})$, but this information is hardly given in practical applications. In addition, even if we can get the explicit form of $G(\boldsymbol{\theta})$, the inversion of $G(\boldsymbol{\theta})$ is necessary in order to get the natural gradient at each learning step, which is very time consuming. To solve these practical problems, Amari et al. (2000) proposed an efficient method of directly obtaining an estimate of $G^{-1}(\boldsymbol{\theta})$ for the stochastic neural network having one output node with Gaussian random noise. The present paper gives a more general perspective on the adaptive natural gradient method. It will be extended to a general form for stochastic neural networks represented by a general conditional pdf of output $\boldsymbol{y}$ given input $\boldsymbol{x}$ specified by the structure of the network. Our final purpose is to develop the explicit algorithms of adaptive natural gradient learning for various stochastic models and loss functions which are widely used in practical applications. To this end, we give a general description of adaptive natural gradient learning for various stochastic neural networks in this section.

Let us assume a stochastic network with L output nodes and M parameters. The network first calculates a deterministic input–output function $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta})$ determined by the network structure, and then the output $\boldsymbol{y}$ is emitted stochastically subject to the conditional probability density of the form

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) = r(\boldsymbol{y}|\boldsymbol{f}((\boldsymbol{x};\boldsymbol{\theta}))), \quad (8)$$

$$f(\boldsymbol{x};\boldsymbol{\theta}) = (f_1(\boldsymbol{x};\boldsymbol{\theta}),\ldots,f_L(\boldsymbol{x};\boldsymbol{\theta})). \quad (9)$$

Here the function r defines the stochastic property of the network.

From the definition of the Fisher information matrix of Eq. (5), taking the expectation on $E_{\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}}[\cdot]$, we have a special type of the Fisher information matrix written as

$$G(\boldsymbol{\theta}) = E_x[\nabla F R (\nabla F)^{\mathrm{T}}], \quad (10)$$

where

$$\nabla F = (\nabla f_1(\boldsymbol{x},\boldsymbol{\theta}),\ldots,\nabla f_L(\boldsymbol{x},\boldsymbol{\theta})), \quad (11)$$

$$R = [R_{ij}] = \left[E_{\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}}\left[\frac{1}{r^2}\frac{\partial r}{\partial z}\left(\frac{\partial r}{\partial z}\right)^{\mathrm{T}} \right] \right], \quad (12)$$

$$z = \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta}). \quad (13)$$

The adaptive method of obtaining an estimate $\hat{G}_{t+1}$ of $G(\boldsymbol{\theta}_t)$ is given by

$$\hat{G}_{t+1} = (1-\epsilon_t)\hat{G}_t + \epsilon_t \nabla F_t R^{1/2}(\nabla F_t R^{1/2})^{\mathrm{T}}, \quad (14)$$

$$\nabla F_t = (\nabla f_1(\boldsymbol{x}_t,\boldsymbol{\theta}_t),\ldots,\nabla f_L(\boldsymbol{x}_t,\boldsymbol{\theta}_t)), \quad (15)$$

where $R^{1/2}$ denotes the square root of the positive symmetric matrix R and ϵ_t is a time dependent learning rate. Typical examples are $\epsilon_t = c/t$ and $\epsilon_t = \epsilon$. From this type of equation, the inverse matrix $\hat{G}_t^{-1}$ can be directly obtained by

$$\hat{G}_{t+1}^{-1} = \frac{1}{1-\epsilon_t}\hat{G}_t^{-1} - \frac{\epsilon_t}{1-\epsilon_t}\hat{G}_t^{-1}\nabla F_t R^{1/2}((1-\epsilon_t)I + \epsilon_t R^{1/2}\nabla F_t^{\mathrm{T}}\hat{G}_t^{-1}\nabla F_t R^{1/2})^{-1}R^{1/2}\nabla F_t^{\mathrm{T}}\hat{G}_t^{-1}. \quad (16)$$

One can see that there is still an inverse operation as well as a matrix square-root operation in Eq. (16). However, the size of matrix $R^{1/2}\nabla F_t^{\mathrm{T}}\hat{G}_t^{-1}\nabla F_t R^{1/2}$ is $L \times L$ whereas the size of the Fisher information matrix is $M \times M$. Since the number of output nodes is much smaller than the number of parameters in general, this inverse operation is not so time consuming. In addition, the square-root part may disappear when we use some types of $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ in practical applications, as we will show later. In addition, in practical applications, ϵ_t is small so that Eq. (16) is well approximated to be the simpler form

$$\hat{G}_{t+1}^{-1} = (1+\epsilon_t)\hat{G}_t^{-1} - \epsilon_t \hat{G}_t^{-1}\nabla F_t R \nabla F_t^{\mathrm{T}}\hat{G}_t^{-1} \quad (17)$$

by neglecting the small value of the order $\mathrm{o}(\epsilon_t)$.

The final form of the adaptive natural gradient $\hat{\nabla} l$ and its learning algorithm can be given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \hat{\nabla} l(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \eta_t \hat{G}_{t+1}^{-1}\nabla l(\boldsymbol{\theta}_t). \quad (18)$$

In the implementation of the adaptive natural gradient, we need to find the explicit form of R in Eq. (14) which is determined by the conditional probability distribution function $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. The function $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ or $r(\boldsymbol{y}|\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\theta}))$ can be defined in various forms according to the problems we want to solve, and we expect a better performance using an appropriate type of $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. Problems can be divided into two classes based on the characteristics of the stochastic model. They are regression problems and classification problems. In this paper, we define the representative types of $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ for regression problems and classification problems, respectively, and give the explicit form of adaptive natural gradient learning for each class of problems.

2.2. Adaptive natural gradient for regression problems

There are many applications which can be considered as regression problems. Function approximation, time series prediction and non-linear system identification are typical examples. A common characteristic of these problems is that the output $\boldsymbol{y}$ takes continuous values. From this fact, we can use the following type of stochastic network:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\xi}, \tag{19}$$

$$\boldsymbol{y} = (y_1, \ldots y_L)^{\mathrm{T}}, \boldsymbol{f} = (f_1, \ldots f_L)^{\mathrm{T}}, \boldsymbol{\xi} = (\xi_1, \ldots \xi_L)^{\mathrm{T}}.$$

with an additive noise or fluctuation $\boldsymbol{\xi}$. The value of each output node y_i is decided by the sum of the output of deterministic function $f_i(\boldsymbol{x}, \boldsymbol{\theta})$ and additional random noise ξ_i which is subject to an input-independent pdf $r_i(\xi_i)$. The deterministic function $f_i(\boldsymbol{x}, \boldsymbol{\theta})$ is the value of the ith output node of the feedforward neural network, and can be written by

$$f_i(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_j^o v_{ij}\sigma(\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x} + b_j) + b_{o_i} \tag{20}$$

where v_{ij}, $\boldsymbol{w}_j$, b_j, b_{o_i} are the weight parameters of the network. If we assume a different structure of the network, $f_i(\boldsymbol{x}, \boldsymbol{\theta})$ can be written in a different form, but the Fisher information matrix $G(\boldsymbol{\theta})$ of Eq. (5) and adaptive estimation method of Eq. (14) do not depend on the explicit form of $f_i(\boldsymbol{x}, \boldsymbol{\theta})$ directly. Assuming that each noise element ξ_i is mutually independent, we can get the conditional probability density function of output $\boldsymbol{y}$ given input $\boldsymbol{x}$ and its logarithm as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \prod_i^L r_i(y_i - f_i(\boldsymbol{x}, \boldsymbol{\theta})), \tag{21}$$

$$\log p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \sum_i^L \log r_i(y_i - f_i(\boldsymbol{x}, \boldsymbol{\theta})). \tag{22}$$

The negative of log likelihood gives a loss function for this model

$$l(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = -\sum_i^L \log r_i(y_i - f_i(\boldsymbol{x}, \boldsymbol{\theta})). \tag{23}$$

Using this stochastic model, we can get the following theorem.

Theorem 1. *Let the conditional probability distribution of the output* $\boldsymbol{y}$ *of a stochastic model given input* $\boldsymbol{x}$ *be*

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \prod_i^L r_i(y_i - f_i(\boldsymbol{x}, \boldsymbol{\theta})). \tag{24}$$

Then the Fisher information matrix $G(\boldsymbol{\theta})$ *and the adaptive estimate* $\hat{G}_{t+1}^{-1}$ *of the inverse of Fisher information matrix is given by*

$$G(\boldsymbol{\theta}) = E_{\boldsymbol{x}}[\nabla\tilde{F}(\nabla\tilde{F})^{\mathrm{T}}], \tag{25}$$

$$\hat{G}_{t+1}^{-1} = \frac{1}{1-\epsilon_t}\hat{G}_t^{-1} - \frac{\epsilon_t}{(1-\epsilon_t)}\hat{G}_t^{-1}\nabla\tilde{F}_t((1-\epsilon_t)I + \epsilon_t\nabla\tilde{F}_t^{\mathrm{T}}\hat{G}_t^{-1}\nabla\tilde{F}_t)^{-1}\nabla\tilde{F}_t^{\mathrm{T}}\hat{G}_t^{-1} \tag{26}$$

or by neglecting the small value of the order $\mathrm{o}(\epsilon_t)$ *under the assumption of small* ϵ_t,

$$\hat{G}_{t+1}^{-1} = (1+\epsilon_t)\hat{G}_t^{-1} - \epsilon_t(\hat{G}_t^{-1}\nabla\tilde{F}_t)(\hat{G}_t^{-1}\nabla\tilde{F}_t)^{\mathrm{T}}, \tag{27}$$

where

$$\nabla\tilde{F}_t = \left(\sqrt{E_{\xi_1}\left[\left(\frac{r_1'}{r_1}\right)^2\right]}\nabla f_1(\boldsymbol{x}_t, \boldsymbol{\theta}_t), \ldots, \sqrt{E_{\xi_L}\left[\left(\frac{r_L'}{r_L}\right)^2\right]}\nabla f_L(\boldsymbol{x}_t, \boldsymbol{\theta}_t)\right). \tag{28}$$

Note that $E_{\xi_i}[(r_i'/r_i)^2]$ does not depend on $\boldsymbol{x}$ and $\boldsymbol{\theta}$, and can be calculated if the noise model is specified.

In addition, we can get a simpler form if we add one more assumption that all $r_i(\xi_i)$ are the same, represented by a common $r(\xi)$. For this case, we can get the following corollary.

Corollary 1. *Let the conditional probability distribution of the output* $\boldsymbol{y}$ *of a stochastic model given input* $\boldsymbol{x}$ *be*

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \prod_i^L r(y_i - f_i(\boldsymbol{x}, \boldsymbol{\theta})). \tag{29}$$

Then the Fisher information matrix $G(\boldsymbol{\theta})$ *and the adaptive estimate* $\hat{G}_{t+1}^{-1}$ *of the inverse of Fisher information matrix is given by*

$$G(\boldsymbol{\theta}) = E_{\xi}[(r'/r)^2]E_{\boldsymbol{x}}[\nabla F(\nabla F)^{\mathrm{T}}], \tag{30}$$

$$\hat{G}_{t+1}^{-1} = \frac{1}{1-\epsilon_t}\hat{G}_t^{-1} - \frac{\epsilon_t}{(1-\epsilon_t)}\hat{G}_t^{-1}\nabla F_t((1-\epsilon_t)I + \epsilon_t\nabla F_t^{\mathrm{T}}\hat{G}_t^{-1}\nabla F_t)^{-1}\nabla F_t^{\mathrm{T}}\hat{G}_t^{-1} \tag{31}$$

or by neglecting the small value of the order $\mathrm{o}(\epsilon_t)$ *under the assumption of small* ϵ_t,

$$\hat{G}_{t+1}^{-1} = (1+\epsilon_t)\hat{G}_t^{-1} - \epsilon_t(\hat{G}_t^{-1}\nabla F_t)(\hat{G}_t^{-1}\nabla F_t)^{\mathrm{T}}, \tag{32}$$

where

$$\nabla F_t = (\nabla f_1(\boldsymbol{x}_t, \boldsymbol{\theta}_t), \ldots, \nabla f_L(\boldsymbol{x}_t, \boldsymbol{\theta}_t)). \tag{33}$$

Eq. (31) can be given by ignoring the constant part $E_{\xi}[(r'/r)^2]$ of Eq. (30) which has no influence on the direction of the gradient. We can see that the matrix square root operation of Eq. (16) disappears in the forms of Eqs. (26) and (31).

For a further simple case where ξ is subject to the

Gaussian distribution with zero mean and variance σ^2, we can get the same Fisher information matrix as Eq. (30) except that $E_{\xi}[(r'/r)^2]$ is replaced by $1/\sigma^2$, and exactly the same form as Eq. (31) for adaptive estimation. We should note here that the explicit form of adaptive natural gradient learning using Eq. (31) has the same form as that of the on-line Gauss–Newton method which is usually applied only to the sum-of-square error function (see LeCun et al., 1998 for details). The sum-of-square error function can be given under the last assumption above, i.e. $\boldsymbol{\xi}$ is subject to the Gaussian noise with a scalar covariance matrix. As we derived above, however, we can get the on-line Gauss–Newton learning method from a more general type of loss function which can be written by

$$l(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}) = -\sum_{i}^{L} \log r(y_i - f_i(\boldsymbol{x},\boldsymbol{\theta})). \tag{34}$$

Of course, if we use a different stochastic model, we can get a different explicit form of adaptive natural gradient learning. The model that we use in this section is clearly sensible for regression problems, but is not always good for the different types of application problem. We may say therefore that the adaptive natural gradient algorithm is a generalization of the Gauss–Newton method with solid theoretical justification.

2.3. Adaptive natural gradient for classification problems

Whereas each output node of a network for regression problems has in general a continuous value decided by a deterministic function and an additional noise, the target output values for classification problems are discrete, representing classes of patterns. The models in Section 2.2 (Eq. (21)) do not provide a good description of their distribution. We therefore need quite a different type of stochastic model. In this section, we use Bayesian stochastic models for classification problems (Bishop, 1995), and give explicit forms of adaptive natural gradient learning for the models.

2.3.1. Case of two classes

Let us first consider relatively simple problems involving only two classes: C_1 and C_2. To categorize $\boldsymbol{x}$ into the two classes, a network with single output would be enough by using a target coding scheme where $y = 1$ is used for class C_1 and $y = 0$ for class C_2. We use the deterministic function $f(\boldsymbol{x}, \boldsymbol{\theta})$ of the form

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(net_o) = \sigma\left(\sum_j v_j \sigma(\boldsymbol{w}_j^{\mathrm{T}}\boldsymbol{x} + b_j) + b_o\right), \tag{35}$$

where $\sigma(\cdot)$ is the logistic sigmoidal function and $v_j, \boldsymbol{w}_j, b_j$ are weight parameters related to hidden node j. We can consider this output value as the posterior probability $P(C_1|\boldsymbol{x})$ for class C_1. The posterior probability for class C_2 is given by $1 - f(\boldsymbol{x}, \boldsymbol{\theta})$, and the conditional probability distribution function of the final output y given $\boldsymbol{x}$ can be written as

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = f(\boldsymbol{x}, \boldsymbol{\theta})^y (1 - f(\boldsymbol{x}, \boldsymbol{\theta}))^{1-y}. \tag{36}$$

By ignoring $\log q(\boldsymbol{x})$ in Eq. (2) which is independent of $\boldsymbol{\theta}$, the loss function can be given by

$$l(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta}) = -y \log f(\boldsymbol{x}, \boldsymbol{\theta}) - (1 - y)\log(1 - f(\boldsymbol{x}, \boldsymbol{\theta})). \tag{37}$$

This type of loss function is called cross entropy function (see Bishop, 1995 for details), and it was confirmed by computational experiments that one can get better performances for classification problems by using the cross entropy loss function than the sum of square loss function (Joost, 1996).

Using this stochastic model, we can get the following theorem.

Theorem 2. *Let the conditional probability distribution of the output y of a stochastic model given input* $\boldsymbol{x}$ *be*

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = f(\boldsymbol{x}, \boldsymbol{\theta})^y (1 - f(\boldsymbol{x}, \boldsymbol{\theta}))^{1-y}. \tag{38}$$

Then, the Fisher information matrix $G(\boldsymbol{\theta})$ *and the adaptive estimate* $\hat{G}_{t+1}^{-1}$ *of the inverse of Fisher information matrix is given by*

$$G(\boldsymbol{\theta}) = E_{\boldsymbol{x}}\left[\frac{1}{f(\boldsymbol{x}, \boldsymbol{\theta})(1 - f(\boldsymbol{x}, \boldsymbol{\theta}))} \frac{\partial f}{\partial \boldsymbol{\theta}} \left(\frac{\partial f}{\partial \boldsymbol{\theta}}\right)^{\mathrm{T}}\right], \tag{39}$$

$$\hat{G}_{t+1}^{-1} = \frac{1}{1 - \epsilon_t}\hat{G}_t^{-1} - \frac{\epsilon_t}{(1 - \epsilon_t)} \frac{\hat{G}_t^{-1}\nabla f_t (\nabla f_t)^{\mathrm{T}} \hat{G}_t^{-1}}{(1 - \epsilon_t) f_t (1 - f_t) + \epsilon_t \nabla f'_t \hat{G}_t^{-1} \nabla f_t} \tag{40}$$

or by neglecting the small value of the order $\mathrm{o}(\epsilon_t)$ *under the assumption of small* ϵ_t,

$$\hat{G}_{t+1}^{-1} = (1 + \epsilon_t)\hat{G}_t^{-1} - \epsilon_t \frac{1}{f_t(1 - f_t)} \hat{G}_t^{-1}\nabla f_t (\nabla f_t)^{\mathrm{T}} \hat{G}_t^{-1}, \tag{41}$$

where

$$\nabla f_t = \frac{\partial f(\boldsymbol{x}_t, \boldsymbol{\theta}_t)}{\partial \boldsymbol{\theta}_t}$$

and $f_t = f(\boldsymbol{x}_t, \boldsymbol{\theta}_t)$.

2.3.2. Case of multiple classes

Let us now consider a more general case that more than two classes are involved. In the case of L-classes classification problems, we need a network with L output nodes so that each output node i represents each class C_i. We use the target coding scheme where $y_i = \delta_{il}$ for class C_l, such that the value of the ith output node, $f_i(\boldsymbol{x}, \boldsymbol{\theta})$, can be considered as representing the posterior probability $P(C_i|\boldsymbol{x})$ for class C_i. The conditional distribution can therefore be written as

$$p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \prod_{i=1}^{L} (f_i(\boldsymbol{x}, \boldsymbol{\theta}))^{y_i}. \tag{42}$$

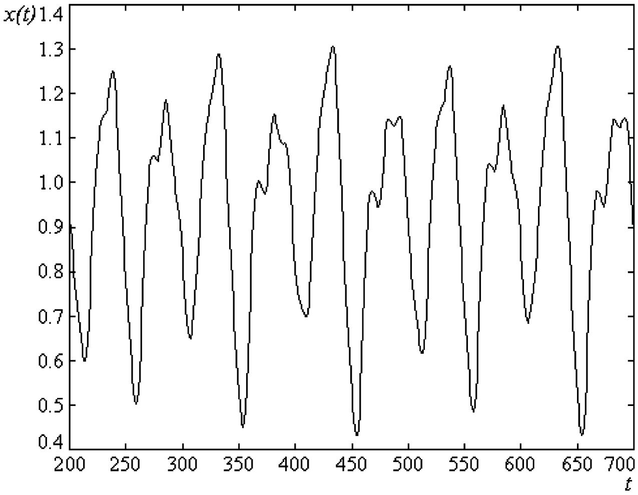

Fig. 1. Mackey–Glass time series.

Since the output values $f_i(\boldsymbol{x}, \boldsymbol{\theta})$ are interpreted as probabilities, they must lie in the range [0,1], and their sum must be equal to 1. This can be achieved by using a generalized form of the logistic function σ_g defined as

$$f_i(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma_g(net_i) = \frac{\exp(net_i)}{\sum_{l=1}^{L} \exp(net_l)}. \tag{43}$$

where net_i is the value of the weighted sum for the ith output node. This scheme was used in Amari (1993). This type of activation function σ_g is known as the normalized exponential or softmax activation function (Bishop, 1995). The difference between the true conditional distribution $p(i|\boldsymbol{x})$ and the distribution $f_i(\boldsymbol{x}, \boldsymbol{\theta})$ given by the network may be measured by the Kullback–Leibler divergence

$$K(\boldsymbol{\theta}) = E_{q(x)}\left[\sum_i p(i|\boldsymbol{x})\log\frac{p(i|\boldsymbol{x})}{f_i(\boldsymbol{x}, \boldsymbol{\theta})}\right]. \tag{44}$$

The expected loss $L(\boldsymbol{\theta})$ or risk function is the expectation of the following loss:

$$l(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = -\sum_{i=1}^{L} y_i \log f_i(\boldsymbol{x}, \boldsymbol{\theta}) \tag{45}$$

except for the entropy term independent of $\boldsymbol{\theta}$. Hence, we have the stochastic descent learning method based on Eq. (45). This was proposed by Amari (1993) (see also Bishop (1995)). Here, we give its adaptive natural gradient version.

Using this stochastic model, we can get the following theorem.

Theorem 3. *Let the conditional probability distribution of output* $\boldsymbol{y}$ *of a stochastic model given input* $\boldsymbol{x}$ *be*

$$p(y|\boldsymbol{x}; \boldsymbol{\theta}) = \prod_{i=1}^{L} (f_i(\boldsymbol{x}, \boldsymbol{\theta}))^{y_i}. \tag{46}$$

Then the Fisher information matrix $G(\boldsymbol{\theta})$ *and the adaptive estimate* $\hat{G}_{t+1}^{-1}$ *of the inverse of Fisher information matrix is given by*

$$G(\boldsymbol{\theta}) = E_{\boldsymbol{x}}\left[\sum_{i=1}^{L} \frac{1}{f_i(\boldsymbol{x}, \boldsymbol{\theta})} \frac{\partial f_i}{\partial \boldsymbol{\theta}}\left(\frac{\partial f_i}{\partial \boldsymbol{\theta}}\right)^{\mathrm{T}}\right], \tag{47}$$

$$\hat{G}_{t+1}^{-1} = \frac{1}{1-\epsilon_t}\hat{G}_t^{-1} - \frac{\epsilon_t}{(1-\epsilon_t)}\hat{G}_t^{-1}\nabla\tilde{F}_t\Big((1-\epsilon_t)I + \epsilon_t\nabla\tilde{F}_t^{\mathrm{T}}\hat{G}_t^{-1}\nabla\tilde{F}_t\Big)^{-1}\nabla\tilde{F}_t^{\mathrm{T}}\hat{G}_t^{-1} \tag{48}$$

or by neglecting the small value of the order $\mathrm{o}(\epsilon_t)$ *under the assumption of small* ϵ_t,

$$\hat{G}_{t+1}^{-1} = (1+\epsilon_t)\hat{G}_t^{-1} - \epsilon_t(\hat{G}_t^{-1}\nabla\tilde{F}_t)(\hat{G}_t^{-1}\nabla\tilde{F}_t)^{\mathrm{T}}, \tag{49}$$

where

$$\nabla\tilde{F}_t = \left(\frac{\nabla f_1(\boldsymbol{x}_t, \boldsymbol{\theta}_t)}{\sqrt{f_1(\boldsymbol{x}_t, \boldsymbol{\theta}_t)}}, \ldots, \frac{\nabla f_L(\boldsymbol{x}_t, \boldsymbol{\theta}_t)}{\sqrt{f_L(\boldsymbol{x}_t, \boldsymbol{\theta}_t)}}\right). \tag{50}$$

3. Experimental results

To check the performance of the proposed algorithms, we conducted computational experiments using the stochastic models that we discussed in the previous section. We used a well-known benchmark problem for each stochastic model, and compared the performance of the adaptive natural gradient learning method with the ordinary gradient learning method with momentum. In each experiment, the initial values of parameters were selected randomly subject to the uniform distribution on $[-0.1, 0.1]^M$, and we conducted ten runs with different initial values for each algorithm and each problem to get an average result. Each learning process stops when the mean square error (MSE) becomes smaller than a desired value, or the number of learning cycle exceeds an appropriate large number. The learning rate η_t for each algorithm and momentum rate α_t for the ordinary gradient learning were adjusted through experiments in order to get fast convergence and a high rate of success. The value of ϵ_t for adaptive natural gradient learning was set to $1/t$.

3.1. Regression problem

For the regression problem, we assume that the random output noise vector is Gaussian with a scalar covariance matrix, so that the loss function becomes the squared-error function. We can therefore use the explicit form of the adaptive natural gradient in Eq. (31). We used a multilayer perceptron with ten hidden units in one hidden layer. The application problem used for the experiment is the Mackey–Glass chaotic time series prediction which is a well-known benchmark problem. The time series data were generated

Table 1
Results on the Mackey–Glass time series prediction problem (OGL: the ordinary gradient learning, ANGL: the adaptive natural gradient learning)

	OGL	ANGL
Learning rate (η)	$\eta = 0.1$, $\alpha = 0.1$	$\eta = 0.005$, $\epsilon_t = 1/t$
Number of hidden nodes	10	10
Rate of success	10/10	10/10
Learning cycle for MSE $<2 \times 10^{-5}$	836,480	502.2
MSE for test data	7.6265×10^{-5}	2.4716×10^{-5}
Processing time (relative to OGL)	1.0	0.064

from the equation

$$x(t+1) = (1-b)x(t) + a\frac{x(t-\tau)}{1 + x(t-\tau)^{10}}, \tag{51}$$

where $a = 0.2$, $b = 0.1$, $\tau = 17$. The input values of the network are given from four previous time series data $x(t)$, $x(t-6)$, $x(t-12)$, $x(t-18)$, and the output value of the network is given from one future time series datum $x(t+6)$. We used 500 data which were generated at $t = 200, \ldots, 700$, for training, and another 500 data at $t = 5000, \ldots, 5500$ were used for test. The time series at $t = 200, \ldots, 700$ is shown in Fig. 1.

We stopped the learning process when the MSE for the training data became to be smaller than 2×10^{-5}. The average results over the 10 independent runs are shown in Table 1. For this problem, the adaptive natural gradient showed more than 1600 times faster convergence than the ordinary gradient learning in the sense of the learning cycle. In the sense of processing time, the proposed learning algorithm was more than 15 times faster than the ordinary one. In addition, the adaptive natural gradient learning method gave a smaller prediction error for the test data. Fig. 2 shows the learning curves of the two algorithms, from which we can see that the plateaus which appear in the ordinary gradient learning mostly do not exist in the adaptive natural gradient learning.

3.2. Classification problem

3.2.1. Case of two classes

Using the stochastic model of Eq. (36), we conducted experiments on an extended exclusive OR problem which is a benchmark problem for pattern classifiers. The pattern set to be classified is shown in Fig. 3. It consists of nine clusters each of which is assigned to one of the two classes marked by the different symbols ○ and +. Each cluster was generated subject to Gaussian distribution specified by a mean and a common covariance matrix. The pattern set used for training has 1800 elements consisting of the nine clusters each of which contains 200 elements generated from respective distributions. The pattern set used for test has 900 elements that were similarly generated.

Since there are some overlapped parts between two different clusters (Fig. 3), it is hard to expect a very small error. Considering the rate of the overlapped part over the whole pattern set, we regarded a learning process as a success when the MSE became smaller than 0.03.

The average results over 10 independent runs are given in Table 2. For this problem, it is shown that the adaptive natural gradient learning method can give more than 250 times faster convergence than the ordinary gradient learning method in the sense of the learning cycle. In the sense of processing time, the proposed learning is more than 10 times faster than the ordinary one. It is also confirmed that this fast convergence speed can be achieved by avoiding plateaus in which the ordinary gradient method was trapped (see Fig. 4). The middle curve in Fig. 4 is the result of the adaptive natural gradient method for the loss function of squared errors which corresponds to the stochastic model of additive Gaussian random noise for regression. This model was used

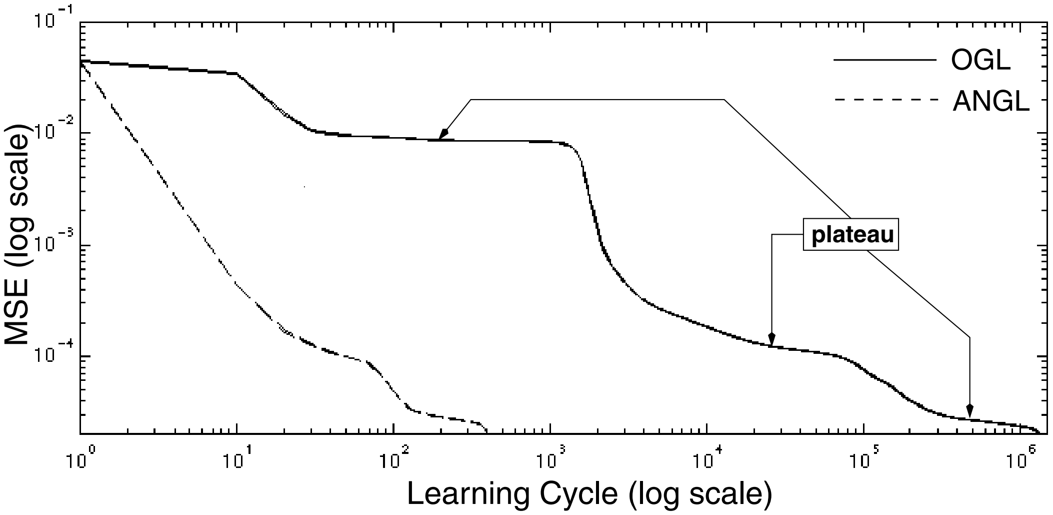

Fig. 2. Learning curves for the Mackey–Glass problem (OGL: the ordinary gradient learning, ANGL: the adaptive natural gradient learning).

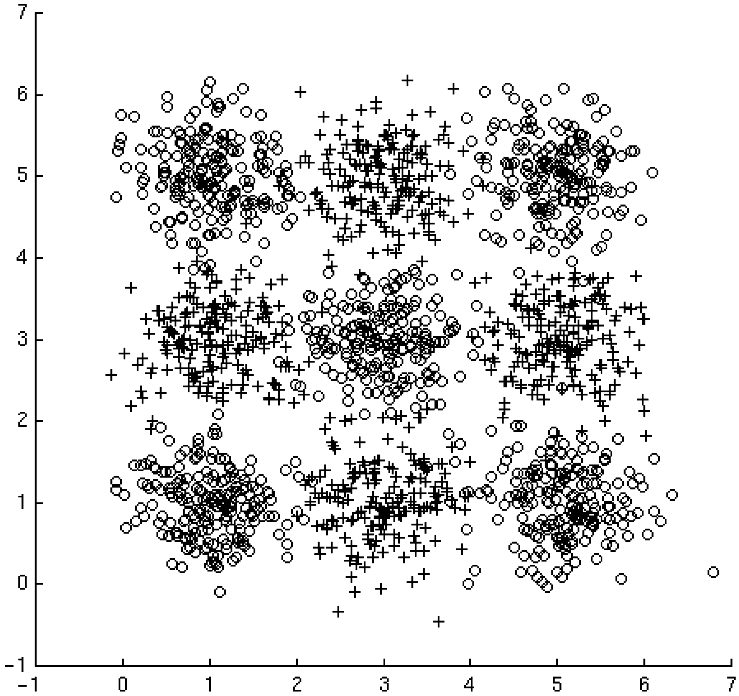

Fig. 3. Extended XOR problem.

in Amari et al. (2000). The present paper used a more appropriate stochastic model for the same problem.

3.2.2. Case of multiple classes

We used the IRIS flower classification problem in order to check the performance of the adaptive natural gradient method derived from the stochastic model (Eq. (42)) for problems involving multiple classes. The IRIS problem, which is a well-known benchmark problem, is to classify three different species of the iris flowers based on the four attributes on the shape of the plant. We therefore need a network structure with four input nodes and three output nodes. From the benchmark data set which consists of 150 data (50 data for each class), we randomly selected 30 data for each class for training, and other 60 data were used for test. We stopped a learning process when MSE became smaller than 10^{-4}.

The average results are compared in Table 3. In the sense of the learning cycle, the adaptive natural gradient method is more than 700 times faster than the ordinary gradient method as well as that the adaptive natural gradient method can give higher classification rates for the test data than the ordinary one. In the sense of the processing time, the proposed method is more than 10 times faster than the ordinary one. Showing the learning curves of the two algorithms in Fig. 5, we confirm that the adaptive natural gradient can avoid or alleviate plateaus in which the ordinary gradient learning is trapped.

Table 2
Results on the extended XOR classification problem (OGL: the ordinary gradient learning, ANGL: the adaptive natural gradient learning)

	OGL	ANGL
Learning rate	$\eta = 0.005$, $\alpha = 0$	$\eta = 0.00002$, $\epsilon_t = 1/t$
Number of hidden nodes	8	8
Rate of success	9/10	10/10
Learning cycle for MSE <0.03	182,440	686.2
Classification rate (Training)	95.39%	96.12%
Classification rate (Test)	94.77%	94.71%
Processing time (relative to OGL)	1.0	0.086

4. Conclusions and discussion

In this paper, we gave the general description of the

Table 3
Results on the IRIS pattern classification problem (OGL: the ordinary gradient learning, ANGL: the adaptive natural gradient learning)

	OGL	ANGL
Learning rate	$\eta = 0.02$, $\alpha = 0$	$\eta = 0.005$, $\epsilon_t = 1/t$
Number of hidden nodes	4	4
Rate of success	10/10	10/10
Learning cycle for MSE $<10^{-4}$	83,586.0	107.8
Classification rate (Training)	100%	100%
Classification rate (Test)	94.38%	94.99%
Processing time (relative to OGL)	1.0	0.097

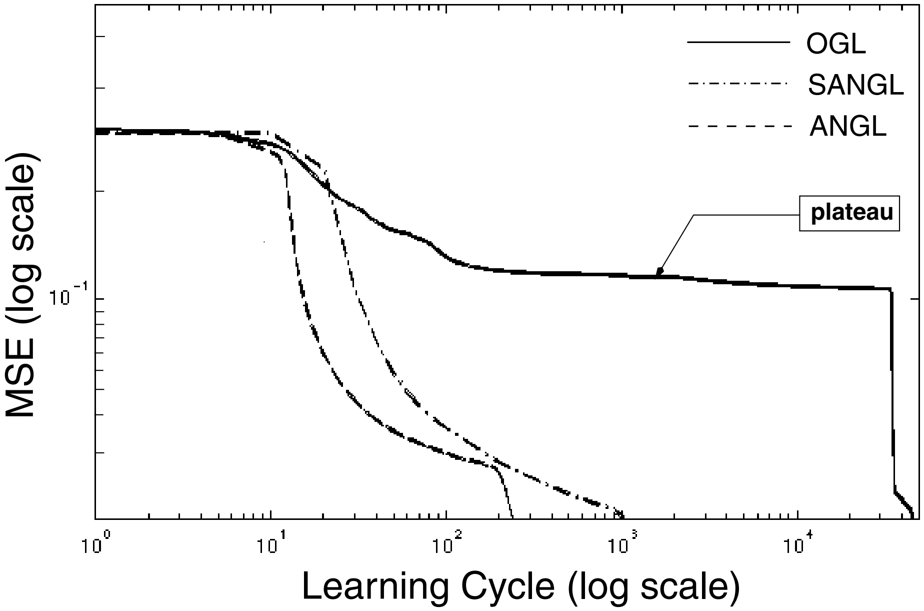

Fig. 4. Learning curves for the extended XOR problem (OGL: the ordinary gradient learning, SANGL: the adaptive natural gradient learning with squared error, ANGL: the adaptive natural gradient learning with cross entropy error).

adaptive natural gradient methods, and developed the explicit algorithms of adaptive natural gradient learning for the representative stochastic models used in practical applications. Using the proposed learning algorithm, we conducted computational experiments on well-known benchmark problems and confirmed that it can be applied to various types of application problem successfully. In the sense of convergence steps, the adaptive natural gradient learning gave remarkable improvement over the ordinary gradient learning, and in the sense of the processing time, the proposed method showed superiority to the ordinary gradient method, too. Through the learning curves given from the experiments, we also confirmed that the adaptive natural gradient learning can avoid or alleviate plateaus.

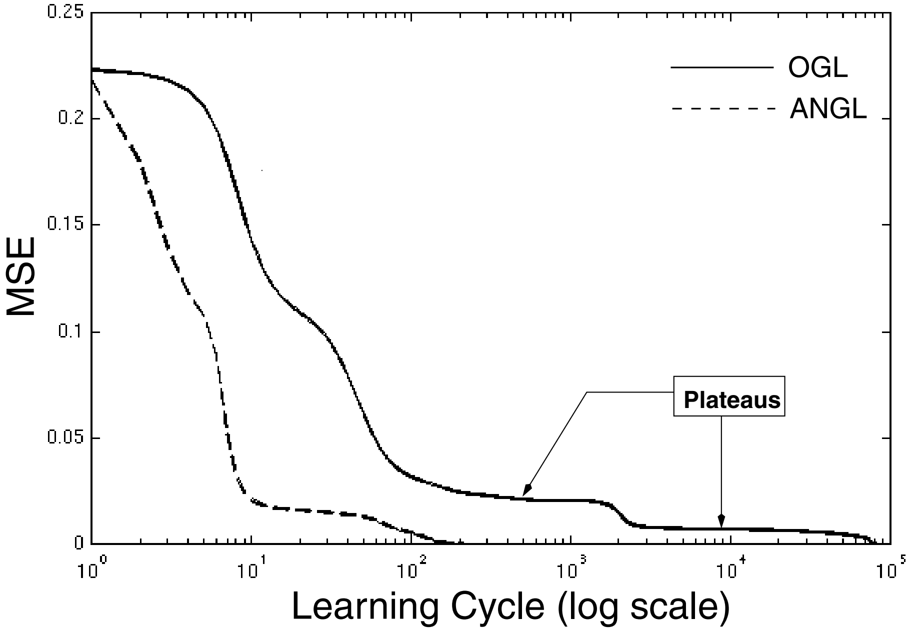

Fig. 5. Learning curves for the IRIS problem (OGL: the ordinary gradient learning, ANGL: the adaptive natural gradient learning).

References

Amari, S. (1985). *Differential-geometrical method in statistics*, *Springer Lecture Note in Statistics*, vol. 28. Berlin: Springer.

Amari, S. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, *5*, 185–196.

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*, 251–276.

Amari, S., & Nagaoka, H. (2000). *Information geometry*, New York: AMS and Oxford University Press.

Amari, S., Park, H., & Fukumizu, F. (2000). Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, *12*, 1399–1409.

Ampazis, N., Perantonis, S. J., & Taylor, J. G. (1999). Dynamics of multilayer networks in the vicinity of temporary minima. *Neural Networks*, *12*, 43–58.

Bishop, C. (1995). *Neural networks for pattern recognition*, New York: Oxford University Press.

Joost, M. (1996). Speeding up backpropagation algorithms by using cross-entropy combined with pattern normalization. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*, 117–126.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. -R. (1998). In G. B. Orr & K. R. Müller, *Neural networks: tricks of the trade*. *Springer Lecture Notes in Computer Sciences*, vol. 1524. Heidelberg: Springer.

Rattray, M., & Saad, D. (1998). Incorporating curvature information into on-line learning. In D. Saad, *On-line learning in neural networks* (pp. 183–207). New York: Cambridge University Press.

Rattray, M., & Saad, D. (2000). Analysis of natural gradient descent for multilayer neural networks. *Physical Review E*, (in press).

Rattray, M., Saad, D., & Amari, S. (1998). Natural gradient descent for on-line learning. *Physical Review Letters*, *81*, 5461–5464.

Saad, D., & Solla, S. A. (1995). On-line learning in soft committee machines. *Physical Review E*, *52*, 4225–4243.

Yang, H. H., & Amari, S. (1998). Complexity issues in natural gradient descent method for training multilayer perceptrons. *Neural Computation*, *10*, 2137–2157.