Natural Gradient Descent for Training Multi-Layer Perceptrons

Article · January 1998

Source: CiteSeer

CITATIONS
18

READS
304

3 authors, including:

Howard H Yang
National Institutes of Health
240 PUBLICATIONS **6,924** CITATIONS
SEE PROFILE

Shun-ichi Amari
RIKEN
564 PUBLICATIONS **34,909** CITATIONS
SEE PROFILE

Natural Gradient Descent for Training Multi-Layer Perceptrons

Howard Hua Yang and Shun-ichi Amari
Lab. for Information Representation, FRP, RIKEN
Hirosawa 2-1, Wako-shi, Saitama 351-01, JAPAN
FAX: +81 48462 4633
E-mails: hhy@koala.riken.go.jp, amari@zoo.riken.go.jp

Submitted to IEEE Tr. on Neural Networks
May 16, 1996

Abstract

The main difficulty in implementing the natural gradient learning rule is to compute the inverse of the Fisher information matrix when the input dimension is large. We have found a new scheme to represent the Fisher information matrix. Based on this scheme, we have designed an algorithm to compute the inverse of the Fisher information matrix. When the input dimension n is much larger than the number of hidden neurons, the complexity of this algorithm is of order $O(n^2)$ while the complexity of conventional algorithms for the same purpose is of order $O(n^3)$. The simulation has confirmed the efficience and robustness of the natural gradient learning rule.

1 Introduction

Inversion of the Fisher information matrix is required to obtain the Cramer-Rao lower bound which is fundamental for analyzing the performance of an unbiased estimator. It is also needed in the natural gradient learning framework [2, 3] to design statistically efficient learning algorithms for parameter estimation in general and for training neural networks in particular. In this paper, we assume a stochastic model for multi-layer perceptrons. Considering the parameter space as a Riemannian space with the Fisher information matrix as its metric, we apply the natural gradient learning rule to train a multi-layer perceptron. The main difficulty encountered is to compute the inverse of the Fisher information matrix of a large size when the input dimension is large. By exploring the structure of the Fisher information matrix and its inverse, we shall design a fast algorithm with lower complexity to implement the natural gradient learning algorithm.

The outline of the paper is as follows. The natural gradient descent method and related problems for training a multi-layer perceptron are described in Section 2. The Fisher information matrix and the scheme to represent this matrix are given in Section 3. The main focus of this paper is to present a fast algorithm to compute the inverse of the Fisher information matrix. Several issues related to this are discussed in Section 4: a group structure for the blocks in the Fisher information matrix is found in Section 4.1. Based on this finding the inverse of the Fisher information matrix is computed in Section 4.2-4.3; the complexity of our algorithm in computing the inverse of the Fisher information matrix is analyzed in Section 4.4; by the

result on the group structure of the Fisher information matrix, the Cramer-Rao lower bound for the committee machine is explicitly given in Section 4.5. The natural gradient vector field is compared with the ordinary vector field in Section 5 via an example to illustrate the effects of the natural gradient learning in training multi-layer perceptrons. The effectiveness of the natural gradient is further demonstrated by some simulation results in Section 6.

2 Natural Gradient Descent Method and Its Efficiency

2.1 Model of a stochastic multi-layer perceptron

Assume the following model of a stochastic multi-layer perceptron:

$$z = \sum_{i=1}^{m} a_i \varphi(\boldsymbol{w}_i^T \boldsymbol{x} + b_i) + \xi \tag{1}$$

where $(\cdot)^T$ denotes the transpose, $\xi \sim N(0, \sigma^2)$ is a Gaussian random variable, and $\varphi(x)$ is a differentiable output function of hidden neurons such as hyperbolic tangent. Assume that the multi-layer network has a n-dimensional input, m hidden neurons, a one dimensional output, and $m \leq n$. Denote $\boldsymbol{a} = (a_1, \cdots, a_m)^T$ the weight vector of the output neuron, $\boldsymbol{w}_i = (w_{1i}, \cdots, w_{ni})^T$ the weight vector of the i-th hidden neuron, and $\boldsymbol{b} = (b_1, \cdots, b_m)^T$ is the vector of thresholds for the hidden neurons. Let $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m]$ be a matrix formed by column weight vectors $\boldsymbol{w}_i$, then (1) can be rewritten as

$$z = \boldsymbol{a}^T \varphi(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}) + \xi.$$

Here, the scalar function φ operates on each component of the vector $\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}$ and this convention will be used throughout this paper. By this convention, the column vector

$$[\varphi(\boldsymbol{w}_1^T \boldsymbol{x} + b_1), \cdots, \varphi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)]^T$$

is written as $\varphi(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})$ and the row vector

$$[\varphi(\boldsymbol{w}_1^T \boldsymbol{x} + b_1), \cdots, \varphi(\boldsymbol{w}_m^T \boldsymbol{x} + b_m)]$$

is written as $\varphi(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b}^T)$.

In this paper, we consider the realizable learning problem. Let

$$\boldsymbol{\theta} = (\boldsymbol{w}_1^T, \cdots, \boldsymbol{w}_m^T, \boldsymbol{a}^T, \boldsymbol{b}^T)^T$$

denote the parameter vector for the student network and

$$\boldsymbol{\theta}^* = (\boldsymbol{w}_1^{*T}, \cdots, \boldsymbol{w}_m^{*T}, \boldsymbol{a}^{*T}, \boldsymbol{b}^{*T})^T$$

for the teacher network.

The problem is to estimate $\boldsymbol{\theta}^*$ based on training examples

$$D_T = \{(\boldsymbol{x}_t, z_t), t = 1, \cdots, T\}.$$

The training examples as well as the future examples for evaluating the generalization error are generated by two steps. First, $\boldsymbol{x}_t$ is randomly generated subject to an unknown probability

density function $p(\boldsymbol{x})$. Second, the corresponding output is generated by the teacher network as

$$z_t = \boldsymbol{a}^{*T}\varphi(\boldsymbol{W}^{*T}\boldsymbol{x}_t + \boldsymbol{b}^*) + \xi_t$$

where $\xi_t \sim N(0, \sigma^2)$. Note the variance of the additive noise ξ_t is the same as the variance of the additive noise ξ in the stochastic perceptron model.

The input-output pairs $\{(\boldsymbol{x}_t, z_t),\ t > T\}$ are the future examples. Define the generalization error

$$E_g(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = E_{\boldsymbol{\theta}^*}[(z - \boldsymbol{a}^T\varphi(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}))^2]$$

where the expectation $E_{\boldsymbol{\theta}^*}[\cdot]$ is taken with respect to the joint probability density function $p(\boldsymbol{x}, z; \boldsymbol{\theta}^*)$.

The purpose of learning is to minimize the generalization error but it cannot be easily evaluated in practice. Instead of the generalization error, we use the training error

$$E_{\rm tr}(\boldsymbol{\theta}, D_T) = \frac{1}{T}\sum_{t=1}^{T}\{z_t - \boldsymbol{a}^T\varphi(\boldsymbol{W}^T\boldsymbol{x}_t + \boldsymbol{b})\}^2$$

to derive a learning algorithm. Minimizing the training error is not equivalent to minimizing the generalization error in general (see, for example, Amari and Murata, 1993)[4]. Some auxiliary methods such as early stopping and regularization are necessary to avoid overtraining or overfitting (see, for example, Ripley, 1996 [10]; Amari et al, 1997 [5]). The overtraining problem is important but it will not be discussed here. We shall only focus the learning algorithm for minimizing the training error.

2.2 Fisher information matrix

The joint probability density function (pdf) of the input and the output is

$$p(\boldsymbol{x}, z; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}) = p(z|\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})p(\boldsymbol{x})$$

where $p(z|\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})$ is the conditional pdf of z when $\boldsymbol{x}$ is input. The loss function is defined as the negative log-likelihood function

$$L(\boldsymbol{x}, z; \boldsymbol{\theta}) = -\log p(\boldsymbol{x}, z; \boldsymbol{\theta}) = l(z|\boldsymbol{x}; \boldsymbol{\theta}) - \log p(\boldsymbol{x})$$

where

$$l(z|\boldsymbol{x}; \boldsymbol{\theta}) = -\log p(z|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{2\sigma^2}(z - \boldsymbol{a}^T\varphi(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}))^2$$

is also a loss function equivalent to $L(\boldsymbol{x}, z; \boldsymbol{\theta})$ since $p(\boldsymbol{x})$ does not depend on $\boldsymbol{\theta}$. Hence, given the training set D_T, minimizing the loss function $L(\boldsymbol{x}, z; \boldsymbol{\theta})$ is equivalent to minimizing the training error.

Since $\frac{\partial L}{\partial \boldsymbol{\theta}} = \frac{\partial l}{\partial \boldsymbol{\theta}}$, the Fisher information matrix is defined by

$$\boldsymbol{G} = \boldsymbol{G}(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}[\frac{\partial L}{\partial \boldsymbol{\theta}}(\frac{\partial L}{\partial \boldsymbol{\theta}})^T] = E_{\boldsymbol{\theta}}[\frac{\partial l}{\partial \boldsymbol{\theta}}(\frac{\partial l}{\partial \boldsymbol{\theta}})^T] \tag{2}$$

where $E_{\boldsymbol{\theta}}[\cdot]$ denotes the expectation with respect to $p(\boldsymbol{x}, z; \boldsymbol{\theta})$.

The inverse of the Fisher information matrix is often used in performance analysis, because the Cramer-Rao states that an unbiased estimator $\widehat{\boldsymbol{\theta}}$ of the true parameter $\boldsymbol{\theta}^*$ satisfies

$$E_{\boldsymbol{\theta}^*}[\|\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*\|^2] \geq {\rm Tr}(\boldsymbol{G}^{-1}(\boldsymbol{\theta}^*))$$

where $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix.

For the on-line estimator $\widehat{\boldsymbol{\theta}}_t$ based on the independent examples $\{(\boldsymbol{x}_s, z_s), s = 1, \cdots, t\}$, the negative log-likelihood function is

$$L_{1,t} = L(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, z_1, \cdots, z_t; \boldsymbol{\theta}) = -\log p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, z_1, \cdots, z_t; \boldsymbol{\theta}) = \sum_{s=1}^{t} L(\boldsymbol{x}_s, z_s; \boldsymbol{\theta}).$$

Since

$$E_{\boldsymbol{\theta}}[\frac{\partial L}{\partial \boldsymbol{\theta}}(\boldsymbol{x}_s, z_s; \boldsymbol{\theta})(\frac{\partial L}{\partial \boldsymbol{\theta}}(\boldsymbol{x}_\tau, z_\tau; \boldsymbol{\theta}))^T] = \delta_{s,\tau} \boldsymbol{G}(\boldsymbol{\theta})$$

due to the independence of the examples,

$$G_t(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}[\frac{\partial L_{1,t}}{\partial \boldsymbol{\theta}}(\frac{\partial L_{1,t}}{\partial \boldsymbol{\theta}})^T] = tG(\boldsymbol{\theta}).$$

So when the examples $\{(\boldsymbol{x}_s, z_s), s = 1, 2 \cdots, \}$ are independently drawn according to the probability law $p(\boldsymbol{x}, z; \boldsymbol{\theta}^*)$, the Cramer-Rao inequality for the on-line estimator is

$$E_{\boldsymbol{\theta}^*}[\|\widehat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*\|^2] \geq \frac{1}{t}\mathrm{Tr}(\boldsymbol{G}^{-1}(\boldsymbol{\theta}^*)) \tag{3}$$

2.3 Natural gradient learning

Let $\boldsymbol{\Theta} = \{\boldsymbol{\theta}\}$ be the parameter space. The divergence between two points $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, or two distributions, $p(\boldsymbol{x}, z; \boldsymbol{\theta}_1)$ and $p(\boldsymbol{x}, z; \boldsymbol{\theta}_2)$, is given by the Kullback-Leibler divergence

$$D(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \mathrm{KL}[p(\boldsymbol{x}, z; \boldsymbol{\theta}_1) \| p(\boldsymbol{x}, z; \boldsymbol{\theta}_2)].$$

When the two points are infinitesimally close, we have the quadratic form

$$D(\boldsymbol{\theta}, \boldsymbol{\theta} + d\boldsymbol{\theta}) = \frac{1}{2} d\boldsymbol{\theta}^T \boldsymbol{G}(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{4}$$

where $\boldsymbol{G}(\boldsymbol{\theta})$ is the Fisher information matrix. This is regarded as the square of the length of $d\boldsymbol{\theta}$. Since $\boldsymbol{G}(\boldsymbol{\theta})$ depends on $\boldsymbol{\theta}$, the parameter space is regarded as a Riemannian space in which the local distance is defined by (4). Here, the Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta})$ plays the role of the Riemannian metric tensor. See (Amari, 1985)[1] and (Murray and Rice, 1993)[9] for information geometry.

The natural gradient descent method proposed by Amari [3, 2] makes use of the Riemannian metric given by the Fisher information matrix to optimize the learning dynamics such that the Cramer-Rao lower bound is achieved asymptotically. The idea is to convert the covariant gradient $\frac{\partial l}{\partial \boldsymbol{\theta}}$ into contravariant form $\boldsymbol{G}^{-1}\frac{\partial l}{\partial \boldsymbol{\theta}}$. It is shown by Amari [2] that the steepest descent direction of a function $C(\boldsymbol{\theta})$ in the Riemannian space $\boldsymbol{\Theta}$ is

$$-\widetilde{\nabla} C(\boldsymbol{\theta}) = -\boldsymbol{G}^{-1}(\boldsymbol{\theta}) \nabla C(\boldsymbol{\theta}).$$

The on-line learning algorithms corresponding to $\frac{\partial l}{\partial \boldsymbol{\theta}}$ and $\boldsymbol{G}^{-1}(\boldsymbol{\theta})\frac{\partial l}{\partial \boldsymbol{\theta}}$ are, respectively, the ordinary gradient descent algorithm:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\mu}{t}\frac{\partial l}{\partial \boldsymbol{\theta}}(z_t | \boldsymbol{x}_t; \boldsymbol{\theta}_t) \tag{5}$$

and the natural gradient descent algorithm:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\mu'}{t}\boldsymbol{G}^{-1}(\boldsymbol{\theta}_t)\frac{\partial l}{\partial \boldsymbol{\theta}}(z_t | \boldsymbol{x}_t; \boldsymbol{\theta}_t) \tag{6}$$

where μ and μ' are learning rates.

It is proved in [2] that the natural gradient learning is Fisher efficient. When the negative log-likelihood function is taken as the loss function, the natural gradient descent algorithm (6) gives an efficient on-line estimator, i.e., the asymptotic variance of $\boldsymbol{\theta}_t$ driven by (6) satisfies

$$E_{\boldsymbol{\theta}^*}[(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*)(\boldsymbol{\theta}_t - \boldsymbol{\theta}^*)^T] \approx \frac{1}{t}\boldsymbol{G}^{-1}(\boldsymbol{\theta}^*) \tag{7}$$

which gives the mean square error

$$E_{\boldsymbol{\theta}^*}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|^2] \approx \frac{1}{t}\mathrm{Tr}(\boldsymbol{G}^{-1}(\boldsymbol{\theta}^*)). \tag{8}$$

However, the algorithm (5) does not attain this limit.

Remark 1 *The equilibrium points are the same for the averaged versions of the dynamics (5) and (6), but the inverse of the Fisher information matrix optimizes the learning dynamics. The local minima problem still exists in the natural gradient learning.*

Remark 2 *The natural gradient learning algorithm is efficient, but its complexity is generally high due to the computation of the matrix inverse. In later sections, we shall explore the structure of the Fisher information matrix for a multi-layer perceptron and design a low complexity algorithm to compute $\boldsymbol{G}^{-1}(\boldsymbol{\theta})$. The dimension of $\boldsymbol{G}(\boldsymbol{\theta})$ is $(nm + 2m) \times (nm + 2m)$, the time complexity of computing $\boldsymbol{G}^{-1}(\boldsymbol{\theta})$ by generally accepted algorithms is $O(n^3)$ which is too high for the on-line natural gradient learning algorithm. Our method reduces the time complexity to $O(n^2)$.*

3 Fisher Information Matrix of A Multi-Layer Perceptron

3.1 Likelihood function of the stochastic multi-layer perceptron

In previous sections, we did not give any specific assumption about the input. To compute an explicit expression of the Fisher information matrix, we now assume that the input $\boldsymbol{x}$ is subject to the standard Gaussian distribution $N(\boldsymbol{0}, \boldsymbol{I})$, where $\boldsymbol{I}$ is the identity matrix.

From $l(z|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{2\sigma^2}(z - \sum_{i=1}^m a_i\varphi(\boldsymbol{w}_i^T\boldsymbol{x} + b_i))^2$, we obtain

$$\frac{\partial l}{\partial w_{ij}} = -\frac{1}{\sigma^2}(z - \sum_{k=1}^m a_k\varphi(\boldsymbol{w}_k^T\boldsymbol{x} + b_k))a_i\varphi'(\boldsymbol{w}_i^T\boldsymbol{x} + b_i)x_j \tag{9}$$

$$\frac{\partial l}{\partial a_i} = -\frac{1}{\sigma^2}(z - \sum_{k=1}^m a_k\varphi(\boldsymbol{w}_k^T\boldsymbol{x} + b_k))\varphi(\boldsymbol{w}_i^T\boldsymbol{x} + b_i) \tag{10}$$

$$\frac{\partial l}{\partial b_i} = -\frac{1}{\sigma^2}(z - \sum_{k=1}^m a_k\varphi(\boldsymbol{w}_k^T\boldsymbol{x} + b_k))a_i\varphi'(\boldsymbol{w}_i^T\boldsymbol{x} + b_i) \tag{11}$$

When $(\boldsymbol{x}, z)$ is the input-output pair of the stochastic multi-layer perceptron, the equations (9)-(11) become

$$\frac{\partial l}{\partial w_{ij}} = -\frac{\xi}{\sigma^2}a_i\varphi'(\boldsymbol{w}_i^T\boldsymbol{x} + b_i)x_j \tag{12}$$

$$\frac{\partial l}{\partial a_i} = -\frac{\xi}{\sigma^2}\varphi(\boldsymbol{w}_i^T\boldsymbol{x} + b_i) \tag{13}$$

$$\frac{\partial l}{\partial b_i} = -\frac{\xi}{\sigma^2}a_i\varphi'(\boldsymbol{w}_i^T\boldsymbol{x} + b_i), \tag{14}$$

and in vector forms

$$\frac{\partial l}{\partial \boldsymbol{w}_i} = -\frac{\xi}{\sigma^2} a_i \varphi'(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)\boldsymbol{x}, i = 1, \cdots, m \tag{15}$$

$$\frac{\partial l}{\partial \boldsymbol{a}} = -\frac{\xi}{\sigma^2} \varphi(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}) \tag{16}$$

$$\frac{\partial l}{\partial \boldsymbol{b}} = -\frac{\xi}{\sigma^2} \{\boldsymbol{a} \odot \varphi'(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})\} \tag{17}$$

where $\odot$ denotes the Hadmard product which is the componentwise product of vectors or matrices. The above three vector forms can be written as one:

$$\frac{\partial l}{\partial \boldsymbol{\theta}} = -\frac{\xi}{\sigma^2} \begin{bmatrix} (\boldsymbol{a} \odot \varphi'(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})) \otimes \boldsymbol{x} \\ \varphi(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}) \\ \boldsymbol{a} \odot \varphi'(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}) \end{bmatrix} \tag{18}$$

where $\otimes$ denotes the Kronecker product. For two matrices $\boldsymbol{A} = (a_{ij})$ and $\boldsymbol{B} = (b_{ij})$, $\boldsymbol{A} \otimes \boldsymbol{B}$ is defined as a larger matrix consisting of the block matrices $[a_{ij}\boldsymbol{B}]$. For two column vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, $\boldsymbol{a} \otimes \boldsymbol{b}$ gives the higher-dimensional column vector $[a_1 \boldsymbol{b}^T, \cdots, a_n \boldsymbol{b}^T]^T$.

3.2 Blocks in the Fisher information matrix

The parameter $\boldsymbol{\theta}$ consists of $m + 2$ blocks

$$\boldsymbol{\theta} = [\boldsymbol{w}_1^T, \cdots, \boldsymbol{w}_m^T, \boldsymbol{a}^T, \boldsymbol{b}^T]^T$$

where the first m blocks $\boldsymbol{w}_i$'s are n-dimensional and the last two blocks $\boldsymbol{a}$ and $\boldsymbol{b}$ are m-dimensional. From (18), the Fisher information matrix $\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{\sigma^2}\boldsymbol{A}(\boldsymbol{\theta})$ is of $(n+2)m \times (n+2)m$ dimensions. Here, the matrix $\boldsymbol{A}(\boldsymbol{\theta})$ does not depend on σ^2. It is partitioned into $(m+2)\times(m+2)$ blocks of sub-matrices corresponding to the partition of $\boldsymbol{\theta}$. We denote each sub-matrix by $\boldsymbol{A}_{ij}$. The Fisher information matrix is written as

$$\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{\sigma^2}[\boldsymbol{A}_{ij}]_{(m+2)\times(m+2)}. \tag{19}$$

The partition diagram for the matrix $\boldsymbol{A}(\boldsymbol{\theta})$ is given below:

$$\boldsymbol{A}(\boldsymbol{\theta}) = \left\{ \begin{array}{c|cccccc} & \boldsymbol{w}_1^T & \cdots & \boldsymbol{w}_m^T & \boldsymbol{a}^T & \boldsymbol{b}^T \\ \hline \boldsymbol{w}_1 & \boldsymbol{A}_{1,1} & \cdots & \boldsymbol{A}_{1,m} & \boldsymbol{A}_{1,m+1} & \boldsymbol{A}_{1,m+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{w}_m & \boldsymbol{A}_{m,1} & \cdots & \boldsymbol{A}_{m,m} & \boldsymbol{A}_{m,m+1} & \boldsymbol{A}_{m,m+2} \\ \boldsymbol{a} & \boldsymbol{A}_{m+1,1} & \cdots & \boldsymbol{A}_{m+1,m} & \boldsymbol{A}_{m+1,m+1} & \boldsymbol{A}_{m+1,m+2} \\ \boldsymbol{b} & \boldsymbol{A}_{m+2,1} & \cdots & \boldsymbol{A}_{m+2,m} & \boldsymbol{A}_{m+2,m+1} & \boldsymbol{A}_{m+2,m+2} \end{array} \right. \tag{20}$$

where

$\boldsymbol{A}_{ij} = a_i a_j E[\varphi'(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)\varphi'(\boldsymbol{w}_j^T \boldsymbol{x} + b_j)\boldsymbol{x}\boldsymbol{x}^T]$, for $1 \le i, j \le m$,
$\boldsymbol{A}_{i,m+1} = a_i E[\varphi'(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)\boldsymbol{x}\varphi(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b})]$ and
$\boldsymbol{A}_{m+1,i} = \boldsymbol{A}_{i,m+1}^T$ for $1 \le i \le m$,
$\boldsymbol{A}_{m+1,m+1} = E[\varphi(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})\varphi(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b})]$,
$\boldsymbol{A}_{i,m+2} = E[a_i \varphi'(\boldsymbol{w}_i^T \boldsymbol{x} + b_i)\boldsymbol{x}(\boldsymbol{a}^T \odot \varphi'(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b}^T))]$ and
$\boldsymbol{A}_{m+2,i} = \boldsymbol{A}_{i,m+2}^T$ for $1 \le i \le m$,
$\boldsymbol{A}_{m+1,m+2} = \boldsymbol{A}_{m+2,m+1}^T = E[\varphi(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})(\boldsymbol{a}^T \odot \varphi'(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b}^T))]$,
$\boldsymbol{A}_{m+2,m+2} = (\boldsymbol{a}\boldsymbol{a}^T) \odot E[\varphi'(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b})\varphi'(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b}^T)]$.

Here, $E[\cdot]$ denotes the expectation with respect to $p(\boldsymbol{x})$. From the above expressions, we know that $\boldsymbol{A}(\boldsymbol{\theta}) = [\boldsymbol{A}_{ij}]$ does not depend on σ^2. The explicit forms of these matrix blocks in the Fisher information matrix are characterized by the three lemmas in the next section.

Remark 3 *Since the loss function $l(z|\boldsymbol{x};\boldsymbol{\theta})$ contains the unknown parameter σ^2, we should use the following modified loss function in practice:*

$$l_1(z|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{2}(z - \boldsymbol{a}^T\varphi(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}))^2.$$

Modifying the learning rule (5), we have the following on-line algorithm:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\mu}{t}\frac{\partial l_1}{\partial \boldsymbol{\theta}}(z_t|\boldsymbol{x}_t;\boldsymbol{\theta}_t). \tag{21}$$

Since $\boldsymbol{G}^{-1}(\boldsymbol{\theta})\frac{\partial l}{\partial \boldsymbol{\theta}} = \boldsymbol{A}^{-1}(\boldsymbol{\theta})\frac{\partial l_1}{\partial \boldsymbol{\theta}}$, the learning rule (6) is exactly the same as the following learning rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\mu'}{t}\boldsymbol{A}^{-1}(\boldsymbol{\theta}_t)\frac{\partial l_1}{\partial \boldsymbol{\theta}}(z_t|\boldsymbol{x}_t;\boldsymbol{\theta}_t). \tag{22}$$

3.3 Representation of blocks in the Fisher information matrix

Lemma 1 *For $i = 1, \cdots, m$, the diagonal blocks are given by*

$$\boldsymbol{A}_{ii} = a_i^2[d_1(w_i, b_i)\boldsymbol{I} + \{d_2(w_i, b_i) - d_1(w_i, b_i)\}\boldsymbol{u}_i\boldsymbol{u}_i^T] \tag{23}$$

where

$$w_i = \|\boldsymbol{w}_i\| \text{ denoting the Euclidean norm of } \boldsymbol{w}_i\ , \boldsymbol{u}_i = \boldsymbol{w}_i/w_i,$$

$$d_1(w, b) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}(\varphi'(wx + b))^2 e^{-\frac{x^2}{2}}dx > 0, \tag{24}$$

$$d_2(w, b) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}(\varphi'(wx + b))^2 x^2 e^{-\frac{x^2}{2}}dx > 0. \tag{25}$$

When $a_i \neq 0$, the inverse of the matrix $\boldsymbol{A}_{ii}$ is given by

$$\boldsymbol{A}_{ii}^{-1} = \frac{1}{a_i^2}[\frac{1}{d_1(w_i, b_i)}\boldsymbol{I} + \{\frac{1}{d_2(w_i, b_i)} - \frac{1}{d_1(w_i, b_i)}\}\boldsymbol{u}_i\boldsymbol{u}_i^T], \quad i = 1, \cdots, m. \tag{26}$$

The proof of Lemma 1 is given in Appendix 1.

In particular, for a single-layer stochastic perceptron

$$z = \varphi(\boldsymbol{w}^T\boldsymbol{x} + b) + \xi,$$

the expressions (23) and (26) give the Fisher information matrix:

$$\boldsymbol{G}(\boldsymbol{w}) = \frac{1}{\sigma^2}[d_1(w, b)\boldsymbol{I} + \{d_2(w, b) - d_1(w, b)\}\boldsymbol{u}_i\boldsymbol{u}_i^T] \tag{27}$$

and its inverse formula

$$\boldsymbol{G}^{-1}(\boldsymbol{w}) = \sigma^2[\frac{1}{d_1(w, b)}\boldsymbol{I} + \{\frac{1}{d_2(w, b)} - \frac{1}{d_1(w, b)}\}\boldsymbol{u}_i\boldsymbol{u}_i^T]. \tag{28}$$

When $b = 0$, the above formula is found by Amari in [3].

To obtain the explicit forms of the other blocks $\boldsymbol{A}_{ij}$ in $\boldsymbol{G}$, we need to introduce two bases in $\Re^n$ which possess certain properties. Except for a set in $\Re^n$ with zero Lebesgue measure, the vectors $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m\}$ are linearly independent. It is easy to supplement $n-m$ vectors $\boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n$ to them, such that they together form a basis in $\Re^n$, $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m; \boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n\}$, having the following properties:

$$\text{for } j > m, \ \ \boldsymbol{u}_j - \mathcal{L}(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m) \tag{29}$$

$$\boldsymbol{u}_j^T \boldsymbol{u}_k = \delta_{j,k} \text{ (delta notation)}, \qquad \text{for } j,k = m+1, \cdots, n, \tag{30}$$

where $\mathcal{L}(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m)$ is the vector space spanned by $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m\}$.

Let $\boldsymbol{U} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_n]$ and $\boldsymbol{V} = (\boldsymbol{U}^{-1})^T = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_n]$. The identity

$$\boldsymbol{V}^T \boldsymbol{U} = \boldsymbol{U}^T \boldsymbol{V} = \boldsymbol{I}$$

implies

$$\boldsymbol{v}_j^T \boldsymbol{u}_k = \boldsymbol{u}_j^T \boldsymbol{v}_k = \delta_{j,k}. \tag{31}$$

Hence, $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$ is the orthogonal conjugate basis of $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$.

From (29), (30) and (31), we can easily prove that

$$\boldsymbol{u}_j = \boldsymbol{v}_j \qquad \text{for } j > m.$$

The random input $\boldsymbol{x}$ is represented in dual ways by

$$\boldsymbol{x} = \sum_{i=1}^{n} x_i \boldsymbol{u}_i = \sum_{i=1}^{n} x_i' \boldsymbol{v}_i$$

where $(x_1, \cdots, x_n)$ and $(x_1', \cdots, x_n')$ are coordinates of $\boldsymbol{x}$ on the bases $(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n)$ and $(\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n)$ respectively. It follows from (31) that

$$x_i = \boldsymbol{v}_i^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{v}_i \quad \text{and} \quad x_i' = \boldsymbol{u}_i^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{u}_i. \tag{32}$$

In this paper, a $m \times n$ matrix is denoted by $(a_{ij})_{m \times n}$. The subscript $m \times n$ may be omitted when the dimension of the matrix can be determined by the context. Define a $n \times n$ matrix $\boldsymbol{R} = (r_{ij}) = (\boldsymbol{u}_i^T \boldsymbol{u}_j)$. Let $\boldsymbol{R}^{-1} = (r^{ij})$. Noticing that $\boldsymbol{R} = (\boldsymbol{u}_i^T \boldsymbol{u}_j) = \boldsymbol{U}^T \boldsymbol{U}$ and $\boldsymbol{V} = (\boldsymbol{U}^{-1})^T$, we have $\boldsymbol{R}^{-1} = \boldsymbol{U}^{-1}(\boldsymbol{U}^{-1})^T = \boldsymbol{V}^T \boldsymbol{V}$, i.e., $r^{ij} = \boldsymbol{v}_i^T \boldsymbol{v}_j$. Due to the properties (29) and (30),

$$\boldsymbol{R} = \begin{bmatrix} (r_{ij})_{m \times m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}_{n \times n}, \quad \text{and} \quad \boldsymbol{R}^{-1} = \begin{bmatrix} (r^{ij})_{m \times m} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}_{n \times n}$$

where $\boldsymbol{0}$ and $\boldsymbol{I}$ are zero and identity matrices respectively with proper dimensions admissible in the above partitions of $\boldsymbol{R}$ and $\boldsymbol{R}^{-1}$. By the definition of $\boldsymbol{R}$, we have $x_j' = \boldsymbol{u}_j^T \boldsymbol{x} = \boldsymbol{u}_j^T \sum_{k=1}^n x_k \boldsymbol{u}_k = \sum_{k=1}^n r_{jk} x_k$ and $x_j = \sum_{k=1}^n r^{jk} x_k'$. Therefore, for $1 \le j \le m$, $x_j' = \sum_{k=1}^m r_{jk} x_k$ and $x_j = \sum_{k=1}^m r^{jk} x_k'$, and for $j = m+1, \cdots, n$, $x_j = x_j'$. With the above notations, we have an expression of the matrix $\boldsymbol{A}_{ij}$ for $1 \le i \ne j \le m$ in the following lemma.

Lemma 2 *For $1 \le i, j \le m$,*

$$\boldsymbol{A}_{ij} = a_i a_j (c_{ij} \Omega_0 + \sum_{l,k=1}^{m} c_{ij}^{lk} \boldsymbol{u}_l \boldsymbol{v}_k^T) \tag{33}$$

where

$$\Omega_0 = \sum_{k=m+1}^{n} \boldsymbol{u}_k \boldsymbol{u}_k^T = \boldsymbol{I} - \sum_{k=1}^{m} \boldsymbol{u}_k \boldsymbol{v}_k^T, \tag{34}$$

$$c_{ij} = E[\varphi'(w_i x_i' + b_i)\varphi'(w_j x_j' + b_j)], \tag{35}$$

$$c_{ij}^{lk} = E[\varphi'(w_i x_i' + b_i)\varphi'(w_j x_j' + b_j)(\sum_{s=1}^{m} r^{ls} x_s')x_k']. \tag{36}$$

The proof of Lemma 2 is given in Appendix 2.

Remark 4 *The vectors $\{\boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n\}$ are only used for theoretical analysis. They are not needed in the equation (33) because of the equation (34). To compute $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m\}$, we define $\boldsymbol{U}_1 = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m]$ and $\boldsymbol{V}_1 = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m]$. From $\boldsymbol{U}^T\boldsymbol{V} = \boldsymbol{I}$, we then have*

$$\boldsymbol{V}_1 = \boldsymbol{U}_1(\boldsymbol{U}_1^T\boldsymbol{U}_1)^{-1} \tag{37}$$

which is the generalized inverse of $\boldsymbol{U}_1$.

Remark 5 *When $i = j$, the equation (33) gives another representation for $\boldsymbol{A}_{ii}$ which is more complicated than the representation (23). But the new representation for $\boldsymbol{A}_{ii}$ is needed to derive some properties of the Fisher information matrix. To compute the inverse of $\boldsymbol{A}_{ii}$, we use the equation (26). To compute the inverse of the Fisher information matrix, we also need to compute the inverse of matrices which have the same structure as $\boldsymbol{A}_{ij}$ for $i \neq j$. This will be explained later in details when we give an algorithm to compute the inverse of the Fisher information matrix.*

It follows from (32) that both x_i and x_i' are Gaussian random variables with a zero mean and

$$E[x_i x_j] = \boldsymbol{v}_i^T\boldsymbol{v}_j = r^{ij}, \qquad E[x_i' x_j'] = \boldsymbol{u}_i^T\boldsymbol{u}_j = r_{ij}. \tag{38}$$

Therefore,

$$(x_1', \cdots, x_n') \sim N(\boldsymbol{0}, \boldsymbol{R}), \tag{39}$$

$$(x_1, \cdots, x_n) \sim N(\boldsymbol{0}, \boldsymbol{R}^{-1}), \tag{40}$$

and from (35) and (36) c_{ij} is a function of b_i, b_j, w_i, w_j and r_{ij}, and c_{ij}^{lk} is a function of b_i, b_j, w_i, w_j and $(r_{ij})_{m\times m}$.

Lemma 3 *For $1 \leq i \leq m$,*

$$\boldsymbol{A}_{i,m+1} = \boldsymbol{A}_{m+1,i}^T = (\sum_{k=1}^{m} c_{i1}^k \boldsymbol{v}_k, \cdots, \sum_{k=1}^{m} c_{im}^k \boldsymbol{v}_k) \tag{41}$$

where

$$c_{ij}^k = E[\varphi'(w_i x_i' + b_i)\varphi(w_j x_j' + b_j)x_k'], \quad 1 \leq i, j, k \leq m. \tag{42}$$

$\boldsymbol{A}_{i,m+2}$ has the same structure as $\boldsymbol{A}_{i,m+1}$:

$$\boldsymbol{A}_{i,m+2} = \boldsymbol{A}_{m+2,i}^T = (\sum_{k=1}^{m} \tilde{c}_{i1}^k \boldsymbol{v}_k, \cdots, \sum_{k=1}^{m} \tilde{c}_{im}^k \boldsymbol{v}_k) \tag{43}$$

where

$$\widetilde{c}_{ij}^{k} = a_i a_j E[\varphi'(w_i x_i' + b_i)\varphi'(w_j x_j' + b_j)x_k'], \quad 1 \le i, j, k \le m.$$

$$\boldsymbol{A}_{m+1,m+1} = (b_{ij})_{m\times m} \tag{44}$$

with $b_{ij} = E[\varphi(w_i x_i' + b_i)\varphi(w_j x_j' + b_j)]$ *is a function of* b_i, b_j, w_i, w_j *and* r_{ij}.

$$\boldsymbol{A}_{m+1,m+2} = \boldsymbol{A}_{m+2,m+1}^T = (\widetilde{b}_{ij})_{m\times m} \tag{45}$$

with $\widetilde{b}_{ij} = a_j E[\varphi(w_i x_i' + b_i)\varphi'(w_j x_j' + b_j)]$.

$$\boldsymbol{A}_{m+2,m+2} = (b_{ij}')_{m\times m} \tag{46}$$

with $b_{ij}' = a_i a_j E[\varphi'(w_i x_i' + b_i)\varphi'(w_j x_j' + b_j)]$.

The proof of Lemma 3 is given in Appendix 3.

Remark 6 *When the input is Gaussian random vector, the matrix* $\boldsymbol{G}$ *can be calculated analytically by Lemma 1-3 except for some numerical integrations for those coefficients* d_1, d_2 *and* c_{ij}^{kl} *etc. When the pdf of the input is unknown, we can estimate the Fisher information matrix based on the empirical distribution of the input. However, this is time consuming and also needs a large number of input examples. It is difficult to implement the natural gradient descent method as an on-line algorithm in this way. When the pdf of the input is known but different from the standard Gaussian distribution, either non-standard Gaussian or non-Gaussian, we need some preprocessing procedures.*

3.4 Preprocessing

In the previous sections, the explicit form of $\boldsymbol{G}$ is found by assuming a standard Gaussian input. This assumption facilitates the computation of $\boldsymbol{G}$. In fact, the explicit form of $\boldsymbol{G}$ is still useful for non-Gaussian inputs if a preprocessing procedure is applied. This is equivalent to adding one preprocessing layer in front of the stochastic multi-layer perceptron defined by (1).

Assume the sampled input $\boldsymbol{x}_t$ is an i.i.d. process. When the input is not a standard Gaussian process, we can use a linear or non-linear mapping to transform the input into a Gaussian process.

If the input is Gaussian but the covariance matrix is not an identity matrix, we can apply a linear transform $\boldsymbol{u}_t = \boldsymbol{B}\boldsymbol{x}_t$ to obtain a standard Gaussian process $\boldsymbol{u}_t$. Here, the matrix $\boldsymbol{B}$ can be found either by a batch algorithm or an on-line algorithm. By the batch algorithm, we first compute the sample covariance matrix

$$\widehat{\boldsymbol{R}}_{\boldsymbol{x}} = \frac{1}{T-1}\sum_{t=1}^{T}(\boldsymbol{x}_t - \overline{\boldsymbol{x}})(\boldsymbol{x}_t - \overline{\boldsymbol{x}})^T,$$

$$\overline{\boldsymbol{x}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{x}_t,$$

then compute the Cholesky factorization $\widehat{\boldsymbol{R}}_{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{F}^T$ and find $\boldsymbol{B} = \boldsymbol{F}^{-1}$. We can also apply the following on-line whitening algorithm in [6]:

$$\boldsymbol{B}_{t+1} = (1+\mu)\boldsymbol{B}_t - \mu\boldsymbol{u}_t\boldsymbol{u}_t^T\boldsymbol{B}_t \tag{47}$$

where $\boldsymbol{u}_t = \boldsymbol{B}_t \boldsymbol{x}_t$ and $\mu > 0$ is a learning rate.

If the input is not Gaussian, we need a non-linear function to transform the input $\boldsymbol{x}_t$ to a Gaussian process. Let $F_{\boldsymbol{x}}(\cdot)$ be the cumulative distribution function (cdf) of $\boldsymbol{x}_t$ and $F_N(\cdot)$ be the cdf of n-dimensional Gaussian r.v., then we obtain a standard Gaussian process

$$\boldsymbol{u}_t = F_N^{-1}(F_{\boldsymbol{x}}(\boldsymbol{x}_t)) \sim N(\boldsymbol{0}, \boldsymbol{I}).$$

If $F_{\boldsymbol{x}}(\cdot)$ is unknown, it can be approximated by the empirical distribution based on the data set. This method is applicable to the input with an arbitrary distribution but it is not applicable to on-line processing since the empirical distribution is computed by a batch algorithm. If the probability density function (pdf) of the input can be approximated by some expansions such as the Gram-Charlier expansion and the Edgeworth expansion [12], $F_{\boldsymbol{x}}(\cdot)$ can be approximated on-line by using an adaptive algorithm to compute the moments or cumulants in these expansions.

After preprocessing, we use the data set $\widetilde{D}_T = \{(\boldsymbol{u}_t, z_t), t = 1, \cdots, T\}$ instead of D_T to train the multi-layer perceptron.

4 Inverse of Fisher Information Matrix

Now we calculate the inverse of the Fisher information matrix which is needed in the natural gradient descent method.

We have already shown the explicit form (26) of the inverse of $\boldsymbol{A}_{ii}$ for $1 \leq i \leq m$. To calculate the entire inverse of $\boldsymbol{G}$, we need to compute the inverse of a matrix with the similar representation as $\boldsymbol{A}_{ij}$ for $1 \leq i \neq j \leq m$.

To this end, let us define

$$\begin{aligned}
&Gl(m, \Re) = \{\boldsymbol{A} \in \Re^{m \times m} : \det(\boldsymbol{A}) \neq 0\}, \\
&\mathcal{M} = \{a_0 \Omega_0 + \sum_{i,j=1}^{m} a_{ij} \boldsymbol{u}_i \boldsymbol{v}_j^T : a_0 \neq 0, \boldsymbol{A} = (a_{ij}) \in Gl(m, \Re)\}, \\
&\overline{\mathcal{M}} = \{a_0 \Omega_0 + \sum_{i,j=1}^{m} a_{ij} \boldsymbol{u}_i \boldsymbol{v}_j^T\} \quad \text{(the closure of } \mathcal{M}), \\
&\widetilde{\mathcal{M}} = \{[a_0, \boldsymbol{A}] : a_0 \neq 0, \boldsymbol{A} \in Gl(m, \Re)\},
\end{aligned}$$

and recall the definition of $\Omega_0 = \sum_{i,j=m+1}^{n} \boldsymbol{u}_i \boldsymbol{u}_j^T$. The $Gl(m, \Re)$ is the set of all non-singular $m \times m$ matrices. The set $\widetilde{\mathcal{M}}$ is a product space of $\Re \backslash \{0\}$ and $Gl(m, \Re)$.

4.1 A group structure

Define a mapping $\psi : \quad \widetilde{\mathcal{M}} \to \Re^{n \times n}$ by

$$\psi(\widetilde{A}) = a_0 \Omega_0 + \sum_{i,j=1}^{m} a_{ij} \boldsymbol{u}_i \boldsymbol{v}_j^T \in \mathcal{M}, \quad \text{for } \widetilde{A} = [a_0, (a_{ij})] \in \widetilde{\mathcal{M}}.$$

Then $\mathcal{M}$ is the image set of the mapping ψ.

For $\widetilde{A} = [a_0, \boldsymbol{A}]$ and $\widetilde{B} = [b_0, \boldsymbol{B}] \in \widetilde{\mathcal{M}}$, we define a multiplication in $\widetilde{\mathcal{M}}$ by

$$\widetilde{A} \star \widetilde{B} = [a_0 b_0, \boldsymbol{A}\boldsymbol{B}].$$

It is easy to verify that $\widetilde{\mathcal{M}}$ is a group. For $\widetilde{A} = [a_0, \boldsymbol{A}] \in \widetilde{\mathcal{M}}$, its inverse element is $\widetilde{A}^{-1} = [\frac{1}{a_0}, \boldsymbol{A}^{-1}]$. The following lemma gives the relation between $\mathcal{M}$ and $\widetilde{\mathcal{M}}$.

Lemma 4 *The mapping ψ is an isomorphism between $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ which implies that $\mathcal{M}$ is sub-group of $Gl(n,\Re)$ and for $\boldsymbol{C} = a_0\Omega_0 + \sum_{i,j=1}^{m} a_{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T \in \mathcal{M}$,*

$$\boldsymbol{C}^{-1} = \frac{1}{a_0}\Omega_0 + \sum_{i,j=1}^{m} a^{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T \tag{48}$$

where the matrix (a^{ij}) is the inverse of the matrix $(a_{ij})^{-1}$.

The proof of Lemma 4 is given in Appendix 4.

Regarding the relation between $\mathcal{M}$ and $\overline{\mathcal{M}}$, we have

Lemma 5

$$\mathcal{M} = \overline{\mathcal{M}} \bigcap Gl(n,\Re) \tag{49}$$

The proof of Lemma 5 is given in Appendix 5.

Let $\boldsymbol{B}_{[m\times m]} = [\boldsymbol{B}_{ij}]_{m\times m}$ be a matrix consisting of $m \times m$ blocks. To invert the Fisher information matrix, we need the following lemma regarding the inverse of a block matrix $\boldsymbol{B}_{[m\times m]}$ whose blocks belong to $\overline{\mathcal{M}}$.

Lemma 6 *If $\boldsymbol{B}_{[m\times m]}$ is a positive definite matrix with the blocks $\boldsymbol{B}_{ij} \in \overline{\mathcal{M}}$, then the inverse $\boldsymbol{B}_{[m\times m]}^{-1}$ has the same structure as $\boldsymbol{B}_{[m\times m]}$, namely, $\boldsymbol{B}_{[m\times m]}^{-1} = [\boldsymbol{B}^{ij}]_{m\times m}$ in which $\boldsymbol{B}^{ij} \in \overline{\mathcal{M}}$.*

The proof of Lemma 6 is given here since it includes procedures to compute the inverse of a block matrix with blocks in $\overline{\mathcal{M}}$. These procedures will be referred to for several times in later sections.

Proof of Lemma 6:
We shall repeatedly apply the following 4-block matrix inverse formula in the current paper.

$$\begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{B}^{11} & \boldsymbol{B}^{12} \\ \boldsymbol{B}^{21} & \boldsymbol{B}^{22} \end{bmatrix}$$

where

$\boldsymbol{B}^{11} = \boldsymbol{B}_{11}^{-1} + \boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}\boldsymbol{B}_{22,1}^{-1}\boldsymbol{B}_{21}\boldsymbol{B}_{11}^{-1}$,
$\boldsymbol{B}_{22,1} = \boldsymbol{B}_{22} - \boldsymbol{B}_{21}\boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}$,
$\boldsymbol{B}^{22} = \boldsymbol{B}_{22,1}^{-1}$, and
$\boldsymbol{B}^{12} = (\boldsymbol{B}^{21})^T = -\boldsymbol{B}_{11}^{-1}\boldsymbol{B}_{12}\boldsymbol{B}_{22,1}^{-1}$.

Since $\boldsymbol{B}_{[m\times m]}$ is positive definite, the diagonal blocks $\boldsymbol{B}_{ii}$ in $\boldsymbol{B}_{[m\times m]}$ and $\boldsymbol{B}_{[m\times m]}^{-1}$ are also positive definite. Let us first compute the inverse of $[\boldsymbol{B}_{ij}]_{2\times 2}$. By Lemma 5, if an invertible matrix belongs to $\overline{\mathcal{M}}$, it also belongs to $\mathcal{M}$. So $\boldsymbol{B}_{11} \in \mathcal{M}$, and by Lemma 4 $\boldsymbol{B}_{11}^{-1} \in \mathcal{M}$. Since the equality (65) in Appendix 4 still holds in $\overline{\mathcal{M}}$, the set $\overline{\mathcal{M}}$ is closed under the matrix product. It is easy to see that $\overline{\mathcal{M}}$ is also closed under the operations such as the matrix summation and scaling. By Lemma 5, $\boldsymbol{B}_{22,1} \in \mathcal{M}$, and so $\boldsymbol{B}_{22,1}^{-1} \in \mathcal{M}$ again by Lemma 4. Hence, $\boldsymbol{B}^{ij} \in \overline{\mathcal{M}}$ for $i \neq j$. The proposition of Lemma 6 is proved for $m = 2$. Assume the proposition is true for $m = k - 1$.

Again, we apply the 4-block matrix inverse formula to the following 2×2 block matrix

$$\boldsymbol{B}_{[k\times k]} = \begin{bmatrix} \boldsymbol{B}'_{11} & \boldsymbol{B}'_{12} \\ \boldsymbol{B}'_{21} & \boldsymbol{B}'_{22} \end{bmatrix}$$

where

$$\begin{aligned}
\boldsymbol{B}'_{11} &= \boldsymbol{B}_{[(k-1)\times(k-1)]} \\
\boldsymbol{B}'_{12} &= \begin{bmatrix} \boldsymbol{B}_{1,k} \\ \vdots \\ \boldsymbol{B}_{k-1,k} \end{bmatrix} \\
\boldsymbol{B}'_{21} &= [\boldsymbol{B}_{k,1}, \cdots, \boldsymbol{B}_{k,k-1}] \\
\boldsymbol{B}'_{22} &= \boldsymbol{B}_{kk}.
\end{aligned}$$

Because $\overline{\mathcal{M}}$ is closed to the matrix product, summation and scaling, $\boldsymbol{B}'_{22,1} = \boldsymbol{B}'_{22} - \boldsymbol{B}'_{21}\boldsymbol{B}'^{-1}_{11}\boldsymbol{B}'_{12}$ belongs to $\overline{\mathcal{M}}$ by the inductive assumption that the proposition is true for $m = k-1$. Since $\boldsymbol{B}_{[k\times k]}$ is positive definite and so is $\boldsymbol{B}'_{22,1}$ and $\boldsymbol{B}'^{22} = \boldsymbol{B}'^{-1}_{22,1}$. Similarly, we can prove that the blocks in $\boldsymbol{B}'^{12}$, $\boldsymbol{B}'^{21}$ and $\boldsymbol{B}'^{11}$ belong to $\overline{\mathcal{M}}$. Therefore, we conclude that the proposition is also true for $m = k$.
Q.E.D.

4.2 Inverse of $\boldsymbol{A}_{ij}$

Define matrices

$$\boldsymbol{C}_{ij} = \begin{bmatrix} c_{ij}^{1,1} & \cdots & c_{ij}^{1,m} \\ \cdots & \cdots & \cdots \\ c_{ij}^{m,1} & \cdots & c_{ij}^{m,m} \end{bmatrix}$$

for $1 \le i \ne j \le m$. Here, c_{ij}^{kl} are defined by (36). Applying Lemma 4-5, from Lemma 2 we have the following theorem.

Theorem 1 *For $1 \le i \ne j \le m$, $\boldsymbol{A}_{ij}$ is invertible iff $c_{ij} \ne 0$ and the matrix $\boldsymbol{C}_{ij}$ is invertible. When $\boldsymbol{A}_{ij}$ is invertible, we have the following inverse formula*

$$\boldsymbol{A}_{ij}^{-1} = \frac{1}{a_i a_j}\left(\frac{1}{c_{ij}}\Omega_0 + \sum_{l,k=1}^{m} d_{ij}^{lk}\boldsymbol{u}_l\boldsymbol{v}_k^T\right) \tag{50}$$

where $(d_{ij}^{lk}) = \boldsymbol{C}_{ij}^{-1}$.

4.3 Inverse of the whole Fisher information matrix

To compute the inverse of the Fisher information matrix $\boldsymbol{G} = \frac{1}{\sigma^2}[\boldsymbol{A}_{ij}]_{(m+2)\times(m+2)}$, we consider the following partition of $\boldsymbol{G}$:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_{11} & \boldsymbol{G}_{12} \\ \boldsymbol{G}_{21} & \boldsymbol{G}_{22} \end{bmatrix} \tag{51}$$

where $\boldsymbol{G}_{11}$ and $\boldsymbol{G}_{22}$ are the $m \times m$ and 2×2 block matrices respectively on the left-upper and right-lower corners of $\boldsymbol{G}$. We first apply the 4-block matrix inverse formula to the above partition of $\boldsymbol{G}$ and then apply the procedures in the proof of Lemma 6 to compute the inverse of $\boldsymbol{G}_{11}$.

4.4 Complexity of computing the inverse of the Fisher information matrix

The method for computing $\boldsymbol{G}^{-1}$ given in the above section has low complexity when $m \ll n$. We define the complexity of an algorithm by the time and space needed to run the algorithm. We count the time and space complexity by flops and units. A flop is one multiplication or one addition. A unit is the memory space for a variable.

One advantage of using the formulas (26) and (50) is to reduce the space complexity for storing $\boldsymbol{G}$. To store $\boldsymbol{G}$ directly, we need $\frac{1}{2}(mn+2m)(mn+2m+1)$ units. But, by using the representation (33), when $m \ll n$ the space complexity for storing $\boldsymbol{G}$ is reduced to $O(n)$ from $O(n^2)$ since the dual bases are shared by all blocks in $\boldsymbol{G}_{11}$. Here, $\boldsymbol{G}_{11}$ is defined in the partition (51) of $\boldsymbol{G}$.

The dual bases $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$ and $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n\}$ are used in the theoretical analysis. The vectors $\{\boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n\}$ are not needed in the representation (33) of $\boldsymbol{A}_{ij}$ and the inverse formula (50). Normalizing the weight vectors $\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_m\}$, we obtain $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m\}$ with the time complexity $O(n)$. By the algorithm given in Remark 4, we can compute $\boldsymbol{V}_1 = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m]$. The time complexity is also $O(n)$.

In the partition (51), the dimension of $\boldsymbol{G}_{11}$ is $mn \times mn$ and that of $\boldsymbol{G}_{22}$ is $2m \times 2m$. When $m \ll n$, the time complexity of computing $\boldsymbol{G}^{-1}$ is determined by that of computing $\boldsymbol{G}_{11}^{-1}$ which is computed by the procedures in the proof of Lemma 6 where the 4-block matrix inverse formula and Lemma 4-5 are repeatedly applied. In this process, $\boldsymbol{G}_{11}^{-1}$ is recursively computed. The size of the working memory needed is still of order $O(n^2)$ since $m \ll n$. But, when $m \ll n$, the time complexity of computing $\boldsymbol{G}_{11}^{-1}$ is significantly reduced from $O(n^3)$ by conventional methods such as the one in [11] to $O(n^2)$ by our method. In summary, we have the following table to compare our method with the conventional methods:

Table 1: The complexity of our method and the conventional methods

Cost	Conventional methods	Our method
Space to store $\boldsymbol{G}$	$O(n^2)$	$O(n)$
Time to compute $\boldsymbol{G}^{-1}$	$O(n^3)$	$O(n^2)$
Working memory needed	$O(n^2)$	$O(n^2)$

Since the explicit expressions (26) and (50) are suitable for parallel matrix computations, we can further reduce the time complexity by using a computer with distributed memory multiprocessors.

4.5 The Cramer-Rao lower bound for a committee machine

For a committee machine, we can further simplify the factor $\mathrm{Tr}(\boldsymbol{G}^{-1}(\boldsymbol{\theta}^*))$ in the Cramer-Rao lower bound inequality (3). When $a_i = 1, b_i = 0, i = 1, \cdots, m$, the multi-layer perceptron (1) becomes a model for a committee machine. The dimension of the parameter space of this committee machine is nm.

The Fisher information matrix for the committee machine is

$$\boldsymbol{G}(\boldsymbol{\theta}) = \frac{1}{\sigma^2}\boldsymbol{A}_{[m\times m]} = \frac{1}{\sigma^2}\begin{bmatrix} \boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{1m} \\ \cdots & \cdots & \cdots \\ \boldsymbol{A}_{m1} & \cdots & \boldsymbol{A}_{mm} \end{bmatrix}$$

where $\boldsymbol{\theta} = (\boldsymbol{w}_1^T, \cdots, \boldsymbol{w}_m^T)^T$ and $\boldsymbol{A}_{ij}$ are the same as those for the multi-layer perceptron except $a_i = 1$.

By Lemma 6,

$$\boldsymbol{G}(\boldsymbol{\theta})^{-1} = \sigma^2 \boldsymbol{A}_{[m\times m]}^{-1} = \sigma^2 \begin{bmatrix} \boldsymbol{A}^{11} & \cdots & \boldsymbol{A}^{1m} \\ \cdots & \cdots & \cdots \\ \boldsymbol{A}^{m1} & \cdots & \boldsymbol{A}^{mm} \end{bmatrix}$$

where $\boldsymbol{A}^{ii} \in \mathcal{M}$ for all i and $\boldsymbol{A}^{ij} \in \overline{\mathcal{M}}$ for $i \neq j$.

Note for $\boldsymbol{A} = a_0\Omega_0 + \sum_{k,l=1}^m a_{kl}\boldsymbol{u}_k\boldsymbol{v}_l^T \in \mathcal{M}$,

$$Tr(\boldsymbol{A}) = a_0(n-m) + \mathrm{Tr}((a_{kl})) = a_0(n-m) + \sum_{k=1}^m a_{kk}$$

because of the properties (30) and (31).

Let $\boldsymbol{A}^{ii} = a_0^i\Omega_0 + \sum_{k,l=1}^m a_{ii}^{kl}\boldsymbol{u}_k\boldsymbol{v}_l^T$, then

$$Tr(\boldsymbol{A}^{ii}) = a_0^i(n-m) + \sum_{k=1}^m a_{ii}^{kk}.$$

Hence,

$$\mathrm{Tr}(\boldsymbol{G}(\boldsymbol{\theta})^{-1}) = \sigma^2\mathrm{Tr}(\boldsymbol{A}_{[m\times m]}^{-1}) = \sigma^2\sum_{i=1}^m \mathrm{Tr}(\boldsymbol{A}^{ii}) = \sigma^2(c_0(\boldsymbol{\theta}) + c_1(\boldsymbol{\theta})n)$$

where

$$c_1(\boldsymbol{\theta}) = \sum_{i=1}^m a_0^i$$
$$c_0(\boldsymbol{\theta}) = \sum_{i,k=1}^m a_{ii}^{kk} - c_1(\boldsymbol{\theta})m.$$

So we have the following theorem which gives an order estimation of the Cramer-Rao lower bound for the committee machine.

Theorem 2 *Assume $m \ll n$ and $\boldsymbol{G}(\boldsymbol{\theta})$ is non-singular around $\boldsymbol{\theta}^*$, then the Cramer-Rao lower bound inequality is*

$$E_{\boldsymbol{\theta}^*}[\|\widehat{\boldsymbol{\theta}}_t - \boldsymbol{\theta}^*\|^2] \geq \frac{\sigma^2}{t}(c_0 + c_1 n) \tag{52}$$

where $c_0 = c_0(\boldsymbol{\theta}^*)$ and $c_1 = c_1(\boldsymbol{\theta}^*)$ are two constants not depending on the input dimension n.

It is shown in [2] that the gradient descent learning rule (6) is Fisher efficient. In particular, for the committee machine, by Theorem 2 the natural gradient descent rule gives the mean square error

$$E_{\boldsymbol{\theta}^*}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|^2] \approx \frac{\sigma^2}{t}(c_0 + c_1 n). \tag{53}$$

5 Natural gradient vector field vs. ordinary vector field

The behaviors of the two learning dynamics (5) and (6) are characterized, respectively, by the mean vector fields

$$\boldsymbol{v}_1(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}^*}[\frac{\partial l}{\partial \boldsymbol{\theta}}(z|\boldsymbol{x};\boldsymbol{\theta})] = E_{\boldsymbol{\theta}^*}[\begin{bmatrix} \frac{\partial l}{\partial \boldsymbol{w}_1}(z|\boldsymbol{x};\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial l}{\partial \boldsymbol{w}_m}(z|\boldsymbol{x};\boldsymbol{\theta}) \\ \frac{\partial l}{\partial \boldsymbol{a}}(z|\boldsymbol{x};\boldsymbol{\theta}) \\ \frac{\partial l}{\partial \boldsymbol{b}}(z|\boldsymbol{x};\boldsymbol{\theta}) \end{bmatrix}] \text{ and}$$

$$\boldsymbol{v}_2(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}^*}[\boldsymbol{G}^{-1}(\boldsymbol{\theta})\frac{\partial l}{\partial \boldsymbol{\theta}}(z|\boldsymbol{x};\boldsymbol{\theta})] = \boldsymbol{G}^{-1}(\boldsymbol{\theta})\boldsymbol{v}_1(\boldsymbol{\theta}).$$

Again, the expectation with respect to $p(\boldsymbol{x}, z; \boldsymbol{\theta}^*)$ is denoted by $E_{\boldsymbol{\theta}^*}[\cdot]$ and the samples $(\boldsymbol{x}, z)$ are independently drawn from the pdf $p(\boldsymbol{x}, z; \boldsymbol{\theta}^*)$ for the teacher network:

$$z = \sum_{j=1}^{m} a_j^* \varphi(\boldsymbol{w}_j^{*T}\boldsymbol{x} + b_j^*) + \xi.$$

From (9)-(11), we have

$$E_{\boldsymbol{\theta}^*}[\frac{\partial l}{\partial \boldsymbol{w}_i}(z|\boldsymbol{x};\boldsymbol{\theta})] = -\frac{1}{\sigma^2}E_{\boldsymbol{\theta}^*}[\sum_{j=1}^{m}(a_j^*\varphi(\boldsymbol{w}_j^{*T}\boldsymbol{x} + b_j^*) - a_j\varphi(\boldsymbol{w}_j^T\boldsymbol{x} + b_j))a_i\varphi'(\boldsymbol{w}_i^T\boldsymbol{x} + b_i)\boldsymbol{x}],$$

$$E_{\boldsymbol{\theta}^*}[\frac{\partial l}{\partial \boldsymbol{a}}(z|\boldsymbol{x};\boldsymbol{\theta})] = -\frac{1}{\sigma^2}E_{\boldsymbol{\theta}^*}[\sum_{j=1}^{m}(a_j^*\varphi(\boldsymbol{w}_j^{*T}\boldsymbol{x} + b_j^*) - a_j\varphi(\boldsymbol{w}_j^T\boldsymbol{x} + b_j))\varphi(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})],$$

$$E_{\boldsymbol{\theta}^*}[\frac{\partial l}{\partial \boldsymbol{b}}(z|\boldsymbol{x};\boldsymbol{\theta})] = -\frac{1}{\sigma^2}E_{\boldsymbol{\theta}^*}[\sum_{j=1}^{m}(a_j^*\varphi(\boldsymbol{w}_j^{*T}\boldsymbol{x} + b_j^*) - a_j\varphi(\boldsymbol{w}_j^T\boldsymbol{x} + b_j))\{\boldsymbol{a} \odot \varphi'(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})\}].$$

The above two equations are used to compute the vector field $\boldsymbol{v}_1(\boldsymbol{\theta})$. Note the σ^2 in $\boldsymbol{v}_1(\boldsymbol{\theta})$ and $\boldsymbol{G}^{-1}(\boldsymbol{\theta})$ are canceled in computing $\boldsymbol{v}_2(\boldsymbol{\theta})$. So the vector field $\boldsymbol{v}_2(\boldsymbol{\theta})$ does not depend on the noise level σ^2.

The following dynamic systems:

$$\frac{d\boldsymbol{\theta}}{dt} = -\eta \boldsymbol{v}_1(\boldsymbol{\theta}) \tag{54}$$

$$\frac{d\boldsymbol{\theta}}{dt} = -\eta \boldsymbol{v}_2(\boldsymbol{\theta}) = -\eta \boldsymbol{G}^{-1}(\boldsymbol{\theta})\boldsymbol{v}_1(\boldsymbol{\theta}) \tag{55}$$

characterize the averaging behaviors of the gradient descent algorithm (5) and the natural gradient descent algorithm (6) respectively.

To show the difference between the two mean vector fields $\boldsymbol{v}_1(\boldsymbol{\theta})$ and $\boldsymbol{v}_2(\boldsymbol{\theta})$, we consider a simple committee machine with 2-dimensional input, 2 hidden neurons, $a_1 = a_2 = 1$ and the thresholds $b_1 = b_2 = 0$, and the 2 weight vectors $\boldsymbol{w}_i$ with unit length $\|\boldsymbol{w}_i\| = 1$. The weight vectors are reparameterized by

$$\boldsymbol{w}_i = \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix}, \quad i = 1, 2.$$

Let $\varphi(x) = \tanh(x)$, $\sigma^2 = 1$ and $\alpha_1^* = 0$ and $\alpha_2^* = \frac{3\pi}{4}$ be the parameters for the teacher. Let $\boldsymbol{\theta} = (\alpha_1, \alpha_2)$. We calculate the mean vector fields $v_1(\alpha_1, \alpha_2)$ and $v_2(\alpha_1, \alpha_2)$ in the same way of deriving (54) and (55). The two mean vector fields are plotted in Figure 1-2.

Although the gradient descent system (54) and the natural gradient descent system (55) have the same set of fixed points, they generate different flows. The convergence of the ordinary gradient flow towards $\boldsymbol{\theta}^*$ is slower than the natural gradient flow especially along the diagonal line in the (α_1, α_2)-space. On average, the learning dynamics of the natural gradient descent algorithm is optimized to achieve the Cramer-Rao lower bound.

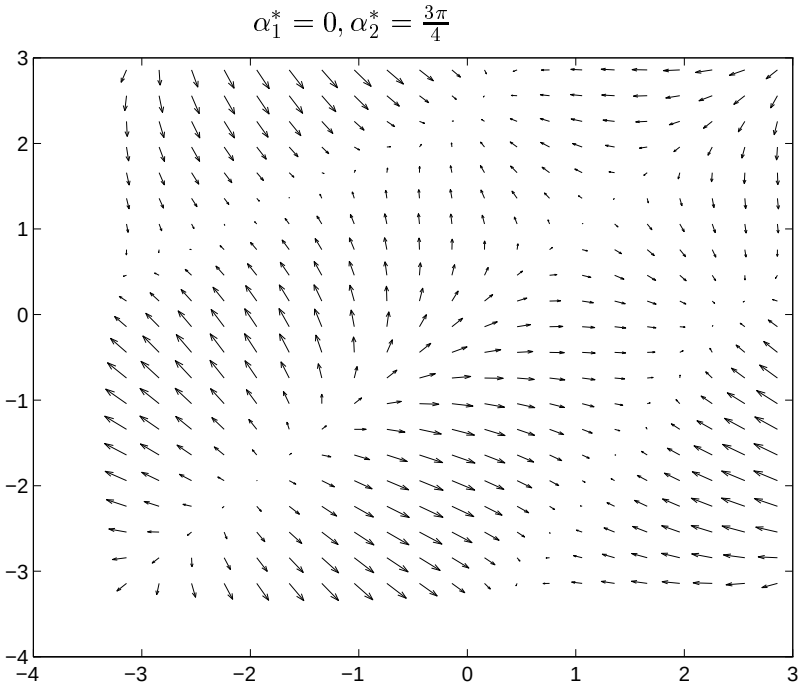

Figure 1: The mean vector field $v_1(\alpha_1, \alpha_2)$

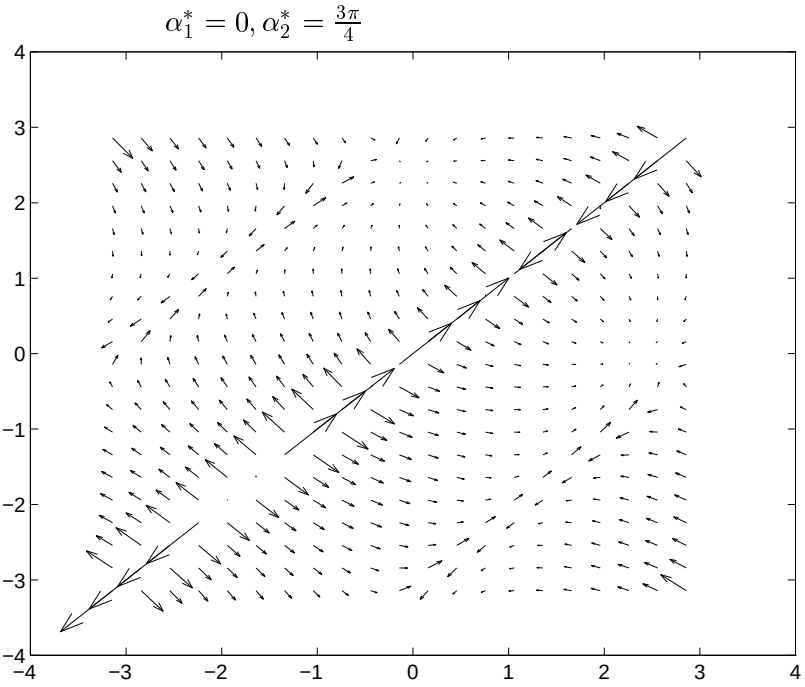

Figure 2: The mean vector field $v_2(\alpha_1, \alpha_2)$

6 Simulation

6.1 Single-layer perceptron

Assume the input $\boldsymbol{x}_t$ is of dimension $n = 7$ subject to the Gaussian distribution $\boldsymbol{x}_t \sim N(\boldsymbol{0}, \boldsymbol{I})$. The output z_t are generated by the model:

$$z_t = \varphi(\boldsymbol{w}^{*T}\boldsymbol{x}_t) + \xi_t$$

where $\varphi(u) = \frac{1-e^{-u}}{1+e^{-u}}$ and $\xi_t \sim N(0, \sigma^2)$. For the single-layer perceptron, $z = \varphi(\boldsymbol{w}^T\boldsymbol{x})$, we have the two on-line algorithms based on the gradient descent (GD) and the natural GD methods in the following forms:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mu_0(t)(z_t - \varphi(\boldsymbol{w}_t^T\boldsymbol{x}_t))\varphi'(\boldsymbol{w}_t^T\boldsymbol{x}_t)\boldsymbol{x}_t \tag{56}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mu_1(t)\boldsymbol{A}^{-1}(\boldsymbol{w}_t)(z_t - \varphi(\boldsymbol{w}_t^T\boldsymbol{x}_t))\varphi'(\boldsymbol{w}_t^T\boldsymbol{x}_t)\boldsymbol{x}_t \tag{57}$$

where

$$\boldsymbol{A}^{-1}(\boldsymbol{w}) = \frac{1}{d_1(w)}\boldsymbol{I} + (\frac{1}{d_2(w)} - \frac{1}{d_1(w)})\frac{\boldsymbol{w}\boldsymbol{w}^T}{w^2}, \tag{58}$$

$$d_i(w) = d_i(w, 0), \quad i = 1, 2,$$

$d_i(w, b)$ was defined previously by (24) and (25), and $\mu_i(t)$ are two learning rate schedules defined by

$$\mu_i(t) = \eta_i \frac{1 + \frac{c_i}{\eta_i}t}{1 + \frac{c_i}{\eta_i}t + t^2}, \quad i = 0, 1. \tag{59}$$

Here t is the iteration index. The learning rate function $\mu_i(t)$ is a special form of the following search-then-converge schedules proposed in [7]:

$$\mu(t) = \eta \frac{1 + \frac{c}{\eta}\frac{t}{\tau}}{1 + \frac{c}{\eta}\frac{t}{\tau} + \frac{t^2}{\tau}}. \tag{60}$$

$t < \tau$ is a "search phase" and $t > \tau$ is a "converge phase". The learning rate functions $\mu_i(t)$ do not have the search phase but they start learning with a weaker converge phase when η_i are small. When t is large, each learning rate function $\mu_i(t)$ decreases as $\frac{c_i}{t}$.

In this example, we choose $\eta_0 = 1.25$, $\eta_1 = 0.05$, $c_0 = 8.75$, and $c_1 = 1$. These parameters are selected by trial and error to optimize the performance of the GD and the natural GD methods at the noise level $\sigma = 0.2$. The learning rate functions $\mu_0(t)$ and $\mu_1(t)$ are compared in Figure 3 with their asymptotic curves $\frac{c_i}{t}$.

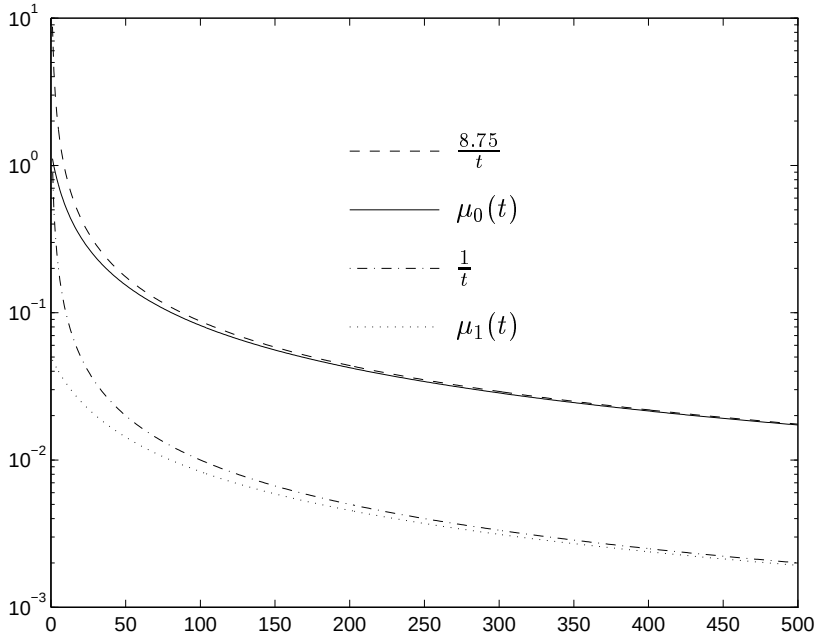

Figure 3: $\mu_0(t)$ vs. $\mu_1(t)$ when $\eta_0 = 1.25$, $\eta_1 = 0.05$, $c_0 = 8.75$ and $c_1 = 1$.

A 7-dimensional vector is randomly chosen as $\boldsymbol{w}^*$ for the teacher network. In this simulation, we choose

$$\boldsymbol{w}^* = [-1.1043, 0.4302, 1.1978, 1.5317, -2.2946, -0.7866, 0.4428]^T.$$

The training examples are the Gaussian random inputs to the teacher network and its outputs in Gaussian noise. Let $\boldsymbol{w}_t$ and $\widetilde{\boldsymbol{w}}_t$ be the weight vectors driven by the equations (56) and (57) respectively. The error functions for the GD and the natural GD algorithms are $\|\boldsymbol{w}_t - \boldsymbol{w}^*\|$ and $\|\widetilde{\boldsymbol{w}}_t - \boldsymbol{w}^*\|$.

Let $w^* = \|\boldsymbol{w}^*\|$. From the equation (58), we obtain the Cramer-Rao Lower Bound (CRLB) for the deviation at the true weight vector $\boldsymbol{w}^*$:

$$\mathrm{CRLB}(t) = \frac{\sigma}{\sqrt{t}} \sqrt{\frac{n-1}{d_1(w^*)} + \frac{1}{d_2(w^*)}}. \tag{61}$$

The error functions are compared with the CRLB in Figure 4.

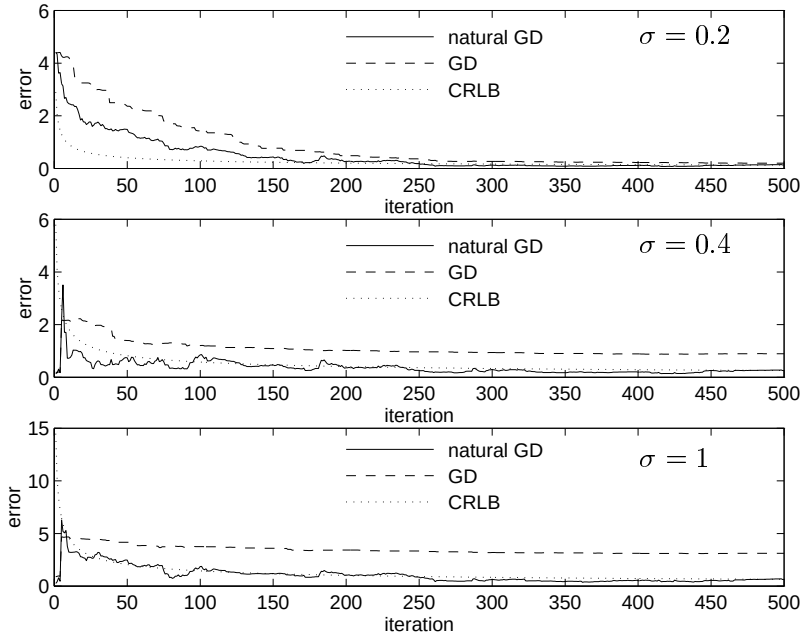

Figure 4: Performance of the GD and the natural GD at different noise levels $\sigma = 0.2, 0.4, 1$.

It is shown in Figure 4 that the natural GD algorithm reaches CRLB at different noise levels while the GD algorithm reaches the CRLB only at the noise level $\sigma = 0.2$. The robustness of the natural gradient descent against the additive noise in the training examples is clearly shown by this example.

6.2 Multi-layer perceptron

Let us reconsider the simple committee machine with 2-dimensional input and 2-hidden neurons given in Section 5. The problem is to train the following multi-layer perceptron:

$$y = \varphi(\boldsymbol{w}_1^T \boldsymbol{x}) + \varphi(\boldsymbol{w}_2^T \boldsymbol{x}) \tag{62}$$

based on the examples $\{(\boldsymbol{x}_t, z_t), t = 1, \cdots, T\}$ generated by the following stochastic committee machine:

$$z_t = \varphi(\boldsymbol{w}_1^{*T} \boldsymbol{x}_t) + \varphi(\boldsymbol{w}_2^{*T} \boldsymbol{x}_t) + \xi_t \tag{63}$$

with Gaussian noise $\xi_t \sim N(0, \sigma^2)$ and Gaussian input $\boldsymbol{x}_t \sim N(\boldsymbol{0}, \boldsymbol{I})$. Assume $\|w_i^*\| = 1$. We can reparameterize the weight vector to decrease the dimension of the parameter space from 4 to 2:

$$\boldsymbol{w}_i = \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \end{bmatrix}, \quad \boldsymbol{w}_i^* = \begin{bmatrix} \cos(\alpha_i^*) \\ \sin(\alpha_i^*) \end{bmatrix}, \quad i = 1, 2.$$

The parameter space is $\{\boldsymbol{\theta} = (\alpha_1, \alpha_2)\}$. Assume the true parameters are $\alpha_1^* = 0$ and $\alpha_2^* = \frac{3\pi}{4}$. Due to the symmetry, both $\boldsymbol{\theta}_1^* = (0, \frac{3\pi}{4})$ and $\boldsymbol{\theta}_2^* = (\frac{3\pi}{4}, 0)$ are true parameters. Let $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}'_t$ be generated by the GD algorithm and the natural GD algorithm respectively. The errors are measured by

$$\varepsilon_t = \min\{\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_1^*\|, \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_2^*\|\},$$
$$\varepsilon'_t = \min\{\|\boldsymbol{\theta}'_t - \boldsymbol{\theta}_1^*\|, \|\boldsymbol{\theta}'_t - \boldsymbol{\theta}_2^*\|\}.$$

In this simulation, we first start the GD algorithm with the initial vector $\boldsymbol{\theta}_0 = (0.1, 0.2)$ for 80 iterations, then we start the natural GD algorithm using the estimates of the parameters obtained by the GD algorithm at the 80-th iteration as the initial for the natural GD algorithm. The noise level is $\sigma = 0.05$. N independent runs are conducted to obtain $\varepsilon_t(j)$ and $\varepsilon'_t(j)$ for $j = 1, \cdots, N$. Averaging these errors, we obtain

$$\overline{\varepsilon}_t = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\varepsilon_t(j))^2}, \quad \text{and} \quad \overline{\varepsilon'}_t = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\varepsilon'_t(j))^2}.$$

The averaging errors $\overline{\varepsilon}_t$ and $\overline{\varepsilon'}_t$ for the two algorithms are compared with the CRLB in Figure 5 for $N = 10$. The search-then-converge learning schedule (60) is used in the GD algorithm while the learning rate for the natural GD algorithm is simply the annealing rate $\frac{1}{k}$. The CRLB is given in Section 4.5 by Theorem 2.

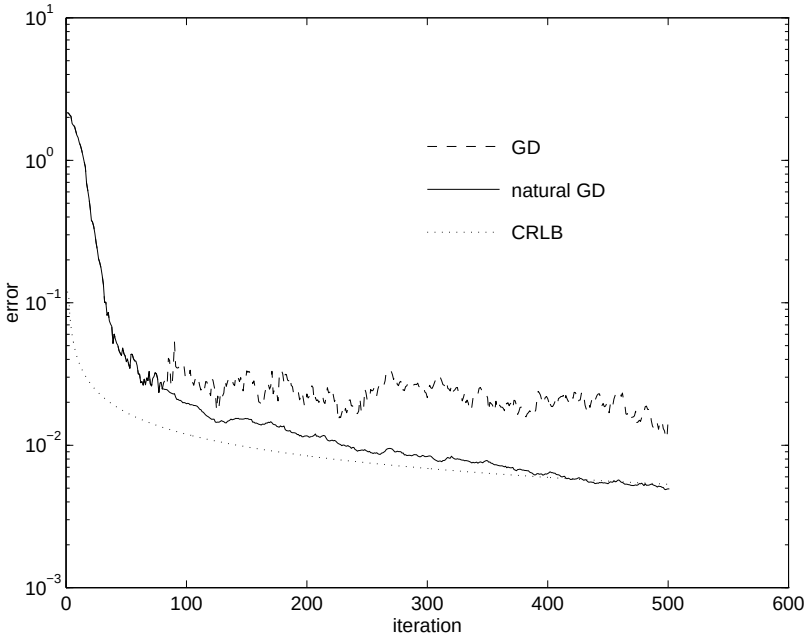

Figure 5: The GD vs. the natural GD

7 Conclusions

The natural gradient descent learning rule is statistically efficient. It can be used to train any adaptive system. But the complexity of this learning rule depends on the architecture of

the learning machine. The main difficulty in implementing this learning rule is to compute the inverse of the Fisher information matrix of a large size. For a multi-layer perceptron, we have shown an efficient scheme to represent the Fisher information matrix based on which the space for storing this large matrix is reduced to $O(n)$ from $O(n^2)$. We have also shown an algorithm to compute the inverse of the Fisher information matrix. Its time complexity is of order $O(n^2)$ when the input dimension n is much larger than the number of hidden neurons while the complexity of conventional algorithms for the same purpose is of order $O(n^3)$. An intermediate result is an explicit CRLB formula for a committee machine.

The simulation results have confirmed the fast convergence and statistical efficiency of the natural gradient descent learning rule. It is also verified that this learning rule is robust against the additive noise in the training examples.

8 Appendix

Appendix 1 *Proof of Lemma 1:*

Let $\boldsymbol{u}_1 = \boldsymbol{w}_i/w_i$ and extend it to $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n\}$ an orthogonal basis of $\Re^n$. The random input $\boldsymbol{x}$ is decomposed as

$$\boldsymbol{x} = \sum_{j=1}^{n} c_j \boldsymbol{u}_j$$

where $c_j = \boldsymbol{x}^T \boldsymbol{u}_j$. Because of $\boldsymbol{x} \sim N(\boldsymbol{0}, \boldsymbol{I})$, c_j are i.i.d. with $N(0,1)$. Noticing

$$\boldsymbol{x}\boldsymbol{x}^T = c_1^2 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \sum_{j=2}^{n} c_j c_1 (\boldsymbol{u}_j \boldsymbol{u}_1^T + \boldsymbol{u}_1 \boldsymbol{u}_j^T) + \sum_{j,k=2}^{n} c_j c_k \boldsymbol{u}_j \boldsymbol{u}_k^T,$$

$E[c_j] = 0$ and $E[c_j c_k] = \delta_{j,k}$, we have
$E[(\varphi'(\boldsymbol{w}_i^T \boldsymbol{x} + b_i))^2 \boldsymbol{x}\boldsymbol{x}^T]$
$= E[(\varphi'(w_i c_1 + b_i))^2 c_1^2] \boldsymbol{u}_1 \boldsymbol{u}_1^T + E[(\varphi'(w_i c_1 + b_i))^2 c_1] \sum_{j=2}^{n} E[c_j] (\boldsymbol{u}_j \boldsymbol{u}_1^T + \boldsymbol{u}_1 \boldsymbol{u}_j^T)$
$+ E[(\varphi'(w_i c_1 + b_i))^2] \sum_{j,k=2}^{n} E[c_j c_k] \boldsymbol{u}_j \boldsymbol{u}_k^T$
$= E[(\varphi'(w_i c_1 + b_i))^2 c_1^2] \boldsymbol{u}_1 \boldsymbol{u}_1^T + E[(\varphi'(w_i c_1 + b_i))^2] \sum_{j=2}^{n} \boldsymbol{u}_j \boldsymbol{u}_j^T$
$= d_2(w_i, b_i) \boldsymbol{u}_1 \boldsymbol{u}_1^T + d_1(w_i, b_i) \sum_{j=2}^{n} \boldsymbol{u}_j \boldsymbol{u}_j^T$.

Since $\{\boldsymbol{u}_i\}$ are orthogonal,

$$\sum_{k=2}^{n} \boldsymbol{u}_i \boldsymbol{u}_i^T = \boldsymbol{I} - \boldsymbol{u}_1 \boldsymbol{u}_1^T.$$

Therefore,

$$E[(\varphi'(\boldsymbol{w}_i^T \boldsymbol{x} + b_i))^2 \boldsymbol{x}\boldsymbol{x}^T] = d_1(w_i, b_i) \boldsymbol{I} + (d_2(w_i, b_i) - d_1(w_i, b_i)) \boldsymbol{u}_1 \boldsymbol{u}_1^T.$$

Q.E.D.

Appendix 2 *Proof of Lemma 2:*

Here we use the notations introduced before Lemma 2.

Since $\boldsymbol{u}_j = \boldsymbol{v}_j$ for $j > m$, we have $x_j = x'_j$ for $j > m$, and $E[x_i x'_j] = E[x_i x_j] = \delta_{i,j}$ for $i, j > m$.

When $k \neq l$, $E[x_k x'_l] = E[\boldsymbol{v}_k^T \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{u}_l] = \boldsymbol{v}_k^T \boldsymbol{u}_l = 0$. Since x_i and x'_i are Gaussian, x_k and x'_l are independent when $k \neq l$. Therefore, $\{x'_1, \cdots, x'_m\}$ and $\{x_{m+1}, \cdots, x_n\}$ are independent. Applying this result, we simplify the following

$E[\varphi'(\boldsymbol{w}_i^T\boldsymbol{x}+b_i)\varphi'(\boldsymbol{w}_j^T\boldsymbol{x}+b_j)\boldsymbol{x}\boldsymbol{x}^T]$
$= E[\varphi'(w_i\boldsymbol{u}_i^T\boldsymbol{x}+b_i)\varphi'(w_j\boldsymbol{u}_j^T\boldsymbol{x}+b_j)(\sum_{l=1}^n x_l\boldsymbol{u}_l)(\sum_{k=1}^n x'_k\boldsymbol{v}_k^T)]$
$= E[\varphi'(w_ix'_i+b_i)\varphi'(w_jx'_j+b_j)(\sum_{l,k=1}^m+\sum_{l=1}^m\sum_{k=m+1}^n+\sum_{l=m+1}^n\sum_{k=1}^m+\sum_{l,k=m+1}^n)x_lx'_k\boldsymbol{u}_l\boldsymbol{v}_k^T]$
$= \sum_{l,k=1}^m c_{ij}^{lk}\boldsymbol{u}_l\boldsymbol{v}_k^T + c_{ij}\sum_{l,k=m+1}^n\delta_{l,k}\boldsymbol{u}_l\boldsymbol{v}_k^T + \text{Term1} + \text{Term2}$
$= c_{ij}\sum_{k=m+1}^n\boldsymbol{u}_k\boldsymbol{u}_k^T + \sum_{l,k=1}^m c_{ij}^{lk}\boldsymbol{u}_l\boldsymbol{v}_k^T + \text{Term1} + \text{Term2}$

where c_{ij} and c_{ij}^{lk} are defined by (35) and (36). Here,
$\text{Term1} = \sum_{l=1}^m\sum_{k=m+1}^n E[\varphi'(w_ix'_i+b_i)\varphi'(w_jx'_j+b_j)x_lx_k]\boldsymbol{u}_l\boldsymbol{u}_k^T$ and
$\text{Term2} = \sum_{l=m+1}^n\sum_{k=1}^m E[\varphi'(w_ix'_i+b_i)\varphi'(w_jx'_j+b_j)x_lx'_k]\boldsymbol{u}_l\boldsymbol{v}_k^T$

are zero due to three reasons: 1) $x_l = \sum_{s=1}^m r^{ls}x'_s$ for $1 \le l \le m$, 2) $\{x'_1,\cdots,x'_m\}$ and $\{x_{m+1},\cdots,x_n\}$ are independent and 3) $E[x_k]=0$. Finally, we obtain (33).

By the definition $\boldsymbol{V}=(\boldsymbol{U}^{-1})^T$, so $\sum_{k=1}^n\boldsymbol{u}_k\boldsymbol{v}_k^T = \boldsymbol{U}\boldsymbol{V}^T = \boldsymbol{I}$ and

$$\Omega_0 = \sum_{k=m+1}^n\boldsymbol{u}_k\boldsymbol{u}_k^T = \boldsymbol{I} - \sum_{k=1}^m\boldsymbol{u}_k\boldsymbol{v}_k^T.$$

Q.E.D.

Appendix 3 *Proof of Lemma 3:*

Again, we shall apply the properties that $\{x'_1,\cdots,x'_m\}$ and $\{x_{m+1},\cdots,x_n\}$ are independent to simplify $\boldsymbol{A}_{i,m+1}$ for $1 \le i \le m$:

$$\begin{aligned}
\boldsymbol{A}_{i,m+1} &= E[\varphi'(\boldsymbol{w}_i^T\boldsymbol{x}+b_i)\boldsymbol{x}\varphi(\boldsymbol{x}^T\boldsymbol{W}+\boldsymbol{b}^T)] \\
&= E[\varphi'(w_ix'_i+b_i)(\sum_{k=1}^m x'_k\boldsymbol{v}_k + \sum_{k=m+1}^n x'_k\boldsymbol{v}_k)(\varphi(w_1x'_1+b_1),\cdots,\varphi(w_mx'_m+b_m)] \\
&= E[\varphi'(w_ix'_i)\sum_{k=1}^m x'_k\boldsymbol{v}_k(\varphi(w_1x'_1+b_1),\cdots,\varphi(w_mx'_m+b_m)] \\
&+E[\varphi'(w_ix'_i+b_i)\sum_{k=m+1}^n x'_k\boldsymbol{v}_k(\varphi(w_1x'_1+b_1),\cdots,\varphi(w_mx'_m+b_m)]. \qquad (64)
\end{aligned}$$

The first term on the right hand side (RHS) of the above equation (64) becomes

$$(\sum_{i=1}^m c_{i1}^k\boldsymbol{v}_k,\cdots,\sum_{i=1}^m c_{im}^k\boldsymbol{v}_k)$$

where c_{ij}^k are defined by (42). Since $E[x_k]=0$, the second term on the RHS of (64) is zero. Denote

$$\widetilde{\varphi}(\boldsymbol{u}^T) = \boldsymbol{a}^T\odot\varphi'(\boldsymbol{u}^T) = [a_1,\cdots,a_m\varphi'(u_m)].$$

Similar to the derivation of $\boldsymbol{A}_{i,m+1}$, we obtain the expression (64) for $\boldsymbol{A}_{i,m+2}$.

It is easy to obtain $\boldsymbol{A}_{m+1,m+1} = (b_{ij})_{m\times m}$ where

$$b_{ij} = E[\varphi(\boldsymbol{w}_i^T\boldsymbol{x}+b_i)\varphi(\boldsymbol{w}_j^T\boldsymbol{x}+b_j)] = E[\varphi(w_ix'_i+b_i)\varphi(w_jx'_j+b_j)].$$

b_{ij} are functions of b_i, b_j, w_i, w_j, and r_{ij} since $E[x'_ix'_j]=r_{ij}$ for $1 \le i,j \le m$. Similarly, we obtain (45) and (46).

Q.E.D.

Appendix 4 *Proof of Lemma 4:*

For $\widetilde{A} = [a_0, (a_{ij})] \in \widetilde{\mathcal{M}}$, define

$$\psi(\widetilde{A}) = a_0\Omega_0 + \sum_{i,j=1}^{m} a_{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T.$$

To show that $\psi : \widetilde{\mathcal{M}} \to \mathcal{M}$ is one-to-one, we assume $\widetilde{B} = [b_0, (b_{ij})] \in \widetilde{\mathcal{M}}$ and $\psi(\widetilde{A}) = \psi(\widetilde{B})$, i.e.,

$$a_0\Omega_0 + \sum_{i,j=1}^{m} a_{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T = b_0\Omega_0 + \sum_{i,j=1}^{m} b_{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T.$$

Since $\Omega_0\boldsymbol{u}_i\boldsymbol{v}_j^T = 0$ for $1 \le i, j \le m$, from the above equation we first obtain $a_0 = b_0$ then $a_{ij} = b_{ij}$ after applying the property (31). Therefore, ψ is a one-to-one mapping from $\widetilde{\mathcal{M}}$ to $\mathcal{M}$. Let $\boldsymbol{C} = \psi(\widetilde{A})$ and $\boldsymbol{C}' = \psi(\widetilde{B})$, then

$$\begin{aligned}
\boldsymbol{C}\boldsymbol{C}' &= a_0b_0\Omega_0 + \sum_{i,j,l,k=1}^{m} a_{ij}b_{lk}\boldsymbol{u}_i\boldsymbol{v}_j^T\boldsymbol{u}_l\boldsymbol{v}_k^T \\
&= a_0b_0\Omega_0 + \sum_{i,j,l,k=1}^{m} a_{ij}b_{lk}\delta_{j,l}\boldsymbol{u}_i\boldsymbol{v}_k^T = a_0b_0\Omega_0 + \sum_{i,j,k=1}^{m} a_{ij}b_{jk}\boldsymbol{u}_i\boldsymbol{v}_k^T \\
&= a_0b_0\Omega_0 + \sum_{i,k=1}^{m} d_{ik}\boldsymbol{u}_i\boldsymbol{v}_k^T, \qquad (65)
\end{aligned}$$

where $(d_{ij}) = (a_{ij})(b_{ij})$. So $\boldsymbol{C}\boldsymbol{C}' = \psi([a_0b_0,\ (a_{ij})(b_{ij})])$, i.e.,

$$\psi(\widetilde{A})\psi(\widetilde{B}) = \psi(\widetilde{A} \star \widetilde{B}).$$

Hence, ψ is an isomorphic mapping between $\widetilde{\mathcal{M}}$ and $\mathcal{M}$, and $\mathcal{M}$ is a sub-group of $Gl(n, \Re)$. For any element in $\mathcal{M}$ with the form $\boldsymbol{C} = a_0\Omega_0 + \sum_{i,j=1}^{m} a_{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T = \psi([a_0,\ (a_{ij})])$, to obtain the inverse of $\boldsymbol{C}$, we first compute the inverse in $\widetilde{\mathcal{M}}$

$$\widetilde{A}^{-1} = [\frac{1}{a_0}, (a_{ij})^{-1}]$$

then the inverse in $\mathcal{M}$

$$\boldsymbol{C}^{-1} = \psi(\widetilde{A}^{-1}) = \frac{1}{a_0}\Omega_0 + \sum_{i,j=1}^{m} a^{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T$$

by the relation $(\psi(\widetilde{A}))^{-1} = \psi(\widetilde{A}^{-1})$, where $(a^{ij}) = (a_{ij})^{-1}$.
Q.E.D.

Appendix 5 *Proof of Lemma 5:*

It follows from Lemma 4:

$$\mathcal{M} \subseteq \overline{\mathcal{M}} \bigcap Gl(n, \Re).$$

Let $\boldsymbol{A} \in \overline{\mathcal{M}}$ and $\det(\boldsymbol{A}) \neq 0$. From $\boldsymbol{A} = a_0\Omega_0 + \sum_{i,j=1}^{m} a_{ij}\boldsymbol{u}_i\boldsymbol{v}_j^T$, we have

$$\begin{aligned}
&\boldsymbol{A}\boldsymbol{u}_k = a_0\boldsymbol{u}_k \text{ for } k = m+1, \cdots, n, \text{ and} \\
&\boldsymbol{A}\boldsymbol{u}_k = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m][a_{1k}, \cdots, a_{mk}]^T \text{ for } k = 1, \cdots, m.
\end{aligned}$$

So $\boldsymbol{A}[\boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n] = a_0[\boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n]$ and $\boldsymbol{A}[\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m] = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m]\boldsymbol{B}$ where $\boldsymbol{B} = (a_{ij})$. Hence,

$$\boldsymbol{AU} = \boldsymbol{A}[\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m, \boldsymbol{u}_{m+1}, \cdots, \boldsymbol{u}_n] = \boldsymbol{U} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & a_0 \boldsymbol{I} \end{bmatrix}.$$

Therefore, $\det(\boldsymbol{B}) a_0^{n-m} = \det(\boldsymbol{AU}) \neq 0$ which implies $\det(\boldsymbol{B}) \neq 0$ and $a_0 \neq 0$. So $\boldsymbol{A} \in \mathcal{M}$. Q.E.D.

References

[1] S. Amari. *Differential-Geometrical Methods in Statistics, Lecture Notes in Statistics vol.28.* Springer, 1985.

[2] S. Amari. Natural gradient works efficiently in learning. *Accepted by Neural Computation*, 1997.

[3] S. Amari. Neural learning in structured parameter spaces – natural Riemannian gradient. In *Advances in Neural Information Processing Systems, 9, MIT Press: Cambridge, MA. (to appear)*, 1997.

[4] S. Amari and N. Murata. Statistical theory of learning curves under entropic loss criterion. *Neural Computation*, 5:140–153, 1993.

[5] S. Amari, N. Murata, K.-R. Müller, M. Finke, and H. H. Yang. Asymptotic statistical theory of overtraining and cross-validation. *IEEE Trans. on Neural Networks*, page to appear, 1997.

[6] J.-F. Cardoso and B. Laheld. Equivariant adaptive source separation. *IEEE Trans. on Signal Processing*, 44(12):3017–3030, December 1996.

[7] C. Darken and J. Moody. Towards faster stochastic gradient search. In *Advances in Neural Information Processing Systems, 4, eds. Moody, Hanson, and Lippmann, Morgan Kaufmann, San Mateo*, pages 1009–1016, 1992.

[8] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.

[9] M. K. Murray and J. W. Rice. *Differential Geometry and Statistics* . New York: Chapman & Hall, 1993.

[10] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[11] G. W. Stewart. *Introduction to Matrix Computations.* New York: Academic Press, 1973.

[12] A. Stuart and J. K. Ord. *Kendall's Advanced Theory of Statistics.* Edward Arnold, 1994.