TR-H-243

On-Line EM Algorithm for the Nomalized Gaussian Network.

Masa-aki SATO and Shin ISHII

1998.4.6

ATR人間情報通信研究所
〒619-0288　京都府相楽郡精華町光台2-2　TEL: 0774-95-1011

ATR Human Information Processing Research Laboratories
2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
Telephone: +81-774-95-1011
Fax : +81-774-95-1008

On-line EM Algorithm for the Normalized Gaussian Network

Masa-aki Sato † and **Shin Ishii** ‡†

† ATR Human Information Processing Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan
TEL: (+81)-774-95-1039 FAX: (+81)-774-95-1008
E-mail: masaaki@hip.atr.co.jp

‡† Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara 630-0101, Japan

Abstract

Normalized Gaussian Network (NGnet) (Moody and Darken 1989) is a network of local linear regression units. The model softly partitions the input space by normalized Gaussian functions and each local unit linearly approximates the output within the partition.

In this article, we propose a new on-line EM algorithm for the NGnet, which is derived from the batch EM algorithm (Xu, Jordan and Hinton 1995) by introducing a discount factor. We show that the on-line EM algorithm is equivalent to the batch EM algorithm if a specific scheduling of the discount factor is employed. In addition, we show that the on-line EM algorithm can be considered as a stochastic approximation method to find the maximum likelihood estimator. A new regularization method is proposed in order to deal with a singular input distribution. In order to manage dynamic environments, where the input-output distribution of data changes with time, unit manipulation mechanisms such as unit production, unit deletion, and unit division are also introduced based on the probabilistic interpretation.

Experimental results show that our approach is suitable for function approximation problems in dynamical environments. We also applied our on-line EM algorithm to a reinforcement learning problem. It is shown that the NGnet, when using the on-line EM algorithm, learns the value function much faster than the method based on the gradient descent algorithm.

1 Introduction

Normalized Gaussian Network (NGnet) (Moody and Darken 1989) is a network of local linear regression units. The model softly partitions the input space by normalized Gaussian functions and each local unit linearly approximates the output within the partition. Since the NGnet is a local model, it is possible to change parameters of several units in order to learn a single datum. Therefore, the learning process becomes easier than that of the global models such as the multi-layered perceptron. In a local model, on the other hand, the number of necessary units grows exponentially as the input dimension increases, if one wants to approximates the input-output relationship over the whole input space. This results in a computational explosion, which is often called the "curse of dimensionality". However, actual data will often distribute in a lower dimensional space than the input space, such as in attractors of dynamical systems.

The NGnet, which is a kind of "Mixtures of Experts" model (Jacobs, Jordan, Nowlan and Hinton 1991 ; Jordan and Jacobs 1994), can be interpreted as an output of a stochastic model with hidden variables. The model parameters can be determined by the maximum likelihood estimation method. In particular, the EM algorithm for the NGnet was derived by Xu, Jordan and Hinton (1995). In this article, we propose a new on-line EM algorithm for the NGnet. The on-line EM algorithm is derived from the batch EM algorithm (Xu, Jordan and Hinton 1995) by introducing a discount factor. Although the derivation is straightforward, important modifications are necessary for practical applications. We will discuss these points in detail and propose a modified version of the on-line EM algorithm. We show that the on-line EM algorithm is equivalent to the batch EM algorithm if a specific scheduling of the discount factor is employed. In addition, we show that the on-line EM algorithm can be considered as a stochastic approximation method to find the maximum likelihood estimator. A new regularization method is proposed in order to deal with a singular input distribution. Unit manipulation mechanisms based on the probabilistic interpretation are also introduced in order to manage dynamic environments. These mechanisms are unit production, unit deletion, and unit division.

In order to investigate the performance of our new on-line EM algorithm, experiments for function approximation problems are conducted for three circumstances. The first one is a usual function approximation problem for a set of observed data. The second one is a function approximation in which the distribution of the input data changes with time. This experiment is performed to check the applicability of our approach to the dynamic environments. The third one is a function approximation in which the distribution of the input data is singular, i.e., the dimension of the input data distribution is smaller than the input space dimension. In this case, a straightforward application of the basic on-line EM algorithm does not work, because the covariance matrices used in the NGnet become singular. Our modified on-line EM algorithm is shown to work well even in this situation.

Encouraged by these positive results, we applied our on-line EM algorithm to a reinforcement learning problem. In the actor-critic model (Barto, Sutton and Anderson 1983), learning of the value function for a current policy can be regarded as a function approximation problem in a dynamic environment, since the policy changes with time as the learning proceeds. As an example, we examined a task for swinging-up an inverted pendulum (Doya 1996). The experimental result shows that the NGnet, when using our new on-line EM algorithm, learns the value function much faster than the method based on the gradient descent algorithm.

The paper is organized as follows. The NGnet and its stochastic model are explained in Section 2. The batch EM algorithm is introduced in Section 3. These sections are based on previous papers (Moody and Darken 1989; Xu, Jordan and Hinton 1995). The basic on-line EM algorithm is derived in Section 4. The modifications of the basic on-line EM algorithm are discussed in Sections 5, 6 and 7. The experimental results are shown in Section 8, and Section 9 sums up the paper.

2 Normalized Gaussian network

2.1 NGnet

The Normalized Gaussian Network (NGnet) model (Moody and Darken 1989), which transforms an N-dimensional input vector x to a D-dimensional output vector y, is defined by the following equations.

$$y = \sum_{i=1}^{M} (W_i x + b_i) \mathcal{N}_i(x) \tag{2.1a}$$

$$\mathcal{N}_i(x) \equiv G_i(x) / \sum_{j=1}^{M} G_j(x) \tag{2.1b}$$

$$G_i(x) \equiv (2\pi)^{-N/2} |\Sigma_i|^{-1/2} \exp\left[-\frac{1}{2} (x - \mu_i)' \Sigma_i^{-1} (x - \mu_i) \right] . \tag{2.1c}$$

M denotes the number of units, and the prime ($'$) denotes a transpose. $G_i(x)$ is an N-dimensional Gaussian function; its center is an N-dimensional vector μ_i and its covariance matrix is an $(N \times N)$-dimensional matrix Σ_i. $|\Sigma_i|$ is the determinant of the matrix Σ_i. $\mathcal{N}_i(x)$ is the i-th normalized Gaussian function. W_i and b_i are a $(D \times N)$-dimensional linear regression matrix and a D-dimensional bias vector, respectively. Subsequently, we use notations: $\tilde{W}_i \equiv (W_i, b_i)$, and $\tilde{x}' \equiv (x', 1)$. With these notations, (2.1a) is rewritten as

$$y = \sum_{i=1}^{M} \mathcal{N}_i(x) \tilde{W}_i \tilde{x} \ . \tag{2.2}$$

The Gaussian function $G_i(x)$ is a kind of radial basis function (Poggio and Girosi 1990). However, the normalized Gaussian functions, $\mathcal{N}_i(x)$ $(i = 1, ..., M)$, do not have a radial symmetry. They softly partition the input space into M regions. The i-th unit linearly approximates its output by $\tilde{W}_i \tilde{x}$ within the corresponding region. An output of the NGnet is given by a summation of these outputs weighted by the normalized Gaussian functions.

2.2 Stochastic model of NGnet

The NGnet (2.1) can be interpreted as a stochastic model, in which a pair of an input and an output (x, y) is a stochastic (incomplete) event. For each event, a single unit is assumed to be selected from a set of classes $\{i | i = 1, ..., M\}$. The unit index i is regarded as a hidden variable. A triplet (x, y, i) is called a complete event. The stochastic model is defined by the

probability distribution for a complete event (Xu, Jordan and Hinton 1995):

$$P(x,y,i|\theta) = (2\pi)^{-(D+N)/2}\sigma_i^{-D}|\Sigma_i|^{-1/2}M^{-1}\exp\left[-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i) - \frac{1}{2\sigma_i^2}(y-\tilde{W}_i\tilde{x})^2\right], \tag{2.3}$$

where $\theta \equiv \{\mu_i, \Sigma_i, \sigma_i^2, \tilde{W}_i \mid i = 1, ..., M\}$ is a set of model parameters. From the probability distribution (2.3), the following probabilities are obtained.

$$P(i|\theta) = 1/M \tag{2.4a}$$

$$P(x|i,\theta) = G_i(x) \tag{2.4b}$$

$$P(y|x,i,\theta) = (2\pi)^{-D/2}\sigma_i^{-D}\exp\left[-\frac{1}{2\sigma_i^2}(y-\tilde{W}_i\tilde{x})^2\right]. \tag{2.4c}$$

These probabilities define a stochastic data generation model. First, a unit is selected randomly with an equal probability (2.4a). If the i-th unit is selected, the input x is generated according to the Gaussian distribution (2.4b). Given the unit index i and the input x, the output y is generated according to the Gaussian distribution (2.4c), which has the mean $\tilde{W}_i\tilde{x}$ and the variance σ_i^2.

When the input x is observed, the probability that the output value becomes y in this model, turns out to be

$$P(y|x,\theta) = \sum_{i=1}^{M}\mathcal{N}_i(x)P(y|x,i,\theta). \tag{2.5}$$

From this conditional probability, the expectation value of the output y for a given input x is obtained as:

$$E[y|x] \equiv \int yP(y|x,\theta)dy = \sum_{i=1}^{M}\mathcal{N}_i(x)\tilde{W}_i\tilde{x}, \tag{2.6}$$

which is equivalent to the output of the NGnet (2.2). Namely, the probability distribution (2.3) provides a stochastic model for the NGnet.

3 EM algorithm

From a set of T events (observed data), $(\{x\},\{y\}) \equiv \{(x(t),y(t)) \mid t = 1, ..., T\}$, the model parameter θ of the stochastic model (2.3) can be determined by the maximum likelihood estimation method. In particular, the EM algorithm (Dempster, Laird and Rubin 1977) can be applied to models having hidden variables. The EM algorithm repeats the following E-step and M-step. Since the likelihood for the set of observations increases (or does not change) after an E- and M-step, the maximum likelihood estimator is asymptotically obtained by repeating the E- and M-steps.

- E (Estimation) step
  Let $\bar{\theta}$ be the present estimator. By using $\bar{\theta}$, the posterior probability that the i-th unit is selected for each observation $(x(t),y(t))$ is calculated according to the Bayes rule.

$$P(i|x(t),y(t),\bar{\theta}) = P(x(t),y(t),i|\bar{\theta})/\sum_{j=1}^{M}P(x(t),y(t),j|\bar{\theta}). \tag{3.1}$$

- M (Maximization) step
  By using the posterior probability (3.1), the expected log-likelihood $Q(\theta|\bar{\theta}, \{x\}, \{y\})$ for the complete events is defined by

$$Q(\theta|\bar{\theta}, \{x\}, \{y\}) = \sum_{t=1}^{T} \sum_{i=1}^{M} P(i|x(t), y(t), \bar{\theta}) \log P(x(t), y(t), i|\theta). \tag{3.2}$$

  On the other hand, the log-likelihood of the observed data $(\{x\}, \{y\})$ is given by

$$L(\theta|\{x\}, \{y\}) = \sum_{t=1}^{T} \log P(x(t), y(t)|\theta) = \sum_{t=1}^{T} \log \left(\sum_{i=1}^{M} P(x(t), y(t), i|\theta) \right). \tag{3.3}$$

  Since an increase of $Q(\theta|\bar{\theta}, \{x\}, \{y\})$ implies an increase of the log-likelihood $L(\theta|\{x\}, \{y\})$ (Dempster, Laird and Rubin 1977), $Q(\theta|\bar{\theta}, \{x\}, \{y\})$ is maximized with respect to the estimator θ. A solution of the necessity condition $\partial Q/\partial\theta = 0$ is given (Xu, Jordan and Hinton 1995) by

$$\mu_i = \langle x \rangle_i(T)/\langle 1 \rangle_i(T) \tag{3.4a}$$

$$\Sigma_i = \langle (x - \mu_i)(x - \mu_i)' \rangle_i(T)/\langle 1 \rangle_i(T) = \langle xx' \rangle_i(T)/\langle 1 \rangle_i(T) - \mu_i(T)\mu_i'(T) \tag{3.4b}$$

$$\tilde{W}_i \langle \tilde{x}\tilde{x}' \rangle_i(T) = \langle y\tilde{x}' \rangle_i(T) \tag{3.4c}$$

$$\sigma_i^2 = \frac{1}{D} \langle |y - \tilde{W}_i \tilde{x}|^2 \rangle_i(T)/\langle 1 \rangle_i(T) = \frac{1}{D} \left[\langle |y^2| \rangle_i(T) - \mathrm{Tr} \left(\tilde{W}_i \langle \tilde{x}y' \rangle_i(T) \right) \right] / \langle 1 \rangle_i(T) \tag{3.4d}$$

  where $\mathrm{Tr}(\cdot)$ denotes a matrix trace. A symbol $\langle \cdot \rangle_i$ denotes a weighted mean with respect to the posterior probability (3.1) and it is defined by

$$\langle f(x, y) \rangle_i(T) \equiv \frac{1}{T} \sum_{t=1}^{T} f(x(t), y(t)) P(i|x(t), y(t), \bar{\theta}). \tag{3.5}$$

If an infinite number of data, which are drawn independently according to an unknown data distribution density $\rho(x, y)$, are given, the weighted mean (3.5) converges to the following expectation value.

$$\langle f(x, y) \rangle_i(T) \overset{T \to \infty}{\longrightarrow} E[f(x, y) P(i|x, y, \bar{\theta})]_\rho, \tag{3.6}$$

where $E[\cdot]_\rho$ denotes the expectation value with respect to the data distribution density $\rho(x, y)$. In this case, the M-step equation (3.4) can be written as

$$g_i \equiv E[P(i|x, y, \bar{\theta})]_\rho \tag{3.7a}$$

$$\mu_i = E[xP(i|x, y, \bar{\theta})]_\rho / g_i \tag{3.7b}$$

$$\Sigma_i = E[xx' P(i|x, y, \bar{\theta})]_\rho / g_i - \mu_i \mu_i' \tag{3.7c}$$

$$\tilde{W}_i E[\tilde{x}\tilde{x}' P(i|x, y, \bar{\theta})]_\rho = E[y\tilde{x}' P(i|x, y, \bar{\theta})]_\rho \tag{3.7d}$$

$$\sigma_i^2 = \frac{1}{D} \left(E[|y|^2 P(i|x, y, \bar{\theta})]_\rho - \mathrm{Tr}(\tilde{W}_i E[\tilde{x}y' P(i|x, y, \bar{\theta})]_\rho) \right) / g_i, \tag{3.7e}$$

where the new parameter set $\theta = \{\mu_i, \Sigma_i, \tilde{W}_i, \sigma_i^2 | i = 1, ..., M\}$ is calculated by using the old parameter set $\bar{\theta} = \{\bar{\mu}_i, \bar{\Sigma}_i, \bar{\tilde{W}}_i, \bar{\sigma}_i^2 | i = 1, ..., M\}$.

At an equilibrium point of this EM algorithm, the old and new parameters become the same, i.e., $\theta = \bar{\theta}$. The equilibrium condition of the EM algorithm, i.e., (3.7) along with $\theta = \bar{\theta}$, is equivalent to the maximum likelihood condition,

$$\partial L(\theta)/\partial\theta = 0, \tag{3.8}$$

where the log-likelihood function is given by

$$L(\theta) = E[\log \sum_{i=1}^{M} P(x, y, i|\theta)]_\rho. \tag{3.9}$$

4 On-line EM algorithm

The EM algorithm introduced in the previous section is based on a batch learning (Xu, Jordan and Hinton 1995), namely, the parameters are updated after seeing all the observed data $(\{x\}, \{y\})$. In this section, we derive an on-line version of the EM algorithm. Since the estimator is changed after each observation, let $\theta(t)$ be the estimator after the t-th observation $(x(t), y(t))$. In the on-line EM algorithm, the weighted mean (3.5) is replaced by:

$$\ll f(x,y) \gg_i (T) \equiv \eta(T) \sum_{t=1}^{T} \lambda^{T-t} f(x(t), y(t)) P(i|x(t), y(t), \theta(t-1)) \tag{4.1a}$$

$$\eta(T) \equiv (\sum_{t=1}^{T} \lambda^{T-t})^{-1} = (1-\lambda)/(1-\lambda^T). \tag{4.1b}$$

Here, the parameter λ $(0 < \lambda < 1)$ is a discount factor, which is introduced for forgetting the effect of the old posterior values employing the earlier inaccurate estimator. $\eta(T)$ is a normalization coefficient and plays a role like a learning rate. The modified weighted mean $\ll \cdot \gg_i$ can be obtained by the step-wise equation:

$$\ll f(x,y) \gg_i (t) = \ll f(x,y) \gg_i (t-1) + \eta(t) \left[f(x(t), y(t)) P_i(t) - \ll f(x,y) \gg_i (t-1) \right], \tag{4.2}$$

where $P_i(t) \equiv P(i|x(t), y(t), \theta(t-1))$. After calculating the modified weighted means, i.e., $\ll 1 \gg_i (t)$, $\ll x \gg_i (t)$, $\ll xx' \gg_i (t)$, $\ll |y|^2 \gg_i (t)$, and $\ll y\tilde{x}' \gg_i (t)$, according to (4.2), the new estimator $\theta(t)$ is obtained by the following equations.

$$\mu_i(t) = \ll x \gg_i (t) / \ll 1 \gg_i (t) \tag{4.3a}$$

$$\Sigma_i^{-1}(t) = [\ll xx' \gg_i (t) / \ll 1 \gg_i (t) - \mu_i(t)\mu_i'(t)]^{-1} \tag{4.3b}$$

$$\tilde{W}_i(t) = \ll y\tilde{x}' \gg_i (t) [\ll \tilde{x}\tilde{x}' \gg_i (t)]^{-1} \tag{4.3c}$$

$$\sigma_i^2(t) = \frac{1}{D} \left[\ll |y|^2 \gg_i (t) - \mathrm{Tr}\left(\tilde{W}_i(t) \ll \tilde{x}y' \gg_i (t) \right) \right] / \ll 1 \gg_i (t). \tag{4.3d}$$

This defines the on-line EM algorithm. In this algorithm, calculations of the inverse matrices are necessary at each time step. A recursive formula for Σ_i^{-1} and $\tilde{W}_i$ can be derived, in which there is no need to calculate the matrix inverse, by using a standard method (Jürgen 1996).

Let us define the weighted inverse covariance matrix of $\tilde{x}$: $\tilde{\Lambda}_i(t) \equiv (\ll \tilde{x}\tilde{x}' \gg_i (t))^{-1}$. This quantity can be obtained by the step-wise equation:

$$\tilde{\Lambda}_i(t) = \frac{1}{1-\eta(t)} \left[\tilde{\Lambda}_i(t-1) - \frac{P_i(t)\tilde{\Lambda}_i(t-1)\tilde{x}(t)\tilde{x}'(t)\tilde{\Lambda}_i(t-1)}{(1/\eta(t)-1) + P_i(t)\tilde{x}'(t)\tilde{\Lambda}_i(t-1)\tilde{x}(t)} \right]. \tag{4.4}$$

$\Sigma_i^{-1}(t)$ can be obtained from the following relation with $\tilde{\Lambda}_i(t)$.

$$\tilde{\Lambda}_i(t) \ll 1 \gg_i (t) = \begin{pmatrix} \Sigma_i^{-1}(t) & -\Sigma_i^{-1}(t)\mu_i(t) \\ -\mu_i'(t)\Sigma_i^{-1}(t) & 1 + \mu_i'(t)\Sigma_i^{-1}(t)\mu_i(t) \end{pmatrix}. \tag{4.5}$$

The estimator for the linear regression matrix, $\tilde{W}_i$, is given by

$$\begin{aligned} \tilde{W}_i(t) &= \ll y\tilde{x}' \gg_i (t)\tilde{\Lambda}_i(t) & \text{(4.6a)} \\ &= \tilde{W}_i(t-1) + \eta(t)P_i(t)(y(t) - \tilde{W}_i(t-1)\tilde{x}(t))\tilde{x}'(t)\tilde{\Lambda}_i(t). & \text{(4.6b)} \end{aligned}$$

Thus, the basic on-line EM algorithm is summarized as follows. For each observation $(x(t), y(t))$, the weighted means, i.e., $\ll 1 \gg_i (t)$, $\ll x \gg_i (t)$, $\ll |y|^2 \gg_i (t)$, and $\ll \tilde{x}y' \gg_i (t)$, are calculated by using the step-wise equation (4.2). $\tilde{\Lambda}_i(t)$ is also calculated by the step-wise equation (4.4). Subsequently, the estimators for the model parameters are obtained by (4.3a), (4.3d), (4.5), and (4.6b).

5 Scheduling of discount factor

5.1 Time-dependent discount factor

In the previous section, we assumed that the discount factor λ is a constant. If this is the case, the estimators obtained by the on-line EM algorithm and the batch EM algorithm differ from each other due to the presence of λ in (4.1). Here, we assume that λ is a function of the time t, so that λ^{T-t} in (4.1a) is replaced by $\Pi_{s=t+1}^{T}\lambda(s)$. The normalization coefficient $\eta(T)$ (4.1b) is redefined by

$$\eta(T) \equiv \left(\sum_{t=1}^{T} \prod_{s=t+1}^{T} \lambda(s) \right)^{-1}. \tag{5.1}$$

It is calculated by the step-wise equation:

$$\eta(t) = (1 + \lambda(t)/\eta(t-1))^{-1}. \tag{5.2}$$

There is no need to redefine the step-wise equations (4.2), (4.4) and (4.6b), if the effective learning coefficient $\eta(t)$ is calculated by (5.2).

The constraint, $0 \leq \lambda(t) \leq 1$, gives the constraint on $\eta(t)$:

$$1 \geq \eta(t) \geq 1/t. \tag{5.3}$$

If this constraint is satisfied, the equation (5.2) can be solved for $\lambda(t)$ as:

$$\lambda(t) = \eta(t-1)(1/\eta(t) - 1). \tag{5.4}$$

5.2 Equivalence between on-line and batch EM algorithms

In the next part, we show that the on-line EM algorithm defined by (3.1), (4.2) and (4.3) is equivalent to the batch EM algorithm defined by (3.1), (3.4) and (3.5), with an appropriate choice of $\lambda(t)$.

It is assumed that the same set of data, $\{(x(t), y(t))|t = 1, ..., T\}$, is repeatedly supplied to the learning system, i.e., $x(t+T) = x(t)$ and $y(t+T) = y(t)$. The time t is represented by an epoch index k $(= 1, 2, ...)$ and a data number index m $(= 1, ..., T)$ within an epoch, i.e., $t = (k-1)T + m$. Let us assume that the model parameter θ is updated at the end of each epoch, i.e., when $m = T$, and it is denoted by $\theta(k)$. Let us suppose that $\lambda(t)$ is given by

$$\lambda(t) = \begin{cases} 0 & \text{if } t = (k-1)T + 1 \\ 1 & \text{otherwise} \end{cases} . \tag{5.5}$$

The corresponding $\eta(t)$ is given by $\eta((k-1)T + m) = 1/m$. Then, the weighted mean $\ll f(x, y) \gg_i (t)$ is initialized to $f(x(1), y(1))P(i|x(1), y(1), \theta(k-1))$ at the beginning of an epoch. At the end of an epoch, $\ll f(x, y) \gg_i (kT) = \langle f(x, y) \rangle_i(T)$ is satisfied for $\bar{\theta} = \theta(k-1)$. This shows that the on-line EM algorithm together with $\lambda(t)$ given by (5.5) is equivalent to the batch EM algorithm. It should be noted that the step-wise equation (4.4) for $\tilde{\Lambda}_i(t)$ can not be used in this case, since the equation (4.4) becomes singular for $\eta(t) = 1$.

5.3 Stochastic approximation

If an infinite number of data, which are drawn independently according to the data distribution density $\rho(x, y)$, are available, the on-line EM algorithm can be considered as a stochastic approximation (Kushner and Yin 1997) for obtaining the maximum likelihood estimator, as demonstrated below.

Let ϕ be a compact notation for the weighted mean, i.e., $\phi(t) \equiv \{\ll 1 \gg_i (t), \ll x \gg_i (t), \ll xx' \gg_i (t), \ll y\tilde{x}' \gg_i (t), \ll |y|^2 \gg_i (t) | i = 1, ..., M\}$. The on-line EM algorithm, (4.2) and (4.3), can be written in an abstract form:

$$\delta\phi(t) \equiv \phi(t) - \phi(t-1) = \eta(t)[F(x(t), y(t), \theta(t-1)) - \phi(t-1)] \tag{5.6a}$$

$$\theta(t) = H(\phi(t)). \tag{5.6b}$$

It can be easily proved that the set of equations:

$$\phi = E[F(x, y, \theta)]_\rho \tag{5.7a}$$

$$\theta = H(\phi) \tag{5.7b}$$

is equivalent to the maximum likelihood condition (3.8). Then, the on-line EM algorithm can be written as

$$\delta\phi(t) = \eta(t)(E[F(x, y, H(\phi(t-1)))]_\rho - \phi(t-1)) + \eta(t)\zeta(x(t), y(t), \phi(t-1)), \tag{5.8}$$

where the stochastic noise term ζ is defined by

$$\zeta(x(t), y(t), \phi(t-1)) \equiv F(x(t), y(t), H(\phi(t-1)) - E[F(x, y, H(\phi(t-1)))]_\rho, \tag{5.9}$$

and it satisfies

$$E[\zeta(x, y, \phi)]_\rho = 0. \tag{5.10}$$

The equation (5.8) has the same form as the Robbins-Monro stochastic approximation (Kushner and Yin 1997), which finds the maximum likelihood estimator given by (5.7). The effective learning coefficient $\eta(t)$ should satisfy the condition

$$\eta(t) \stackrel{t\to\infty}{\longrightarrow} 0, \quad \sum_{t=1}^{\infty} \eta(t) = \infty, \quad \sum_{t=1}^{\infty} \eta^2(t) < \infty. \tag{5.11}$$

Typically, $\eta(t)$, which satisfies the conditions (5.3) and (5.11), is given by

$$\eta(t) \stackrel{t\to\infty}{\longrightarrow} \frac{1}{at + b} \quad (1 > a > 0). \tag{5.12}$$

The corresponding discount factor is given by

$$\lambda(t) \stackrel{t\to\infty}{\longrightarrow} 1 - \frac{1 - a}{at + (b - a)}, \tag{5.13}$$

namely, $\lambda(t)$ is increased such that $\lambda(t)$ approaches 1 as $t \to \infty$.

For the convergence proof of the stochastic approximation (5.8), the boundedness of the noise variance is necessary (Kushner and Yin 1997). The noise variance is given by

$$E[\zeta(x, y, \phi)^2]_\rho = E[F(x, y, H(\phi))^2]_\rho - E[F(x, y, H(\phi))]_\rho^2. \tag{5.14}$$

Both terms on the right hand side in (5.14) are finite, if we assume that the data distribution density $\rho(x, y)$ has a compact support. Consequently, the noise variance becomes finite under this assumption. This assumption is not so restrictive, because actual data always distribute in a finite domain. We can weaken this assumption such that $\rho(x, y)$ decreases exponentially as $|x|$ or $|y|$ goes to infinity.

It should be noted that the stochastic approximation (5.8) for finding the maximum likelihood estimator is not a stochastic gradient ascent algorithm for the log-likelihood function (3.9). The on-line EM algorithm (5.8) is faster than the stochastic gradient ascent algorithm, because the M-step is solved exactly. We have so far used the on-line EM algorithm defined by (4.2) and (4.3), which is equivalent to the basic on-line EM algorithm defined by (4.2), (4.3a), (4.3d), (4.4), (4.5), and (4.6b), if $\eta(t) \neq 1$. Therefore, the basic on-line EM algorithm is also a stochastic approximation.

In practical applications, the learning coefficient $\eta(t)$ given by (5.12) becomes too small for a very large t, such that the parameters could not be changed significantly in a finite period. In order to avoid this situation, $\eta(t)$ is often set at a small positive value η_{min} for a very large t. This clamp corresponds to $\lambda(t) = 1 - \eta_{min}$. If the input and/or output distribution for the observed data change with time, it is not necessary that $\eta(t)$ converges to zero. This dynamical situation will be considered later in our experiments.

6 Regularization

6.1 Regularization of the covariance matrix

We have assumed so far that the covariance matrix of the input data is not singular in each region, namely, the inverse matrix for every Σ_i $(i = 1, ..., M)$ exists. In actual applications of

the algorithm, however, this assumption often fails. A typical case occurs when the dimension of the input data distribution is smaller than the dimension of the input space. In such a case, Σ_i^{-1} can not be calculated and $\tilde{\Lambda}_i$ diverges exponentially with the time t.

There are several methods for dealing with this problem. They are the introduction of Bayes priors, the singular value decomposition, the ridge regression, etc. However, they are not satisfactory for our purpose. Here, we will propose a simple new method that performs well.

Let us first consider a regularization of an $(N \times N)$-dimensional covariance matrix, Σ, for the observed data. Its eigen values and normalized eigen vectors are denoted by ξ_n (≥ 0) and ψ_n $(n = 1, ..., N)$, respectively.

$$\Sigma\psi_n \equiv \xi_n\psi_n \qquad (n = 1, ..., N). \tag{6.1}$$

The set of the eigen vectors, $\{\psi_n | n = 1, ..., N\}$, forms a set of orthonormal bases. The condition number of the covariance matrix Σ is defined by

$$\upsilon \equiv \xi_{min}/\xi_{max}, \tag{6.2}$$

where ξ_{min} and ξ_{max} are the minimum and maximal eigen values of Σ, respectively. So that $0 \leq \upsilon \leq 1$. If $\upsilon = 0$, the covariance matrix Σ is singular. If $\upsilon \approx 1$, the covariance matrix Σ is regular and the calculation of its inverse is numerically stable. If $0 < \upsilon \ll 1$, the covariance matrix is regular and Σ^{-1} is given by

$$\Sigma^{-1} = \sum_{n=1}^{N} \xi_n^{-1}\psi_n\psi_n'. \tag{6.3}$$

The inverse covariance matrix Σ^{-1} (6.3) is dominated by ξ_{min}, which may contain a numerical noise. As a consequence, the inverse matrix Σ^{-1} becomes sensitive to numerical noises.

In order to deal with this problem, we propose the following regularized covariance matrix Σ_R for the data covariance matrix Σ.

$$\Sigma_R = \Sigma + \alpha\Delta^2 I_N \qquad (0 < \alpha < 1) \tag{6.4a}$$

$$\Delta^2 = \mathrm{Tr}(\Sigma)/N, \tag{6.4b}$$

where I_N is an $(N \times N)$-dimensional identity matrix. The data variance Δ^2 satisfies the inequality

$$\xi_{max} \geq \Delta^2 = \frac{1}{N}\sum_{n=1}^{N} \xi_n \geq \xi_{max}/N. \tag{6.5}$$

The eigen values and eigen vectors of the regularized matrix Σ_R are given by $(\xi_n + \alpha\Delta^2)$ and ψ_n, respectively. From the inequality (6.5), it can be proved that the condition number of the regularized matrix Σ_R satisfies the condition:

$$\upsilon_R \geq \alpha/(N(1+\alpha)). \tag{6.6}$$

Then, the condition number of Σ_R is bounded below and it is controlled by the small constant α. The rate of change for the regularized eigen value is defined by

$$R(\xi_n) \equiv ((\xi_n + \alpha\Delta^2) - \xi_n)/\xi_n = \alpha(\Delta^2/\xi_n). \tag{6.7}$$

From the inequality (6.5), the ratio (6.7) satisfies the condition:

$$R(\xi_n) \geq R(\xi_{max}) \geq \alpha/N \tag{6.8a}$$

$$R(\xi_n) \leq R(\xi_{min}) \leq \alpha/v. \tag{6.8b}$$

If all the eigen values of Σ are the same order, i.e., $v \approx O(1)$, the rate of change becomes small, i.e., $R(\xi_n) \approx O(\alpha)$, so that the effect of the regularization is negligible. If the data covariance matrix Σ is singular, i.e., $v = 0$, the zero eigen value of Σ is replaced by $\alpha\Delta^2$. Thus, the regularized matrix Σ_R becomes regular and its condition number is bounded by (6.6). In general, the eigen values, which are larger than their average, are slightly affected, while the very small eigen values are changed so as to be nearly equal to $\alpha\Delta^2$.

By introducing a Bayes prior for a regularized matrix Σ_B as

$$P(\Sigma_B) = (\kappa/2)^N \exp\left(-(\kappa/2)\mathrm{Tr}(\Sigma_B^{-1})\right), \tag{6.9}$$

one can get a similar regularization equation

$$\Sigma_B = \Sigma + \kappa I_N, \tag{6.10}$$

where the regularization parameter κ is a given constant. However, it is rather difficult to determine the value of κ without knowledge of the data covariance matrix. It is especially difficult for the NGnet, since there are M independent local covariance matrices Σ_i $(i = 1, ..., M)$. The proposed method (6.4) automatically adjusts this constant by using the data variance Δ^2, which is easily calculated in an on-line manner.

If $\Sigma = 0$, Δ^2 becomes zero and the regularization (6.4) does not work. Therefore, if Δ^2 is smaller than a threshold value Δ^2_{min}, Δ^2 is set to Δ^2_{min}. This prevents Σ_R from being singular even when $\Sigma = 0$.

6.2 Regularization of on-line EM algorithm

Based on the consideration above, the i-th weighted covariance matrix is redefined in our on-line EM algorithm as:

$$\Sigma_i^{-1}(t) = \left[\left(\ll xx' \gg_i (t) - \mu_i(t)\mu_i'(t) \ll 1 \gg_i (t) + \alpha \ll \Delta_i^2 \gg_i (t) I_N\right) / \ll 1 \gg_i (t)\right]^{-1}, \tag{6.11}$$

where

$$\ll \Delta_i^2 \gg_i (t) \equiv \ll |x - \mu_i(t)|^2 \gg_i (t)/N = \left(\ll |x|^2 \gg_i (t) - |\mu_i(t)|^2 \ll 1 \gg_i (t)\right)/N. \tag{6.12}$$

The regularized Σ_i^{-1} (6.11) can be obtained from the relation (4.5) by using the following regularized $\tilde{\Lambda}_i$:

$$\tilde{\Lambda}_i(t) = \left(\ll \tilde{x}\tilde{x}' \gg_i (t) + \alpha \ll \Delta_i^2 \gg_i (t) \tilde{I}_N\right)^{-1}. \tag{6.13}$$

The $((N+1) \times (N+1))$-dimensional matrix, $\tilde{I}_N$, is defined by

$$\tilde{I}_N \equiv \begin{pmatrix} I_N & 0 \\ 0 & 0 \end{pmatrix} = \sum_{n=1}^{N} \tilde{e}_n \tilde{e}_n', \tag{6.14}$$

where $\tilde{e}_n$ is an $(N+1)$-dimensional unit vector; its n-th element is equal to 1 and the other elements are equal to 0. The regularization term in (6.13) can be calculated in an on-line manner. From the definition (6.12), $\ll \Delta_i^2 \gg_i (t)$ can be written as

$$\ll \Delta_i^2 \gg_i (t) = \ll \Delta_i^2 \gg_i (t-1) + \eta(t)(\Delta_i^2(t)P_i(t) - \ll \Delta_i^2 \gg_i (t-1)), \tag{6.15}$$

where $\Delta_i^2(t)$ is given by

$$\eta(t)P_i(t)\Delta_i^2(t) = \eta(t)P_i(t)|x(t) - \mu_i(t)|^2/N + (1-\eta(t))|\mu_i(t) - \mu_i(t-1)|^2 \ll 1 \gg_i (t-1)/N. \tag{6.16}$$

The second term on the right hand side in (6.16) comes from the difference between $\ll |x - \mu_i(t)|^2 \gg_i (t-1)$ and $\ll |x - \mu_i(t-1)|^2 \gg_i (t-1)$. Using (6.14) and (6.15), (6.13) can be rewritten as

$$\tilde{\Lambda}_i(t) = \left[(1-\eta(t))\tilde{\Lambda}_i^{-1}(t-1) + \eta(t)\left(\tilde{x}(t)\tilde{x}'(t) + \sum_{n=1}^{N} \tilde{\nu_n}(t)\tilde{\nu_n}'(t)\right) P_i(t)\right]^{-1} \tag{6.17a}$$

$$\tilde{\nu_n}(t) \equiv \sqrt{\alpha}\Delta_i(t)\tilde{e}_n. \tag{6.17b}$$

As a result, the regularized $\tilde{\Lambda}_i(t)$ (6.13) can be calculated as follows. For a given input data $x(t)$, $\tilde{\Lambda}_i(t)$ is calculated using the step-wise equation (4.4). After that, $\tilde{\Lambda}_i(t)$ is updated by

$$\tilde{\Lambda}_i(t) := \tilde{\Lambda}_i(t) - \frac{\eta(t)P_i(t)\tilde{\Lambda}_i(t)\tilde{\nu_n}(t)\tilde{\nu_n}'(t)\tilde{\Lambda}_i(t)}{1 + \eta(t)P_i(t)\tilde{\nu_n}'(t)\tilde{\Lambda}_i(t)\tilde{\nu_n}(t)}, \tag{6.18}$$

using the virtual data $\{\tilde{\nu_n}(t)|n = 1, ..., N\}$.

After calculating the regularized $\tilde{\Lambda}_i(t)$ (6.13), the linear regression matrix $\tilde{W}_i$ is obtained by using (4.6b), in which the regularized $\tilde{\Lambda}_i(t)$ is used. In the calculation of (4.6b), only the observed data $\{(x(t), y(t))|t = 1, 2, ...\}$ are used and the virtual data $\{\tilde{\nu_n}(t)|n = 1, ..., N; t = 1, 2, ...\}$, which have been used in the calculation of the regularized $\tilde{\Lambda}_i(t)$, must not be used. Therefore, equation (4.6a) does not hold in our regularization method. Since the regularized $\tilde{\Lambda}_i(t)$ is positive definite, the linear regression matrix $\tilde{W}_i$ at an equilibrium point of (4.6b) satisfies the condition

$$\tilde{W}_i E[\tilde{x}\tilde{x}'P(i|x, y, \theta)]_\rho = E[y\tilde{x}'P(i|x, y, \theta)]_\rho, \tag{6.19}$$

which is identical to the maximum likelihood equation, (3.7d) along with $\theta = \bar{\theta}$. Although the matrix $E[\tilde{x}\tilde{x}'P(i|x, y, \theta)]_\rho$ may not have the inverse, the relation (6.19) still has a meaning. Therefore, our method does not introduce a bias on the estimation of $\tilde{W}_i$.

Let us compare our regularization method, (4.6b) along with (6.13), to the Bayes prior method. By introducing a Bayes prior for $\tilde{W}_i$ as

$$P(\tilde{W}_i) = (\kappa/2\pi)^{D(N+1)/2} \exp\left(-(\kappa/2)\mathrm{Tr}(\tilde{W}_i'\tilde{W}_i)\right), \tag{6.20}$$

the regularized equation for $\tilde{W}_i$ is obtained:

$$\tilde{W}_i = < y\tilde{x}' >_i \left(\langle \tilde{x}\tilde{x}' \rangle_i + \kappa I_{N+1}\right)^{-1}. \tag{6.21}$$

This equation is a regularized version of the equation (4.6a) instead of the equation (4.6b). At an equilibrium point of the equation (6.21), $\tilde{W}_i$ satisfies

$$\tilde{W}_i \left(E[\tilde{x}\tilde{x}'P(i|x, y, \theta)]_\rho + \kappa I_{N+1}\right) = E[y\tilde{x}'P(i|x, y, \theta)]_\rho, \tag{6.22}$$

implying that this Bayes prior method introduces a bias on the estimation of $\tilde{W}_i$.

7 Unit manipulation

Since the EM algorithm only guarantees local optimality, the obtained estimator depends on its initial value. The initial allocation of the units in the input space is especially important for attaining a good approximation. If the initial allocation is quite different from the input distribution, much time is needed to achieve proper allocation. In order to overcome this difficulty, we introduce dynamic unit manipulation mechanisms, which are also effective for dealing with dynamic environments. These mechanisms are unit production, unit deletion, and unit division, and they are conducted in an on-line manner after observing each datum $(x(t), y(t))$.

- **Unit production**
  The probability $P(x(t), y(t), i \mid \theta(t-1))$ indicates how probable the i-th unit produces the datum $(x(t), y(t))$ with the present parameter $\theta(t-1)$. Let $0 < P_{produce} \ll 1/M$. When $\max_{i=1}^{M} P(x(t), y(t), i \mid \theta(t-1)) < P_{produce}$, the datum is too distant from the present units to be explained by the current stochastic model. In this case, a new unit is produced to account for the new datum. The initial parameters of the new unit are given by:

$$\mu_{M+1} = x(t) \tag{7.1a}$$
$$\Sigma_{M+1}^{-1} = \chi_{M+1}^{-2} I_N \qquad \chi_{M+1}^2 = \beta_1 \min_{i=1}^{M} |x(t) - \mu_i|^2 / N \tag{7.1b}$$
$$\sigma_{M+1}^2 = \beta_2 \max_{i=1}^{M} \sigma_i^2 \tag{7.1c}$$
$$\tilde{W}_{M+1} \equiv (W_{M+1}, b_{M+1}) = (0, y(t)), \tag{7.1d}$$

  where β_1 and β_2 are appropriate positive constants.

- **Unit deletion**
  The weighted mean $\ll 1 \gg_i (t)$, which is calculated by (4.2), indicates how much the i-th unit has been used to account for the data until t. Let $0 < P_{delete} \ll 1/M$. If $\ll 1 \gg_i (t) < P_{delete}$, the unit has rarely been used. In this case, the i-th unit is deleted.

- **Unit division**
  The unit error variance $\sigma_i^2(t)$ (4.3d) indicates the squared error between the i-th unit's prediction and the actual output. Let D_{divide} be a specific positive value. If $\sigma_i^2(t) > D_{divide}$, the unit's prediction is insufficient, probably because the partition in charge is too large to make a linear approximation. In this case, the i-th unit is divided into two units and the partition in charge is divided into two. The initial parameters of the two units are given by:

$$\mu_i(new) = \mu_i(old) + \beta_3 \sqrt{\xi_1} \psi_1 \qquad \mu_{M+1}(new) = \mu_i(old) - \beta_3 \sqrt{\xi_1} \psi_1 \tag{7.2a}$$
$$\Sigma_{M+1}^{-1}(new) = \Sigma_i^{-1}(new) = 4\xi_1^{-1} \psi_1 \psi_1' + \sum_{n=2}^{N} \xi_n^{-1} \psi_n \psi_n' \tag{7.2b}$$
$$\sigma_i^2(new) = \sigma_{M+1}^2(new) = \sigma_i^2(old)/2 \tag{7.2c}$$
$$\tilde{W}_i(new) = \tilde{W}_{M+1}(new) = \tilde{W}_i(old), \tag{7.2d}$$

  where ξ_n and ψ_n denote the eigen value and the eigen vector of the covariance matrix $\Sigma_i(old)$, respectively, and $\xi_1 = \xi_{max}$. β_3 is an appropriate positive constant.

Although similar unit manipulation mechanisms have been proposed in (Platt 1991; Schaal and Atkeson 1997), these mechanisms can be conducted with a probabilistic interpretation in our on-line EM algorithm.

Finally, we would like to comment on the extrapolation done by the NGnet after the learning phase. If an input x that is far from all the unit centers is given, the output of the NGnet is approximately $\tilde{W}_i\tilde{x}$, where i is the index of the closest unit to the input x. This implies that the output of the NGnet linearly diverges as $|x| \to \infty$ where the NGnet has never experienced the training data. A simple way to prevent this undesirable behavior is given as follows. If $\sum_{j=1}^{M} G_j(x) \leq G_{min}$ for a small threshold value G_{min}, the normalized Gaussian function $\mathcal{N}_i(x)$ defined by (2.1b) is replaced by $G_i(x)/G_{min}$. This prescription, however, has not been used in the following experiments, because the input spaces in those experiments are bounded.

8 Experiments

8.1 Function approximation in static environment

Applicability of our algorithm is investigated using the following function ($N = 2$ and $D = 1$), which was used by Schaal and Atkeson (1997).

$$y = \max\left\{e^{-10x_1^2}, e^{-50x_2^2}, 1.25e^{-5(x_1^2+x_2^2)}\right\} \qquad (-1 \leq x_1, x_2 \leq 1)\ . \tag{8.1}$$

Figure 1 shows the function shape.

By sampling the input variable vector, $x \equiv (x_1, x_2)$, uniformly from its domain, we prepared 500 input-output pairs, $\{(x(t), y(t))|t = 1, ..., 500\}$, as a training data set. A fairly large Gaussian noise $N(0, 0.1)$ is added to the outputs, where $N(0, 0.1)$ denotes a Gaussian distribution whose mean and standard deviation are 0 and 0.1, respectively. Figure 2 shows the function shape with the noise. The problem task is for the NGnet to approximate function (8.1) from the 500 noisy data. We prepared 41×41 mesh grids on the input domain, and the approximation accuracy was evaluated by means of the averaged squared error $nMSE$ on the grids (Schaal and Atkeson 1997). Here, $nMSE$ was normalized by the variance of the desired outputs (8.1). We compared the batch EM algorithm and the on-line EM algorithm. In this experiment, the discount factor $\lambda(t)$ was scheduled for approaching 1 as in (5.13). Figure 3 shows the time-series of $nMSE$. The abscissa denotes the learning epochs, and the 500 data points were supplied once in each epoch. The same training data set was used through the whole epochs. In the figure, we can see that both the batch EM algorithm and the on-line EM algorithm are able to approximate the target function in a small number of epochs. Although the so-called "overlearning" can be seen in both learning algorithms, it is more noticeable in the batch EM algorithm than in the on-line EM algorithm. The number of the units used in both algorithms was 50. In this task, the data distribution does not change over time. In this case, it can be thought that the batch learning is more suitable than the on-line learning because the batch learning can process the whole data at once. However, our on-line EM algorithm has ability similar to the batch algorithm. In addition, our on-line EM algorithm achieves a faster and more accurate approximation ability for this task than the RFWR (Receptive Field Weighted Regression) model proposed by Schaal and Atkeson (1997).

8.2 Function approximation in dynamic environments

The on-line EM algorithm is effective for a function approximation in a dynamic environment where the input-output distribution changes with time.

For an experiment, the distribution of the input variable x_1 in (8.1) was continuously changed in 500 epochs from the uniform distribution in the interval $[-1, -0.2]$ to that in the interval $[0.2, 1]$. At each epoch, the input variables were generated in the current domain. In the applications of the on-line EM algorithm for such dynamic environments, the learning behavior changes according to the scheduling of the discount factor $\lambda(t)$. When the discount factor is relatively small, the model tends to forget the past learning result, and to quickly adapt to the present input-output distribution. We show two typical behavioral patterns in the learning phase.

1. **Fast adaptation with forgetting past**
   In the first case, $\lambda(t)$ is initially set at a relatively small value and it is slowly increased, i.e., the effective learning coefficient $\eta(t)$ is relatively large throughout the experiment. The NGnet adapts to the input distribution change by means of relocation of the units' center. During the course of this relocation, the units' center moves to the new region, and consequently the past approximation in the old region is forgotten. Figures 4(a) and 4(b) show the center and the covariance (,i.e., two-dimensional display of the standard deviation) of all the units for the 50-th epoch and the 500-th epoch, respectively. The dots denote the 500 input points in each epoch. We can see that the NGnet adapts to the input distribution change by means of the drastic relocation of the units' center. Figure 5 shows the time-series of $nMSE$ on the current input region during the course of this learning task. We can see that the error does not grow large throughout the task. In this experiment, the number of the units does not change.

2. **Slow adaptation without forgetting past**
   In the second case, $\lambda(t)$ is rapidly increased, i.e., $\eta(t)$ rapidly approaches to zero. The NGnet adapts to the input distribution change without forgetting the past approximation result. In Figure 6, the solid, dashed, and dotted lines denote the time-series of $nMSE$ on the whole input domain, $nMSE$ on the current input domain at each epoch, and the number of units, respectively. Figure 7 shows the center and the covariance of all the units at the end of the learning task. Since the unit relocation is slow, the model adapts to the input distribution change mainly by the unit production mechanism. Consequently, the units in the region where no more input data appear remain, and the function approximation on the whole input domain is accurately maintained.

8.3 Singular input distribution

In order to evaluate our regularization method, we prepared a function approximation problem where the input variables are linearly dependent and there is an irrelevant variable. In such a case, the basic on-line EM algorithm without the regularization method obtains a poor result, because the input distribution becomes singular.

In this experiment, the output y is given by the same function as (8.1), while the input variables are given by $x \equiv (x_1, x_2, x_3, x_4, x_5) = (x_1, x_2, (x_1 + x_2)/2, (x_1 - x_2)/2, 0.1)$. When the

input data are generated, a Gaussian noise $N(0, 0.05)$ is added to x_3, x_4 and x_5. For a training set, we prepared 500 data, where x_1 and x_2 were uniformly taken from their domain, and the output y was added by a Gaussian noise $N(0, 0.05)$. For evaluation, a test set was prepared by setting x_1 and x_2 on the 41×41 mesh grids and the other variables were generated according to the above prescription. During the learning, the same training set was repeatedly supplied. In Figure 8, the solid, dashed, and dotted lines are the learning curves for $\alpha = 0.23$, $\alpha = 0.1$ and $\alpha = 0$, respectively. If no regularization method is employed ($\alpha = 0$), the error becomes large after the early learning stage. Comparing the cases of $\alpha = 0.23$ and $\alpha = 0.1$, the regularization effect seems fairly robust with respect to the parameter α value.

Let us consider another situation, where the input data distribute on a curved manifold. We consider a 3-D input space. Each input datum (x_1, x_2, x_3) is restricted on the unit sphere, i.e., $x_1^2 + x_2^2 + x_3^2 = 1$. A function defined on this unit sphere is given by

$$(x_1, x_2, x_3) = (\cos\theta_1 \cos\theta_2, \cos\theta_1 \sin\theta_2, \sin\theta_1) \tag{8.2a}$$

$$y = \cos(\theta_1)\cos(\theta_2/2) \max\left\{e^{-10(2\theta_1/\pi)^2}, e^{-50(\theta_2/\pi)^2}, 1.25e^{-5((2\theta_1/\pi)^2+(\theta_2/\pi)^2)}\right\}, \tag{8.2b}$$

where the range of the spherical coordinate is given by $-\pi/2 \leq \theta_1 \leq \pi/2$ and $-\pi \leq \theta_2 < \pi$. The output y does not include a noise. The function (8.2b) is similar to the function (8.1), but it is changed so as to satisfy the consistency of the spherical coordinate. We prepared 2000 data points uniformly on the sphere. In each learning epoch, the same data set is repeatedly used.

The covariance matrix of the whole input data is not singular. However, the covariance matrix of each local unit is nearly degenerate, so that the calculation of the inverse covariance matrix and the linear regression matrix may include a noise without the help of the regularization method. In figure 9, the solid and dashed lines are the learning curves for $\alpha = 0.023$ and $\alpha = 0$, respectively. If no regularization method is employed ($\alpha = 0$), the error does not become small. Note, however, that the calculation of the inverse covariance matrices is possible without the regularization, since the input data is not linearly degenerate. Our regularization method ($\alpha = 0.023$) provides a fairly good result compared to the basic method without the regularization ($\alpha = 0$). The condition numbers defined by (6.2) with the regularization and without the regularization are 0.0191 ± 0.0042 and 0.0071 ± 0.0038, respectively.

The local covariance matrix, Σ_i, of each unit represents the local input data distribution fairly well in our regularized method. It has two principal components which span the tangential plane to the unit sphere at each local unit position. The third eigen value is very small and it is bounded below by the regularization term. In figure 10, receptive fields of the local units are shown. The receptive field of each unit is defined by an ellipse whose axes correspond to the two principal components of the local covariance matrix Σ_i. The center of this ellipse is set at the center of the local unit, μ_i. One can see that the receptive fields appropriately cover the unit sphere. The average cosine between the eigen vectors, which corresponds to the third eigen values of the covariance matrices, and the spherical normal vectors was 0.9966 ± 0.0157. This implies that the receptive fields are almost tangential to the unit sphere.

8.4 Reinforcement learning

We apply our new approach to a reinforcement learning problem. The task is to swing the pendulum upward by a restricted torque controller and stabilize the pendulum near the up-

right position (Doya 1996). An actor-critic model proposed by Barto et al. (Barto, Sutton & Anderson, 1983) is used for the learning system. For the current state, $x_c(t)$, of the controlled system, the actor outputs an control signal (action) $u(t)$, which is given by the policy function $\Omega(x_c(t))$, i.e., $u(t) = \Omega(x_c(t))$. The controlled system changes its state to $x_c(t+1)$ after receiving the control signal $u(t)$. Following that, a reward $r(x_c(t+1))$ is given to the learning system. It is assumed that there is no knowledge of the controlled system.

The objective of the learning system is to find the optimal policy function $\Omega^*(x_c)$ that maximizes the discounted future return defined by

$$V(x_c) \equiv \sum_{t=0}^{\infty} \gamma^t r(x_c(t+1))\Big|_{x_c(0)=x_c}, \tag{8.3}$$

where $0 < \gamma < 1$ is a discount factor and $V(x_c)$ is called the value function. The value function $V(x_c)$ is defined for the current policy function $\Omega(x_c)$ employed by the actor.

The Q-function is defined by

$$Q(x_c, u) = \gamma V(x_c(t+1)) + r(x_c(t+1)), \tag{8.4}$$

where $x_c(t) = x_c$ and $u(t) = u$ are assumed. The value function can be obtained from the Q-function:

$$V(x_c) = Q(x_c, u = \Omega(x_c)). \tag{8.5}$$

The Q-function should satisfy the consistency condition

$$Q(x_c(t), u(t)) = \gamma Q(x_c(t+1), \Omega(x_c(t+1))) + r(x_c(t+1)). \tag{8.6}$$

The critic estimates the Q-function that satisfies the consistency condition (8.6). The Q-function is approximated by the NGnet, which is called the critic-network. The input to the critic-network is the current system state $x_c(t)$ and the control signal $u(t)$. The target output for the critic-network is given by the right hand side of (8.6), in which the Q-function is calculated using the current critic-network. After obtaining the new state $x_c(t+1)$, the parameters of the critic-network are updated using the on-line EM algorithm. This learning scheme is different from TD-learning (Sutton 1988) or Q-learning (Watkins 1989), because it directly uses the target Q-function value.

In the task for swinging up the pendulum, the control signal $u(t)$ represents a torque which is applied to the controlled system, i.e., the pendulum. The policy function is approximated by an actor-network, which is a variation of the Normalized Gaussian Network:

$$u = \Omega(x_c) = u_{max} \cdot \tanh\left(\sum_{i=1}^{M_0} \omega_i \mathcal{N}_i(x_c) + \epsilon\right), \tag{8.7}$$

where a random noise ϵ is added in the training phase in order to explore the state space. Since the maximal torque is fixed at u_{max}, the output of the actor-network is filtered through the sigmoidal function, $\tanh(\cdot)$. The centers of the units are fixed at the mesh grid points in the input space. The covariance matrices are also fixed to the univariate covariance matrices with the same variance. There is no linear term. Only the bias parameters ω are updated by the gradient ascent method so that the Q-function value increases (Sofge and White 1992):

$$\Delta\omega \propto \frac{\partial \Omega}{\partial \omega}(x_c(t)) \cdot \frac{\partial Q}{\partial u}(x_c(t), u(t)). \tag{8.8}$$

The reward for the inverted pendulum is given by

$$r(x_c) = \exp(-(\dot{\theta})^2/2\iota_1^2 - \theta^2/2\iota_2^2), \tag{8.9}$$

where θ and $\dot{\theta}$ denote the angle from the upright position and the angular velocity of the pendulum, respectively, i.e., $x_c \equiv (\dot{\theta}, \theta)$. $\iota_1 (= 3\pi/5)$ and $\iota_2 (= \pi/5)$ are constant values. The reward (8.9) encourages the pendulum to stay near the upright position.

After releasing the pendulum from a vicinity of the upright position, the control and the learning process of the actor-critic network is conducted for 7 seconds. This is a single episode. The reinforcement learning is done by repeating these episodes. As the learning proceeds, the initial position of the pendulum is gradually moved away from the upright position. In order to see the learning performance, we prepared three testbeds.

- Easy initial setting: $0 \leq |\dot{\theta}| \leq 3\pi/5, 0 \leq |\theta| \leq \pi/5$
- Medium initial setting: $0 \leq |\dot{\theta}| \leq 6\pi/5, \pi/5 \leq |\theta| \leq 2\pi/5$
- Difficult initial setting: $0 \leq |\dot{\theta}| \leq 9\pi/5, 2\pi/5 \leq |\theta| \leq 3\pi/5$

After each episode, the actor-critic network is tested under the above three testbeds. Figure 11 shows the time-series of the success rate for the three testbeds. A success is determined when the final reward is larger than 0.99. In order to achieve this reward value, the pendulum should stay near the upright position, because the reward (8.9) includes a penalty term for a large velocity. After about 100 episodes, the system is able to make the pendulum achieve an upright position for the easy initial setting. After this learning stage, the success rate for easy and medium initial settings slightly decreases, because the initial position at the training session moves away from the upright position. In this learning period, the system is mainly adapting to initial states fairly distant from the upright position. This adaptation is conducted by relocation of the critic-network units. After 350 episodes, the system is able to make the pendulum achieve an upright position from almost every initial state. Since the maximal torque generated by the controller is limited, the system inverts the pendulum after swinging it several times. According to our previous experiments, in which the critic-network is the NGnet trained by the gradient descent learning algorithm, a good control was obtained after about 2000 episodes. Therefore, our new approach based on the on-line EM algorithm is able to obtain a good control much faster than that based on the gradient descent algorithm.

9 Conclusion

In this article, we proposed a new on-line EM algorithm for the NGnet. We showed that the on-line EM algorithm is equivalent to the batch EM algorithm if a specific scheduling of the discount factor is employed. In addition, we showed that the on-line EM algorithm can be considered as a stochastic approximation method to find the maximum likelihood estimator.

A new regularization method was proposed in order to deal with a singular input distribution. In order to manage the dynamic environments, unit manipulation mechanisms such as unit production, unit deletion, and unit division were also introduced based on the probabilistic interpretation.

Experimental results showed that our approach is suitable for dynamical environments where the input-output distribution of data changes with time.

We also applied our on-line EM algorithm to a reinforcement learning problem. It has been shown that the NGnet, when using the on-line EM algorithm, learns the value function much faster than the method based on the gradient descent algorithm.

In forthcoming papers, we will discuss applications for the reinforcement learning in more detail.

References

[1] Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-13**, 834-846.

[2] Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, **39**, 1-22.

[3] Doya, K. (1996). Temporal difference learning in continuous time and space. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8* (pp. 1073-1079), Cambridge, MA: MIT Press.

[4] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, **3**, 79-87.

[5] Jordan, M. I., & Jacobs, R. A. (1994). Hierachical mixtures of experts and the EM algorithm. *Neual Computation*, **6**, 181-214.

[6] Jürgen, S. (1996). *Pattern Classification: A Unified View of Statistical and Neural Approaches*, New York: John Wiley & Sons.

[7] Kushner, H. J., & Yin, G. G. (1997). *Stochastic Approximation Algorithms and Applications*, New York: Springer-Verlag.

[8] Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, **1**, 281-294.

[9] Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, **3**, 213-225.

[10] Poggio, T., & Girosi, F. (1990). Networks for approximation and learning, *Proceedings of the IEEE*, **78**, 1481-1496.

[11] Schaal, S., & Atkeson, C. G. (1997). Constructive incremental learning from only local information. preprint.

[12] Sofge, D. A., & White, D. A. (1992). Applied learning - optimal control for manufacturing. In D. A. White & D. A. Sofge (Eds.), *Handbook of Intelligent Control* (pp. 259-282), New York: Van Nostrand Reinhold.

[13] Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, **3**, 9-44.

[14] Watkins, C. (1989). *Learning from Delayed Rewards*, Ph. D. thesis, Cambridge, England: Cambridge University.

[15] Xu, L., Jordan, M. I., & Hinton, G. E. (1995). An alternative model for mixtures of experts. In G. Tesauro, D. S. Touretzky, & T. K.Leen (Eds.), *Advances in Neural Information Processing Systems 7* (pp. 633-640), Cambridge, MA: MIT Press.

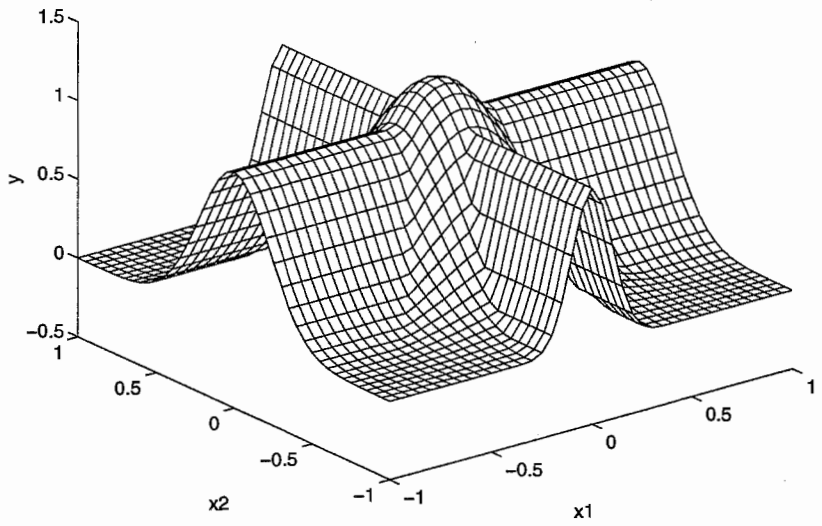


Figure 1 Function shape of equation (8.1).

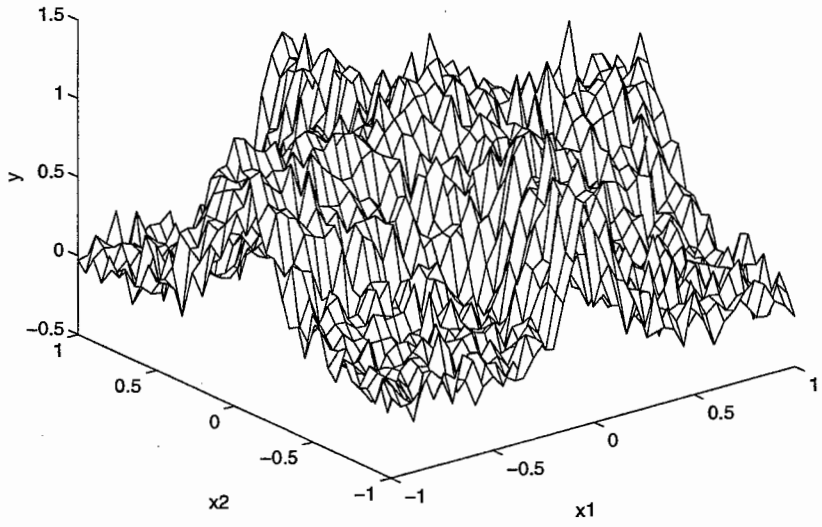


Figure 2 Function shape of equation (8.1) with a Gaussian noise whose mean and standard deviation are 0 and 0.1, respectively. This figure shows the noisy function values on the mesh grids, while the training data are randomly sampled from input domain. Therefore, this figure does not directly show the training data.

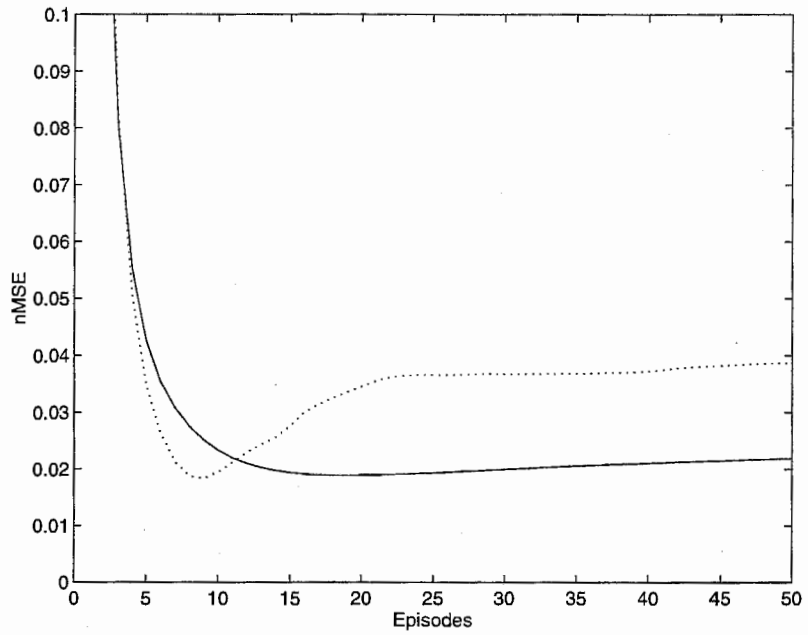


Figure 3 Learning processes of batch EM algorithm (dotted line) and on-line EM algorithm (solid line). 500 training data are supplied once in each epoch. Ordinate denotes squared error normalized by variance of desired outputs. Error is evaluated on 41×41 mesh grids on input domain.

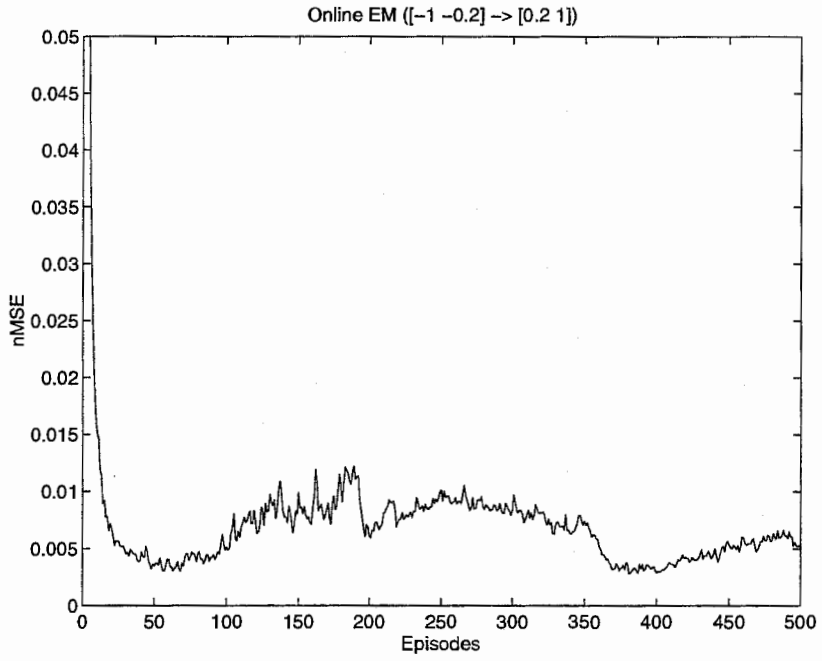
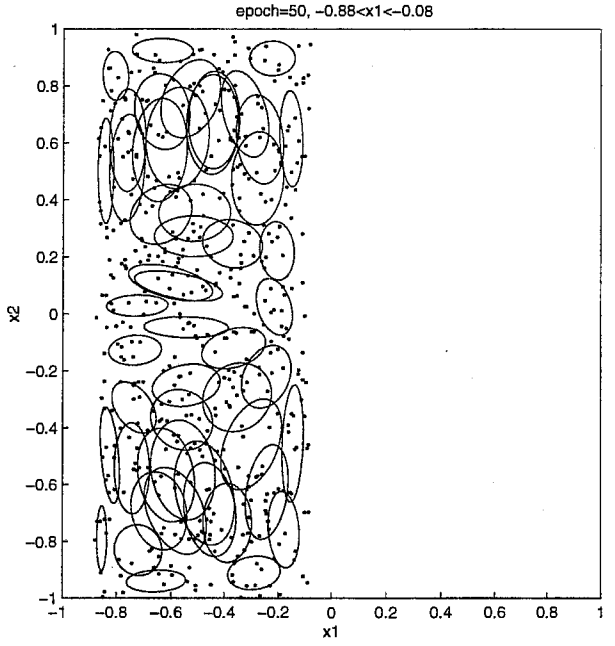
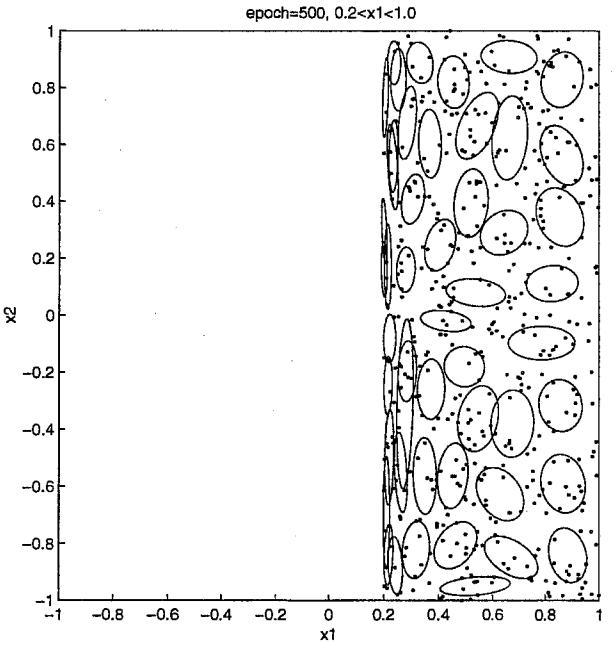


Figure 5 Learning process of on-line EM algorithm which quickly adapts to dynamic environment with forgetting past. Ordinate denotes $nMSE$ evaluated on current input domain at each epoch.



(a)



(b)

Figure 4 Dynamic relocation of the units for dynamic environment. Each ellipse denotes receptive field of a unit; ellipse corresponds to covariance (i.e., two-dimensional display of standard deviation). Ellipse center is set at center of the unit, μ_i. Dots denote 500 inputs in current epoch. (a) At 50-th epoch. (b) At 500-th epoch.

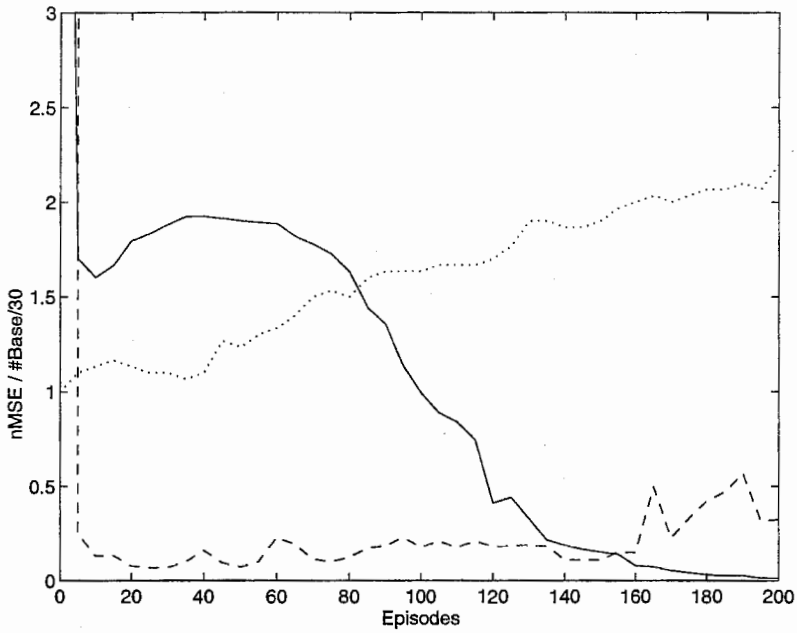


Figure 6 Learning process of on-line EM algorithm, which slowly adapts to dynamic environment without forgetting past. Solid, dashed, and dotted lines denote $nMSE$ on whole input domain, $nMSE$ on current input domain at each epoch, and number of units, respectively, (unit number is normalized by its initial value 30 for convenience).

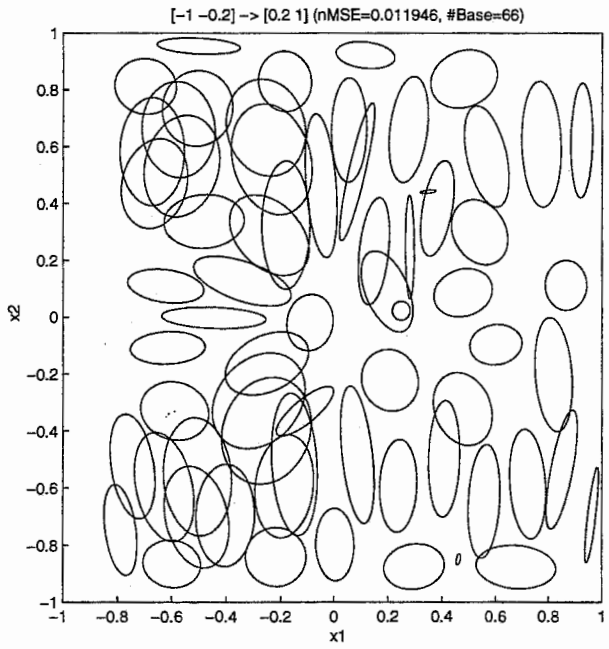


Figure 7 Receptive fields after learning without forgetting past. At end of the learning, the input data distribute in region: $0.2 \leq x_1 \leq 1$.

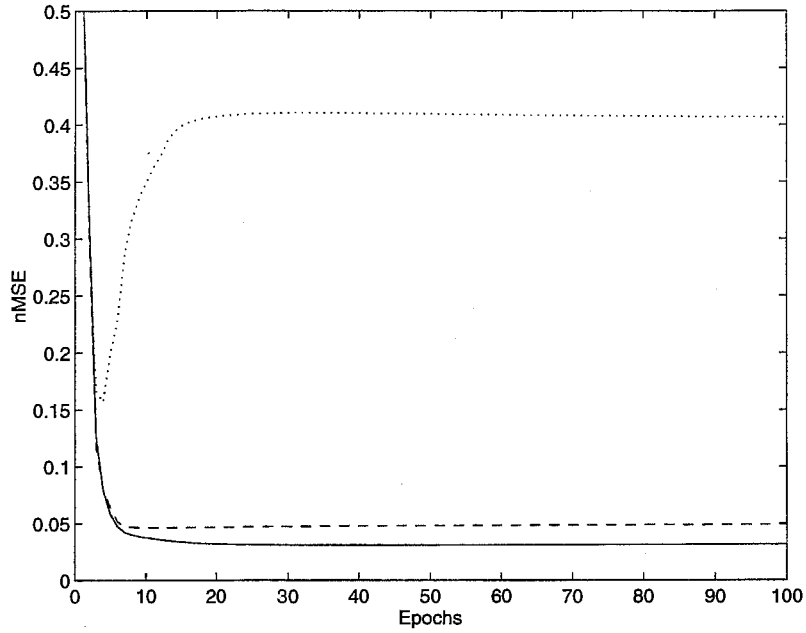


Figure 8 Learning processes of the case where the output is given by (8.1) for the input $x \equiv (x_1, x_2, x_3, x_4, x_5) = (x_1, x_2, (x_1 + x_2)/2 + N(0, 0.05), (x_1 - x_2)/2 + N(0, 0.05), N(0.1, 0.05))$ Solid, dashed, and dotted lines are the learning curves for $\alpha = 0.23$, $\alpha = 0.1$, and $\alpha = 0$, respectively.

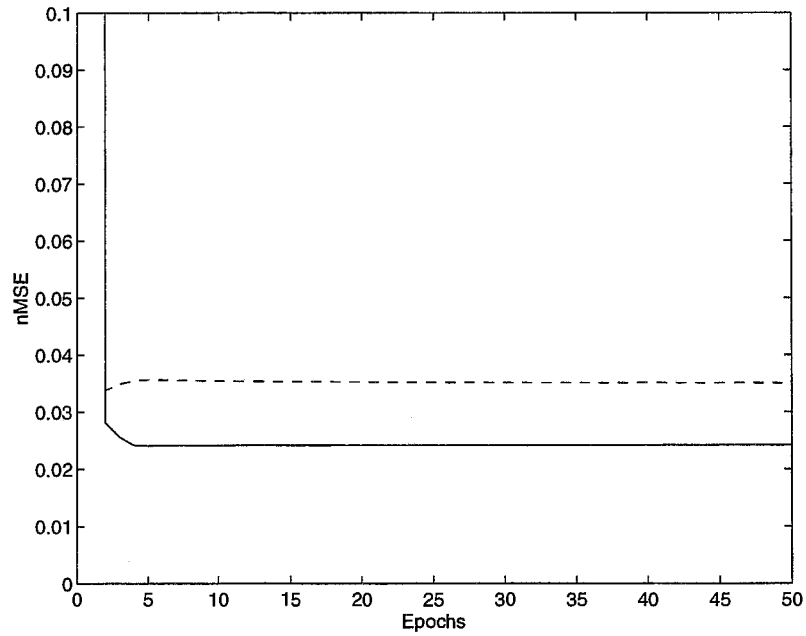


Figure 9 Learning processes for the function (8.2). Solid and dashed lines are the learning curves for the cases of $\alpha = 0.023$ and $\alpha = 0$, respectively.

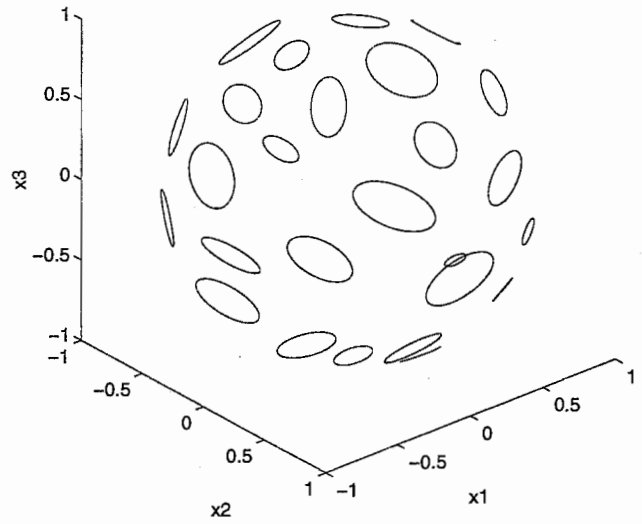


Figure 10 Receptive fields of the local units at the 50-th epoch after learning function (8.2). The receptive field of each unit is defined by an ellipse whose axes correspond to the two principal components of the local covariance matrix Σ_i. Ellipse center is set at the center of the local unit, μ_i. This figure shows the 3-D view of a hemisphere.

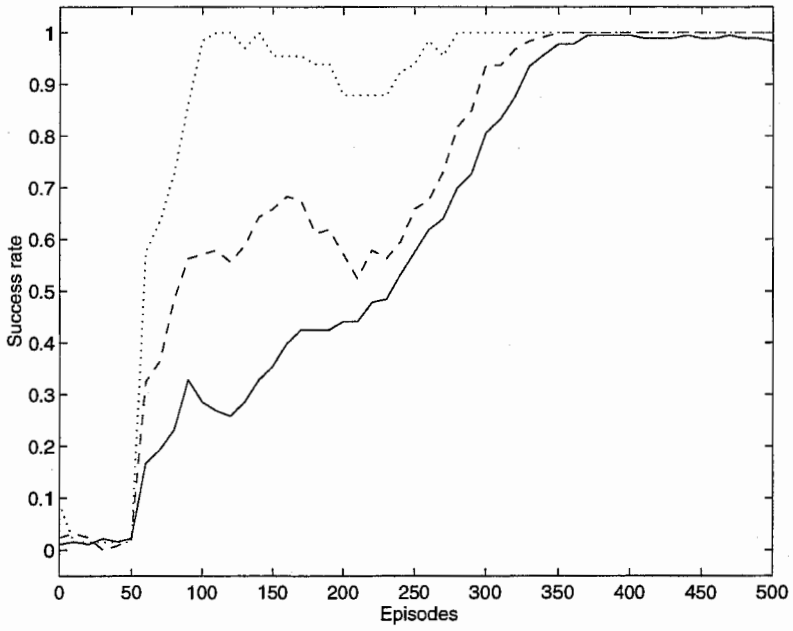


Figure 11 Time-series of success rate. Abscissa denotes number of episodes used for training the actor-critic network. Dotted, dashed and solid lines denote success rates for easy, medium and difficult initial settings, respectively.