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
TEL: (+81)-743-72-5251 FAX: (+81)-743-72-5259
E-mail: ishii@is.aist-nara.ac.jp

Abstract

A 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 over 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 dynamic environments. We also apply our on-line EM algorithm to robot dynamics problems and compare our algorithm with the Mixtures-of-Experts family.

1 Introduction

The main aim of this work is to develop a fast on-line learning method which can deal with dynamic environments. We use a Normalized Gaussian Network (NGnet) (Moody and Darken 1989) for this purpose. The NGnet 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 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 approximate 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. 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. Similar on-line EM algorithms have been proposed by Nowlan (1991), and Jordan and Jacobs (1994) for mixture models. However, there have been no convergence proof for these on-line EM algorithms. Neal and Hinton (1998) proposed a wide variety of incremental EM algorithms and proved the convergence of their algorithms. One drawback of their algorithms is that they either need additional storage variables for all the training data or need to see all the training data at each iteration. This prevents these algorithms from being applied to real time on-line settings where new data are supplied indefinitely each time.

For practical applications, important modifications to the on-line EM algorithm are necessary. 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 over time, unit manipulation mechanisms such as unit production, unit deletion, and unit division are also introduced based on the probabilistic interpretation.

Applicability of our new on-line EM algorithm is investigated using function approximation problems in three circumstances. The first one is a usual function approximation for a set of observed data. The second one is a function approximation in which the distribution of the input data changes over time. This experiment is performed to check the applicability of our approach to 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 works well even in this situation. Finally, we applied our on-line EM algorithm to realistic and difficult problems in robot dynamics. The performance of our on-line EM algorithm is compared with those of the Mixtures-of-Experts family.

2 Normalized Gaussian network

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.

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

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

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

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. $\tilde{W}_i \equiv (W_i, b_i)$ is a $(D \times (N+1))$-dimensional linear regression matrix, $\tilde{x}' \equiv (x', 1)$, and $\tilde{W}_i \tilde{x} \equiv W_i x + b_i$. 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.

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):

$$P(x, y, i|\theta) = (2\pi)^{-(D+N)/2} \sigma_i^{-D} |\Sigma_i|^{-1/2} M^{-1} \exp\left[-\frac{1}{2}(x-\mu_i)' \Sigma_i^{-1} (x-\mu_i) - \frac{1}{2\sigma_i^2}(y - \tilde{W}_i \tilde{x})^2 \right], \tag{2.2}$$

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 (2.2), the following probabilities are obtained.

$$P(i|\theta) = 1/M \tag{2.3a}$$

$$P(x|i,\theta) = G_i(x) \tag{2.3b}$$

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

These probabilities define a stochastic data generation model. First, a unit is randomly selected with an equal probability (2.3a). If the i-th unit is selected, the input x is generated according to the Gaussian distribution (2.3b). Given the unit index i and the input x, the output y is generated according to the Gaussian distribution (2.3c), 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

$$P(y|x,\theta) = \sum_{i=1}^{M} \mathcal{N}_i(x) P(y|x,i,\theta). \tag{2.4}$$

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

$$E[y|x] \equiv \int y P(y|x,\theta) dy = \sum_{i=1}^{M} \mathcal{N}_i(x) \tilde{W}_i \tilde{x}, \tag{2.5}$$

which is equivalent to the output of the NGnet (2.1a). Namely, the probability distribution (2.2) 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 (2.2) 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 (<u>E</u>stimation) 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 (<u>M</u>aximization) 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.

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

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

$$g_i \equiv E[P(i|x,y,\bar{\theta})]_\rho \tag{3.7a}$$
$$\mu_i = E[xP(i|x,y,\bar{\theta})]_\rho/g_i \tag{3.7b}$$
$$\Sigma_i = E[xx'P(i|x,y,\bar{\theta})]_\rho/g_i - \mu_i\mu_i' \tag{3.7c}$$
$$\tilde{W}_iE[\tilde{x}\tilde{x}'P(i|x,y,\bar{\theta})]_\rho = E[y\tilde{x}'P(i|x,y,\bar{\theta})]_\rho \tag{3.7d}$$
$$\sigma_i^2 = \frac{1}{D}\left(E[|y|^2P(i|x,y,\bar{\theta})]_\rho - \mathrm{Tr}(\tilde{W}_iE[\tilde{x}y'P(i|x,y,\bar{\theta})]_\rho)\right)/g_i, \tag{3.7e}$$

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

4.1 Basic 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

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

$$\eta(T) \equiv \left(\sum_{t=1}^{T} \prod_{s=t+1}^{T} \lambda(s) \right)^{-1} . \tag{4.1b}$$

Here, the parameter $\lambda(t)$ $(0 \le \lambda(t) \le 1)$ is a time-dependent 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:

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

$$\eta(t) = (1 + \lambda(t)/\eta(t-1))^{-1}, \tag{4.2b}$$

where $P_i(t) \equiv P(i|x(t), y(t), \theta(t-1))$. The constraint, $0 \le \lambda(t) \le 1$, gives the constraint on $\eta(t)$:

$$1 \ge \eta(t) \ge 1/t. \tag{4.3}$$

If this constraint is satisfied, the equation (4.2b) can be solved for $\lambda(t)$ as

$$\lambda(t) = \eta(t-1)(1/\eta(t) - 1). \tag{4.4}$$

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.

$$\mu_i(t) = \ll x \gg_i (t) / \ll 1 \gg_i (t) \tag{4.5a}$$

$$\Sigma_i^{-1}(t) = [\ll xx' \gg_i (t) / \ll 1 \gg_i (t) - \mu_i(t)\mu_i'(t)]^{-1} \tag{4.5b}$$

$$\tilde{W}_i(t) = \ll y\tilde{x}' \gg_i (t) [\ll \tilde{x}\tilde{x}' \gg_i (t)]^{-1} \tag{4.5c}$$

$$\sigma_i^2(t) = \frac{1}{D} \left[\ll |y|^2 \gg_i (t) - \mathrm{Tr} \left(\tilde{W}_i(t) \ll \tilde{x}y' \gg_i (t) \right) \right] / \ll 1 \gg_i (t). \tag{4.5d}$$

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 using a standard method (Jürgen 1996), so that there is no need to calculate the matrix inverse. 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:

$$\tilde{\Lambda}_i(t) = \frac{1}{1-\eta(t)} \left[\tilde{\Lambda}_i(t-1) - \frac{P_i(t)\tilde{\Lambda}_i(t-1)\tilde{x}(t)\tilde{x}'(t)\tilde{\Lambda}_i(t-1)}{(1/\eta(t) - 1) + P_i(t)\tilde{x}'(t)\tilde{\Lambda}_i(t-1)\tilde{x}(t)} \right]. \tag{4.6}$$

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

$$\tilde{\Lambda}_i(t) \ll 1 \gg_i (t) = \begin{pmatrix} \Sigma_i^{-1}(t) & -\Sigma_i^{-1}(t)\mu_i(t) \\ -\mu_i'(t)\Sigma_i^{-1}(t) & 1 + \mu_i'(t)\Sigma_i^{-1}(t)\mu_i(t) \end{pmatrix}. \tag{4.7}$$

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

$$\begin{aligned}\tilde{W}_i(t) &= \ll y\tilde{x}' \gg_i (t)\tilde{\Lambda}_i(t) & (4.8a)\\ &= \tilde{W}_i(t-1) + \eta(t)P_i(t)(y(t) - \tilde{W}_i(t-1)\tilde{x}(t))\tilde{x}'(t)\tilde{\Lambda}_i(t). & (4.8b)\end{aligned}$$

Thus, the basic on-line EM algorithm is summarized as follows. For the t-th 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 with the step-wise equation (4.6). Subsequently, the estimators for the model parameters are obtained by (4.5a), (4.5d), (4.7), and (4.8b).

4.2 Equivalence of on-line and batch EM algorithms

In this part, we show that the basic on-line EM algorithm defined by (3.1), (4.2) and (4.5) 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

$$\lambda(t) = \begin{cases} 0 & \text{if } t = (k-1)T + 1 \\ 1 & \text{otherwise} \end{cases} . \tag{4.9}$$

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 (4.9) is equivalent to the batch EM algorithm. Consequently, the estimator of the on-line EM algorithm converges to the maximum likelihood estimator. It should be noted that the step-wise equation (4.6) for $\tilde{\Lambda}_i(t)$ can not be used in this case, since the equation (4.6) becomes singular for $\eta(t) = 1$.

4.3 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 (4.5), can be written in an abstract form:

$$\begin{aligned}&\delta\phi(t) \equiv \phi(t) - \phi(t-1) = \eta(t)[F(x(t), y(t), \theta(t-1)) - \phi(t-1)] & (4.10a)\\ &\theta(t) = H(\phi(t)). & (4.10b)\end{aligned}$$

It can be easily proved that the set of equations:

$$\begin{aligned}&\phi = E[F(x, y, \theta)]_\rho & (4.11a)\\ &\theta = H(\phi) & (4.11b)\end{aligned}$$

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

$$\delta\phi(t) = \eta(t)(E[F(x,y,H(\phi(t-1)))]_\rho - \phi(t-1)) + \eta(t)\zeta(x(t),y(t),\phi(t-1)), \tag{4.12}$$

where the stochastic noise term ζ is defined by

$$\zeta(x(t),y(t),\phi(t-1)) \equiv F(x(t),y(t),H(\phi(t-1)) - E[F(x,y,H(\phi(t-1)))]_\rho, \tag{4.13}$$

and it satisfies

$$E[\zeta(x,y,\phi)]_\rho = 0. \tag{4.14}$$

The equation (4.12) has the same form as the Robbins-Monro stochastic approximation (Kushner and Yin 1997), which finds the maximum likelihood estimator given by (4.11). The effective learning coefficient $\eta(t)$ should satisfy the condition

$$\eta(t) \overset{t\to\infty}{\longrightarrow} 0, \quad \sum_{t=1}^{\infty}\eta(t) = \infty, \quad \sum_{t=1}^{\infty}\eta^2(t) < \infty. \tag{4.15}$$

Typically, $\eta(t)$, which satisfies the conditions (4.3) and (4.15), is given by

$$\eta(t) \overset{t\to\infty}{\longrightarrow} \frac{1}{at+b} \quad (1 > a > 0). \tag{4.16}$$

The corresponding discount factor is given by

$$\lambda(t) \overset{t\to\infty}{\longrightarrow} 1 - \frac{1-a}{at+(b-a)}, \tag{4.17}$$

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

For the convergence proof of the stochastic approximation (4.12), the boundedness of the noise variance is necessary (Kushner and Yin 1997). The noise variance is given by

$$E[\zeta(x,y,\phi)^2]_\rho = E[F(x,y,H(\phi))^2]_\rho - E[F(x,y,H(\phi))]_\rho^2. \tag{4.18}$$

Both terms on the right hand side in (4.18) 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 (4.12) 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 (4.12) 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 (4.5), which is equivalent to the basic on-line EM algorithm defined by (4.2), (4.5a), (4.5d), (4.6), (4.7), and (4.8b), if $\eta(t) \neq 1$. Therefore, the basic on-line EM algorithm is also a stochastic approximation.

5 Regularization

5.1 Regularization of 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 such as the introduction of Bayes priors, the singular value decomposition, the ridge regression, etc. However, these methods 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 eigenvalues and normalized eigenvectors are denoted by ξ_n (≥ 0) and ψ_n $(n = 1, ..., N)$, respectively. The set of the eigenvectors, $\{\psi_n | n = 1, ..., N\}$, forms a set of orthonormal bases. The condition number of the covariance matrix Σ is defined by

$$v \equiv \xi_{min}/\xi_{max}, \tag{5.1}$$

where ξ_{min} and ξ_{max} are the minimum and maximal eigenvalues of Σ, respectively. So that $0 \leq v \leq 1$. If $v = 0$, the covariance matrix Σ is singular. If $v \approx 1$, the covariance matrix Σ is regular and the calculation of its inverse is numerically stable. If $0 < v \ll 1$, the covariance matrix is regular and Σ^{-1} is given by $\Sigma^{-1} = \sum_{n=1}^{N} \xi_n^{-1} \psi_n \psi_n'$. It is dominated by ξ_{min}^{-1}, 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.

$$\Sigma_R = \Sigma + \alpha \Delta^2 I_N \qquad (0 < \alpha < 1) \tag{5.2a}$$

$$\Delta^2 = \mathrm{Tr}(\Sigma)/N, \tag{5.2b}$$

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

$$\xi_{max} \geq \Delta^2 = \frac{1}{N} \sum_{n=1}^{N} \xi_n \geq \xi_{max}/N. \tag{5.3}$$

The eigenvalues and eigenvectors of the regularized matrix Σ_R are given by $(\xi_n + \alpha\Delta^2)$ and ψ_n, respectively. From the inequality (5.3), it can be proved that the condition number of the regularized matrix Σ_R satisfies the condition:

$$v_R \geq \alpha/(N(1+\alpha)). \tag{5.4}$$

Then, the condition number of Σ_R is bounded below and controlled by the small constant α. The rate of change for the regularized eigenvalue is defined by $R(\xi_n) \equiv ((\xi_n + \alpha\Delta^2) - \xi_n)/\xi_n = \alpha(\Delta^2/\xi_n)$. From the inequality (5.3), this ratio satisfies the condition: $R(\xi_n) \geq R(\xi_{max}) \geq \alpha/N$ and $R(\xi_n) \leq R(\xi_{min}) \leq \alpha/v$. If all the eigenvalues 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 eigenvalue of Σ is replaced by $\alpha\Delta^2$. Thus, the regularized matrix Σ_R becomes regular and its condition number is bounded by (5.4). In general, the eigenvalues, which are larger than their average, are slightly affected, while the very small eigenvalues are changed so as to be nearly equal to $\alpha\Delta^2$. If $\Sigma = 0$, Δ^2 becomes zero and the regularization (5.2) 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 $\Sigma = 0$.

By introducing a Bayes prior for a regularized matrix Σ_B as $P(\Sigma_B) = (\kappa/2)^N \exp\left(-(\kappa/2)\mathrm{Tr}(\Sigma_B^{-1})\right)$, one can get a similar regularization equation

$$\Sigma_B = \Sigma + \kappa I_N, \tag{5.5}$$

where the regularization parameter κ is a given constant. However, it is rather difficult to determine the κ value without the 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 (5.2) automatically adjusts this constant by using the data variance Δ^2, which is easily calculated in an on-line manner.

5.2 Regularization of on-line EM algorithm

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

$$\Sigma_i^{-1}(t) = \left[\left(\ll xx' \gg_i (t) - \mu_i(t)\mu_i'(t) \ll 1 \gg_i (t) + \alpha \ll \Delta_i^2 \gg_i (t) I_N\right) / \ll 1 \gg_i (t)\right]^{-1}, \quad (5.6)$$

where

$$\ll \Delta_i^2 \gg_i (t) \equiv \ll |x - \mu_i(t)|^2 \gg_i (t)/N = \left(\ll |x|^2 \gg_i (t) - |\mu_i(t)|^2 \ll 1 \gg_i (t)\right)/N. \quad (5.7)$$

The regularized Σ_i^{-1} (5.6) can be obtained from the relation (4.7) by using the following regularized $\tilde{\Lambda}_i$:

$$\tilde{\Lambda}_i(t) = \left(\ll \tilde{x}\tilde{x}' \gg_i (t) + \alpha \ll \Delta_i^2 \gg_i (t) \tilde{I}_N\right)^{-1}. \quad (5.8)$$

The $((N+1) \times (N+1))$-dimensional matrix, $\tilde{I}_N$, is defined by $\tilde{I}_N \equiv \begin{pmatrix} I_N & 0 \\ 0 & 0 \end{pmatrix} = \sum_{n=1}^N \tilde{e}_n \tilde{e}_n'$, 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. From the definition (5.7), $\ll \Delta_i^2 \gg_i (t)$ can be written as

$$\ll \Delta_i^2 \gg_i (t) = \ll \Delta_i^2 \gg_i (t-1) + \eta(t)(\Delta_i^2(t) P_i(t) - \ll \Delta_i^2 \gg_i (t-1)), \quad (5.9)$$

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

$$\eta(t)P_i(t)\Delta_i^2(t) = \eta(t)P_i(t)|x(t) - \mu_i(t)|^2/N + (1-\eta(t))|\mu_i(t) - \mu_i(t-1)|^2 \ll 1 \gg_i (t-1)/N. \quad (5.10)$$

The second term on the right hand side in (5.10) 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 (5.9), (5.8) can be rewritten as

$$\tilde{\Lambda}_i(t) = \left[(1-\eta(t))\tilde{\Lambda}_i^{-1}(t-1) + \eta(t)\left(\tilde{x}(t)\tilde{x}'(t) + \sum_{n=1}^{N} \tilde{\nu_n}(t)\tilde{\nu_n}'(t)\right) P_i(t)\right]^{-1} \quad (5.11a)$$

$$\tilde{\nu_n}(t) \equiv \sqrt{\alpha}\Delta_i(t)\tilde{e}_n. \quad (5.11b)$$

As a result, the regularized $\tilde{\Lambda}_i(t)$ (5.8) can be calculated as follows. For a given input data $x(t)$, $\tilde{\Lambda}_i(t)$ is calculated using the step-wise equation (4.6). After that, $\tilde{\Lambda}_i(t)$ is updated by

$$\tilde{\Lambda}_i(t) := \tilde{\Lambda}_i(t) - \frac{\eta(t)P_i(t)\tilde{\Lambda}_i(t)\tilde{\nu_n}(t)\tilde{\nu_n}'(t)\tilde{\Lambda}_i(t)}{1 + \eta(t)P_i(t)\tilde{\nu_n}'(t)\tilde{\Lambda}_i(t)\tilde{\nu_n}(t)}, \quad (5.12)$$

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

After calculating the regularized $\tilde{\Lambda}_i(t)$ (5.8), the linear regression matrix $\tilde{W}_i$ is obtained by using (4.8b), in which the regularized $\tilde{\Lambda}_i(t)$ is used. Only the observed data $\{(x(t), y(t)) | t = 1, 2, ...\}$ are used in the calculation of (4.8b), 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 (4.8a) 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 (4.8b) satisfies the condition

$$\tilde{W}_i E[\tilde{x}\tilde{x}' P(i|x,y,\theta)]_\rho = E[y\tilde{x}' P(i|x,y,\theta)]_\rho, \tag{5.13}$$

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 (5.13) still has a meaning. Therefore, our method does not introduce a bias on the estimation of $\tilde{W}_i$.

6 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

$$\mu_{M+1} = x(t) \tag{6.1a}$$

$$\Sigma_{M+1}^{-1} = \chi_{M+1}^{-2} I_N \qquad \chi_{M+1}^2 = \beta_1 \min_{i=1}^M |x(t) - \mu_i|^2 / N \tag{6.1b}$$

$$\sigma_{M+1}^2 = \beta_2 \max_{i=1}^M \sigma_i^2 \tag{6.1c}$$

$$\tilde{W}_{M+1} \equiv (W_{M+1}, b_{M+1}) = (0, y(t)), \tag{6.1d}$$

  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 $\sigma_i^2(t)$ (4.5d) 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

$$\mu_i(new) = \mu_i(old) + \beta_3 \sqrt{\xi_1}\psi_1 \qquad \mu_{M+1}(new) = \mu_i(old) - \beta_3 \sqrt{\xi_1}\psi_1 \tag{6.2a}$$

$$\Sigma_{M+1}^{-1}(new) = \Sigma_i^{-1}(new) = 4\xi_1^{-1}\psi_1\psi_1' + \sum_{n=2}^N \xi_n^{-1}\psi_n\psi_n' \tag{6.2b}$$

$$\sigma_i^2(new) = \sigma_{M+1}^2(new) = \sigma_i^2(old)/2 \tag{6.2c}$$

$$\tilde{W}_i(new) = \tilde{W}_{M+1}(new) = \tilde{W}_i(old), \tag{6.2d}$$

where ξ_n and ψ_n denote the eigenvalue and the eigenvector 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.

7 Experiments

7.1 Function approximation in static environment

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

$$y = \max\left\{e^{-10x_1^2}, e^{-50x_2^2}, 1.25e^{-5(x_1^2+x_2^2)}\right\} \qquad (-1 \le x_1, x_2 \le 1)\ . \tag{7.1}$$

We prepared 500 input-output pairs, $\{(x(t), y(t))|t = 1, ..., 500\}$, as a training data set, by uniformly sampling the input variable vector, $x \equiv (x_1, x_2)$, from its domain. 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. The problem task is for the NGnet to approximate the function (7.1) from the 500 noisy data. We compared the batch EM algorithm and the on-line EM algorithm. In this experiment, the discount factor $\lambda(t)$ was scheduled to approach 1 as in (4.17). Figure 1 shows the time-series of the normalized mean squared error, $nMSE$, which is normalized by the variance of the desired outputs (7.1). 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. The data distribution does not change over time in this task. 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 an ability similar to the batch algorithm. In addition, the on-line EM algorithm achieves a faster and more accurate approximation for this task than the RFWR (Receptive Field Weighted Regression) model proposed by Schaal and Atkeson (1997).

7.2 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 over time. For an experiment, the distribution of the input variable x_1 in (7.1) 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 to 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.

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 a 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 2(b) and 2(c) show the center and the covariance of all the units for the 50-th epoch and the 500-th epoch, respectively. The NGnet adapts to the input distribution change by the drastic relocation of the units' center. Figure 2(a) shows that $nMSE$ on the current input region does not grow large throughout the task. In this experiment, the number of the units does not change.

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. Figure 3(a) shows this learning process. Figure 3(b) shows the center and the covariance of all the units at the end of the learning task. Since the unit relocation is slow, the NGnet 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.

7.3 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 (7.1), 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. The same training set was repeatedly supplied during the learning. In Figure 4(a), 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. Each input datum in the three dimensional space, (x_1, x_2, x_3), is restricted on the unit sphere, i.e., $(x_1, x_2, x_3) = (\cos\theta_1 \cos\theta_2, \cos\theta_1 \sin\theta_2, \sin\theta_1)$. A function defined on this unit sphere is given by

$$y = \cos(\theta_1)\cos(\theta_2/2)\max\left\{e^{-10(2\theta_1/\pi)^2}, e^{-50(\theta_2/\pi)^2}, 1.25e^{-5((2\theta_1/\pi)^2+(\theta_2/\pi)^2)}\right\}, \tag{7.2}$$

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 (7.2) is similar to the function (7.1), but it is changed so as to satisfy the consistency of the spherical coordinate. We prepared 2000 data points uniformly on the sphere. 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 4(b), 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. 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 (5.1) 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 eigenvalue is very small and it is bounded below by the regularization term. In figure 4(c), receptive fields of the local units are shown. One can see that the receptive fields appropriately cover the unit sphere.

7.4 Robot dynamics

Finally, we examine realistic and difficult problems. The problems are taken from DELVE (Rasmussen, Neal, Hinton, van Camp, Revow, Ghahramani, Kustra and Tibshirani 1996), which is a new standard collection of neural network benchmark problems. The "Pumadyn" datasets are a family of datasets synthetically generated from a realistic simulation of the dynamics of a Puma 560 robot arm (Ghahramani 1996; Corke 1996). The Puma robot has six revolving joints. The state vector is defined by $\{\theta_i, \dot{\theta}_i | i = 1, ..., 6\}$, where θ_i and $\dot{\theta}_i$ represent the angle and the angular velocity of the i-th joint, respectively. The applied torque to the i-th joint is denoted by τ_i. The task of the Pumadyn-8 family is to predict the third joint acceleration $\ddot{\theta}_3$ given the 8-dimensional input vector $x \equiv \{\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, \theta_3, \dot{\theta}_3, \tau_1, \tau_2\}$. Other variables are fixed to zero. The input data are randomly generated by sampling the input variables uniformly from the range $(|\theta|/\pi, |\dot{\theta}|/\pi, |\tau| \leq 0.6)$. The datasets are characterized by nonlinearity, noise level and size of training data. We examined six training datasets: "nm64", "nm256" and "nm1024", which are nonlinear and medium noisy datasets consisting of 64, 256 and 1024 training data, as well as "nh64", "nh256" and "nh1024", which are nonlinear and highly noisy datasets consisting of 64, 256 and 1024 training data. Gaussian noise with standard deviations of 0.12 and 0.4 were added to both the input and the output variables for the medium and the highly noisy datasets, respectively. The test dataset size was 1024 when the training dataset size was 1024, and 512 for the other cases.

We compared our on-line EM (OEM) algorithm with the Mixtures-of-Experts (ME) methods registered in DELVE (Waterhouse 1997) and the Moody-Darken (MD) method (Moody and Darken 1989). There are nine competitor methods,i.e., on-line EM algorithm (OEM), on-line EM algorithm with Neal early stopping (OEM-ese)(Neal 1998), mixtures-of-experts trained by ensemble learning (ME-el), hierarchical mixtures-of-experts trained by ensemble learning (HME-el), mixtures-of-experts trained by committee early stopping (ME-ese), hierarchical mixtures-of-experts trained by committee early stopping (HME-ese), hierarchical mixtures-of-experts grown via committee early stopping (HME-grow), Moody-darken (MD) method, and modified Moody-darken (MD-l) method.

The committee early stopping method uses a committee of (hierarchical) mixtures-of-experts models. The training data are randomly divided into a training set and a validation set, with 2/3 and 1/3 of the total data, respectively. The training is done for each member of the committee using different training/validation sets and terminated at the minimum of the validation error. The prediction is given by the average of the outputs for all committee members. The number of committee members depends on the training time. A minimum of 3 members and a maximum of 50 can be created. If the training time is greater than a given threshold, which is proportional to the amount of data, no more members can be created.

The HME-grow method constructs a hierarchical mixtures-of-experts from data, learning both the structure as well as the parameters of the model. In the growing process, one of the experts is split into two experts and a gate. The committee early stopping method is also used to control the size of the hierarchical trees.

The ensemble learning estimates the posterior probability distribution for the model parameters rather than estimates the optimal parameter value. The Gaussian prior distribution on the gate and the experts parameters are assumed to introduce hyper parameters. The optimal hyper parameters are estimated by an alternating minimization procedure that combines the standard EM algorithm for parameter estimation with re-estimation of hyper-parameters. The algorithm is computationally highly demanding.

The training of the OEM method is terminated at a prespecified number of epochs, which is 5 percent of the data size. The OEM-ese uses a simpler version of the committee early stopping proposed by Neal (1998). The training data is partitioned into four equal portions. The committee consists of four networks. Each network uses three of four portions as a training set and the remaining portion as a validation set.

The MD method uses two-stage learning. It first determines the unit centers according to the input distribution by using the k-means clustering algorithm. Subsequently, the bias coefficients b_i are determined by using the least mean squared error method while fixing the unit centers. The NGnet used in the original MD model has isometric covariance matrices and no linear coefficients, i.e., $W_i \equiv 0$. The MD-l method, in contrast, uses the NGnet with linear coefficients.

The main differences among the OEM, the ME and the MD methods are their criterion functions. The OEM method maximizes the log-likelihood of the joint probability distribution, $\sum_{t=1}^{T} \log P(x(t), y(t)|\theta)$. Therefore, the units are allocated according to the input-output data distribution. The ME method maximizes the log-likelihood of the conditional probability, $\sum_{t=1}^{T} \log P(y(t)|x(t), \theta)$. Therefore, the specialization process of each expert is done according to the output performance and does not take account of the input data distribution. The first stage learning of the MD method can be considered as the maximization of the log-likelihood of the input probability, $\sum_{t=1}^{T} \log P(x(t)|\theta)$. The second stage learning minimizes the mean squared error of the network output. Therefore, the optimality of the total algorithm is not guaranteed.

The simulation results are summarized in Fig 5. Table I shows the computation time needed for training by these algorithms. The results depend very much on the training size. The most striking differences can be seen for the nm256 dataset. The best performance is achieved by (H)ME-el and followed by OEM (-ese), (H)ME-ese, HME-grow and MD(-l) in that order. The performance of the ME family and the OEM family are almost the same for the nm64 and nh64 datasets while those of the MD family are slightly worse. For the nm1024 and nh1024 datasets, (H)ME-el and OEM(-ese) show comparable performances. The performances of (H)ME-ese and MD-l are slightly worse. HME-grow and MD each show a poor performance.

Some comments can be made on the above results. The amount of data is very small compared to the dimensionality of the input space, namely, the noisy data are sparsely distributed in a high dimensional space. Therefore, the reliable estimation of the optimal parameter is very difficult. This situation favors the ensemble learning. The disadvantage of the ensemble learning is that it requires a huge amount of time as shown in Table I. The OEM method achieves comparable learning ability with the (H)ME-el method within a significantly smaller amount of computation time for the 1024 datasets.

The original MD method showed a poor performance despite being very fast. The modified MD method with linear coefficients showed a better performance than the original MD method. However, its performance is worse than the OEM method's. The performance of the MD methods depended on the number of units. In Fig 5, the results of the best MD models are shown. For example, the best number of units for MD and MD-l are 140 and 33 for the nm1024 dataset, respectively. The OEM method automatically selects the number of units. The final numbers of units for the nm1024, nm256 and nm64 datasets are 6, 5 and 1, respectively, although the initial number of units is set to 2 percent of the data size.

We considered a more realistic situation. For real-time robot control, the training data are generated along the robot trajectory. Therefore, we generated the training data by freely operating the Puma robot from the upright position. Because of the inertia, the robot exhibited chaotic oscillations. The task was to predict the third joint acceleration $\ddot{\theta}_3$ given the 10-dimensional input vector $x \equiv \{\theta_i, \dot{\theta}_i | i = 1, ..., 5\}$. Since θ_6 and $\dot{\theta}_6$ were almost zero throughout the trajectory, we omitted these variables and the applied torque was zero. The size of training data was 7000 (the ME family could not operate more than 7000 data in our computer). The 7000 test data were generated in the neighborhood of the trajectory. The results are summarized in Table II. The OEM method achieves a better performance than the ME family with a significantly smaller amount of time. The performance of the MD method is worse than that of the (H)ME-ese method.

The computation times of the OEM, the MD and the MD-l methods scale as $(N+1)^2 M$, M^3, and $((N+1)M)^3$, respectively, where M is the number of units and N is the input dimension. The best performance of MD(-l) was obtained with 420 (50) units, while the final number of units in OEM became 230 after 4 epochs. When the number of units in MD(-l) was larger than 1000 (100), the MD(-l) method became slower than the OEM method and the performance was degraded.

In summary, we confirmed that the on-line EM algorithm achieves a good performance in a short amount of time even if the amount of data becomes large.

Table I

Computation time comparison for nine methods. Used dataset was "nm1024" with 1024 training data. Computer used was Silicon Graphics Octane (CPU: R10000 175MHz $\times$ 2).

Method	ME-el	HME-el	OEM-ese	OEM	ME-ese	HME-ese	HME-grow	MD-l	MD
Time (sec)	16500	23580	395	40	2129	2127	2059	16	5

Table II

Learning results for robot trajectory datasets. Computation time and $nMSE$ for six methods are shown. Ensemble learning, (H)ME-el, does not end within a week.

Method	(H)ME-el	OEM	ME-ese	HME-ese	HME-grow	MD-l	MD
nMSE (%)	-	1.8	2.3	2.8	22.2	3.9	4.4
Time (sec)	-	1085	84600	63600	36780	220	130

8 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 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 dynamic environments where the input-output distribution of data changes over time. We also applied our on-line EM algorithm to robot dynamics problems. Our algorithm achieved a comparable learning ability to the Mixtures-of-Experts (ME) methods for a small amount of data, and achieved a better performance than the ME methods for a large amount of data within a significantly smaller amount of computation time. Our method can also be applied to reinforcement learning (Sato & Ishii 1998) and the reconstruction of nonlinear dynamics (Ishii & Sato 1998).

References

[1] Corke, P.I. (1996). A Robotics Toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, **3**, 24-32.

[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] Ghahramani, Z. (1996). The pumadyn dataset. http://www.cs.toronto.edu/~delve/data/pumadyn/pumadyn.ps.gz.

[4] Ishii, S. & Sato, M. (1998). On-line EM algorithm and reconstruction of chaotic dynamics. in Proceedings of NNSP'98.

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

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

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

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

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

[10] Neal, R. M. (1998). Assessing relevance determination methods using DELVE. In C. M. Bishop (Ed.), *Generalization in Neural Networks and Machine Learning* (pp. 97-129). Berlin: Springer-Verlag.

[11] Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in Graphical Models* (pp. 355-368). Norwell, MA:Kluwer Academic Press.

[12] Nowlan, S. J. (1991). Soft competitive adaptation: neural network learning algorithms based on fitting statistical mixtures. *CMU Technical Report*, **CS-91-126**, Pittsburgh: Carnegie Mellon University.

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

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

[15] Rasmussen, C. E., Neal, R. M., Hinton, G. E., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., & Tibshirani, R. (1996). The DELVE manual. ftp://ftp.cs.utoronto.ca/pub/neuron/delve/doc/manual.ps.gz.

[16] Sato, M., & Ishii, S. (1999). Reinforcement learning based on on-line EM algorithm, In M. S. Kearns, S. A. Solla, and D. A. Cohn, (Eds.), *Advances in Neural Information Processing Systems 11* (pp. 1052-1058). Cambridge, MA: MIT Press.

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

[18] Waterhouse S.R. (1997). *Classification and Regression using Mixtures of Experts.* PhD Thesis, Cambridge University.

[19] 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.

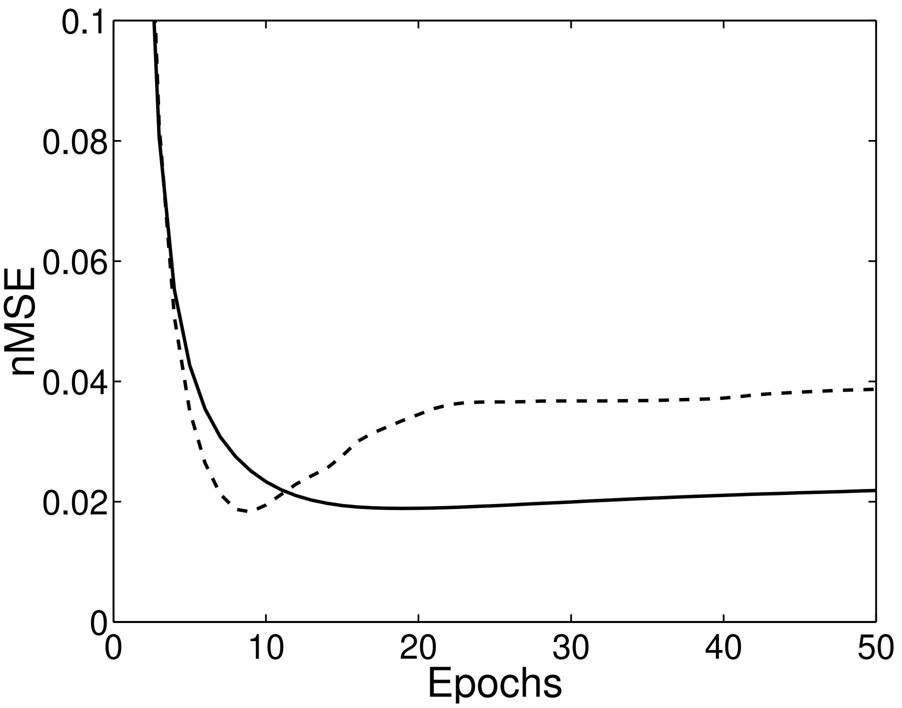
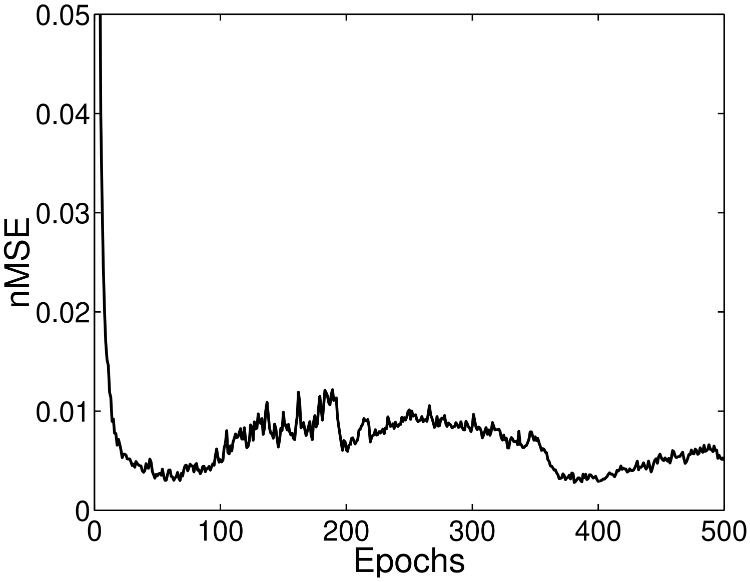
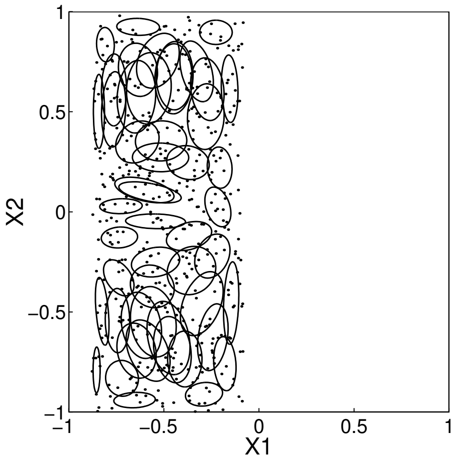


Figure 1

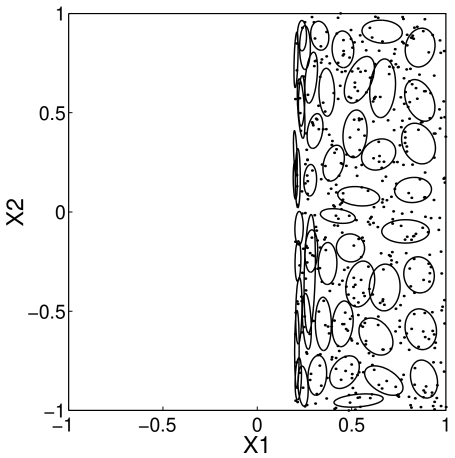
Learning processes of batch EM algorithm (dotted line) and on-line EM algorithm (solid). 500 training data are supplied once in each epoch. Ordinate denotes $nMSE$, i.e., mean squared error normalized by variance of desired outputs. Error is evaluated on 41×41 mesh grids on input domain.



(a)



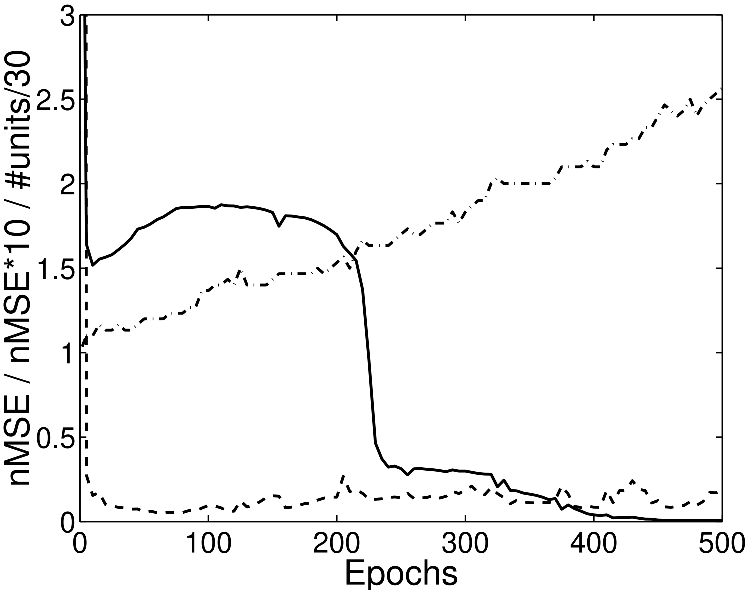
(b)



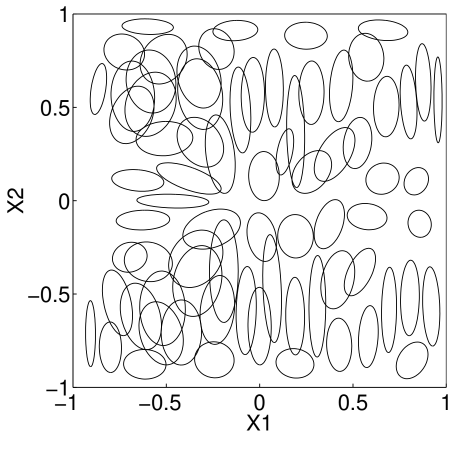
(c)

Figure 2

(a) Learning process of on-line EM algorithm that quickly adapts to dynamic environment with forgetting past. Ordinate denotes $nMSE$ evaluated on current input domain at each epoch. (b) and (c): Dynamic unit relocation for dynamic environment. Each ellipse denotes receptive field of a unit; ellipse corresponds to covariance (two-dimensional display of standard deviation). Ellipse center is set at unit center, μ_i. Dots denote 500 inputs in current epoch. (b) At 50-th epoch. (c) At 500-th epoch.



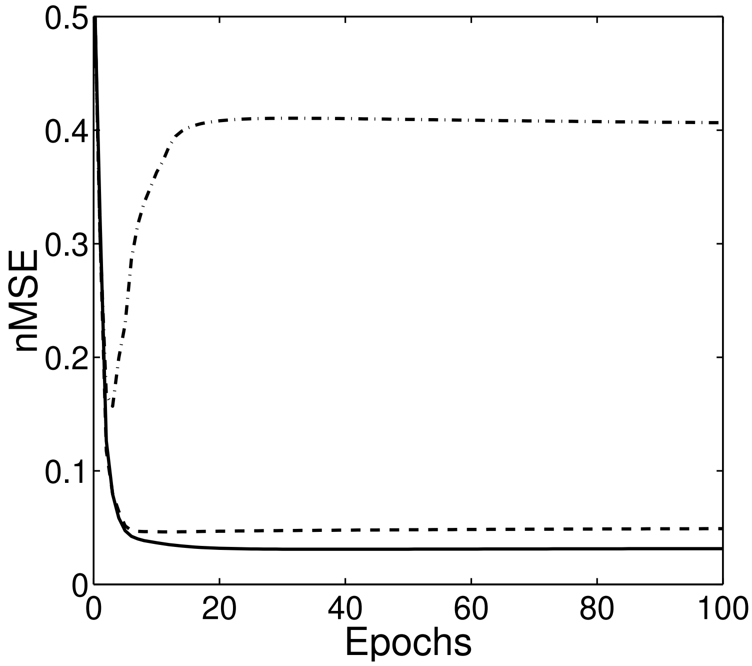
(a)



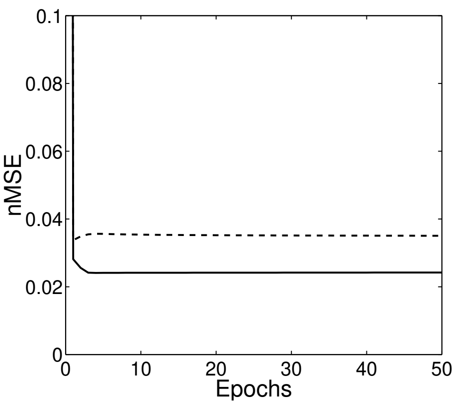
(b)

Figure 3

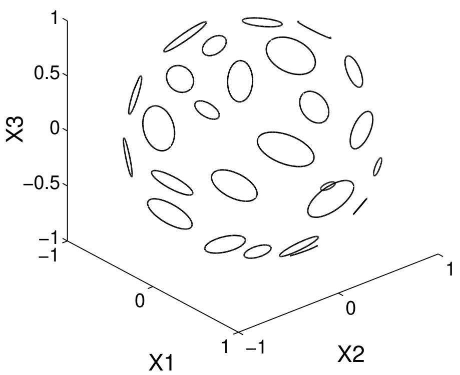
(a) Learning process of on-line EM algorithm that slowly adapts to dynamic environment without forgetting past. Solid line: $nMSE$ on whole input domain, dashed line: $nMSE$ on current input domain at each epoch, dotted line: number of units, which is normalized by its initial value 30 for convenience. (b) Receptive fields after learning.



(a)



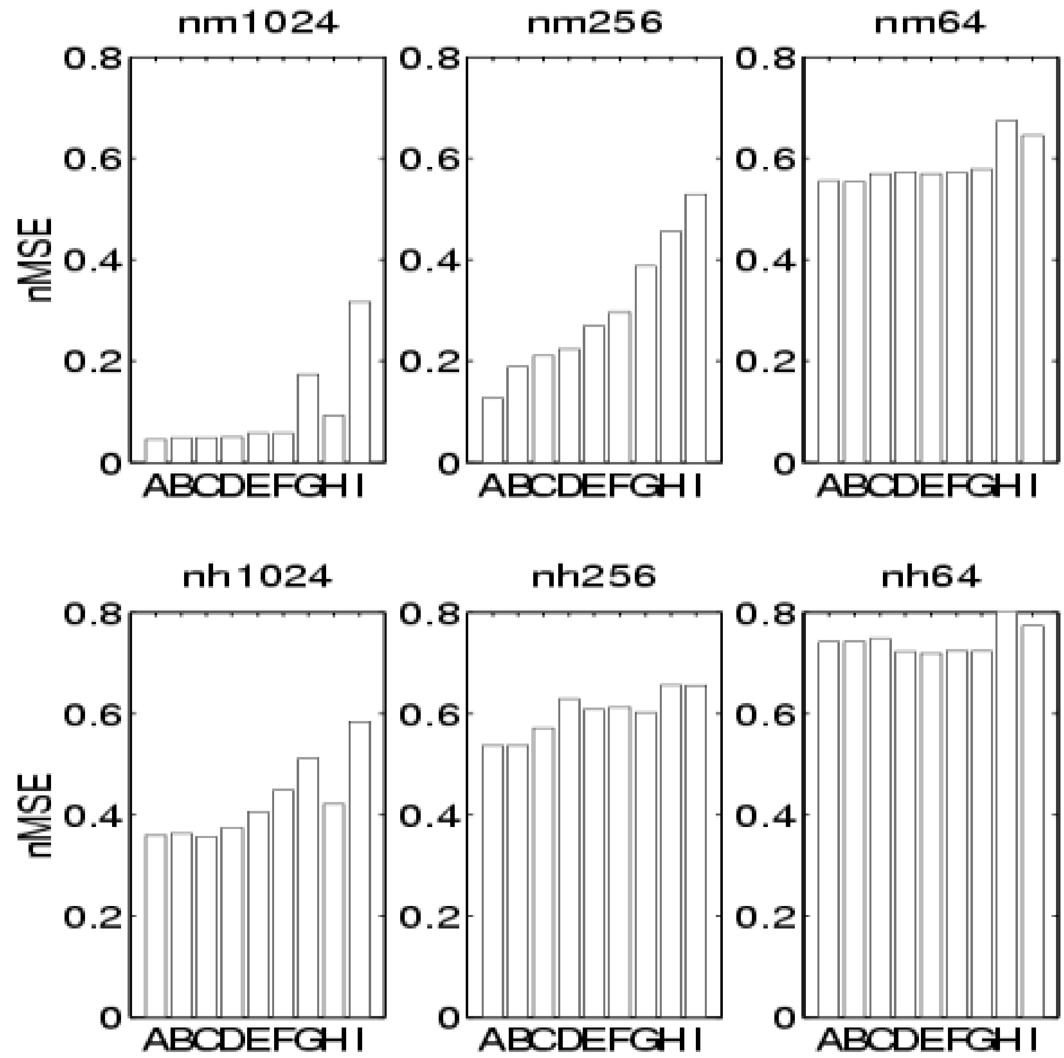
(b)



(c)

Figure 4

(a) Learning processes when output is given by (7.1) for input $x \equiv (x_1, x_2, x_3, x_4, x_5) = (x_1, x_2, (x_1 + x_2)/2 + N(0, 0.05), (x_1 - x_2)/2 + N(0, 0.05), N(0.1, 0.05))$. Learning curves for $\alpha = 0.23$ (solid line), $\alpha = 0.1$ (dashed), and $\alpha = 0$ (dotted). (b) Learning curves for $\alpha = 0.023$ (solid) and $\alpha = 0$ (dashed). Output is given by (7.2). (c) Receptive fields at 50-th epoch, each of which is defined by an ellipse with axes for two principal components of local covariance matrix Σ_i. Ellipse center is set at unit center, μ_i. This figure shows 3-D view of a hemisphere.



Label	A	B	C	D	E
Method	ME-el	HME-el	OEM-ese	OEM	ME-ese
Label	F	G	H	I	
Method	HME-ese	HME-grow	MD-1	MD	

Figure 5

Learning results for "Pumadyn" datasets: "nm64", "nm256", "nm1024", "nh64", "nh256" and "nh1024". Bars represent the expected $nMSE$,which is calculated by an average of $nMSE$ for different test datasets, for nine methods shown at the bottom of the figure.