A Recurrent Log-Linearized Gaussian Mixture Network

Toshio Tsuji, *Member, IEEE*, Nan Bu, Osamu Fukuda, and Makoto Kaneko, *Senior Member, IEEE*

***Abstract*—Context in time series is one of the most useful and interesting characteristics for machine learning. In some cases, the dynamic characteristic would be the only basis for achieving a possible classification. A novel neural network, which is named "a recurrent log-linearized Gaussian mixture network (R-LLGMN)," is proposed in this paper for classification of time series. The structure of this network is based on a hidden Markov model (HMM), which has been well developed in the area of speech recognition. R-LLGMN can as well be interpreted as an extension of a probabilistic neural network using a log-linearized Gaussian mixture model, in which recurrent connections have been incorporated to make temporal information in use. Some simulation experiments are carried out to compare R-LLGMN with the traditional estimator of HMM as classifiers, and finally, pattern classification experiments for EEG signals are conducted. It is indicated from these experiments that R-LLGMN can successfully classify not only artificial data but real biological data such as EEG signals.**

***Index Terms*—EEG, Gaussian mixture model, hidden Markov model (HMM), log-linearized model, neural networks (NNs), pattern classification, recurrent neural networks (RNNs).**

I. Introduction

ALTHOUGH pattern classification has been one of the most actively researched fields for some years, even now various investigations are carried out to attain higher classification performance. The pattern classification problem is a kind of decision-making problem that can be described as follows: Given an input vector X (or input series X $[t]$) and several output classes which are known *a priori*, decide to which class the input belongs. In other words, it is a problem of a mapping input vectors into output classes. Therefore, the essential point to consider in order to achieve high classification performance is how to estimate the mapping for classification from given data.

The 1980s witnessed the resurgence of neural networks (NNs), and the so-called backpropagation NNs [1] were shown to be capable of representing any nonlinear mapping using nonlinear transducers and layers with variable sizes. Inspired by this, the backpropagation NN was recognized as one of the most attractive principles for learning classifiers. In principle, NN can solve the classification problem by determining weights even in an extremely high-dimensional space, so it was considered that all the characteristics were *learned* automatically through minimization, or search for the global minimum, of the output error (cost) function [2]. Some drawbacks, however, have been pointed out, which can be summarized as follows.

1) NN needs a large amount of training data.
2) A large-scale network structure is necessary.
3) To achieve good convergence, it takes too many learning iterations.
4) There are likely to be many local minima for the learning of NNs.

In order to deal with these problems, a number of observations have been made following investigation into integrating domain/task specific knowledge into the architecture of NN, since the generic NN does not have any mechanisms for incorporating any additional knowledge which can place constraints on NN. This kind of NN can be named as model-based neural networks (MNNs) [2]. It extends NN functionality to include more explicit constraints on network geometry and connection weights. Therefore, it is possible to construct networks that respond to intrinsic features of the input data that are known *a priori*. Consequently, the problem becomes much easier, and this may allow the reduction of the network dimensionality and the learning difficulties.

In the meantime, many researchers have studied Bayesian classifiers, which can deal with pattern classification by the estimation of probability density function (pdf). The pioneering work by Richard and Lippman [3] demonstrated that outputs of NNs, if estimated accurately, could estimate Bayesian *a posteriori* probabilities. Then, by replacing the sigmoid activation function often used in NNs with an exponential function, the probabilistic neural network (PNN) was developed [4]. For realization of PNN, the following three approaches have been suggested: parametric, nonparametric, and semiparametric. For parametric approaches, a specific type of pdf is assumed for each event. The neural network is constructed by transforming this statistical model, and each component of the NN has specific interpretation [5]. As for nonparametric techniques, such as those described in published material, the pdf can be approximated by simply summing up small multivariate Gaussian distributions centered at each training sample point [4].

The semiparametric estimation of the pdf, having a flexible structure that can represent any distribution and include a set of parameters for particular distributions, is considered as one of the most successful classifiers. The unknown distribution is defined as a weighted sum of a number of component distributions (e.g., Gaussian distribution). The pdf of input patterns can be calculated from this mixture model. The development of NNs based on the Gaussian mixture model (GMM) that uses Gaussian component densities has been carried out in tandem:

Manuscript received March 21, 2001; revised November 9, 2001 and August 6, 2002. This work was supported in part by the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

T. Tsuji, N. Bu, and M. Kaneko are with the Department of Artificial Complex Systems Engineering, Hiroshima University, Higashi-Hiroshima 739-8527, Japan (e-mail: tsuji@bsys.hiroshima-u.ac.jp).

O. Fukuda is with the National Institute of Advanced Industrial Science and Technology, Tsukuba 305-8564, Japan.

Digital Object Identifier 10.1109/TNN.2003.809403

Tråvén [6], Perlovsky and McManus [7], Tsuji *et al.* [8], [9], Lee and Shimoji [10], Streit and Luginbuhl [11] and Bishop [12]. Particularly, Tsuji *et al.* proposed an NN, a log-linearized Gaussian mixture network (LLGMN), which estimates the pdf based on the GMM and a log-linear model [9]. The weight coefficients of LLGMN include the parameters of the log-linearized GMM that are the nonlinear combination of the GMM parameters, such as the mixture coefficients, mean values, and standard deviations of each component. In addition, these weights are trained in the same manner as the error backpropagation rule. LLGMN is successfully applied to the EMG pattern classification [13], where six motions of forearm and hand were classified using EMG signal measured from several pairs of electrodes. However, because this NN is based on a *static* model, it does not take context of time into consideration. In dealing with signals of dynamic characteristics, the classification results of LLGMN could lack consistency. In order to deal with this problem and to obtain a higher classification rate, it is necessary to develop a *dynamic* NN.

Unfortunately, the structure of such feedforward NNs is not appropriate for processing temporal sequences in practice. There are two main reasons, namely, 1) it is difficult to store past internal states and 2) they treat each input pattern as independent events. Addressing these problems, many researchers introduced recurrent neural networks (RNNs) [1], [14], [15] into the field of pattern classification.

It was Hopfield who first claimed an NN with feedback connection in 1982 [16]. Later, he showed the ability of the Hopfield NN, providing a solution to the "Traveling-Salesman Problem" (TSP), for which the computational difficulty has been much studied. The Hopfield NN is a highly connected network, but it is usually not necessary to feed the overall network output back into the input layer. In some other cases where the multilayer perceptron (MLP) [17] is used, recurrent connection can be made between the hidden layers to encapsulate information. Also, Lin *et al.* [18] claimed that by embedding the delay memory in the RNN architecture, the NN could use the information at previous steps, and the NN would be made less prone to the problem of long-term dependency learning. Recently, there have been many publications showing that RNN has been successfully used to learn various temporal sequences and applied to temporal pattern recognition. Petrosian *et al.* [19] addressed the successful RNN for predicting the onset of epileptic seizure. In the work of Aussem [20], a dynamical recurrent neural networks (DRNN) is used for time series prediction and modeling of small dynamical systems. Zhang *et al.* [21] proposed the mixed order locally recurrent neural networks to build long-term prediction models for nonlinear processes. In this RNN, the output of a hidden neuron is fed back to its input through several units with time delay, and different hidden neurons can have different numbers of feedbacks. Schittenkopf *et al.* [22] extended the mixture density networks (MDNs) [12] in a *recurrent* way to take into account the previous conditional variances as in the GARCH framework.

In this paper, we propose a novel NN, a recurrent log-linearized Gaussian mixture network (R-LLGMN), which is based on the algorithm of hidden Markov model (HMM). This network can also be regarded as an extension of LLGMN, introducing recurrent connections into LLGMN. Using GMM, the *a posteriori* probability can be estimated and, simultaneously, the recurrent connection makes use of available information on the time context. The weight coefficients of R-LLGMN correspond to the nonlinear combination of the HMM parameters, such as the mixing coefficient, mean vector, covariance matrix and transition probability. The weight coefficients of R-LLGMN, however, has no constraints as the parameter in the statistical model. Therefore, the representation ability of R-LLGMN should be higher than that of HMM, and R-LLGMN is expected to have better performance in the case of temporal pattern classification.

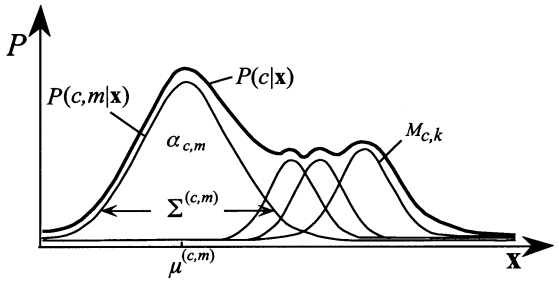


Fig. 1. Gaussian mixture model.

This paper is organized as follows: Section II gives a brief introduction on LLGMN. HMM as well as the algorithm and architecture of R-LLGMN is described in Section III. The results of computer simulation and pattern classification experiments of the EEG are presented in Sections IV and V, respectively. Section VI concludes the paper.

II. LLGMN

A. Log-Linearized Gaussian Mixture Model (LLGMM)

In terms of classifying an observed vector $\mathbf{x}$ into one of the C given classes, the *a posteriori* probability $P(c|\mathbf{x})$ is examined, and the class with the highest one is determined according to the Bayes' rule. In the LLGMM, the pdf of class c $(c = 1, \ldots, C)$ is approximated with a GMM as shown in Fig. 1 [9], and it can be described as follows:

$$\begin{aligned} P(c|\mathbf{x}) &= \sum_{m=1}^{M_c} P(c, m|\mathbf{x}) = \sum_{m=1}^{M_c} \frac{P(c, m)P(\mathbf{x}|c, m)}{P(\mathbf{x})} \\ &= \sum_{m=1}^{M_c} \frac{P(c, m)P(\mathbf{x}|c, m)}{\sum_{c'=1}^{C}\sum_{m'=1}^{M_c} P(c', m')P(\mathbf{x}|c', m')} \\ &= \sum_{m=1}^{M_c} \frac{\alpha_{c,m} g\left(\mathbf{x}; \boldsymbol{\mu}^{(c,m)}, \Sigma^{(c,m)}\right)}{\sum_{c'=1}^{C}\sum_{m'=1}^{M_c} \alpha_{c',m'} g\left(\mathbf{x}; \boldsymbol{\mu}^{(c',m')}, \Sigma^{(c',m')}\right)} \end{aligned} \tag{1}$$

where M_c $(c = 1, \ldots, C)$ denotes the number of components of class c, $\alpha_{c,m} \equiv P(c, m)$ is the *a priori* probability (or the mixture coefficient) for each component $\{c, m\}$. $P(\mathbf{x}|c, m)$ is the probability for $\mathbf{x}$ to be generated from the component m in class c, which is expressed using $g(\mathbf{x}; \boldsymbol{\mu}^{(c,m)}, \Sigma^{(c,m)})$, the

d-dimensional Gaussian distribution, with mean vector $\boldsymbol{\mu}^{(c,m)}$ and covariance matrix $\Sigma^{(c,m)}$ of each component.

Extending $g(\mathbf{x}; \boldsymbol{\mu}^{(c,m)}, \Sigma^{(c,m)})$ with the mean vector $\boldsymbol{\mu}^{(c,m)} = (\mu_1^{(c,m)}, \ldots, \mu_d^{(c,m)})^{\mathrm{T}}$ and the inverse of the convariance matrix $\Sigma^{(c,m)-1} = [s_{ij}^{(c,m)}]$, the numerator in (1) is in the form as

$$\begin{aligned}\alpha_{c,m} g\left(\mathbf{x}; \boldsymbol{\mu}^{(c,m)}, \Sigma^{(c,m)}\right) &= \alpha_{c,m}(2\pi)^{-(d/2)} \left|\Sigma^{(c,m)}\right|^{-(1/2)} \\ &\times \exp\left[-\tfrac{1}{2}\sum_{j=1}^{d}\sum_{l=1}^{j}(2-\delta_{jl}) s_{jl}^{(c,m)} x_j x_l \right. \\ &\quad + \sum_{j=1}^{d}\sum_{l=1}^{d} s_{jl}^{(c,m)} \mu_j^{(c,m)} x_l \\ &\quad \left. - \tfrac{1}{2}\sum_{j=1}^{d}\sum_{l=1}^{d} s_{jl}^{(c,m)} \mu_j^{(c,m)} \mu_l^{(c,m)}\right] \end{aligned} \quad (2)$$

where x_i, $i = 1, 2, \ldots, d$, is the elements of $\mathbf{x}$ and δ_{jl} is the Kronecker delta: $\delta_{jl} = 1$ when $i = j$ and $\delta_{jl} = 0$ otherwise.

Applying a *log-linearization* process to (2), we get

$$\xi_{c,m} \triangleq \log \alpha_{c,m} g\left(\mathbf{x}; \boldsymbol{\mu}^{(c,m)}, \Sigma^{(c,m)}\right) = \boldsymbol{\beta}^{(c,m)^{\mathrm{T}}} \mathbf{X} \quad (3)$$

where $\mathbf{X} \in \Re^H$ and $\boldsymbol{\beta}^{(c,m)} \in \Re^H$ are defined as

$$\begin{aligned}\mathbf{X} = (1, \mathbf{x}^{\mathrm{T}}, {x_1}^2, x_1x_2, \ldots, x_1x_d, {x_2}^2, x_2x_3, \ldots \\ x_2x_d, \ldots, {x_d}^2)^{\mathrm{T}} \end{aligned} \quad (4)$$

$$\begin{aligned}\boldsymbol{\beta}^{(c,m)} = \left(\beta_0^{(c,m)}, \sum_{j=1}^{d} s_{j1}^{(c,m)} \mu_j^{(c,m)}, \ldots, \sum_{j=1}^{d} s_{jd}^{(c,m)} \mu_j^{(c,m)} \right. \\ -\tfrac{1}{2} s_{11}^{(c,m)}, -s_{12}^{(c,m)}, \ldots, s_{1d}^{(c,m)}, \ldots \\ \left. -\tfrac{1}{2}(2-\delta_{jl}) s_{jl}^{(c,m)}, \ldots, -\tfrac{1}{2} s_{dd}^{(c,m)}\right)^{\mathrm{T}} \end{aligned} \quad (5)$$

$$\begin{aligned}\beta_0^{(c,m)} = -\frac{1}{2}\sum_{j=1}^{d}\sum_{l=1}^{d} s_{jl}^{(c,m)} \mu_j^{(c,m)} \mu_l^{(c,m)} - \frac{d}{2}\log 2\pi \\ - \tfrac{1}{2}\log\left|\Sigma^{(c,m)}\right| + \log \alpha_{c,m} \end{aligned} \quad (6)$$

and the dimension H is defined as $H = 1 + d(d+3)/2$. Thus, $\xi_{c,m}$ is expressed as the product of the coefficient vector $\boldsymbol{\beta}^{(c,m)}$ and the modified input vector $\mathbf{X}$. To remove the effect of $\boldsymbol{\beta}^{(c,m)}$, which is due to the statistical constrains of GMM, a new variable $Y_{c,m}$ and coefficient vector $\mathbf{w}^{(c,m)}$ are introduced as

$$\begin{aligned} Y_{c,m} &\equiv \xi_{c,m} - \xi_{C,M_C} \\ &= \left(\boldsymbol{\beta}^{(c,m)} - \boldsymbol{\beta}^{(C,M_C)}\right)^{\mathrm{T}} \mathbf{X} = \mathbf{w}^{(c,m)^{\mathrm{T}}} \mathbf{X}. \end{aligned} \quad (7)$$

The coefficient vector $\mathbf{w}^{(c,m)}$ is defined as the difference between $\boldsymbol{\beta}^{(c,m)}$ and $\boldsymbol{\beta}^{(C,M_C)}$, and $\mathbf{w}^{(C,M_C)} = 0$. Using the coefficient vetor $\mathbf{w}^{(c,m)}$ as the weight coefficients, the model described above is transformed to a feedforward NN, that is, LLGMN.

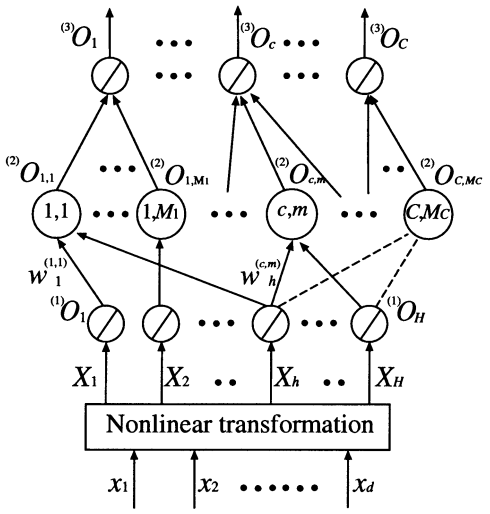


Fig. 2. Structure of LLGMN.

B. NN Structure

By applying the log-linear model, GMM is incorporated into the three-layer feedforward NN shown in Fig. 2. Also, a simple learning algorithm based on the backpropagation is employed [9].

In the preprocess, the input vector $\mathbf{x}$ is converted into the modified vector $\mathbf{X}$, according to (4). The first layer consists of H units corresponding to the dimension of $\mathbf{X}$ and the identity function is used for activation of each unit. ${}^{(1)}O_h$ denotes the output of the hth unit in the first layer.

In the second layer, each unit receives the output of the first layer weighted by the coefficient $w_h^{(c,m)}$ and outputs the *a posteriori* probability of each component. The relationships between the input of unit $\{c, m\}$ in the second layer (${}^{(2)}I_{c,m}$) and the output (${}^{(2)}O_{c,m}$) are defined as

$${}^{(2)}I_{c,m} = \sum_{h=1}^{H} {}^{(1)}O_h w_h^{(c,m)} \quad (8)$$

$${}^{(2)}O_{c,m} = \frac{\exp\left[{}^{(2)}I_{c,m}\right]}{\sum_{c'=1}^{C}\sum_{m'=1}^{M_C}\exp\left[{}^{(2)}I_{c',m'}\right]} \quad (9)$$

where $w_h^{(C,M_C)} = 0$ $(h = 1, \ldots, H)$.

Finally, the third layer consists of C units corresponding to the number of classes and outputs the *a posteriori* probability for class c $(c = 1, \ldots, C)$. The unit c integrates the outputs of M_c units $\{c, m\}$ $(m = 1, \ldots, M_c)$ in the second layer. The function between the input and the output is described as

$${}^{(3)}O_c = {}^{(3)}I_c = \sum_{m=1}^{M_c} {}^{(2)}O_{c,m} \quad (10)$$

where the output ${}^{(3)}O_c$ of the last layer corresponds to the *a posteriori* probability of class c.

Although LLGMN is based on a static model in which the characteristics of the pdf do not alter through time, it achieves high performance in classification by incorporating the statistical structure in the network. Since the weight coefficients have no constraints and are mutually independent, the learning

process is made flexible and the *answer* can be searched within a much larger space. However, for time series signals, LLGMN does not achieve a sufficient classification performance. To overcome this difficulty, it is necessary to develop some techniques to incorporate a *dynamic* statistical model into the NN.

III. R-LLGMN

A. HMM

As for temporal classification, the HMM [26], [27] is a well-developed technology, which has been successfully used most particularly in the domain of speech recognition [27].

A Markov process is a stochastic process for which probability distribution of the present state in a sequence is a function of model parameters and the previous state, and is independent of all history prior to that. The HMM assumes that a Markov process can only be observed via another stochastic process which produces a sequence of *observations* or outputs resulting from the underlying Markov process. Therefore, a complete specification of an HMM requires two model parameters N and M, which denote the numbers of states and observations, respectively, and three probability matrices $\mathbf{A}$ (state transition probability), $\mathbf{B}$ (observation probability) and $\boldsymbol{\pi}$ (initial state probability). As for classification we need one HMM for each class c, then the probability of the particular stream $\tilde{\mathbf{x}} = (\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(T))$ we observed, $P(\tilde{\mathbf{x}}|c)$, is computed using the probability matrices $\mathbf{A}$, $\mathbf{B}$ and $\boldsymbol{\pi}$, finally the most probable one is chosen. The computation of probability is made by employing a forward-algorithm [27]. The Baum–Welth algorithm [26], [27] or equivalently the EM (expectation-modification) method [28], provides a way of estimation of the probabilities of $\mathbf{A}$, $\mathbf{B}$ and $\boldsymbol{\pi}$ from training data.

On the other hand, a continuous density HMM (CDHMM) was introduced for continuous signals (or vectors) in many practical problems. The observation probability matrix $\mathbf{B}$ is replaced by continuous probability density function, usually a Gaussian density is used in CDHMM. Since Gaussian mixture density can be used to approximate any continuous probability density function, the modeling ability of the hidden Markov processes has, thus, been greatly enhanced. Baum *et al.* [29] extended the Baum–Welth algorithm to CDHMM, with some limitations. Juang *et al.* [30] further expanded the estimation algorithm, and their group has applied it to speech recognition.

The HMM have been proved to be very effective in practice, producing high levels of classification accuracy. However, the structure of HMM is not always known *a priori*, which depends on problems. In the field of acoustic speech recognition (ASR), for example, the *a priori* model topology (e.g., a left-to-right HMM) is chosen to ease the computation. Sometimes there are complex tradeoffs that have to be made between model complexity and the difficulty of training. Also both the discrete HMM and the CDHMM consist of many parameters, so that the estimation process is usually very sensitive to initialization. Rabiner *et al.* [31] combined a segmental *k*-means procedure to initialize estimates of model parameters, and then the Baum–Welth algorithm is used as a "model refinement tool." Furthermore, a large quantity of training data are required to train HMM, which does not result in good adaptability.

Because of the desirable properties of NNs, the combined architecture of HMM and NNs (a so called hybrid HMM/NN) has widely spread in the field of ASR. The hybrid HMM/NN can be divided into two types. In the first type, the standard framework of HMM is kept intact, but the observation probabilities are computed by an NN. Bourlard and Wellekens [32] as well as Cohen *et al.* [33] provided such methods using MLPs, while Robinson [34], Mitchell *et al.* [35], and Ström [36] developed this kind of architectures with the RNNs. It is easy to imagine that if NN just plays a part in the work of HMM, then the system consequently gets to be complex, as does the training algorithm as well. Some of the weaknesses of HMM still remain. Alternatively, Bridle [37] proposed a "Alpha-Net" that treats the *forward-algorithm* computation as a recurrent network. In his study, the Alpha computation of HMM is considered as a network, so all the parameters in HMM are transformed into the parameters in the network, and they can be modified with the NNs training method. However, the Alpha-Net just develops HMM formally to a NN architecture, namely it is just a *copy* of HMM. In the rest of this section, we will give the description of the R-LLGMN, and it can be regarded as an NN which introduces a log-linear Gaussian mixture model into CDHMM.

B. Log-Linearized CDHMM

Let us consider a kind of CDHMM, which is shown in Fig. 3, where there are C classes in this model and the class c ($c \in \{1, \ldots, C\}$) is composed of K_c states. The observation probability of state k in class c is approximated with Gaussian mixture model. The system undergoes a change of state (possibly back to the same state) in each class. Suppose that, for a time series $\mathbf{x}(t) \in \Re^d$ ($t = 1, \ldots, T$), at any time $\mathbf{x}(t)$ must occur from one state k of class c in the model, where $k \in \{1, \ldots, K_c\}$.

According to this model, given a time series $\tilde{\mathbf{x}}$, the *a posteriori* probability for class c, $P(c|\tilde{\mathbf{x}})$, is derived as

$$P(c|\tilde{\mathbf{x}}) = \sum_{k=1}^{K_c} P(c, k|\tilde{\mathbf{x}}) = \sum_{k=1}^{K_c} \frac{\alpha_T^c(k)}{\sum_{c'=1}^{C} \sum_{k'=1}^{K_{c'}} \alpha_T^{c'}(k')} \tag{11}$$

viz. summation of the *a posteriori* probabilities for all the state k in class c. Here, $\alpha_t^c(k)$ is the forward variable, which is defined as the probability for partial time series $(\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(t))$ to be generated from class c and input vector $\mathbf{x}(t)$ occurs from state k in class c. According to the forward algorithm, $\alpha_T^c(k)$ can be computed as follows:

$$\alpha_1^c(k) = \pi_k^c b_k^c(\mathbf{x}(1)) \tag{12}$$

$$\alpha_t^c(k) = \sum_{k'=1}^{K_c} \alpha_{t-1}^c(k') \gamma_{k',k}^c b_k^c(\mathbf{x}(t)), \qquad 1 < t \le T \tag{13}$$

where $\gamma_{k',k}^c$ is the probability for state changing from k' to k in class c, and $b_k^c(\mathbf{x}(t))$ is defined as the *a posteriori* probability for state k in class c corresponding to $x(t)$. In addition, (12) illustrates the initial phase, where the *a priori* probability π_k^c equals to $P(c, k)|_{t=0}$, although in most practical problems π_k^c

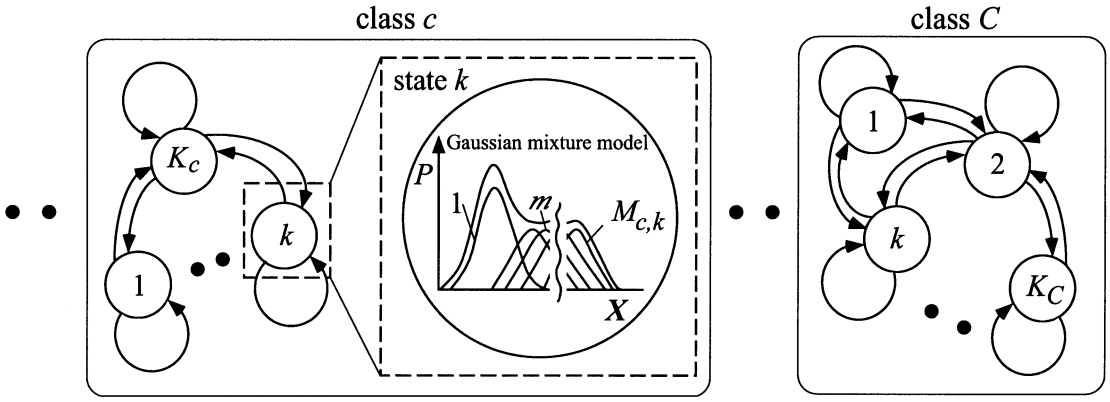


Fig. 3. CDHMM with C classes and K_c states in class c.

is unknown. With GMM, $\gamma^c_{k',k} b^c_k(\mathbf{x}(t))$ in the right side of (13) can be derived with the form

$$\begin{aligned}
&\gamma^c_{k',k} b^c_k(\mathbf{x}(t)) \\
&\quad = \sum_{m=1}^{M_{c,k}} \gamma^c_{k',k} r_{c,k,m} g\left(\mathbf{x}(t);\, \boldsymbol{\mu}^{(c,k,m)},\, \Sigma^{(c,k,m)}\right) \\
&\quad = \sum_{m=1}^{M_{c,k}} \gamma^c_{k',k} r_{c,k,m} (2\pi)^{-(d/2)} \left|\Sigma^{(c,k,m)}\right|^{-(1/2)} \\
&\qquad \times \exp\Bigg[-\tfrac{1}{2} \sum_{j=1}^{d} \sum_{l=1}^{j} (2-\delta_{jl}) s^{(c,k,m)}_{jl} x_j(t) x_l(t) \\
&\qquad\quad + \sum_{j=1}^{d} \sum_{l=1}^{d} s^{(c,k,m)}_{jl} \mu^{(c,k,m)}_j x_l(t) \\
&\qquad\quad - \tfrac{1}{2} \sum_{j=1}^{d} \sum_{l=1}^{d} s^{(c,k,m)}_{jl} \mu^{(c,k,m)}_j \mu^{(c,k,m)}_l \Bigg] \qquad (14)
\end{aligned}$$

where $r_{c,k,m}$, $\boldsymbol{\mu}^{(c,k,m)} = (\mu^{(c,k,m)}_1, \ldots, \mu^{(c,k,m)}_d)^{\mathrm{T}}$, $\Sigma^{(c,k,m)} \in \Re^{d\times d}$, $s^{(c,k,m)}_{ij}$ and $x_i(t)$ stands for the mixing proportion, the mean vector, the covariance matrix of each component $\{c, k, m\}$, element of the inverse of covariance matrix $\Sigma^{(c,k,m)-1}$ and element of $\mathbf{x}(t)$, respectively.

Taking the *log-linearization* (see II-A) of $\gamma^c_{k',k} r_{c,k,m} g(\mathbf{x}(t); \boldsymbol{\mu}^{(c,k,m)}, \Sigma^{(c,k,m)})$, we get

$$\begin{aligned}
\xi^c_{k',k,m}(t) &\triangleq \log \gamma^c_{k',k} r_{c,k,m} g\left(\mathbf{x}(t);\, \boldsymbol{\mu}^{(c,k,m)},\, \Sigma^{(c,k,m)}\right) \\
&= \boldsymbol{\beta}^c_{k',k,m}{}^{\mathrm{T}} \mathbf{X}(t) \qquad (15)
\end{aligned}$$

where $\mathbf{X}(t) \in \Re^H$ and $\boldsymbol{\beta}^c_{k',k,m} \in \Re^H$ are defined similarly as those in LLGMN (4)(5)(6). We can see that $\xi^c_{k',k,m}$ can be expressed as the product of the coefficient vector $\boldsymbol{\beta}^c_{k',k,m}$ and the modified input vector $\mathbf{X} \in \Re^H$, where the element of the vector $\boldsymbol{\beta}^c_{k',k,m}$ consists of the parameters of the statistic model, and the modified input vector $\mathbf{X}(t)$ includes the product of the elements of the input vector $\mathbf{x}(t)$.

Hence, the model can be developed as the network structure, using $\boldsymbol{\beta}^c_{k',k,m}$ as the weight coefficients. However, most elements of $\boldsymbol{\beta}^c_{k',k,m}$ are constrained by the statistical properties of the parameters in the model. This constraint may cause a difficult problem in the learning procedure: how to satisfy the constraints during the learning of the weight coefficients. Therefore, the new variable $Y^c_{k',k,m}$ and the new coefficient vector $\mathbf{w}^c_{k',k,m}$ are introduced as follows, similarly to LLGMN:

$$\begin{aligned}
Y^c_{k',k,m}(t) &\equiv \xi^c_{k',k,m}(t) - \xi^C_{K_C,K_C,M_{C,K}}(t) \\
&= \left(\boldsymbol{\beta}^c_{k',k,m} - \boldsymbol{\beta}^C_{K_C,K_C,M_{C,K}}\right)^{\mathrm{T}} \mathbf{X}(t) \\
&= \mathbf{w}^c_{k',k,m}{}^{\mathrm{T}} \mathbf{X}(t). \qquad (16)
\end{aligned}$$

The weight coefficient $\mathbf{w}^c_{k',k,m}$ is defined as the difference between $\boldsymbol{\beta}^c_{k',k,m}$ and $\boldsymbol{\beta}^C_{K_C,K_C,M_{C,K}}$, so $\mathbf{w}^C_{K_C,K_C,M_{C,K}} = 0$. Because of this transformation, the new parameter $\mathbf{w}^c_{k',k,m}$ has no constraints as the statistical parameter, and the constraints in $\xi^c_{k',k,m}(t)$ and $\boldsymbol{\beta}^c_{k',k,m}$ such as the positive definiteness of the covariance matrices are ignored. Therefore, the parameter space of $\mathbf{w}^c_{k',k,m}$ becomes larger than that of $\boldsymbol{\beta}^c_{k',k,m}$, and the weight coefficient $\mathbf{w}^c_{k',k,m}$ can have any real number. Note that this transformation does not result any loss of information in spite of $Y^C_{K_C,K_C,M_{C,K}}(t) = 0$, since the variable $\xi^c_{k',k,m}(t)$ in (15) is redundant because of $\sum_{c=1}^{C} \sum_{k=1}^{K_c} P(c, k|\mathbf{x}(t)) = 1$. Subsequently, (13) can be rewritten in the form

$$\begin{aligned}
\alpha^c_t(k) &= \sum_{k'=1}^{K_c} \alpha^c_{t-1}(k') \gamma^c_{k',k} b^c_k(\mathbf{x}(t)) \\
&= \sum_{k'=1}^{K_c} \alpha^c_{t-1}(k') \exp\left[Y^c_{k',k,m}(t)\right], \qquad t > 1. \qquad (17)
\end{aligned}$$

On the other hand, when $t = 1$ in (12), employing GMM as well as the log-linear model, we can derive $\xi^{ini,c}_{k',k,m}$, $\boldsymbol{\beta}^{ini,c}_{k',k,m}$, $\mathbf{w}^{ini,c}_{k',k,m}$ and $Y^{ini,c}_{k',k,m}$ similarly. It is the same as the case $t > 1$, by replacing $\gamma^c_{k',k}$ in (14) (15) with π^c_k. Then (12) would be expressed as

$$\begin{aligned}
\alpha^c_1(k) &= \pi^c_k b^c_k(\mathbf{x}(1)) = \exp\left[Y^{ini,c}_{k',k,m}\right] \\
&= \exp\left[\mathbf{w}^{ini,c}_{k',k,m}{}^{\mathrm{T}} \mathbf{X}(1)\right]. \qquad (18)
\end{aligned}$$

In this paper, we regard $\mathbf{w}^{ini,c}_{k',k,m} = \mathbf{w}^c_{k',k,m}$, because both of $\mathbf{w}^{ini,c}_{k',k,m}$ and $\mathbf{w}^c_{k',k,m}$ have no constraints and include many unknown statistical parameters. Consequently, many parameters of the probabilistic model such as the mixing coefficient $r_{c,k,m}$, the mean vector $\boldsymbol{\mu}^{(c,k,m)}$, the covariance matrix $\Sigma^{(c,k,m)}$ and

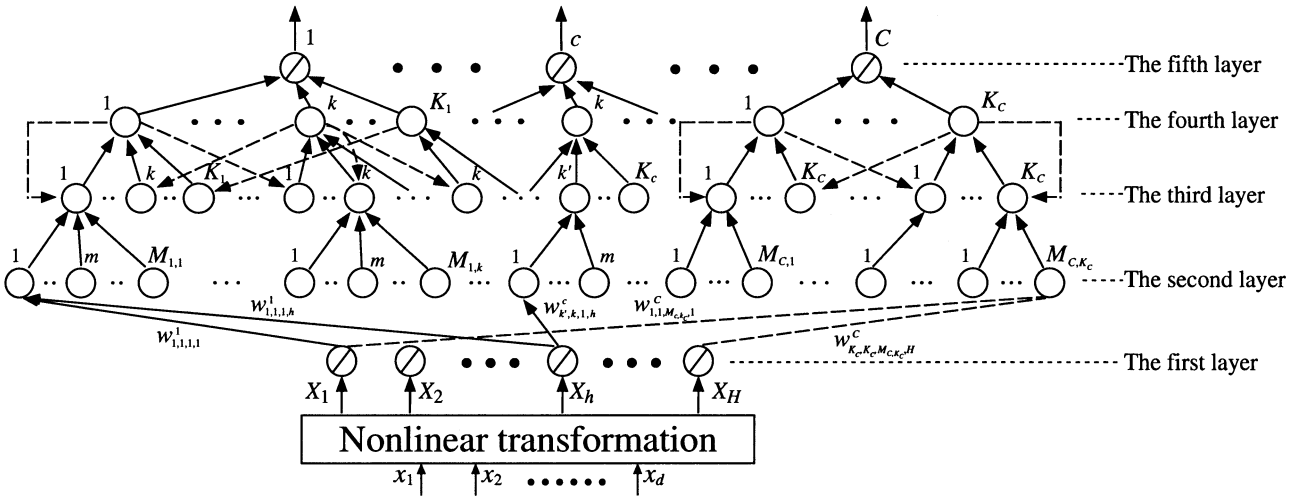


Fig. 4. NN structure.

the transition probability $\gamma^c_{k',k}$ are replaced by arbitrary parameters $\mathbf{w}^c_{k',k,m}$. In Section III-C, we transform this model to the network structure, in which the coefficient vector $\mathbf{w}^c_{k',k,m}$ is used as the weight vector.

C. NN Structure

The structure of the proposed NN is shown in Fig. 4. It is a five-layer recurrent NN with a feedback connection between the fourth layer and the third layer. First of all, we define the number of units in each layer. The fifth layer contains C units, and for each unit c $(c = 1, \ldots, C)$ there are K_c *branch* connections. Then, $\sum_{c=1}^{C} K_c$ units corresponding to these *branches* form the fourth layer. The unit $\{c, k\}$ $(k = 1, \ldots, K_c)$ in the fourth layer connects with K_c units in the third layer which consists of $\sum_{c=1}^{C} {K_c}^2$ units. There are $M_{c,k}$ components set for the units $\{c, k, k'\}$ $(k' = 1, \ldots, K_c)$ in the third layer, so the total units in the second layer are $\sum_{c=1}^{C}\sum_{k=1}^{K_c} {K_c}^2 M_{c,k}$ units. Meanwhile, the input vector series $\mathbf{x}(t) \in \Re^d$ $(t = 1, \ldots, T)$ is modified in the same way as LLGMN, then the vector $\mathbf{X}(t) \in \Re^H$ acts as the input of the first layer. Therefore, the first layer consists of H units, and the identity function is used for activation of each unit as well. ${}^{(1)}I_h$ and ${}^{(1)}O_h$ denote the input and the output, respectively, of the hth unit in the first layer.

Unit $\{c, k, k', m\}$ $(m = 1, \ldots, M_{c,k})$ in the second layer receives the output of the first layer weighted by the coefficient $w^c_{k',k,m,h}$. The input ${}^{(2)}I^c_{k',k,m}(t)$ and the output ${}^{(2)}O^c_{k',k,m}(t)$ are defined as

$$ {}^{(2)}I^c_{k',k,m}(t) = \sum_{h=1}^{H} {}^{(1)}O_h(t) w^c_{k',k,m,h} \tag{19} $$

$$ {}^{(2)}O^c_{k',k,m}(t) = \exp\left({}^{(2)}I^c_{k',k,m}(t)\right). \tag{20} $$

The output of the second layer is added up and input into the third layer. Also, the output of the fourth layer is fed back to the third layer. These are expressed as follows:

$$ {}^{(3)}I^c_{k',k}(t) = \sum_{m=1}^{M_{c,k}} {}^{(2)}O^c_{k',k,m}(t) \tag{21} $$

$$ {}^{(3)}O^c_{k',k}(t) = {}^{(4)}O^c_{k'}(t-1)\,{}^{(3)}I^c_{k',k}(t) \tag{22} $$

where ${}^{(4)}O^c_{k'}(0) = 1.0$ for the initial phase.

The activation functions in the fourth layer are described in the form

$$ {}^{(4)}I^c_k(t) = \sum_{k'=1}^{K_c} {}^{(3)}O^c_{k',k}(t) \tag{23} $$

$$ {}^{(4)}O^c_k(t) = \frac{{}^{(4)}I^c_k(t)}{\sum_{c'=1}^{C}\sum_{k'=1}^{K_{c'}} {}^{(4)}I^{c'}_{k'}(t)}. \tag{24} $$

At last, the unit c in the fifth layer integrates the outputs of K_c units $\{c, k\}$ $(k = 1, \ldots, K_c)$ in the fourth layer. The relationship in the fifth layer is defined as

$$ {}^{(5)}I^c(t) = \sum_{k=1}^{K_c} {}^{(4)}O^c_k(t) \tag{25} $$

$$ {}^{(5)}O^c(t) = {}^{(5)}I^c(t). \tag{26} $$

Let us consider the case where the length of time series T is one, and $K_c = 1$ for each unit in the fifth layer. As $t = 1$ for all the time, the recurrent connection in (22) does not work any more. In this case, using $K_c = 1$, the relationship from the second layer to the fifth layer [(19)–(26)] can be reorganized as follows:

$$ {}^{(2)}I^c_{1,1,m}(1) = \sum_{h=1}^{H} {}^{(1)}O_h(t) w^c_{1,1,m,h} \tag{27} $$

$$ {}^{(5)}O^c(1) = \frac{\sum_{m=1}^{M_c} \exp\left({}^{(2)}I^c_{1,1,m}(1)\right)}{\sum_{c'=1}^{C}\sum_{m'=1}^{M_c} \exp\left({}^{(2)}I^{c'}_{1,1,m'}(1)\right)} \tag{28} $$

which is exactly the same relationship as the one between the second and third layers in LLGMN (see (9) and (10)). As is to say, when it is not necessary to consider the dynamic properties in the data sequence $(T = 1)$ or the recurrent connections in the proposed NN are not significant, it reduces to LLGMN. In other words, LLGMN has been extended and affixed with recurrent

TABLE I
FORWARD ALGORITHM OF R-LLGMN AND THE CORRESPONDING COMPUTATION OF CDHMM. R-LLGMN INCLUDES THE FORWARD COMPUTATION OF CDHMM AS A SPECIAL CASE

	R-LLGMN	CDHMM
Input vector	$\mathbf{x}(t)\,(t=1,\ldots,T)$	$\mathbf{x}(t)\,(t=1,\ldots,T)$
2nd layer	${}^{(2)}O^{c}_{k,k,m}(t)$	$g(\mathbf{x}(t);\mu^{(c,k,m)},\Sigma^{(c,k,m)})$
3rd layer	${}^{(3)}I^{c}_{k',k}(t) = \sum_{m=1}^{M_{c,k}} {}^{(2)}O^{c}_{k',k,m}(t)$	$b^{c}_{k}(x(t)) = \sum_{m=1}^{M_{c,k}} r_{c,k,m}\, g(x(t);\mu^{(c,k,m)},\Sigma^{(c,k,m)})$
3rd & 4th layer	${}^{(4)}I^{c}_{k}(t) = \sum_{k'=1}^{K_c} {}^{(3)}O^{c}_{k',k}(t) = \sum_{k'=1}^{K_c} {}^{(4)}O^{c}_{k'}(t\text{-}1)\,{}^{(3)}I^{c}_{k',k}(t)$	$\alpha_t(k) = \sum_{k'=1}^{K_c} \alpha_{t-1}(k')\gamma^{c}_{k',k}\, b^{c}_{k}(x(t))$
5th layer	${}^{(5)}O^{c}(T)$	$P(c\|\tilde{\mathbf{x}})$
Coefficients	$w^{c}_{k',k,m,h}$	$r_{c,k,m}$; $\mu^{(c,k,m)}$; $\Sigma^{(c,k,m)}$; $\gamma^{c}_{k',k}$

connections, so we named the NN presented in this paper the R-LLGMN.

Table I shows the correspondence of R-LLGMN to CDHMM. The input vector series, which is transferred into the first layer of R-LLGMN, is exactly the same observation sequence in the model of CDHMM. The units in the second layer act alike as the mixture components of the continuous observation probability density function in CDHMM [see (14)]. The calculation in the third and fourth layers, associated with the feedback connections, represents the forward algorithm (or Alpha computation) [27]. Finally, units in the fifth layer output the *a posteriori* probability $P(c|\tilde{\mathbf{x}})$ of class c for $\tilde{\mathbf{x}}$. On the other hand, the weight coefficients $\mathbf{w}^{c}_{k',k,m}$, between the first layer and the second layer, correspond to the transferred parameters used in the forward computation, such as the mixing coefficient $r_{c,k,m}$, the mean vector $\boldsymbol{\mu}^{(c,k,m)}$, the covariance matrix $\Sigma^{(c,k,m)}$ and the transition probability $\gamma^{c}_{k',k}$. With respect to these, R-LLGMN can be interpreted as a hidden Markov neural network (HMN), based on CDHMM, and can model the observation sequence through learning only the weight coefficients $\mathbf{w}^{c}_{k',k,m}$.

However, R-LLGMN is not just a *copy* of CDHMM: it is superior because of a better parameterization. An essential point is that R-LLGMN replaces all of the parameters in CDHMM with the weight coefficients $\mathbf{w}^{c}_{k',k,m}$, and this replacement removes restrictions of the statistical parameters in CDHMM, e.g., $0 \le r_{c,k,m} \le 1$, standard deviations of GMMs >0, and so on. Therefore, the learning algorithm of R-LLGMN is simplified and is expected of higher generalization ability than that of CDHMM. Another important distinction between R-LLGMN and CDHMM is the number of parameters used in these methods. The number of parameters of R-LLGMN (N_R) and that of CDHMM (N_C), which is the sum of element numbers of $\gamma^{c}_{k',k}$, $r_{c,k,m}$, $\boldsymbol{\mu}^{(c,k,m)}$ and $\Sigma^{(c,k,m)}$, are given as follows:

$$N_R = \sum_{c=1}^{C} K_c \sum_{k=1}^{K_c} M_{c,k} H = \sum_{c=1}^{C} K_c \sum_{k=1}^{K_c} M_{c,k}\left(1+\tfrac{3}{2}d+\tfrac{1}{2}d^2\right) \quad (29)$$

$$N_C = \sum_{c=1}^{C} {K_c}^2 + \sum_{c=1}^{C}\sum_{k=1}^{K_c} M_{c,k}(1+d+d^2). \quad (30)$$

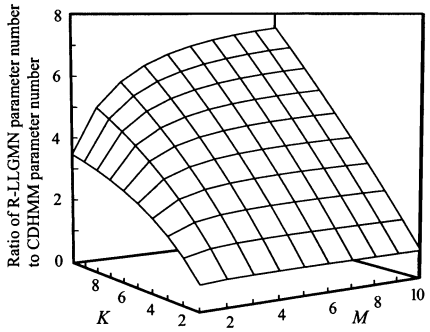


Fig. 5. Examples of the ratio of N_R to N_C with $d = 2$.

Under $K_1 = K_2 = \cdots = K_C = K$ and $M_{1,1} = M_{1,2} = \cdots = M_{1,K_1} = M_{2,1} = \cdots = M_{c,k} = \cdots = M_{C,K_C} = M$, (29) and (30) result in

$$N_R = CK^2M\left(1+\tfrac{3}{2}d+\tfrac{1}{2}d^2\right) \quad (31)$$

$$N_C = CK^2 + CKM(1+d+d^2). \quad (32)$$

Fig. 5 shows the ratio of N_R to N_C with $d = 2$. It should be noted that in the experiments in Section V, the dimension of input is two. It is obvious that N_R is larger than N_C in most cases. This may indicate that R-LLGMN has a better representation ability. Although R-LLGMN has more parameters, we can train it with only one sample (time series) using a gradient learning method, while for CDHMM, at least two samples are needed to calculate mean and standard deviation for each Gaussian component. It could be expected that even if the training date in the learning process is of small size, R-LLGMN achieves a better estimation than CDHMM.

D. Learning Algorithm

A set of vector streams $\mathbf{x}(t)^{(n)}$ $(t = 1, \ldots, T_n;\ n = 1, \ldots, N)$ are given for training R-LLGMN with teacher vector $\mathbf{T}^{(n)} = (T_1^{(n)}, \ldots, T_c^{(n)}, \ldots, T_C^{(n)})^{\mathrm{T}}$ $(n = 1, \ldots, N)$ for the nth input stream $\tilde{\mathbf{x}}^{(n)}$. Then, these N vector streams are divided into L subsets, while each set consists of C stream classes $(N = L \times C)$. If the vector stream $\tilde{\mathbf{x}}^{(n)}$ is set for the class $\hat{c}$ in subset l $(l = 1, \ldots, L)$, then $T_{\hat{c}}^{(n)} = 1$, and $T_c^{(n)} = 0$ for all the other classes in this subset. It is supposed that the network catches the character of the data set, if for all the streams the last output of stream $\tilde{\mathbf{x}}^{(n)}$, namely, ${}^{(5)}O^{c}(T_n)$, is close enough to the teacher signal $\mathbf{T}^{(n)}$. In this paper an energy function for the network is defined as

$$J = \sum_{n=1}^{N} J_n = -\sum_{n=1}^{N}\sum_{c=1}^{C} T_c^{(n)} \log {}^{(5)}O^{c}(T_n). \quad (33)$$

The learning process is to minimize J, that is, to maximize the likelihood that each teacher vector $\mathbf{T}^{(n)}$ is obtained for the input stream $\tilde{\mathbf{x}}^{(n)}$.

Usually, the weight modification $\Delta w^{c}_{k',k,m,h}$ for $w^{c}_{k',k,m,h}$ is defined as

$$\Delta w^{c}_{k',k,m,h} = -\eta \sum_{n=1}^{N} \frac{\partial J_n}{\partial w^{c}_{k',k,m,h}} \quad (34)$$

in a collective learning scheme with a fixed $\eta > 0$ as the learning rate. Because of the recurrent connection in R-LLGMN, the backpropagation-through-time (BPTT) algorithm [1], [23] is used. It is supposed that the error gradient within a stream (block) is accumulated and weight modifications are only computed at the end of each block; the error is then propagated backward to the beginning of the block. So, using the chain rule for the stream $\tilde{\mathbf{x}}^{(n)}$, $\partial J_n / \partial w^c_{k',k,m,h}$ in (34) can be expanded in the following way:

$$\begin{aligned}
&\frac{\partial J_n}{\partial w^c_{k',k,m,h}} \\
&= -\sum_{t=0}^{T_n-1}\sum_{c'=1}^{C}\sum_{k''=1}^{K_{c'}} {}^{(n)}\Delta^{c'}_{k''}(t)\,\frac{\partial {}^{(4)}O^{c'}_{k''}(T_n-t)}{\partial {}^{(4)}I^c_k(T_n-t)} \\
&\quad\times \frac{\partial {}^{(4)}I^c_k(T_n-t)}{\partial {}^{(3)}O^c_{k',k}(T_n-t)}\,\frac{\partial {}^{(3)}O^c_{k',k}(T_n-t)}{\partial {}^{(3)}I^c_{k',k}(T_n-t)} \\
&\quad\times \frac{\partial {}^{(3)}I^c_{k',k}(T_n-t)}{\partial {}^{(2)}O^c_{k',k,m}(T_n-t)}\,\frac{\partial {}^{(2)}O^c_{k',k,m}(T_n-t)}{\partial {}^{(2)}I^c_{k',k,m}(T_n-t)} \\
&\quad\times \frac{\partial {}^{(2)}I^c_{k',k,m}(T_n-t)}{\partial w^c_{k',k,m,h}} \\
&= -\sum_{t=0}^{T_n-1}\sum_{c'=1}^{C}\sum_{k''=1}^{K_{c'}} {}^{(n)}\Delta^{c'}_{k''}(t) \\
&\quad\times \left(\Gamma_{(c',k''),(c,k)} - {}^{(4)}O^{c'}_{k''}(T_n-t)\right)\frac{{}^{(4)}O^{c'}_{k''}(T_n-t)}{{}^{(4)}I^{c'}_{k''}(T_n-t)} \\
&\quad\times {}^{(4)}O^c_{k'}(T_n-t-1)\,{}^{(2)}O^c_{k',k,m}(T_n-t)X_h(T_n-t)
\end{aligned} \tag{35}$$

where ${}^{(n)}\Delta^{c'}_{k''}(t)$ is defined as the partial differentiation of J_n to ${}^{(4)}O^{c'}_{k''}(T_n-t)$

$${}^{(n)}\Delta^{c'}_{k''}(t) = \frac{\partial J_n}{\partial {}^{(4)}O^{c'}_{k''}(T_n-t)} \tag{36}$$

and $\Gamma_{(c',k''),(c,k)}$ is defined as

$$\Gamma_{(c',k''),(c,k)} = \begin{cases} 1, & (c'=c;\ k''=k) \\ 0, & \text{(otherwise).} \end{cases} \tag{37}$$

(36) can be derived as follows:

$$\begin{aligned}
{}^{(n)}\Delta^{c'}_{k''}(0) &= \frac{\partial (T^{(n)}_{c'}\log {}^{(5)}O^{c'}(T_n))}{\partial {}^{(5)}O^{c'}(T_n)}\,\frac{\partial {}^{(5)}O^{c'}(T_n)}{\partial {}^{(4)}O^{c'}_{k''}(T_n)} \\
&= \frac{T^{(n)}_{c'}}{{}^{(5)}O^{c'}(T_n)}
\end{aligned} \tag{38}$$

$$\begin{aligned}
{}^{(n)}\Delta^{c'}_{k''}(t+1) &= \sum_{c''=1}^{C}\sum_{k'''=1}^{K_{c''}} {}^{(n)}\Delta^{c''}_{k'''}(t)\sum_{k''''=1}^{K_{c''}} \\
&\quad\times \frac{\partial {}^{(4)}O^{c''}_{k'''}(T_n-t)}{\partial {}^{(4)}I^{c'}_{k''''}(T_n-t)}\,\frac{\partial {}^{(4)}I^{c'}_{k''''}(T_n-t)}{\partial {}^{(3)}O^{c'}_{k'',k''''}(T_n-t)} \\
&\quad\times \frac{\partial {}^{(3)}O^{c'}_{k'',k''''}(T_n-t)}{\partial {}^{(4)}O^{c'}_{k''}(T_n-(t+1))} \\
&= \sum_{c''=1}^{C}\sum_{k'''=1}^{K_{c''}} {}^{(n)}\Delta^{c''}_{k'''}(t)\sum_{k''''=1}^{K_{c''}} \\
&\quad\times \left(\Gamma_{(c'',k'''),(c',k'''')} - {}^{(4)}O^{c''}_{k'''}(T_n-t)\right) \\
&\quad\times \frac{{}^{(4)}O^{c''}_{k'''}(T_n-t)}{{}^{(4)}I^{c''}_{k'''}(T_n-t)}\,{}^{(3)}I^{c'}_{k'',k''''}(T_n-t).
\end{aligned} \tag{39}$$

In this paper, the dynamics of the terminal attractor (TA) [24] is incorporated in the learning rule in order to regulate the convergence time of the learning. The differential equation of TA is defined as

$$\dot{u} = -u^{\beta}. \tag{40}$$

When the parameter β is determined as $0 < \beta < 1$, u is a monotonically nonincreasing function, and always converges stably to the equilibrium point in a finite time, since the Lipschitz conditions are violated at $u = 0$

$$\left.\frac{d\dot{u}}{du}\right|_{u=0} = \left.-\beta u^{\beta-1}\right|_{u=0} = -\infty \tag{41}$$

where β determines how the dynamics converges, such as smooth or sharp, although the convergence time is fixed depending on the initial condition $u = u_0$

$$t_f = \int_0^{t_f} dt = \int_{u_0}^{u\to 0}\frac{du}{\dot{u}} = \frac{u_0^{1-\beta}}{(1-\beta)} < \infty. \tag{42}$$

If TA, which is defined above, is incorporated into the energy function (33) of R-LLGMN, the convergence time of the learning can be regulated [25].

Let us consider the incorporation of TA into R-LLGMN, in the proposed learning method, the weight coefficients of R-LLGMN are considered as the time dependent continuous variables and the time derivative of $w^c_{k',k,m,h}$ is defined as

$$\dot{w}^c_{k',k,m,h} = -\eta_{ta}\gamma\frac{\partial J}{\partial w^c_{k',k,m,h}} \tag{43}$$

$$\gamma = \frac{J^{\beta}}{\sum_{c=1}^{C}\sum_{k=1}^{K_c}\sum_{k'=1}^{K_c}\sum_{m=1}^{M_{c,k}}\sum_{h=1}^{H}\left(\frac{\partial J}{\partial w^c_{k',k,m,h}}\right)^2} \tag{44}$$

where $\eta_{ta} > 0$ is positive, and γ is calculated using constant β. The time derivative of the energy function J can be calculated as

$$\begin{aligned}
\dot{J} &= \sum_{c=1}^{C}\sum_{k=1}^{K_c}\sum_{k'=1}^{K_c}\sum_{m=1}^{M_{c,k}}\sum_{h=1}^{H}\left(\frac{\partial J}{\partial w^c_{k',k,m,h}}\dot{w}^c_{k',k,m,h}\right) \\
&= -\eta_{ta}J^{\beta} \le 0.
\end{aligned} \tag{45}$$

Thus, the convergence time can be given as

$$t_f = \int_0^{t_f} dt = \int_{J_0}^{J_f}\frac{dJ}{\dot{J}} = \frac{J_0^{1-\beta} - J_f^{1-\beta}}{\eta_{ta}(1-\beta)} \le \frac{J_0^{1-\beta}}{\eta_{ta}(1-\beta)} \tag{46}$$

where J_0 is an initial value of the energy function J calculated using initial weights, and J_f is the final value of J at the equilib-

TABLE II
PARAMETERS OF THE HMMs USED IN THE EXPERIMENTS

	π_c	A_c	B_c
Class I (c=1)	[1.0, 0.0, 0.0]	0.154 0.423 0.423 0.079 0.314 0.607 0.079 0.607 0.314	0.730 0.227 0.000 0.033 0.075 0.215 0.312 0.398 0.075 0.215 0.312 0.398
Class II (c=2)	[1.0, 0.0, 0.0]	0.454 0.273 0.273 0.172 0.294 0.534 0.412 0.294 0.294	0.040 0.244 0.269 0.447 0.484 0.228 0.080 0.208 0.484 0.228 0.080 0.208

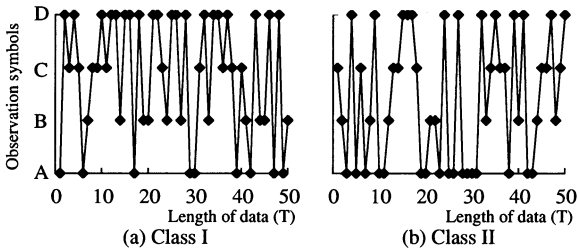


Fig. 6. Examples of the data used. (a) Class I. (b) Class II.

rium point. In the case of $J_f = 0$, the equal sign of (46) is held. Thus the convergence time can be specified by learning rate η_{ta}. On the other hand, in the case of $J_f \neq 0$, the convergence time is always less than the upper limit of (46).

The learning is carried out by a discrete form, derived from (43)

$$w^c_{k',k,m,h}(t+\Delta t) = w_{k',k,m,h}(t) + \frac{\Delta t}{2} \times (\dot{w}^c_{k',k,m,h}(t) + \dot{w}^c_{k',k,m,h}(t+\Delta t)) \quad (47)$$

where Δt denotes the sampling time. The total number of learning iterations becomes $t_f/\Delta t$, and the computation time depends on this number. If Δt is determined as a small value, the energy function decreases accurately according to (45).

IV. SIMULATION EXPERIMENTS

Simulation experiments were performed to explore the ability of R-LLGMN, comparing the classification performance of R-LLGMN with the one of HMM for experimental data generated with HMMs. It should be noted that the computer program of a Baum–Welth algorithm for comparison, "Myers's hidden Markov model software," was downloaded from the internet website.[1]

The time series $(d = 1)$ were generated for two classes $(C = 2)$ using two different HMMs set for each class (see Table II). The input series are one-dimensional and are encoded to four symbols, A, B, C, D, which correspond to 0, 1/3, 2/3, and 1, respectively. These real numbers are used for the numerical calculation. The HMMs used are full connective, and there are three states for each model and four distinct observation symbols (output of the model). Fig. 6 shows examples of the generated data with a length of 50.

The R-LLGMN was set as follows: $C = 2, K_1 = K_2 = 3, H = 3$, and $M_{c,k} = 1$ (see III-C). The Baum-Welth algorithm was prepared to estimate models of the same size as those defined in Table III. Classification experiments carried out using various sizes of training data. The training data includes N vector streams which are further divided into L subsets, where each set consists of C stream classes $(N = L \times C)$. The number of subsets L and the length of the time series T $(T = T_1 = T_2 = \cdots = T_N)$ changed from 1 to 10 and from 20 to 100, respectively. The R-LLGMN and the Baum–Welth algorithm were trained five times with different data of the same size, then the five sets of coefficients were examined. The R-LLGMN learned according to the dynamics of the terminal attractor incorporated $(t_f = 1$ s and $\Delta t = 0.000\,25$ s, that is, 4000 iterations), and the learning of Baum-Welth algorithm would terminate when the change of the parameters per iteration becomes less than a threshold of 0.000 01.

As to the recognition process, each set of coefficients was used to recognize five sets of data, which comprises 400 series (200 for each class) with the same data length as the corresponding training data. Then the rates of classification were calculated over 10 000 results ($5 \times 5 \times 400$) for each size of the training data. The mean values and the standard deviations of the classification rates for each size are shown in Figs. 7 and 8. Please note that the directions of axes of T and L are reversed to make the figures shown clearly. For cases of large data size, the classification rates of the R-LLGMN are almost the same

[1]ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/recognition/

TABLE III
CLASSIFICATION RESULTS OF EYE STATE USING LLGMN, LLGMN WITH RNF, HMM, CDHMM, AND R-LLGMN

Type of the methods		LLGMN	LLGMN with NF	HMM	CDHMM	R-LLGMN
Subject A (male)	*R*	91.1	94.5	95.7	95.7	95.7
	SD	0.4	0.3	0.0	0.0	0.0
Subject B (male)	*R*	93.2	93.4	88.1	87.3	92.4
	SD	0.6	0.1	0.0	0.0	0.0
Subject C (male)	*R*	81.3	89.7	93.8	93.4	95.2
	SD	1.1	0.5	0.0	0.0	0.0
Total	*R*	88.5	92.5	92.5	92.1	94.4
	SD	1.7	0.3	0.0	0.0	0.0

R : Classification rate [%] *SD* : Standard deviation [%]

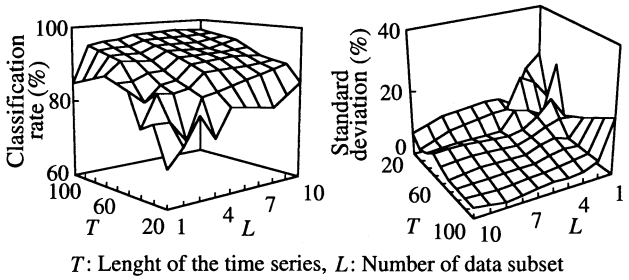


Fig. 7. Classification rates of the R-LLGMN for different data size.

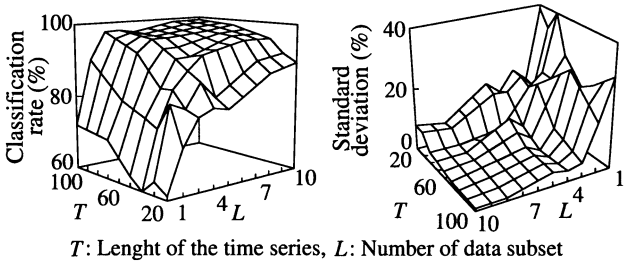


Fig. 8. Classification rates of the HMM for different data size.

as the HMM, indicating that R-LLGMN can work as an estimator of HMM. Alternatively, given training data of small size, the Baum–Welth algorithm results in a worse classification rate than the R-LLGMN. R-LLGMN can achieve high classification performance for the small size of data because of the inheritance of the NNs.

V. EEG Pattern Classification

Bioelectric signals such as EEG and EMG are typical time series with dynamic characteristic, which are expected to be control signal for a new type of a man–machine interface. The bioelectric signal ranges widely in frequency-domain and contains high frequency components, so adequate signal processing is necessary. In particular, it should be filtered through a well-designed low-pass filter to remove the high frequency components, while the pattern classification of the signal must be done to reveal the operator's intention. However, it is very difficult to perform the filtering and the classification simultaneously. The authors had tried to classify EEG signals with recurrent neural networks such as Jordan's and Ellman's networks [38], [39]. It was, however, too difficult for only use of them to achieve high classification accuracy because of the considerable time-varying characteristics of EEG signals. To overcome this difficulty, Tsuji *et al.* [8], [9], [13], [25] investigated the pattern classification problem of EEG (EMG) signals using a static probabilistic NN, LLGMN, and a recurrent neural filter (RNF). Although this method attained relatively high classification rates, it is necessary to train two different types of NNs, that is, LLGMN and RNF, therefore the learning procedure becomes quite complicated and general optimization is almost impossible. Alternatively, R-LLGMN ensures that the filtering process and the pattern classification can be achieved at the same time.

In this section, as an application of R-LLGMN, the EEG signal classification has been done using the same data in [9] for comparison. R-LLGMN is based on CDHMM, and inherits lots of advantages from it. It can be expected R-LLGMN realizes higher learning/classification performance using a one-network structure and a simple learning algorithm.

A. Experimental Apparatus and Conditions

Fig. 9 shows the experimental apparatus. A simple and handy electroencephalograph (IBVA, Random ELECTRONICES DESIGN) was used to measure EEG signals. The experimental system consists of the headband, transmitter and receiver. The transmitter was attached to the headband. The EEG signals measured from the electrodes were digitized by an A/D converter after they were amplified and filtered through low-cut (3 Hz) and high-cut (40 Hz) analogue filters. The noise in the EEG signals can be reduced significantly by the bipolar derivation between the two electrodes located at Fp1 and Fp2.

The EEG signals were measured in the following conditions:

1) Photic stimulation by opening and closing eyes.

   Subjects were seated in a well-lighted room. First, EEG signals were measured with both eyes opened and closed (60 s for each). The measured signals were used as learning data. Next, subjects were asked to switch their eye states alternatively according to a pseudorandom series for 450 s.

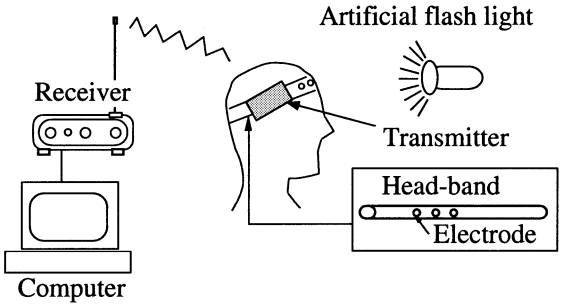


Fig. 9. Structure of EEG determination system.

2) Photic stimulation by opening, closing eyes and an artificial light (opening).

   Subjects were seated in a dark room. There is an additional state in this condition, an artificial light is used while the subjects open their eyes. A flash light (xenon, illuminating power: 0.176 J) was set at a distance of 50 cm from the subject's eyes. Learning data and classification data are recorded for three states of EEG signals in the same manner as the condition 1.

Although the input length of the time series is fixed in the experiment, the duration of meaningful and effective EEG signal is not always fixed indeed, but changes depending on classes and subjects. It is considered that if the length of the input signal is long enough, the learning itself can select an appropriate length for the discrimination automatically. The fixed duration used in the experiments is just a length of the input signal, which means the upper limit of the EEG duration.

B. Feature Extraction of EEG Signals

The electroencephalograph used in the experiments has one pair of electrodes, so that the spatial information of the EEG signals on the location of the electrodes cannot be utilized. The frequency characteristics of EEG signals, however, significantly changes depending on the eye states. Therefore, the spectral information of the measured EEG signals were used as follows. The power spectral density function of the measured EEG signal was estimated using fast Fourier transform (FFT) for every 128 sampled data. The function was divided into several ranges (from 0 to 35 Hz). The frequency bands of this range were determined based on the clinical use of the brain wave (delta, theta, alpha, beta). Time series of the mean values of the power spectral density function within each frequency ranges were calculated and normalized between [0, 1] in each range. Thus, the two-dimensional data (corresponding to frequency range [0∼8], [9∼35] [Hz]) were obtained and used as the input vector to the networks.

C. Classification Result for Opening/Closing EEG Signals

The classification experiments were performed using five methods: R-LLGMN, LLGMN (II-B), LLGMN with RNF [9], Baum–Welth algorithm (HMM and CDHMM).[2] In R-LLGMN, parameters of the network architecture are set as: $C = 2$, $K_1 = K_2 = 1$, $H = 6$, and component for each unit in the third layer is one. The parameters used are chosen to make

[2]The CDHMM computer program is based on "Speech recognition system (SRS-V1.1)" (Electrical Engineering and Computer Engineering Department, University of Newcastle, Australia).

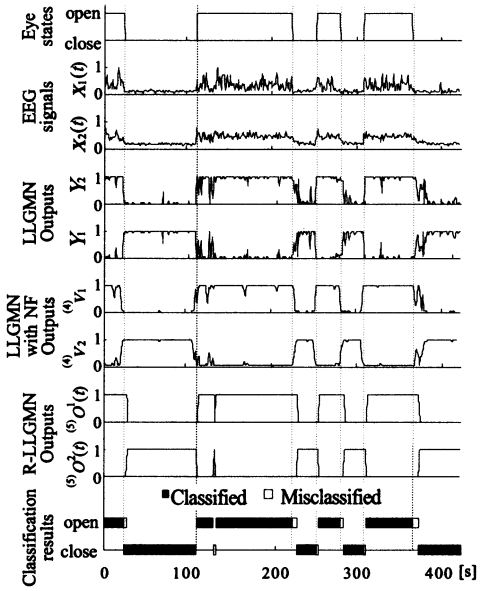


Fig. 10. Example of the classification result of eye states.

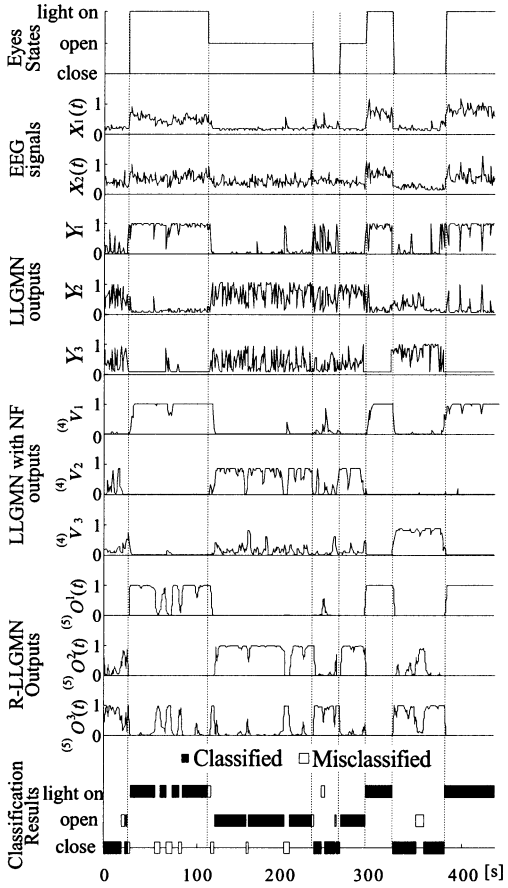


Fig. 11. Example of the classification result for three types of the photic stimulation.

experimental conditions as equal as possible. In LLGMN, $H = 6$ and $C = 2$ in the first and third layer, and the second layer consists of six units. As to LLGMN with RNF, there are eight units in RNF, which is connected to the same LLGMN explained above. In the RNF, fully interconnected units in the second layer keep the internal representation, so that the time history of the input data can be considered. Therefore it shows an effect as a filter and makes the *a posteriori* probability from the LLGMN smoother. For HMM, it is used to estimate models with one state $(N = 1)$. The structure of CDHMM used in the experiments is settled with the same condition as the one of R-LLGMN.

Experiments were performed for three subjects (A, B, C: males). First, Fig. 10 shows an example of the classification results of LLGMN, LLGMN with RNF and R-LLGMN (subject A). In this figure, the timing of the switching eye states, the input EEG signals, the outputs of LLGMN, LLGMN with RNF and R-LLGMN, and the classification results of R-LLGMN are plotted. As can be seen, the R-LLGMN performs at a very high classification rate of 97.6%.

Table III shows classification results for all subjects. The mean values and the standard deviations of the classification rate are computed for ten kinds of initial weights, which are randomly chosen. According to the results for the three subjects, except for LLGMN, all the other methods attained high classification rates. Generally, LLGMN based on a static Gaussian mixture model is not suitable for classification of dynamic signals like EEG, while the other methods contain the dynamic statistical model. It should be noted that LLGMN with RNF has a rather complicated construction, so that the difficulty of learning this method may be a critical problem. In addition, although training for HMM and CDHMM can be carried out easily, a large amount of training data is needed. On the other hand, R-LLGMN can train the static (a Gaussian mixture model) and dynamic (recurrent connections) parts at the same time even with a small amount of data (see III-C). Because the task of discriminating two eye states (open and close) is relatively easy for HMM, CDHMM, and R-LLGMN, through learning states of open and close can be easily divided into two regions, thus all the three methods achieve results with S.D. equal to zero.

D. Classification Results for Three States of EEG Signals

Fig. 11 shows an example of the classification result of subject A. In this experiment, the TA learning was repeated five times to gain a better convergence. Although it can be seen that the classification became difficult to classify compared to the results in Section V-C, a classification rate of 87.4% was still achieved.

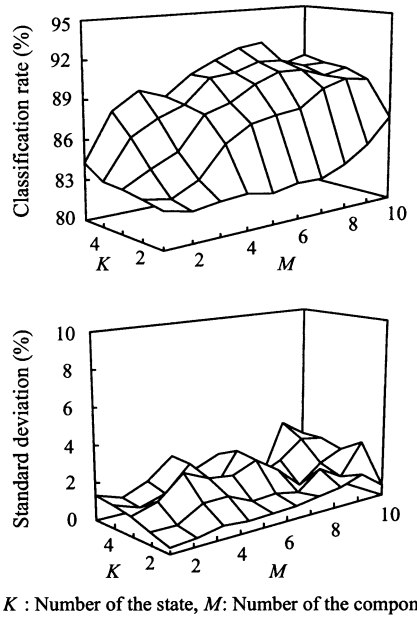
Table IV shows the classification results for all subjects. Although the results were worse than those of the classification for opening/closing the eyes, R-LLGMN realized almost the best classification performance.

As a real biological data, EEG signals are very complicated because no one can simply give the exact number of states and components of them. This motivated the experiments examining changes of the classification rates of R-LLGMN depending on the number of states and components. In the experiments, the number of subsets $L = 5$ and the length of time series $T = 5$ were used, and the number of states K varies from one to five, the number of components M from one to ten. The results of subject B are plotted in Fig. 12. It indicates that the classification rates can be improved by increasing the number of components and states. A further investigation is worthy to study how R-LLGMN can cope with a more complicated model, which

TABLE IV
CLASSIFICATION RESULTS FOR THE PHOTIC STIMULATION USING LLGMN, LLGMN WITH RNF, HMM, CDHMM, AND R-LLGMN

Type of the methods		LLGMN	LLGMN with NF	HMM	CDHMM	R-LLGMN
Subject A (male)	*R*	75.8	84.4	82.9	85.5	91.0
	SD	0.5	0.0	0.0	0.0	0.0
Subject B (male)	*R*	77.2	87.1	84.8	83.8	84.3
	SD	1.9	0.9	0.0	0.0	0.0
Subject C (male)	*R*	65.8	74.6	76.7	78.6	81.4
	SD	1.7	1.8	0.0	0.0	0.0
Total	*R*	72.9	82.0	81.2	82.7	85.5
	SD	1.4	0.9	0.0	0.0	0.0

R : Classification rate [%] *SD* : Standard deviation [%]



K : Number of the state, *M*: Number of the component

Fig. 12. Change of the classification rates depending on the numbers of the states and the components of R-LLGMN (subject B).

contains more states and stronger connection. We will make an additional report on this in the future.

VI. CONCLUSION

In this paper, a new model-based NN, R-LLGMN, has been proposed to deal with time series classification. R-LLGMN is derived through the modification of HMM, and includes HMM in its structure, so R-LLGMN can be considered as an HMN. Furthermore, R-LLGMN can be interpreted as an extension of LLGMN, where recurrent connections are embedded to approximate the inherent dynamic characteristics in the time series signals, and the LLGMM successfully used in LLGMN is also utilized to compute the pdf of input pattern. Simulations and experiments have been carried out to examine the classification capability of the proposed network.

In this paper, as the first stage of our research, the comparison experiments between R-LLGMN and other classification methods were carried out, and high learning/classification performances of R-LLGMN were confirmed. The results of the EEG pattern classification experiments showed that R-LLGMN can realize a relatively high classification rate, and differences among subjects are not significant because of NNs incorporated. It has been shown that R-LLGMN is suitable for the classification of bioelectric signals such as EEG, since the filtering process as well as the discrimination have been merged together in the same network architecture.

In our future research, we would like to conduct a theoretical analyzes on the recurrent capabilities of R-LLGMN. The connections between the third and the fourth layers represent Alpha computation in CDHMM, and it can be expected that R-LLGMN can acquire any structures of a Markov model through learning, such as the ergotic model, the left-right model and so on. Also, our future research will be directed toward revealing potential ability of R-LLGMN comparing with CDHMM and improving the learning algorithm.

REFERENCES

[1] D. E. Rumelhart and J. L. McClelland, "Learning internal representations by error propagation," in *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, vol. 1, pp. 318–362.
[2] T. Caelli, D. M. Squire, and T. P. J. Wild, "Model-based neural networks," *Neural Networks*, vol. 6, no. 5, pp. 613–625, 1993.
[3] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate Bayesian *a posteriori* probabilities," *Neural Comput.*, vol. 3, pp. 461–483, 1991.
[4] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.
[5] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, Architectures, and Applications*, S. F. Fogelman and J. Hault, Eds. New York: Springer-Verlag, 1989, pp. 227–236.
[6] H. G. C. Tråvén, "A neural network approach to statistical pattern classification by 'semiparametric' estimation of probability density functions," *IEEE Trans. Neural Networks*, vol. 2, pp. 366–377, May 1991.
[7] L. I. Perlovsky and M. M. McManus, "Maximum likelihood neural networks for sensor fusion and adaptive classification," *Neural Networks*, vol. 4, no. 1, pp. 89–102, 1991.
[8] T. Tsuji, D. Mori, and T. Ito, "Motion discrimination method from EMG signal using statistically structured neural networks" (in Japanese), *Trans. Inst. Elect. Eng. Japan*, vol. 112-C, no. 8, pp. 465–473, 1992.
[9] T. Tsuji, O. Fukuda, H. Ichinobe, and M. Kaneko, "A log-linearized Gaussian mixture network and its application to EEG pattern classification," *IEEE Trans. Syst., Man, Cybern. C*, vol. 29, pp. 60–72, Feb. 1999.
[10] S. Lee and S. Shimoji, "Self-organization of Gaussian mixture model for learning class pdfs in pattern classification," in *Proc. IEEE Int. Joint Conf. Neural Networks*, vol. III, 1993, pp. 2492–2495.
[11] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood training of probabilistic neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 764–783, Sept. 1994.
[12] C. Bishop, *Neural Network for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.
[13] O. Fukuda, T. Tsuji, A. Ohtsuka, and M. Kaneko, "EMG-based human-robot interface for rehabilitation aid," in *Proc. IEEE Int. Conf. Robotics Automation*, Leuven, Belgium, 1998, pp. 3492–3497.
[14] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, 1989.
[15] H. Bersini, M. Saerens, and L. G. Sotelino, "Hopfield net generation, encoding and classification of temporal trajectories," *IEEE Trans. Neural Networks*, vol. 5, pp. 945–953, Nov. 1994.
[16] J. J. Hopfield, "Neural Networks and physical systems with emergent collective computational abilities," *Proc. Nat. Academy Sci. USA*, vol. 79, pp. 2554–2558, 1982.
[17] N. G. Bourbakis, C. Koutsougeras, and A. Jameel, "Handwritten character recognition using low resolutions," *Eng. Applicat. Artif. Intell.*, vol. 12, pp. 139–147, 1999.
[18] T. Lin, B. G. Horne, and C. L. Giles, "How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies," *Neural Networks*, vol. 11, no. 5, pp. 861–868, 1998.
[19] A. Petrosian, D. Prokhorov, R. Homan, R. Dasheiff, and D. Wunsch, II, "Recurrent neural network based prediction of epileptic seizures in intra- and extracranial EEG," *Neurocomputing*, vol. 30, pp. 201–218, 2000.

[20] A. Aussem, "Dynamical recurrent neural networks toward prediction and modeling of dynamical systems," *Neurocomputing*, vol. 28, pp. 207–232, 1999.
[21] J. Zhang, A. J. Morris, and E. B. Martin, "Long-term prediction models based on mixed order locally recurrent neural networks," *Comput. Chem. Eng.*, vol. 22, no. 7-8, pp. 1051–1063, 1998.
[22] C. Schittenkopf, G. Dorffner, and E. Dockner, "Fat tails and nonlinearity in volatility models: What is more important?,", Tech. Rep., OEFAI-99-05, 1999.
[23] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, pp. 1550–1560, Oct. 1990.
[24] M. Zak, "Terminal attractors for addressable memory in neural networks," *Phys. Lett. A*, vol. 133, no. 1,2, pp. 18–22, 1988.
[25] T. Tsuji, O. Fukuda, M. Kaneko, and K. Ito, "Pattern classification of time-series EMG signals using neural networks," *Int. J. Adaptive Contr. Signal Processing*, vol. 14, no. 8, pp. 829–848, 2000.
[26] L. E. Baum and T. Petrie, "Statistical inference for probabilistic function of finite state Markov chains," *Ann. Math. Stat.*, vol. 37, no. 6, pp. 1554–1563, 1966.
[27] L. R. Rabiner, "A tutorial on hidden Markov model and selected applications in speech recognition," *Proc. IEEE*, vol. 77, pp. 257–286, Feb. 1989.
[28] A. P. Dempster, N. M. Larid, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. Ser. B*, vol. 39, no. 1, pp. 1–38, 1977.
[29] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A minimization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Statist.*, vol. 41, pp. 164–171, 1970.
[30] B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 307–309, Mar. 1986.
[31] L. R. Rabiner, J. G. Wilpon, and B. H. Juang, "A segmental k-means training procedure for connected word recognition," *AT&T Tech. J.*, vol. 65, no. 3, pp. 21–32, 1986.
[32] H. Bourlard and C. J. Wellekens, "Links between Markov models and multilayer perceptrons," in *Advances in Neural Information Processing Systems I*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 502–510.
[33] M. Cohen, H. Franco, N. Morgan, D. Rumelhart, and V. Abrash, "Hybrid neural network/hidden Markov model continuous speech recognition," in *Proc. Int. Conf. Spoken Language Processing 1992*, Banff, AB, Canada, 1992, pp. 915–918.
[34] A. J. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Trans. Neural Networks*, vol. 5, pp. 298–305, Mar. 1994.
[35] C. D. Mitchell, M. P. Harper, and L. H. Jamieson, "Stochastic observation hidden Markov models," in *Proc. IEEE ICASSP'96*, 1996, pp. 617–620.
[36] N. Ström, "Phoneme probability estimation with dynamic sparsely connected artificial neural networks," *Free Speech J.*, vol. 1, no. 5, 1997.
[37] J. S. Bridle, "Alpha-nets: A recurrent 'Neural' network architecture with a hidden Markov model interpretation," *Speech Commun.*, vol. 9, no. 1, pp. 83–92, 1990.
[38] M. I. Jordan, "Serial order: A parallel distributed processing approach," Institute for Cognitive Science, University of California, San Diego, Tech. Rep. 8604, 1986.
[39] J. L. Elman, "Finding structure in time," *Cognitive Sci.*, vol. 14, pp. 179–211, 1990.

Toshio Tsuji (A'88–M'99) was born in Kyoto, Japan, on December 25, 1959. He received the B.E. degree in industrial engineering in 1982, the M.E. and Doctor of Engineering degrees in systems engineering in 1985 and 1989, all from Hiroshima University, Hiroshima, Japan.

He was a Research Associate from 1985 to 1994, and an Associate Professor from 1994 to 2002, with the Faculty of Engineering, Hiroshima University. He was a Visiting Professor with the University of Genova, Genova, Italy, from 1992 to 1993. He is currently a Professor with the Department of Artificial Complex Systems Engineering, Hiroshima University. He has been interested in various aspects of motor control in robot and human movements. His current research interests have focused on the control of EMG-controlled prostheses, and computational neural sciences, in particular, biological motor control.

Dr. Tsuji is a member of the Japan Society of Mechanical Engineers, Robotics Society of Japan, and Japanese Society of Instrumentation and Control Engineers.

Nan Bu was born in HangZhou, China, on November 9, 1975. He received the B.E. and M.E. degrees in mechanical engineering from Dalian University of Technology, Dalian, China, in 1998 and 2001, respectively. He is currently pursuing the Ph.D. degree in systems engineering at Hiroshima University, Hiroshima, Japan.

His research interests include the neural network, pattern discrimination, and bioelectric signal analysis.

Osamu Fukuda was born in Fukuoka, Japan, on September 30, 1969. He received the B.E. degree in mechanical engineering from Kyushu Institute of Technology, Iizuka, Japan, in 1993, and the M.E. and Ph.D. degrees in information engineering from Hiroshima University, Hiroshima, Japan, in 1997 and 2000, respectively.

From 1997 to 1999, he was a Research Fellow of the Japan Society for the Promotion of Science. He joined the Mechanical Engineering Laboratory, Agency of Industrial Science and Technology, Ministry of International Trade and Industry, Japan, in 2000. Since 2001, he has been a Member of the Assistive Device Technology Group, Institute for Human Science and Biomedical Engineering, National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan. His main research interests are human interfaces and neural networks.

Dr. Fukuda is a member of the Japan Society of Mechanical Engineers and the Robotics Society of Japan.

Makoto Kaneko (A'84–M'87–SM'00) received the B.S. degree in mechanical engineering from Kyushu Institute of Technology, Iizuka, Japan, in 1976, and the M.S. and Ph.D. degrees in mechanical engineering from Tokyo University, Tokyo, Japan, in 1978 and 1981, respectively.

From 1981 to 1990, he was a Researcher with the Mechanical Engineering Laboratory (MEL), Ministry of International Trade and Industry (MITI), Tsukuba Science City, Japan. From 1988 to 1989, he was a Postdoctoral Fellow with the Technical University of Darmstadt, Darmstadt, Germany. From 1990 to 1993, he was an Associate Professor with the Department of Computer Science and System Engineering, Kyushu Institute of Technology. Since October 1993, he has been with Hiroshima University, Hiroshima, Japan, as a Professor with Graduate School of Engineering. His research interests include tactile-based active sensing, grasping strategy, and medical robotics.

Dr. Kaneko received eight academic awards, including the Humboldt Research Award, IEEE ICRA the Best Manipulation Award, and IEEE IASTP the Outstanding Paper Award. He served as a Technical Editor of IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION from 1990 to 1994. He is a Member of the IEEE Robotics and Automation, Systems, Man, and Cybernetics, and Industrial Engineering Societies.