A Maximum Likelihood Neural Network Based on a Log-Linearized Gaussian Mixture Model

Toshio TSUJI*, Hiroyuki ICHINOBE*, Osamu FUKUDA* and Makoto KANEKO*

*Faculty of Engineering
Hiroshima University
1-4-1 Kagamiyama, Higashi-Hiroshima, 739 Japan
email: tsuji@huis.hiroshima-u.ac.jp

ABSTRACT

The present paper proposes a new probabilistic neural network based on a log-linearized Gaussian mixture model, which can estimate a posteriori probability for pattern classification problems. Although a structure of the proposed network represents a statistic model, a forward calculation and a backward learning rule based on the maximum likelihood estimation can be defined in the same manner as the error back propagation neural network model. It is shown from experiments that considerably high classification performance for small sample size of training data can be realized and a structure of the network is easily determined by an incorporated statistical model.

1. Introduction

Generally, an input for a pattern classification problem can be considered as a stochastic variable with a certain distribution. In this case, the pattern classification problem usually reduces to an estimation problem of a probability density function (pdf), since the classification can be performed according to the Bayes decision rule if a posteriori probability of the input pattern is obtained accurately.

Tråvén [1] proposed a neural network based on a semiparametric estimation of a pdf, which has a flexible structure to represent any distribution and includes a set of parameters of the specific distribution. A mixture model is a key of the semiparametric method, which approximates an unknown distribution by a weighted sum of a finite number of component densities. The Gaussian mixture model (GMM) using Gaussian component densities has been often utilized to combine with neural networks by some researchers (Tråvén [1], Perlovsky and McManus [2] and Lee and Shimoji [3]), and it was shown that the parameters of the GMM can be estimated through learning and the pdf can be approximated accurately. Most of these methods, however, only rearranged an iterative procedure of the maximum likelihood estimation and a forward computation of the posteriori probability in a way imitating the neural network.

On the other hand, Jordan and Jacobs [4] proposed the Hierarchical Mixture-of-Expert (HME) that incorporated a generalized linear model in the neural network. The HME consists of two kinds of subnetworks: expert networks and gating networks. For pattern classification problems, outputs of the gating network and the expert network are corresponding to the a priori probability and the pdf of the component of the mixture model. The a priori probability of each component varies depending on the input data, so that the HME can address a more flexible statistical model comparing with the mixture model. By introducing the gating networks, however, the HME includes a hierarchical internal structure and much more parameters than the ones of the GMM, which may lead to introduce a complicated learning algorithm and loose simplicity that is one of the attractive features of the neural network model.

The present paper proposes a new type of a feedforward probabilistic neural network based on the GMM and the log-linear model for pattern classification problems. By applying the log-linear model to a product of the mixture coefficient and the mixture component of the GMM, a semiparametric model of the pdf can be incorporated into the feedforward neural network and a simple learning algorithm based on the back propagation is still applicable. Also, the network structure such as an activation function of each unit, a number of layers and a number of units can be determined by the corresponding structure of the GMM incorporated in the network.

2. Log-linearized Gaussian Mixture Model

2.1. Gaussian Mixture Model

Here, a pdf $f(x)$ of a feature vector $x \in \Re^d$ is represented by a GMM with K classes:

$$f(x) = \sum_{k=1}^{K} \sum_{m=1}^{M_k} \alpha_{k,m} g(x\,;\mu^{(k,m)}, \Sigma^{(k,m)}) \quad , \tag{1}$$

$$\sum_{k=1}^{K} \sum_{m=1}^{M_k} \alpha_{k,m} = 1 \quad , \tag{2}$$

$$g(x\,;\mu^{(k,m)}, \Sigma^{(k,m)}) = (2\pi)^{-\frac{d}{2}} \left| \Sigma^{(k,m)} \right|^{-\frac{1}{2}} \times \exp\left[-\frac{1}{2} (x-\mu^{(k,m)})^{\mathrm{T}} (\Sigma^{(k,m)})^{-1} (x-\mu^{(k,m)}) \right] \quad , \tag{3}$$

where M_k ($k = 1, \cdots, K$) denotes the number of components of the class k; $\alpha_{k,m}$ denotes a mixture coefficient or a mixing proportion of each component $\{k, m\}$; and $\mu^{(k,m)} \in \Re^{d}$ and $\Sigma^{(k,m)} \in \Re^{d \times d}$ represent the mean vector and the covariance matrix of each component $\{k, m\}$. Note that $|\cdot|$ represents the determinant.

Now, let us consider a problem to classify an observed vector x into one of K classes. The Bayes decision theory determines a specific class if a posteriori probability of the vector belonging to the class is larger than the ones to any other classes. Using the GMM of the pdf of x, the posteriori probability $P(k \mid x)$ $(k = 1, \cdots, K)$ is given as

$$P(k \mid x) = \sum_{k=1}^{K} P(k, m \mid x) = \sum_{m=1}^{M_k} \frac{P(k, m)\, P(x \mid k, m)}{P(x)} \tag{4}$$

where $P(k, m)$ is the a priori probability of the class k and the component m, which corresponds to the mixing coefficient $\alpha_{k,m}$; and $P(x \mid k, m)$ is the pdf of x conditioned by the class k and the component m. Then, using (1), the posteriori probability $P(k, m \mid x)$ can be expressed as

$$P(k, m \mid x) = \frac{P(k, m)\, P(x \mid k, m)}{\sum_{k'=1}^{K} \sum_{m'=1}^{M_{k'}} P(k', m')\, P(x \mid k', m')} = \frac{\alpha_{k,m} g(x\,;\mu^{(k,m)}, \Sigma^{(k,m)})}{\sum_{k'=1}^{K} \sum_{m'=1}^{M_{k'}} \alpha_{k',m'} g(x\,;\mu^{(k',m')}, \Sigma^{(k',m')})} \quad . \tag{5}$$

2.2. Log-Linearization

Since $g(x\,;\mu^{(k,m)}, \Sigma^{(k,m)})$ is the d-dimensional Gaussian distribution given as (3), using the mean vector $\mu^{(k,m)} = (\mu_1^{(k,m)}, \cdots, \mu_d^{(k,m)})^{\mathrm{T}}$ and the inverse of the covariance matrix $\Sigma^{(k,m)-1} = [s_{ij}^{(k,m)}]$, the numerator of the right side of (5) can be represented as

$$\begin{aligned} &\alpha_{k,m} g(x\,;\mu^{(k,m)}, \Sigma^{(k,m)}) \\ &= \exp\Big[-\frac{1}{2} \sum_{j=1}^{d} \sum_{l=1}^{j} (2-\delta_{jl})\, s_{jl}^{(k,m)} x_j x_l + \sum_{j=1}^{d} \sum_{l=1}^{d} s_{jl}^{(k,m)} \mu_j^{(k,m)} x_l \\ &\quad - \frac{1}{2} \sum_{j=1}^{d} \sum_{l=1}^{d} s_{jl}^{(k,m)} \mu_j^{(k,m)} \mu_l^{(k,m)} - \frac{d}{2} \log 2\pi \\ &\quad - \frac{1}{2} \log \left| \Sigma^{(k,m)} \right| + \log \alpha_{k,m} \Big] \quad , \end{aligned} \tag{6}$$

where δ_{ij} is the Kronecker delta: $\delta_{ij} = 1$ when $i = j$ and $\delta_{ij} = 0$ when $i \neq j$.

Let us consider to linearize the right side of (6). Taking a logarithm of (6), we can get

$$\xi_{k,m} \triangleq \log \alpha_{k,m} g(x\,;\mu^{(k,m)}, \Sigma^{(k,m)}) = \beta^{(k,m)\mathrm{T}} X \quad , \tag{7}$$

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

$$X = (1, x^{\mathrm{T}}, x_1^2, x_1x_2, \cdots, x_1x_d, x_2^2, x_2x_3, \cdots, x_2x_d, \cdots, x_d^2)^{\mathrm{T}} \quad , \tag{8}$$

$$\beta^{(k,m)} = (\beta_0^{(k,m)}, \sum_{j=1}^{d} s_{j1}^{(k,m)} \mu_j^{(k,m)}, \cdots, \sum_{j=1}^{d} s_{jd}^{(k,m)} \mu_j^{(k,m)}, -\frac{1}{2} s_{11}^{(k,m)}, -s_{12}^{(k,m)}, \cdots, s_{1d}^{(k,m)}, \cdots, -\frac{1}{2} s_{dd}^{(k,m)})^{\mathrm{T}} \quad , \tag{9}$$

$$\beta_0^{(k,m)} = -\frac{1}{2} \sum_{j=1}^{d} \sum_{l=1}^{d} s_{jl}^{(k,m)} \mu_j^{(k,m)} \mu_l^{(k,m)} - \frac{d}{2} \log 2\pi - \frac{1}{2} \log \left| \Sigma^{(k,m)} \right| + \log \alpha_{k,m} \quad , \tag{10}$$

and the dimensionality H is defined as $H = 1 + d(d + 3)\,/\,2$. We can see that $\xi_{k,m}$ can be expressed as a product of the coefficient vector $\beta^{(k,m)}$ and the modified input vector $X \in \Re^{H}$.

However, since $\sum_{k=1}^{K} \sum_{m=1}^{M_k} P(k, m \mid x) = 1$, the variable $\xi_{k,m}$ is redundant. Then, a new variable $Y_{k,m}$ and a new coefficient vector $w^{(k,m)} \in \Re^{H}$ are introduced:

$$Y_{k,m} \triangleq \xi_{k,m} - \xi_{K,M_K} = (\beta^{(k,m)} - \beta^{(K,M_K)})^{\mathrm{T}} X = w^{(k,m)\mathrm{T}} X \quad , \tag{11}$$

where $w^{(K,M_K)} = \mathbf{0}$. It should be noted that $w^{(k,m)}$ becomes a weight coefficient with no constraints, which is a desirable feature for learning scheme of the neural network. Then the posteriori probability $P(k, m \mid x)$ of (5) can be computed as

$$P(k, m \mid x) = \frac{\exp[Y_{k,m}]}{\sum_{k'=1}^{K} \sum_{m'=1}^{M_{k'}} \exp[Y_{k',m'}]} \quad . \tag{12}$$

As mentioned above, by taking a logarithm of the pdf of each component, the posteriori probability can be expressed using the variable $Y_{k,m}$ that is a linear sum of the modified input vector X and the coefficient vector $w^{(k,m)}$: that is, the GMM is log-linearized. In the next section, the log-linearized Gaussian mixture model (LLGM) derived here is developed in a form of a feedforward neural network.

3. Neural Network Model

3.1. Network Structure

Figure 1 shows the neural network proposed in this paper, which is of a feedforward one with four layers. First, the input feature vector $x \in \Re^d$ is preprocessed and converted into the modified input vector $X \in \Re^H$ according to (8). The first layer consists of H units corresponding to the dimensionality of X and the identity function is used for the activation function of each unit. The relationship between input and output of each unit in the first layer is defined as:

$$ {}^{(1)}I_j = X_j \quad \text{and} \quad {}^{(1)}O_j = {}^{(1)}I_j \ , \qquad (13)$$

where ${}^{(1)}I_j$ and ${}^{(1)}O_j$ denote the input and output of the j-th unit in the first layer.

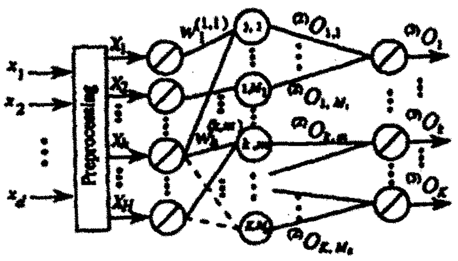


Fig.1 Structure of a LLGMN

The second layer consists of the same number of units as the total component number of the GMM $\sum_{k=1}^{K} M_k$. Each unit receives the output of the first layer weighted by a coefficient $w_h^{(k,m)}$ and outputs the posteriori probability of each component according to (12). The input to the unit $\{k, m\}$ in the second layer, $Y_{k,m}$, and the output, ${}^{(2)}O_{k,m}$, is defined as

$$ Y_{k,m} = \sum_{h=1}^{H} {}^{(1)}O_h w_h^{(k,m)} \ , \qquad (14)$$

$$ {}^{(2)}O_{k,m} = \frac{\exp[Y_{k,m}]}{\sum_{k'=1}^{K} \sum_{m'=1}^{M_{K}} \exp[Y_{k',m'}]} \ , \qquad (15)$$

where $w_h^{(K,M_K)} = 0$ $(h = 1, \cdots, H)$. It should be noted that (15) can be considered as a kind of generalized sigmoid functions.

Finally, the third layer consists of K units corresponding to the number of classes and outputs the posteriori probability of the class k $(k = 1, \cdots, K)$. The unit k integrates the outputs of M_k units $\{k, m\}$ $(m = 1, \cdots, M_k)$ in the second layer. The relationship between input and output is defined as

$$ {}^{(3)}I_k = \sum_{m=1}^{M_k} {}^{(2)}O_{k,m} \quad \text{and} \quad {}^{(3)}O_k = {}^{(3)}I_k \ . \qquad (1$$

In the log-linearized Gaussian mixture netwo (LLGMN) defined above, the posteriori probability each class can be calculated based on the log-lineariz Gaussian mixture structure incorporated in th network by learning only the weight coefficient $w_h^{(k,}$ between the first layer and the second layer.

3.2. Learning Rule

Now, let us consider a supervised learning with teacher vector $T^{(n)} = (T_1^{(n)}, \cdots, T_k^{(n)}, \cdots, T_K^{(n)})^T$ for th n-th feature vector $x^{(n)}$. If a teacher provides a perfe classification, $T_k^{(n)} = 1$ for the particular class k an $T_k^{(n)} = 0$ for all other classes. The network is traine using a given set including N data $x^{(n)} (n = 1, \cdots, N)$ Using the training data, a log-likelihood function L can be derived as

$$ L = \sum_{n=1}^{N} \sum_{k=1}^{K} T_k^{(n)} \log {}^{(3)}O_k \ , \qquad (17$$

where the output ${}^{(3)}O_k$ of the LLGMN corresponds t $P(k \mid x^{(n)})$. As an energy function for the network, w use

$$ J = \sum_{n=1}^{N} J_n = -\sum_{n=1}^{N} \sum_{k=1}^{K} T_k^{(n)} \log {}^{(3)}O_k \ , \qquad (18)$$

and the learning is performed to minimize it, that is, maximize the likelihood. For $x^{(n)}$, a weight modification $\Delta w_h^{(k,m)}$ of the corresponding weight $w_h^{(k,m)}$ $(h = 1, \cdots, H)$ is defined as

$$ \Delta w_h^{(k,m)} = -\eta \frac{\partial J_n}{\partial w_h^{(k,m)}} \ , \qquad (19)$$

in a sequential learning scheme, where η is a positive learning rate and

$$ \frac{\partial J_n}{\partial w_h^{(k,m)}} = \frac{\partial}{\partial w_h^{(k,m)}} \left(-\sum_{k=1}^{K} T_k^{(n)} \log {}^{(3)}O_k \right) $$
$$ = \left({}^{(2)}O_{k,m} - \frac{{}^{(2)}O_{k,m}}{{}^{(3)}O_k} T_k^{(n)} \right) X_h^{(n)} \ . \qquad (20)$$

It can be seen from the learning rule (20) that the teacher signal given for classes are distributed into each component according to the ratio of the posteriori probability ${}^{(2)}O_{k,m}$ of each component to the posteriori probability ${}^{(3)}O_k$ of the class k.

The learning rule derived in this paper can be applied using not only the perfect teacher signal of $\{0,1\}$ but also a probabilistic or fuzzy teacher signal which takes a continuous value of $[0,1]$. It should be noted that $\sum_{k=1}^{K} T_k^{(n)} = 1$ and $T_{k,m}^{(n)} \geq 0$ are required. Now, when the teacher signal vector $T^{(n)}$ and the output vector ${}^{(3)}O = ({}^{(3)}O_1, \cdots\}, {}^{(3)}O_k, \cdots\}, {}^{(3)}O_K)^T$ of the LLGMN are given, using the Kullback information that is often used as a pseudodistance between two pdfs, an energy function for learning is defined as

$$J' = \sum_{n=1}^{N} I_K(T^{(n)}; {}^{(3)}O) = \sum_{n=1}^{N}\sum_{k=1}^{K} T_k^{(n)} \log \frac{T_k^{(n)}}{{}^{(3)}O_k}$$
$$= \sum_{n=1}^{N}\sum_{k=1}^{K} T_k^{(n)} \log T_k^{(n)} - \sum_{n=1}^{N}\sum_{k=1}^{K} T_k^{(n)} \log {}^{(3)}O_k \geq 0 \ , \qquad (21)$$

where $I_K(\cdot;\cdot)$ represents the Kullback information. The first term of the right side of (21) is a constant, so that the second term should be minimized in order to come close ${}^{(3)}O$ to $T^{(n)}$. Since the second term is equal to the energy function J of (18), the learning rules (19), (20) derived for the perfect teacher signal also minimize the Kullback information defined as (21).

4. Simulation Experiments

4.1. Generalization Ability

To compare the LLGMN with the error back propagation neural network (BPN) [5] on generalization ability, pattern classification experiments are carried out using two dimensional data ($d = 2$, $H = 6$) for two classes ($K = 2$) which are artificially generated using the Gaussian mixture pdf with two components ($M_1 = M_2 = 2$). Table 1 indicates the parameters used in the GMM.

The LLGMN includes four units in the second layer which is corresponding to the total component number; six in the input layer; and two in the output layer. On the other hand, the BPN includes two units with the identity activation functions in the input layer corresponding to the dimension of x; ten units with the sigmoid functions in each of two hidden layers; and two units with the sigmoid functions in the output layer. Learning for both the networks is carried out until the energy function J_n of (18) is less than 0.1 for all training data. The teacher signal is given for each class (see (19), (20); $\eta = 0.0001$) and, for the BPN, two outputs are normalized to make the sum of the outputs 1.0.

Figure 2 indicates changes of the classification rate with the number of the training data. Both networks are trained using five sets of the training data independently, the numbers of data of which are 10, 20, 30, 40 and 50. Then, the ratio of the correct classification to 2000 data that are not used in learning (1000 for each class) are computed. In Fig. 2, the mean values and the standard deviations of the classification rate for ten kinds of initial weights which are randomly chosen are plotted. Although both networks can achieve high classification rate for large number of training data, a deference becomes clear as the number of the training data decreases. The LLGM keeps the classification rate high even for small sample size of the training data, whereas the classification rate of the BPN decreases.

Table 1 Parameters of the Gaussian mixture model used in the experiments

		$\alpha_{k,m}$		$\mu^{(k,m)T}$		$\Sigma^{(k,m)}$	
component, m		1	2	1	2	1	2
class, k	1	0.3	0.2	[0.0,0.0]	[5.0,7.0]	$\begin{bmatrix} 9.0 & 0.63 \\ 0.63 & 0.09 \end{bmatrix}$	$\begin{bmatrix} 1.0 & -0.5 \\ -0.5 & 1.0 \end{bmatrix}$
	2	0.25	0.25	[2.0,2.0]	[-6.0,2.0]	$\begin{bmatrix} 1.0 & 2.7 \\ 2.7 & 9.0 \end{bmatrix}$	$\begin{bmatrix} 0.09 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$

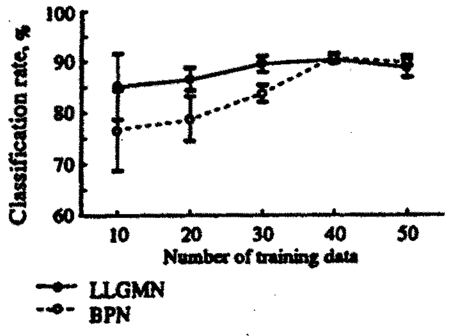
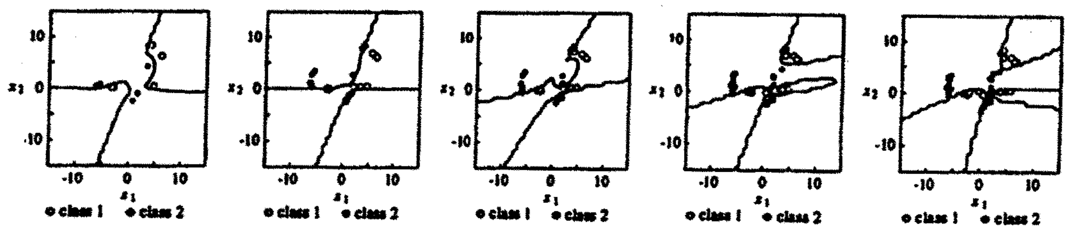
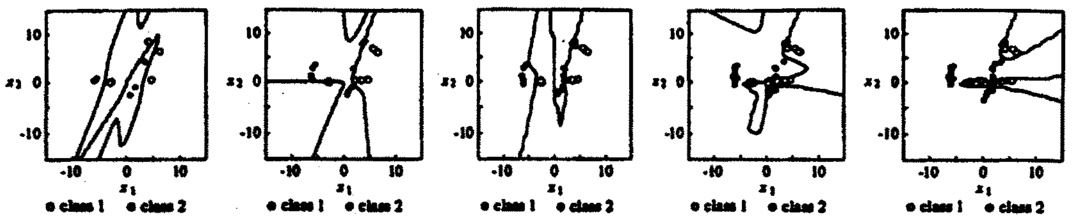


Fig.2 Effect of the number of training data on classification ability



(a) Number of training data : 10 (b) Number of training data : 20 (c) Number of training data : 30 (d) Number of training data : 40 (e) Number of training data : 50

Fig.3 Scatter diagram of training data and decision-region boundaries learned in the LLGMN



(a) Number of training data : 10 (b) Number of training data : 20 (c) Number of training data : 30 (d) Number of training data : 40 (e) Number of training data : 50

Fig.4 Scatter diagram of training data and decision region boundaries learned in the BPN

Decision region boundaries learned by the networks, on which the posteriori probabilities of both class become equal, are shown in Fig. 3 for the LLGMN and Fig. 4 for the BPN. In the BPN, the decision region boundaries are varied largely with the number of the training data. On the other hand, similar decision region boundaries can be obtained by the LLGMN in spite of changes of the number of the training data.

4.2. Representation Ability

The parameters of the GMM such as the mixing coefficient $\alpha_{k,m}$, the mean vector $\mu^{(k,m)}$and the covariance matrix $\Sigma^{(k,m)}$ include several constraints. For example, the covariance matrix must be invertible and the mixing coefficient $\alpha_{k,m}$ must be positive and the total sum of the mixing coefficients must be 1.0. On the other hand, the weight coefficients $w_h^{(k,m)}$ used in the LLGMN have no such constraints and are mutually independent. To evaluate this difference, the LLGMN is compared with the MLANS (Maximum Likelihood Artificial Neural System) [2] which was developed by a direct use of the GMM.

Classification capability of two networks are evaluated in a two dimensional feature space ($d = 2$, $H = 6$) for three classes ($K = 3$), in which classes A or B has a rectangular region and class C has two regions. The class pdf is constant in every region and the a priori probabilities of three classes are the same. Also, two regions belonging to class C have the same a priori probabilities. An example of training data is shown in Fig. 5 with four rectangular regions.

The LLGMN includes six units in the input layer; three in the output layer; and the same number of units as the total number of the components used in the MLANS in the second layer. The teacher signal is given for each class ((19), (20); $\eta = 0.5$) and learning is carried out until the mean value of the energy function J_n of (18) for all training data becomes less than 0.5. Note that for three training sets including 210, 270 and 330 data, dynamics of a terminal attractor is incorporated in the learning rule in order to speed it up [6]. On the other hand, in MLANS, learning procedure is continued until a Bhattacharyya distance of the posteriori probability with an iteration becomes less than 0.0001. For evaluating the classification ability, 3000 data (1000 for each class) are artificially generated, which are different from the training data.

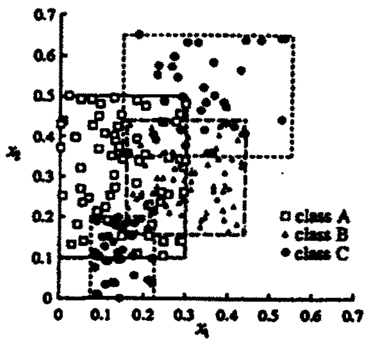


Fig.5 Scatter diagram of 210 training data

Figure 6 shows a classification result when the number of learning data is varied from 30 to 330, where the mean values and the standard deviations of the classification rates for 10 kinds of initial weights are plotted. A solid line and a dashed line show the results of the LLGMN and the MLANS, respectively. Note that both the number of components used in the MLANS and the number of units in the second layer of the LLGMN are nine, i.e., three for each class. When the number of the training data is sufficiently large, the classification rates of both networks are almost the same. As the number of the training data decreases, however, the classification rate of the MLANS becomes worse than the one of the LLGMN. Note that the covariance matrices included in the MLANS could not be estimated in the case of 30 training data, because the number of the data belonging to each component decreases remarkably when the number of the training data is small.

Next, Fig. 7 shows a classification result when the total number of components is varied from 3 to

12, where the same number of components in each class is prepared. The number of the training data is 210, i.e., 70 for each class, and the same convergence conditions as Fig. 6 are used except for the case of the LLGMN with three components in which the learning is carried out until the mean value of the energy function J becomes less than 0.6. By using the MLANS (a dotted line), classification is performed successfully when the number of components is enough large. For the small number of components, however, it becomes difficult to represent the distribution of data adequately, then the classification rate decreases. The standard deviation of the classification rate of the MLANS becomes large when the number of components is six, since three different learning results have been obtained depending on the initial weights. On the other hand, by the LLGMN, classification is still successful even if the number of components is small.

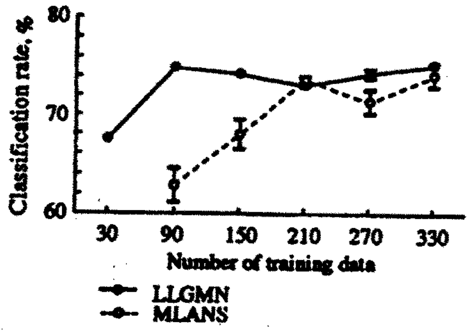


Fig.6 Effect of the number of training data on classification ability

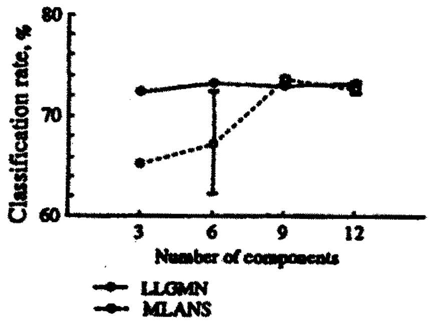


Fig.7 Effect of the number of components on classification ability

5. Conclusion

The present paper proposed a new neural network based on the log-linearized Gaussian mixture model, which can estimate the posteriori probability for pattern classification problems. Advantages of the LLGMN proposed here are summarized as follows: 1) the statistical structure incorporated in the LLGMN realizes considerably high classification ability for a small sample size of training data; 2) the LLGMN can achieve more general ability than the one of the GMM by relieving parameters of the GMM from their constraints; 3) the network structure such as an activation function of each unit, number of layers and number of units can be determined easily corresponding to the GMM incorporated in the network; and 4) the output from the LLGMN can be treated as a probability.

Future research will be directed to developing some techniques to regulate a component number.

This work was supported in part by Tateisi Science and Technology Foundation.

6. References

[1] H. G. C. Tråvén, "A Neural Network Approach to Statistical Pattern Classification by "Semiparametric" Estimation of Probability Density Functions," *IEEE Trans. on Neural Networks*, 2, 3, pp. 366 - 377, 1991.

[2] L. I. Perlov:ky and M. M. McManus, "Maximum Likelihood Neural Networks for Sensor Fusion and Adaptive Classification", *Neural Networks*, 4, pp. 89 - 102, 1991.

[3] S. Lee and S. Shimoji, "Self-Organization of Gaussian Mixture Model for Learning Class pdfs in Pattern Classification", *Proc. IJCNN*, 3, pp. 2492 - 2495, 1993.

[4] M. I. Jordan and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM algorithm", *Proc. IJCNN*, 2, pp. 1339 - 1344, 1993.

[5] D. E. Rumelhart, J. L. McClelland and R. J. Williams, "Learning Internal Representations by Error propagation," in *Parallel Distributed Processing vol.I*, pp. 318 - 362, MIT Press. 1986.

[6] O. Fukuda, T. Tsuji and M. Kaneko, "Pattern Classification of EEG Signals Using a Log-linearized Gaussian Mixture Neural Networks," *ICNN*, 1995 (submitting).