PERGAMON

Neural Networks 12 (1999) 1229–1252

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

# Improved learning algorithms for mixture of experts in multiclass classification

K. Chen[a, b], L. Xu[b,*], H. Chi[b]

[a]Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, NT, Hong Kong, People's Republic of China
[b]National Laboratory of Machine Perception and Center for Information Science, Peking University, Beijing 100871, People's Republic of China

Received 21 October 1996; received in revised form 26 February 1999; accepted 26 February 1999

**Abstract**

Mixture of experts (ME) is a modular neural network architecture for supervised learning. A double-loop Expectation-Maximization (EM) algorithm has been introduced to the ME architecture for adjusting the parameters and the iteratively reweighted least squares (IRLS) algorithm is used to perform maximization in the inner loop [Jordan, M.I., Jacobs, R.A. (1994). Hierarchical mixture of experts and the EM algorithm, *Neural Computation*, *6*(2), 181–214]. However, it is reported in literature that the IRLS algorithm is of instability and the ME architecture trained by the EM algorithm, where IRLS algorithm is used in the inner loop, often produces the poor performance in multiclass classification. In this paper, the reason of this instability is explored. We find out that due to an implicitly imposed incorrect assumption on parameter independence in multiclass classification, an incomplete Hessian matrix is used in that IRLS algorithm. Based on this finding, we apply the Newton–Raphson method to the inner loop of the EM algorithm in the case of multiclass classification, where the exact Hessian matrix is adopted. To tackle the expensive computation of the Hessian matrix and its inverse, we propose an approximation to the Newton–Raphson algorithm based on a so-called generalized Bernoulli density. The Newton–Raphson algorithm and its approximation have been applied to synthetic data, benchmark, and real-world multiclass classification tasks. For comparison, the IRLS algorithm and a quasi-Newton algorithm called BFGS have also been applied to the same tasks. Simulation results have shown that the use of the proposed learning algorithms avoids the instability problem and makes the ME architecture produce good performance in multiclass classification. In particular, our approximation algorithm leads to fast learning. In addition, the limitation of our approximation algorithm is also empirically investigated in this paper. © 1999 Published by Elsevier Science Ltd. All rights reserved.

*Keywords:* Mixture of experts; Multiclass classification; Multinomial density; Generalized Bernoulli density; Expectation-Maximization (EM) algorithm; Newton–Raphson method; Iterative reweighted least squares (IRLS) algorithm; BFGS algorithm

## 1. Introduction

There have recently been widespread interests in the use of multiple models for pattern classification and regression in statistics and neural network communities. The basic idea underlying these methods is the application of a so-called divide-and-conquer principle that is often used to tackle a complex problem by dividing it into simpler problems whose solutions can be combined to yield a final solution. Utilizing this principle, Jacobs, Jordan, Nowlan and Hinton (1991) proposed a modular neural network architecture called mixture of experts (ME). It consists of several expert networks trained on different partitions of the input space. The ME weights the input space by using the posterior probabilities that expert networks generated for getting the output from the input. The outputs of expert networks are combined by a gating network simultaneously trained in order to stochastically select the expert that is performing the best at solving the problem. The gating network is realized by the multinomial logit or softmax function (Bridle, 1989). As pointed out by Jordan and Jacobs (1994), the gating network performs a typical multiclass classification task. The ME architecture has been extended to a hierarchical structure called hierarchical mixtures of experts (HME) (Jordan & Jacobs, 1994). Moreover, Jordan and Jacobs (1994) have introduced the Expectation-Maximization (EM) algorithm (Dempster, Laird & Rubin, 1977) to both the ME and the HME architecture so that the learning process is decoupled in a manner that fits well with the modular structure. The favorable properties of the EM algorithm have been shown by theoretical analyses (Jordan & Xu, 1995; Xu & Jordan, 1996). In the ME architecture, the EM algorithm makes the original complicated maximum likelihood problem decomposed into several

* Corresponding author. Tel.: + 852-2609-8423; fax: + 852-2603-5024.
*E-mail address:* lxu@cse.cuhk.edu.hk (L. Xu)

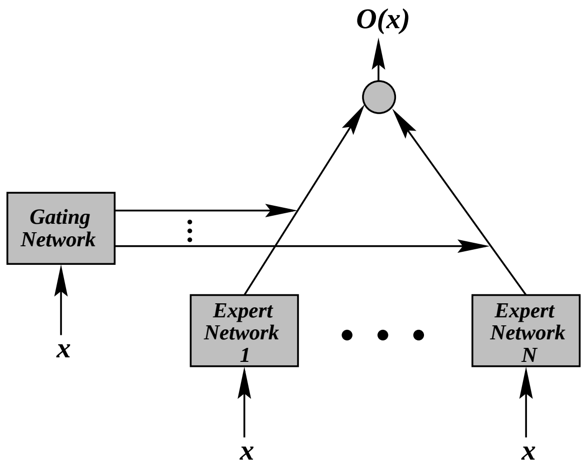


Fig. 1. The architecture of mixture of experts.

separate maximum likelihood problems in the E-step and solve these problems in the M-step (Jordan & Jacobs, 1994). Since these optimization problems are not usually analytically solvable, the EM algorithm is a double-loop procedure in general. To tackle these separate optimization problems in the inner loop, Jordan and Jacobs (1994) proposed an iteratively reweighted least squares (IRLS) algorithm for both regression and pattern classification.

The ME architecture with the EM algorithm has already been applied to both regression and pattern classification (for a review see Waterhouse, 1997). However, empirical studies indicate that the use of the IRLS algorithm in the inner loop of the EM algorithm causes the ME architecture to produce the instable performance. In particular, the problem becomes rather serious in multiclass classification. Our earlier studies show that an ME or an HME architecture cannot reach the steady state often when the IRLS algorithm is used to solve the maximization problems in the inner loop of the EM algorithm. The further observation shows that when the IRLS algorithm is used in the inner loop the-log-likelihood corresponding to an ME architecture does not monotonically increase during parameter estimation (Chen, Xie & Chi, 1995, 1996b). Similar problems have also been mentioned in the literature (Ramamurti & Ghosh, 1996; Ramamurti & Ghosh, 1997). As a result, Waterhouse (1993) transferred a multiclass classification task into several binary classification subtasks for speech recognition to implicitly avoid the instability problem. Alternatively, Chen et al. (1996b) used a so-called generalized Bernoulli density as the statistical model of expert networks for multiclass classification and applied such ME and HME classifiers to speaker identification.

Xu and Jordan (1994), and Xu, Jordan and Hinton (1995) have proposed an alternative ME model, where a localized gating network is employed so that parameter estimation in the gating network can be analytically solvable. For a regression task, the IRLS algorithm is avoided in the alternative ME model so that the EM algorithm becomes a single-loop procedure, and empirical studies show that the alternative ME model can reach the steady state and yield fast learning (Xu et al., 1995; Ramamurti & Ghosh, 1997). It has also been shown that, as a special case, the alternative ME model covers a class of radial basis function networks such that these networks can be trained by either the batch way or adaptive EM-like algorithm (Xu, 1996, 1998). In principle, the alternative ME model is also applicable to a multiclass classification task with the stable solution, as long as each expert has a structure similar to that of the gating network proposed by Xu and others (Xu et al., 1995; Xu & Jordan, 1994).

Since the original ME model with the IRLS algorithm for learning has been widely used in literature, and the reason behind the instability of the IRLS algorithm for multiclass classification still remains unknown, it is undoubtedly important to investigate the intrinsic reason. In this paper, we find out that an incorrect assumption on the parameter independence for multiclass classification is implicitly imposed and results in the use of an incomplete Hessian matrix in the IRLS algorithm, which causes the aforementioned instability of learning. On the basis of the investigation, we propose a Newton–Raphson algorithm to replace the original IRLS algorithm (Jordan & Jacobs, 1994) in the inner loop of the EM algorithm for multiclass classification. Using the proposed learning algorithm, we show that the use of the exact Hessian matrix makes the ME architecture perform well in multiclass classification. However, the use of the exact Hessian matrix could lead to expensive computation during learning. To speed up learning, we propose an approximation to the Newton–Raphson algorithm by introducing an approximate statistical model to expert networks for multiclass classification. In order to demonstrate their effectiveness, we have used the proposed learning algorithms in the inner loop of the EM algorithm to perform synthetic data, benchmark, and real-world multiclass classification tasks. Simulation results have shown that the proposed learning algorithms make the ME architecture produce the satisfactory performance in those multiclass classification tasks. In particular, the proposed approximation algorithm yields significantly faster learning. For comparison, we have also applied the IRLS algorithm and a quasi-Newton algorithm called BFGS in the inner loop of the EM algorithm, respectively, to train the ME architecture for the same tasks. Comparative results show that the ME architecture yields the poor performance when the IRLS algorithm is used in the inner loop of the EM algorithm and the BFGS algorithm does not yield significantly faster learning in contrast to the proposed learning algorithms.

The remainder of this paper is organized as follows. Section 2 briefly reviews the mixture of experts architecture and the EM algorithm. Section 3 analyzes the reason why the IRLS algorithm causes the ME architecture to produce the poor performance in multiclass classification. Section 4 proposes a Newton–Raphson algorithm used in the inner loop of the EM algorithm for multiclass classification and relates it to the IRLS algorithm. Section 5 presents an approximation to the Newton–Raphson algorithm to speed up learning. Simulation results are reported in Section 6. Further discussions are given in Section 7, and conclusions are drawn in the last section.

## 2. Mixtures of experts and EM algorithm

To make this paper self-contained, we briefly review the ME architecture (Jacobs et al., 1991) and the EM algorithm (Jordan & Jacobs, 1994) in this section.

As illustrated in Fig. 1, the ME architecture is composed of a gating network and several expert networks. The gating network receives the vector **x** as input and produces scalar outputs that are partition of unity at each point in the input space. Each expert network produces an output vector for an input vector. The gating network provides linear combination coefficients as veridical probabilities for expert networks and, therefore, the final output of the ME architecture is a convex weighted sum of all the output vectors produced by expert networks. Suppose that there are $N$ expert networks in the ME architecture. All the expert networks are linear with a single output nonlinearity that is also referred to as 'generalized linear' (McCullagh & Nelder, 1983). The $i$th expert network produces its output $\mathbf{o}_i(\mathbf{x})$ as a generalized linear function of the input $\mathbf{x}$:

$$\mathbf{o}_i(\mathbf{x}) = f(\mathbf{W}_i\mathbf{x}), \tag{1}$$

where $\mathbf{W}_i$ is a weight matrix and $f(\cdot)$ is a fixed continuous nonlinearity. The gating network is also generalized linear function, and its $i$th output, $g(\mathbf{x},\mathbf{v}_i)$, is the multinomial logit or softmax function of intermediate variables $\xi_i$ (Bridle, 1989; McCullagh & Nelder, 1983):

$$g(\mathbf{x},\mathbf{v}_i) = \frac{e^{\xi_i}}{\sum_{k=1}^{N} e^{\xi_k}}, \tag{2}$$

where $\xi_i = \mathbf{v}_i^{\mathrm{T}}\mathbf{x}$ and $\mathbf{v}_i$ is a weight vector. The overall output $\mathbf{o}(\mathbf{x})$ of the ME architecture is

$$\mathbf{o}(\mathbf{x}) = \sum_{k=1}^{N} g(\mathbf{x},\mathbf{v}_k)\mathbf{o}_k(\mathbf{x}). \tag{3}$$

The ME architecture can be given a probabilistic interpretation. For an input–output pair (**x**,**y**), the values of $g(\mathbf{v}_i,\mathbf{x})$ are interpreted as the multinomial probabilities associated with the decision that terminates in a regressive process that maps **x** to **y**. Once the decision has been made, resulting in a choice of regressive process $i$, the output **y** is then chosen from a probability density $P(y|x, W_i)$, where $\mathbf{W}_i$ denotes the set of parameters or weight matrix of the $i$th expert network in the model. Therefore, the total probability of generating **y** from **x** is the mixture of the probabilities of generating **y** from each component densities, where the mixing proportions are multinomial probabilities:

$$P(\mathbf{y}|\mathbf{x},\Phi) = \sum_{k=1}^{N} g(\mathbf{x},\mathbf{v}_k)P(\mathbf{y}|\mathbf{x},\mathbf{W}_k), \tag{4}$$

where $\Phi$ is the set of all the parameters including both expert and gating network parameters. Moreover, the probabilistic component of the model is generally assumed to be a Gaussian distribution in the case of regression, a Bernoulli distribution in the case of binary classification, and a multinomial distribution in the case of multiclass classification.

Based on the probabilistic model in Eq. (4), learning in the ME architecture is treated as a maximum likelihood problem. Jordan and Jacobs (1994) have proposed an EM algorithm for adjusting the parameters of the architecture. Suppose that the training set is given as $\chi = \{(\mathbf{x}_t,\mathbf{y}_t)\}_{t=1}^{T}$. The EM algorithm consists of two steps. For the $s$th epoch, the posterior probabilities $h_i^{(t)}$ $(i = 1,\ldots,N)$, which can be interpreted as the probabilities $P(i|\mathbf{x}_t,\mathbf{y}_t)$, are computed in the E-step as

$$h_i^{(t)} = \frac{g(\mathbf{x}_t,\mathbf{v}_i^{(s)})P(\mathbf{y}_t|\mathbf{x}_t,\mathbf{W}_i^{(s)})}{\sum_{k=1}^{N} g(\mathbf{x}_t,\mathbf{v}_k^{(s)})P(\mathbf{y}_t|\mathbf{x}_t,\mathbf{W}_k^{(s)})}. \tag{5}$$

The M-step solves the following maximization problems:

$$\mathbf{W}_i^{(s+1)} = \arg\max_{\mathbf{W}_i} \sum_{t=1}^{T} h_i^{(t)} \log P(\mathbf{y}_t|\mathbf{x}_t,\mathbf{W}_i), \tag{6}$$

and

$$V^{(s+1)} = \arg\max_{V} \sum_{t=1}^{T}\sum_{k=1}^{N} h_k^{(t)} \log g(\mathbf{x}_t,\mathbf{v}_k), \tag{7}$$

where $V$ is the set of all the parameters in the gating network. Therefore, the EM algorithm is summarized as

EM algorithm

1. For each data pair $(\mathbf{x}_t,\mathbf{y}_t)$, compute the posterior probabilities $h_i^{(t)}$ using the current values of the parameters.
2. For each expert network $i$, solve a maximization problem in Eq. (6) with observations $\{(\mathbf{x}_t,\mathbf{y}_t)\}_{t=1}^{T}$ and observation weights $\{h_i^{(t)}\}_{t=1}^{T}$.
3. For the gating network, solve the maximization problem in Eq. (7) with observations $\{(\mathbf{x}_t,h_k^{(t)})\}_{t=1}^{T}$.
4. Iterate by using the updated parameter values.

## 3. The IRLS algorithm and its problem in multiclass classification

### 3.1. The IRLS algorithm

Apparently, the performance of an EM algorithm highly depends upon solutions to those separate maximization problems. As pointed out by Jordan and Jacobs (1994), the separate maximization problems in Eqs. (6) and (7) belong to the IRLS problem (McCullagh & Nelder, 1983). To tackle those separate optimization problems, Jordan and Jacobs (1994) propose an IRLS algorithm for all the generalized linear models used in the ME architecture. To explore the reason of instability of the IRLS algorithm in multiclass classification, here we first briefly review the IRLS algorithm.[1]

The likelihood in the generalized linear models is a product of densities from the exponential family distributions. The general form of a density in the exponential family is denoted as

$$P(y,\eta,\phi) = \exp\left\{\frac{\eta y - b(\eta)}{\phi} + c(y,\phi)\right\}, \tag{8}$$

where $\eta$ is known as the 'natural parameter' and $\phi$ is the dispersion parameter. Note that the form in Eq. (8) is for scalar-valued random variables. In a generalized linear model, moreover, the parameter $\eta$ is modeled as a linear function of the input $\mathbf{x}$:

$$\eta = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{x},$$

where $\boldsymbol{\beta}$ is a parameter vector. For a data set $\chi = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, according to Eq. (8), the log-likelihood is

$$l(\boldsymbol{\beta},\chi) = \sum_{t=1}^{T}\left\{\frac{\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}_t y_t - b(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}_t)}{\phi} + c(y_t,\phi)\right\}. \tag{9}$$

The observations $y_t$ are assumed to be sampled independently from densities $P(y,\eta_t,\phi)$, where $\eta_t = \boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}_t$. Based on the log-likelihood, the link function in the generalized linear model is defined as

$$f(\eta_t) = E[y_t] = b'(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}_t), \tag{10}$$

and the variance function is defined as

$$\mathrm{Var}[y_t] = \phi b''(\boldsymbol{\beta}^{\mathrm{T}}\mathbf{x}_t). \tag{11}$$

As a result, the gradient of the log-likelihood is

$$\frac{\partial l(\boldsymbol{\beta},\chi)}{\partial\boldsymbol{\beta}} = \mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{e}, \tag{12}$$

and the Hessian matrix of the log-likelihood is

$$\frac{\partial^2 l(\boldsymbol{\beta},\chi)}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^{\mathrm{T}}} = -\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}, \tag{13}$$

where $\mathbf{e}$ is the vector whose components are

$$e_t = \frac{y_t - f(\eta_t)}{f'(\eta_t)},$$

and $\mathbf{X}$ is the matrix whose rows are the input vector $\mathbf{x}_t$ and $\mathbf{W}$ is a diagonal matrix whose diagonal elements are $w_t$ in the following form:

$$w_t = \frac{[f'(\eta_t)]^2}{\mathrm{Var}[y_t]}.$$

The IRLS algorithm updates the parameter estimates $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} - \left[\frac{\partial^2 l(\boldsymbol{\beta},\chi)}{\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^{\mathrm{T}}}\right]^{-1}\frac{\partial l(\boldsymbol{\beta},\chi)}{\partial\boldsymbol{\beta}}$$
$$= \boldsymbol{\beta}^{(s)} + (\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{e}. \tag{14}$$

As pointed out by Jordan and Jacobs (1994), it is easy to generalize the algorithm to allow additional fixed observation weights to be associated with the data pairs. Such weights simply multiply the iteratively varying weights $w_t$, resulting in an iteratively reweighted weighted least squares algorithm. Such a generalization is necessary for the problems described in Eq. (6). That is, the EM algorithm defines an observation weights in the outer loop that the IRLS algorithm must treat as fixed during the inner loop (Jordan & Jacobs, 1994).

For the IRLS algorithm, furthermore Jordan and Jacobs (1994) point out that it is straightforward to generalize the algorithm to the case of vector outputs. As a result, each row of the weight matrix is a separate parameter vector corresponding to the aforementioned parameter vector $\beta$ in the case of vector outputs, and these row vectors are updated independently and in parallel (Jordan & Jacobs, 1994).

### 3.2. Problem in multiclass classification

In this subsection, we show that the IRLS algorithm cannot be straightforward generalized to the case of vector outputs from the scalar valued derivation in the case of multiclass classification.

Consider a multiclass classification problem on $K$ variables, $y_1, y_2, \ldots, y_K$ $(K > 2)$. A natural probabilistic model for multiclass classification is the multinomial density as follows,

$$P(y_1,y_2,\ldots,y_K) = \frac{M!}{(y_1!)(y_2!)\cdots(y_K!)}p_1^{y_1}p_2^{y_2}\cdots p_K^{y_K}, \tag{15}$$

where $p_k (k = 1,\ldots,K)$ are the multinomial probabilities associated with the different classes and $\sum_{k=1}^{K} p_k = 1$. As a member of the exponential family, the multinomial density can be written as

$$> P(y_1,y_2,\ldots,y_K) = \exp\left\{\log\left[\frac{M!}{(y_1!)(y_2!)\cdots(y_K!)}\right] + \sum_{k=1}^{K} y_k \log p_k\right\} = \exp\left\{\log\left[\frac{M!}{(y_1!)(y_2!)\cdots(y_K!)}\right] + \sum_{k=1}^{K-1} y_k \log\left[\frac{p_k}{p_K}\right] + M\log p_K\right\}. \tag{16}$$

We define $\eta_k = \log(p_k/p_K)$. Due to $\sum_{k=1}^{K} p_k = 1$, each $p_k$ in

[1] For details, one should be referred to Appendix A (Jordan & Jacobs, 1994).

the multinomial logit model is expressed as

$$p_k = \begin{cases} \dfrac{e^{\eta_k}}{1 + \sum_{i=1}^{K-1} e^{\eta_i}} & k \neq K, \\[2ex] \dfrac{1}{1 + \sum_{i=1}^{K-1} e^{\eta_i}} & k = K. \end{cases} \tag{17}$$

Accordingly, the link function for the multinomial logit model[2] is

$$f(\eta_k) = Mp_k, \tag{18}$$

for all $k$. For multiclass classification, we usually take $M = \sum_{k=1}^{K} y_k$ to equal one since $y_k$ $(k = 1,\ldots,K)$ are veridical probabilities in this case. Therefore, the link function becomes $f(\eta_k) = p_k$. Obviously, it is just the softmax function (Bridle, 1989) when $\eta_K = 0$. Accordingly, the variance function is

$$\mathrm{Var}[y_k] = \phi f'(\eta_k) = \phi p'_k. \tag{19}$$

Since $\phi = 1$ in the multinomial logit model according to Eq. (16), we have $\mathrm{Var}[y_k] = p'_k$. In a generalized linear model, each $\eta_k$ can be modeled as a linear function of the input $\mathbf{x}$, i.e. $\eta_k = \boldsymbol{\beta}_k^{\mathrm{T}}\mathbf{x}$, where $k \neq K$ for the multinomial logit model. Note that the multinomial logit model merely consists of $K - 1$ independent parameter vectors, say $\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{K-1}$, instead of $K$ independent variables due to the constraint $\sum_{k=1}^{K} p_k = 1$.

Let us denote all the parameters in the multinomial logit model as $\Theta$, i.e. $\boldsymbol{\Theta} = [\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{K-1}]^{\mathrm{T}}$. For a sample, $\mathbf{x}_t$, in a data set $\chi = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$, we rewrite $p_{\mathrm{k}}$ in Eq. (17) as $g_k(\boldsymbol{\Theta}, \mathbf{x}_t)$ with the explicit parameter form:

$$g_k(\boldsymbol{\Theta}, \mathbf{x}_t) = \begin{cases} \dfrac{e^{\boldsymbol{\beta}_k^{\mathrm{T}}\mathbf{x}_t}}{1 + \sum_{i=1}^{K-1} e^{\boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}_t}} & k \neq K, \\[2ex] \dfrac{1}{1 + \sum_{i=1}^{K-1} e^{\boldsymbol{\beta}_i^{\mathrm{T}}\mathbf{x}_t}} & k = K \end{cases}. \tag{20}$$

In multiclass classification, therefore, the probabilistic model of an expert network is

$$P(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\Theta}) = \prod_{k=1}^{K} [g_k(\boldsymbol{\Theta}, \mathbf{x}_t)]^{y_{tk}}, \tag{21}$$

where $y_t = \{y_{t1}, y_{t2},\ldots,y_{tK}\}, \sum_{k=1}^{K} y_{tk} = 1$, and $y_{tk} \geq 0$. For the data set $\chi$, the objective function corresponding to the maximization problem in Eq. (6) can be written as

$$\begin{aligned} Q_i^e(\boldsymbol{\Theta}, \chi) &= \sum_{t=1}^{T} h_i^{(t)} \log P(\mathbf{y}_t|\mathbf{x}_t, \boldsymbol{\Theta}) \\ &= \sum_{t=1}^{T}\sum_{k=1}^{K} h_i^{(t)} y_{tk} \log g_k(\boldsymbol{\Theta}, \mathbf{x}_t) = \sum_{k=1}^{K} l_{ik}(\boldsymbol{\Theta}, \chi), \end{aligned} \tag{22}$$

where $l_{ik}(\boldsymbol{\Theta}, \chi) = \sum_{t=1}^{T} h_i^{(t)} y_{tk} \log g_k(\boldsymbol{\Theta}, \mathbf{x}_t)$. For $k \neq K$, obviously, $l_{ik}(\boldsymbol{\Theta}, \chi)$ is a part of the log-likelihood $Q_i^e(\boldsymbol{\Theta}, \chi)$ corresponding to the $k$th component of vector outputs. According to Eq. (20), we observe that $l_{ik}(\boldsymbol{\Theta}, \chi)$ is relevant not only to the parameter $\beta_{\mathrm{k}}$, i.e. the $k$th row vector of the weight matrix in an expert network, but also to other parametrs in $\Theta$, i.e. those row vectors corresponding to other output components. In other words, each row of weight matrix for an expert network is not a separate parameter vector only corresponding to the vector $\beta_k$, and these row vectors cannot be updated independently and in parallel in multiclass classification. Thus, the independence assumption on parameter vectors, used implicitly in Jordan and Jacobs (1994), is incorrect for multiclass classification and, therefore, the derivation of scalar output cannot be straightforward generalized to the case of vector outputs though indeed the multinomial density belongs to the exponential family.

Similarly, the objective function in Eq. (7), corresponding to the maximization problem associated with the gating network, can be also written as

$$Q^g(v) = \sum_{t=1}^{T} \log\left(\prod_{k=1}^{N} [g(\mathbf{x}_t, \mathbf{v}_k)]^{h_k^{(t)}}\right), \tag{23}$$

where $h_k^{(t)}$ is the posterior probability for expert $k, \sum_{i=1}^{N} h_k^{(t)} = 1$, and $h_k^{(t)} \geq 0$. If the IRLS algorithm is used for parameter estimation in the gating network, the same problem will occur since its statistical model is also a multinomial logit model in the original ME architecture.

Here, we emphasize that it is this incorrect independence assumption on parameter vectors in the IRLS algorithm that results in an incomplete Hessian matrix used and thus causes the aforementioned instability in learning.

## 4. A Newton–Raphson algorithm and its relation to the IRLS algorithm

In this section, we first propose a learning algorithm based on the Newton–Raphson method for use in the inner loop of the EM algorithm, and then discuss the relation between the IRLS algorithm and the proposed learning algorithm. It is followed by a multiclass classification example to demonstrate that the use of the exact Hessian matrix makes the ME architecture perform well, while the use of the IRLS algorithm suggested by Jordan and Jacobs (1994) causes the ME architecture to produce the poor performance.

[2] For details, one should be referred to Appendix B (Jordan & Jacobs, 1994).

## 4.1. Newton–Raphson algorithm

For the multinomial logit model, here we derive a learning algorithm for the maximization problems in Eqs. (6) and (7) based on the Newton–Raphson method. Note that the derivation of the Newton–Raphson algorithm is merely for those mixture components belonging to generalized linear models. It is not difficult to extend the proposed learning algorithm to a general case that the mixture components are not generalized linear models[3].

Let $\chi = \{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$ is a given data set, where $\mathbf{y}_t = [y_{t1},\ldots,y_{tK}]^{\mathrm{T}}$, $\sum_{k=1}^{K} y_{tk} = 1$, and $y_{tk} \geq 0$. For the multinomial logit model, its log-likelihood for the data set $\chi$ can be written as

$$l(\mathbf{\Theta}, \chi) = \sum_{t=1}^{T} \sum_{k=1}^{K} y_{tk} \log g_k(\mathbf{\Theta}, \mathbf{x}_t), \tag{24}$$

where $g_k(\mathbf{\Theta}, \mathbf{x}_t)$ is the same as described in Eq. (20) and $\mathbf{\Theta} = [\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{K-1}]^{\mathrm{T}}$.

Before the derivation of the Newton–Raphson algorithm, we first introduce a set of quantities $\eta_{\mathrm{tq}}$ defined by

$$\eta_{tq} = \boldsymbol{\beta}_q^{\mathrm{T}} \mathbf{x}_t, \quad q = 1,\ldots,K-1. \tag{25}$$

To evaluate the derivatives of the log-likelihood in Eq. (24), the derivative of the softmax function in Eq. (20) is useful. To facilitate the presentation, we give its calculation method in Appendix. According to the derivative of the softmax function, we can obtain

$$\frac{\partial[\log g_k(\mathbf{\Theta}, \mathbf{x}_t)]}{\partial \beta_q} = \frac{1}{g_k(\mathbf{\Theta}, \mathbf{x}_t)} \frac{\partial g_k(\mathbf{\Theta}, \mathbf{x}_t)}{\partial \beta_q} \frac{\partial \eta_{tq}}{\partial \beta_q}$$

$$= [\delta_{kq} - g_q(\mathbf{\Theta}, \mathbf{x}_t)]\mathbf{x}_t, \tag{26}$$

where $k = 1,2,\ldots,K$and $q = 1,2,\ldots,K-1$. Therefore, the derivatives of $l(\mathbf{\Theta}, \chi)$ on $\beta_q$ $(q = 1,\ldots K-1)$ are

$$J_q(\mathbf{\Theta}, \chi) = \frac{\partial l(\mathbf{\Theta}, \chi)}{\partial \boldsymbol{\beta}_q} = \sum_{t=1}^{T} \sum_{k=1}^{K} y_{tk} \frac{\partial l[\log g_k(\mathbf{\Theta}, \mathbf{x}_t)]}{\partial \boldsymbol{\beta}_q}$$

$$= \sum_{t=1}^{T} \sum_{k=1}^{K} y_{tk}[\delta_{kq} - g_q(\mathbf{\Theta}, \mathbf{x}_t)]\mathbf{x}_t$$

$$= \sum_{t=1}^{T} [y_{tq} - g_q(\mathbf{\Theta}, \mathbf{x}_t)]\mathbf{x}_t. \tag{27}$$

Here we have used the constraint $\sum_{k=1}^{K} y_{tk} = 1$ in the last step.

In the Newton–Raphson method (Fletcher, 1987), The Hessian matrix needs to be calculated based on a given data set. Since there are $K-1$ parameter vectors, $\boldsymbol{\beta}_1,\ldots,\boldsymbol{\beta}_{K-1}$, in the multinomial logit model described in Eq. (16), the Hessian matrix $\mathbf{H}(\mathbf{\Theta},\chi)$ consists of $(K-1) \times (K-1)$ block matrices $H_{qr}$ $(q, r = 1,\ldots,K-1)$. Based on Eqs. (27) and (A.4) in Appendix, we achieve each block matrix $\mathbf{H}_{\mathrm{qr}}$ as

$$\mathbf{H}_{qr} = \frac{\partial^2 l(\mathbf{\Theta}, \chi)}{\partial \boldsymbol{\beta}_\gamma^{\mathrm{T}} \beta_r^{\mathrm{T}}} = -\sum_{t=1}^{T} \frac{\partial [g_q(\mathbf{\Theta}, \mathbf{x}_t)\mathbf{x}_t]}{\partial \boldsymbol{\beta}_r^{\mathrm{T}}}$$

$$= -\sum_{t=1}^{T} g_q(\mathbf{\Theta}, \mathbf{x}_t)[\delta_{qr} - g_r(\mathbf{\Theta}, \mathbf{x}_t)]\mathbf{x}_t \mathbf{x}_t^{\mathrm{T}}. \tag{28}$$

We now assemble the various pieces for use in the Newton–Raphson method. First of all, Eq. (27) is utilized to represent the overall gradient vector of $l(\mathbf{\Theta}, \chi)$ on all the parameters in $\mathbf{\Theta}$, $\mathbf{J}(\mathbf{\Theta}, \chi)$, as

$$\mathbf{J}(\mathbf{\Theta}, \chi) = \frac{\partial l(\mathbf{\Theta}, \chi)}{\partial \mathbf{\Theta}} = [J_1(\mathbf{\Theta}, \chi),\ldots,J_{K-1}(\mathbf{\Theta}, \chi)]^{\mathrm{T}}. \tag{29}$$

Then, the Hessian matrix is denoted based on Eq. (28) as follows:

$$\mathbf{H}(\mathbf{\Theta}, \chi) = \frac{\partial^2 l(\mathbf{\Theta}, \chi)}{\partial \mathbf{\Theta} \partial \mathbf{\Theta}^{\mathrm{T}}}$$

$$= \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \cdots & \mathbf{H}_{1(K-1)} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \cdots & \mathbf{H}_{2(K-1)} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{H}_{(K-1)1} & \mathbf{H}_{(K-1)2} & \cdots & \mathbf{H}_{(K-1)(K-1)} \end{bmatrix}. \tag{30}$$

The Newton–Raphson method (Fletcher, 1987) updates the parameter estimates $\mathbf{\Theta}$ as follows:

$$\mathbf{\Theta}^{(S+1)} = \mathbf{\Theta}^{(s)} - \alpha \mathbf{H}^{-1}(\mathbf{\Theta}^{(s)}, \chi)\mathbf{J}(\mathbf{\Theta}^{(s)}, \chi), \tag{31}$$

where $\alpha$ is the learning rate and $\alpha \leq 1$. Note that the learning rate is unnecessary in the standard Newton–Raphson method. Here, the learning rate is adopted to speed up learning so that $\mathbf{\Theta}^{(s+1)}$ can be still in the neighborhood of $\mathbf{\Theta}^{(s)}$ for convergence (Minoux, 1986).

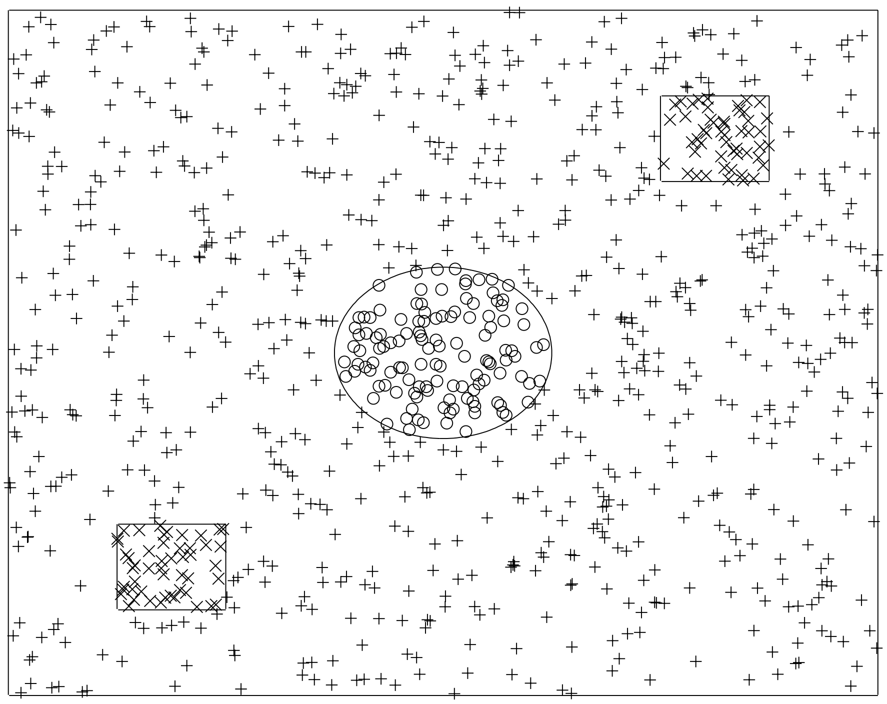
Like the IRLS algorithm (Jordan & Jacobs, 1994), it is easy to generalize the derivation to allow fixed weights to be associated with data pairs. As mentioned before, such a generalization is necessary for the ME architecture with the EM algorithm since the EM algorithm defines observation weights or posterior probabilities in the outer loop that the algorithm must treat as fixed during the inner loop. For problems in Eq. (6), therefore, Eq. (27) becomes

$$\frac{\partial l(\mathbf{\Theta}, \chi)}{\partial \boldsymbol{\beta}_q} = \sum_{t=1}^{T} h_i^{(t)}[y_{tq} - g_q(\mathbf{\Theta}, \mathbf{x}_t)]\mathbf{x}_t, \tag{32}$$

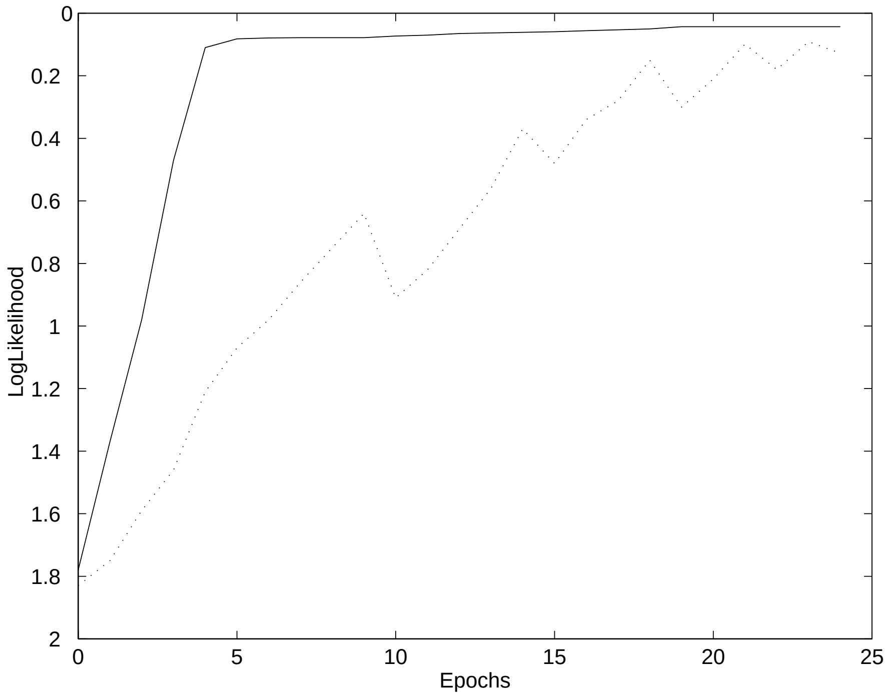
and Eq. (28) becomes

$$\frac{\partial^2 l(\mathbf{\Theta}, \chi)}{\partial \boldsymbol{\beta}_q \boldsymbol{\beta}_r^{\mathrm{T}}} = -\sum_{t=1}^{T} h_i^{(t)} g_q(\mathbf{\Theta}, \mathbf{x}_t)[\delta_{qr} - g_r(\mathbf{\Theta}, \mathbf{x}_t)]\mathbf{x}_t \mathbf{x}_t^{\mathrm{T}}. \tag{33}$$

[3] In this case, one may be referred to the work (Bishop, 1991, 1992) for the exact calculation of the Hessian matrix on the parameters relevant to hidden layers of a neural network in the multinomial logit model.



(a)



(b)

Fig. 2. The synthetic three-category problem. (a) The training set, where $\times$, $\bigcirc$, and $+$ stand for samples belonging to classes 1, 2, and 3, respectively. (b) Evolution of log-likelihood vs. epochs, where solid and dotted curves denote results produced by the proposed Newton–Raphson algorithm and the IRLS algorithm.

### *4.2. Relation between the IRLS algorithm and the Newton–Raphson algorithm*

From the derivation of the proposed learning algorithm, furthermore we find out that for the multinomial logit model the independence assumption on parameter vectors in the IRLS algorithm causes the Hessian matrix of the log-likelihood $l(\boldsymbol{\Theta}, \chi)$ to be incomplete; that is, the diagonal block

Table 1
Simulation results of a synthetic three-category problem: classification results and relative learning time when the IRLS algorithm and the proposed Newton–Raphson algorithm are used in the inner loop of the EM algorithm, respectively. Correct no. and error no. denote the number of classified and misclassified samples, respectively, and the notation is also used in other tables

| | Training set | | Testing set | | |
|---|---|---|---|---|---|
| Algorithm | Correct no. | Error no. | Correct no. | Error no. | Relative time |
| Newton–Raphson | 928 | 22 | 2265 | 235 | 1.0000 |
| IRLS algorithm | 869 | 81 | 1971 | 529 | 0.3514 |

matrices merely remain while all off-diagonal block matrices are treated as zero block matrices. In the sequel, we show such a relation between the IRLS algorithm and the proposed learning algorithm.

First, we define a set of quantities $w_{tq}$ *as*

$$w_{tq} = \frac{\partial g_q(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \boldsymbol{\beta}_q} = g'_q(\boldsymbol{\Theta}, \mathbf{x}_t) = \frac{[g'_q(\boldsymbol{\Theta}, \mathbf{x}_t)]^2}{\mathrm{Var}[y_{tq}]},$$

where $t = 1, \ldots, T$ and $q = 1, \ldots, K-1$. From Eqs. (19) and (20), it is known that $\mathrm{Var}[y_{tq}] = g'_q(\boldsymbol{\Theta}, \mathbf{x}_t)$ in the multinomial logit model. Therefore, we can rewrite Eq. (27) as

$$\mathbf{J}_q(\boldsymbol{\Theta}, \chi) = \sum_{t=1}^{T} [y_{tq} - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)]\mathbf{x}_t = \sum_{t=1}^{T} [y_{tq} - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)]\mathbf{x}_t \frac{w_{tq}}{g'_q(\boldsymbol{\Theta}, \mathbf{x}_t)} = \mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{e}, \quad (34)$$

where $\mathbf{X}$ is the matrix whose rows are the input vectors $\mathbf{x}_t$ and $\mathbf{W}$ is a diagonal matrix whose diagonal elements are $w_{\mathrm{tq}}$. Here $\mathbf{e}$ is the vector whose components are

$$e_{tq} = \frac{y_{tq} - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)}{g'_q(\boldsymbol{\Theta}, \mathbf{x}_t)}$$

Similarly, we rewrite $\mathbf{H}_{qq}$ as

$$\mathbf{H}_{qq} = \frac{\partial^2 l(\boldsymbol{\Theta}, \chi)}{\partial \boldsymbol{\beta}_q \boldsymbol{\beta}_q^{\mathrm{T}}} = -\sum_{t=1}^{T} g_q(\boldsymbol{\Theta}, \mathbf{x}_t)[1 - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)]\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}} = -\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X}. \quad (35)$$

Here we have used the fact of $g'_q(\boldsymbol{\Theta}, \mathbf{x}_t) = g_q(\boldsymbol{\Theta}, \mathbf{x}_t) \times [1 - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)]$ in the second step. Now we treat all off-diagonal block matrices as zero block matrices and neglect them. Utilizing Eqs. (34) and (35), we apply the standard Newton–Raphson method to update parameters $\boldsymbol{\Theta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{K-1}]^{\mathrm{T}}$ as

$$\beta_q^{(s+1)} = \beta_q^{(s)} - [\mathbf{H}_{qq}(\boldsymbol{\Theta}^{(s)}, \chi)]^{-1}\mathbf{J}_q(\boldsymbol{\Theta}^{(s)}, \chi) = \beta_q^{(s)} + (\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{W}\mathbf{e}, \quad (36)$$

where $q = 1, \ldots, K-1$. Obviously, Eq. (36) provides exactly the update formula used in the IRLS algorithm (Jordan & Jacobs, 1994).

## 4.3. Example: A three-category problem

In the subsection, we use both the IRLS algorithm and our learning algorithm in the inner loop of the EM algorithm for a synthetic three-category problem in order to illustrate the instability problem caused by the IRLS algorithm from an empirical perspective.

In this realistic problem, there are three categories in the data set. As illustrated in Fig. 2(a), two small rectangles, an ellipses, and other regions in the large rectangle constitute three different categories denoted classes 1, 2, and 3, respectively. As a training set, 950 points or 2D vectors are produced by a uniformly distributed random number producer. Among these points, 100, 122, and 738 points belong to classes 1, 2, and 3, respectively, indicated by different symbols in Fig. 2(a). The task is to construct a classifier which can distinguish among the three classes. Obviously, the classification task is a nonlinearly separable problem and can be used to evaluate the capability of a classifier in nonlinear separability and generalization.

In simulations, the ME architecture consisting of 12 experts was chosen as the classifier and randomly initialized. Each epoch was composed of a complete *E*-step and *M*-step in the EM algorithm, where both the IRLS algorithm and the Newton–Raphson algorithm were used in the inner loop, respectively. In the *M*-step, the termination condition is that either the log-likelihood reaches the steady state or the number of iterations is above 30. In our learning algorithm, the learning rate, $\alpha$, used was 1.0. As pointed out by Jordan and Jacobs (1994), the learning rate is not required in the IRLS algorithm. The EM learning was terminated after 24 epochs. For evaluating the generalization capability, we used a data set of $50 \times 50 = 2500$ points uniformly distributed in the large rectangle for test. As a result, Table 1 shows classification results on both the training and the testing sets as well as the relative learning time. In addition, we also show the evolution of log-likelihood vs. epochs in Fig. 2(b). It is evident from simulation results that the use of our learning algorithm in the inner loop of EM algorithm makes the ME classifier produce the significantly better performance. From Fig. 2(b), we observe that the log-likelihood monotonically increases and reaches a relatively steady state after six epochs when our learning algorithm is used in the inner loop of the EM algorithm, while the log-likelihood is of an oscillatory or instable behavior when the IRLS algorithm is used. In terms of simulation results, it is justified from an empirical perspective that the use of the incomplete Hessian matrix causes the ME architecture to produce the poor performance. From Table 1, however, we also observe that the use of the exact Hessian matrix results in expensive computation during learning. Basically, the learning time taken by the Newton–Raphson algorithm is about three times more than that of the IRLS algorithm for the synthetic data task.

## 5. Fast learning algorithm

It is well known that the Newton–Raphson method suffers from a high computational burden in general since the second derivatives of objective function and inverse of the Hessian matrix are necessary for updating parameters. Our simulation results presented in Section 4 also indicates that the expensive computation could be involved in our learning algorithm. To tackle the expensive computation problem in the Newton–Raphson algorithm, we propose an approximation algorithm for the ME architecture in this section. The basic idea underlying the approximation algorithm is to introduce an approximate statistical model called generalized Bernoulli density for expert networks in multiclass classification, and then the approximate statistical model makes the Newton–Raphson algorithm simplified in multiclass classification so that all off-diagonal block matrices in the Hessian matrix are naturally zero matrices without the independence assumption of parameter vectors. In the sequel, we first present the generalized Bernoulli density and then derive the approximation algorithm from the Newton–Raphson algorithm on the basis of the generalized Bernoulli density.

### *5.1. Generalized Bernoulli density*

We define the generalized Bernoulli density as

$$P(y_1, y_2, \ldots, y_K) = \prod_{k=1}^{K} p_k^{y_k}(1-p_k)^{1-y_k}, \tag{37}$$

where $y_k \in \{0,1\}$ and $p_k$ $(k = 1, \ldots, K)$ are the Bernoulli probabilities associated with the different classes. The proposed density can be merely used for modeling the probabilistic model of an expert network rather than the gating network in the ME architecture since the problem of multiclass classification in the gating network is a 'soft' classification problem (Jacobs et al., 1991; Jordan & Jacobs, 1994).

The proposed density is a member of the exponential family. The fact is shown as follows:

$$P(y_1, y_2, \ldots, y_K) = \exp\left\{\log \prod_{k=1}^{K} p_k^{y_k}(1-p_k)^{1-y_k}\right\}$$

$$= \exp\left\{\sum_{k=1}^{K} y_k \log\left[\frac{p_k}{1-p_k}\right] + \sum_{k=1}^{K} \log(1-p_k)\right\}. \tag{38}$$

As pointed out by McCullagh and Nelder (1983), the natural parameters in an exponential family density are those quantities that appear linearly in the $y_k$. We define

$$\eta_k = \log\left[\frac{p_k}{1-p_k}\right]. \tag{39}$$

Therefore, Eq. (39) can be inverted to yield

$$p_k = \frac{e^{\eta_k}}{1+e^{\eta_k}}. \tag{40}$$

Inserting Eqs. (39) and (40) to Eq. (38), we achieve

$$P(y_1, y_2, \ldots, y_K) = \exp\left\{\sum_{k=1}^{K} y_k \eta_k - \sum_{k=1}^{K} \log(1+e^{\eta_k})\right\}. \tag{41}$$

Let $\eta = [\eta_1, \eta_2, \ldots, \eta_K]^T$. Thus Eq. (41) implies that $b(\eta)$ should be defined as follows (cf. Eq. (8)):

$$b(\eta) = \sum_{k=1}^{K} \log(1+e^{\eta_k}). \tag{42}$$

Eq. (42) can be used to achieve the link function in the following way:

$$f(\eta_k) = \frac{\partial b(\eta)}{\partial \eta_k} = \frac{e^{\eta_k}}{1+e^{\eta_k}} = p_k. \tag{43}$$

We denote all the parameters in the generalized Bernoulli model as $\mathbf{\Theta}$, i.e. $\mathbf{\Theta} = [\beta_1, \beta_1, \ldots, \beta_K]^T$. Note that unlike the multinomial density there are $K$ independent parameter vectors in the generalized Bernoulli density. For an input sample $\mathbf{x}_t$ in $\chi$, we rewrite $p_k$ in Eq. (43) as $f(\boldsymbol{\beta}_k, \mathbf{x}_t)$ with the explicit parameter form:

$$f(\boldsymbol{\beta}_k, \mathbf{x}_t) = \frac{e^{\boldsymbol{\beta}_k^T \mathbf{x}_t}}{1+e^{\boldsymbol{\beta}_k^T \mathbf{x}_t}}. \tag{44}$$

According to Eq. (44), the link function of generalized Bernoulli density is a sigmoid function. Based on the generalized Bernoulli density, thus, the probabilistic model of an expert network for multiclass classification is

$$P(y_t|\mathbf{x}_t, \mathbf{\Theta}) = \prod_{k=1}^{K} [f(\boldsymbol{\beta}_k, \mathbf{x}_t)]^{y_{tk}}[1 - f(\boldsymbol{\beta}_k, \mathbf{x}_t)]^{1-y_{tk}}, \tag{45}$$

where $y_t = \{y_{t1}, y_{t2}, \ldots, y_{tK}\}$ and $y_{tk} \in \{0,1\}$.

Here we emphasize the generalize Bernoulli density is only an approximation of the multinomial density even for multiclass classification in expert networks. The generalized Bernoulli density implies that all the samples in a given multiclass classification task should be separable. It is not always guaranteed that an expert network modeled by the generalized Bernoulli density can produce Bayesian posterior probabilities, since the summation of all the $p_k$ may not be one when the inseparability of samples reaches some extent. As a result, the output of expert networks in this circumstance can be merely viewed as an approximation to Bayesian posterior probabilities.

### 5.2. Approximation to the Newton–Raphson algorithm

Using the generalized Bernoulli density, we can derive an approximation algorithm from the Newton–Raphson algorithm for expert networks in multiclass classification. For the $i$th expert network, based on the generalized Bernoulli model, its log-likelihood for the data set $\chi$ can be written as

$$l(\mathbf{\Theta}_i, \chi) = \sum_{t=1}^{T}\sum_{k=1}^{K} h_i^{(t)}\{y_{tk}\log[f(\beta_k, \mathbf{x}_t)] + (1 - y_{tk})\log[1 - f(\beta_k, \mathbf{x}_t)]\}, \quad (46)$$

where $f(\beta_k, \mathbf{x}_t)$ is the same as described in Eq. (44) and $\mathbf{\Theta}_i = [\beta_1, \beta_2, \ldots, \beta_K]^{\mathrm{T}}$. For $q = 1, 2, \ldots, K$, the derivative of $l(\mathbf{\Theta}_i, \chi)$ with respect to $\beta_q$ is

$$\mathbf{J}_q(\mathbf{\Theta}_i, \chi) = \frac{\partial l(\mathbf{\Theta}_i, \chi)}{\partial \beta_q} = \sum_{t=1}^{T} h_i^{(t)}[y_{tq} - f(\beta_q, \mathbf{x}_t)]\mathbf{x}_t. \quad (47)$$

Accordingly, the Hessian matrix $\mathbf{H}(\mathbf{\Theta}_i, \mathbf{X})$ consists of $K \times K$ block matrices $H_{qr}$ $(q, r = 1, 2, \ldots, K)$. Based on Eq. (47), each $\mathbf{H}_{qr}$ is

$$\mathbf{H}_{qr} = \frac{\partial l(\mathbf{\Theta}_i, \chi)}{\partial \boldsymbol{\beta}_q \partial \boldsymbol{\beta}_r^{\mathrm{T}}} = -\sum_{t=1}^{T} \delta_{qr} h_i^{(t)} f'(\boldsymbol{\beta}_q, \mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}, \quad (48)$$

where $\delta_{\mathrm{qr}}$ is the Kronecker delta and $f'(\boldsymbol{\beta}_q, \mathbf{x}_t) = f(\boldsymbol{\beta}_q, \mathbf{x}_t) \times [1 - f(\boldsymbol{\beta}_q, \mathbf{x}_t)]$.

We now assemble the various pieces. Using Eq. (47), the gradient vector of $l(\mathbf{\Theta}, \chi)$ on all the parameters in $\mathbf{\Theta}_i$ is $\mathbf{J}(\mathbf{\Theta}_i, \chi)$ as follows,

$$\mathbf{J}(\mathbf{\Theta}_i, \chi) = \frac{\partial l(\mathbf{\Theta}_i, \chi)}{\partial \mathbf{\Theta}_i} = [\mathbf{J}_1(\mathbf{\Theta}_i, \chi), \mathbf{J}_2(\mathbf{\Theta}_i, \chi), \ldots, \mathbf{J}_K(\mathbf{\Theta}_i, \chi)]^{\mathrm{T}}. \quad (49)$$

According to Eq. (48), all the off-diagonal block matrices in the Hessian matrix are zero matrices. Therefore, we have

$$\mathbf{H}(\mathbf{\Theta}_i, \chi) = \frac{\partial^2 l(\mathbf{\Theta}_i, \chi)}{\partial \mathbf{\Theta}_i \partial \mathbf{\Theta}_i^{\mathrm{T}}} = \mathrm{diag}[\mathbf{H}_{11}, \mathbf{H}_{22}, \ldots, \mathbf{H}_{KK}]. \quad (50)$$

In this case, the update formula in the Newton–Raphson method (Fletcher, 1987) is simplified as

$$\boldsymbol{\beta}_q^{(s+1)} = \boldsymbol{\beta}_q^{(s)} - \alpha[\mathbf{H}_{qq}(\mathbf{\Theta}_i, \chi)]^{-1}\mathbf{J}_q(\mathbf{\Theta}_i, \chi) = \boldsymbol{\beta}_q^{(s)} - \alpha\left[\frac{\partial^2 l(\mathbf{\Theta}_i, \chi)}{\partial \boldsymbol{\beta}_q \boldsymbol{\beta}_q^{\mathrm{T}}}\right]^{-1} \frac{\partial l(\mathbf{\Theta}_i, \chi)}{\partial \boldsymbol{\beta}_q} \quad (51)$$

where $q = 1, \ldots, K$. It indicates that each row of the weight matrix $\mathbf{W}_i$ in the $i$th expert network is a separate parameter vector corresponding to a vector $\beta_q$ and these row vectors will be updated independently and in parallel if the statistical model of the $i$th expert network is the generalized Bernoulli density.

## 6. Simulations

In this section, we report simulation results on synthetic data, benchmark, and real-world multiclass classification problems using the ME architecture along with the EM algorithm where the Newton–Raphson algorithm and its approximation were used in the inner loop, respectively. For comparison, the IRLS method (Jordan & Jacobs, 1994) was also used in the inner loop of the EM algorithm to train the ME architecture for the same problems. On the other hand, a class of methods called quasi-Newton methods have been proposed in optimization theory to reduce computational burden of the Newton–Raphson method from a general perspective. The BFGS algorithm is a typical quasi-Newton method and turns out to be one of the most efficient quasi-Newton methods (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970; Fletcher, 1987). Therefore, we also adopted the BFGS algorithm as another approximation to the Newton–Raphson algorithm. As a result, the ME architecture trained by the EM algorithm where the BFGS algorithm was used in the inner loop was also applied to the same problems for comparison, in particular, in computational efficiency. The purpose of simulations is three-fold, i.e. evaluating the performance of our learning algorithms, exploring the limitation of our approximation algorithm, and, from an empirical perspective, demonstrating the instability problem caused by the IRLS algorithm as pointed out in Section 4. All simulations were conducted in a Sun SPARC-20 workstation. Note that the weight matrices of gating and expert networks in the ME architecture were randomly initialized in all the simulations.

Before presenting simulation results, we first define some measure to evaluate the performance of learning algorithms in the ME architecture. For a data set $\chi = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$, the log-likelihood of an ME architecture consisting of $N$ expert networks, $L$, is defined as

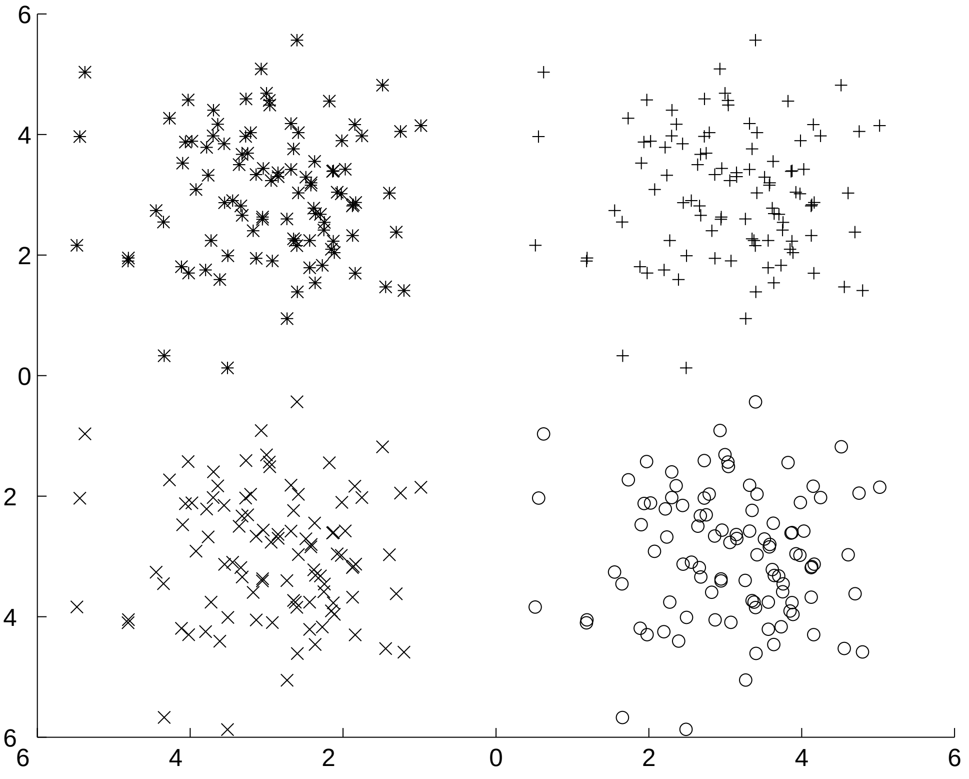
$$L = \frac{1}{T}\sum_{t=1}^{T}\log\left(\sum_{i=1}^{N} g(\mathbf{v}_i, \mathbf{x}_t)P(\mathbf{y}_t|\mathbf{x}_t, \mathbf{W}_i)\right), \quad (52)$$

where all the parameters in Eq. (52) are the same as defined before. In the EM algorithm, we define an epoch as a complete $E$-step and $M$-step though there are several iterations in an $M$-step on the data set $\chi$. Finally, we define the decision rule for classification as

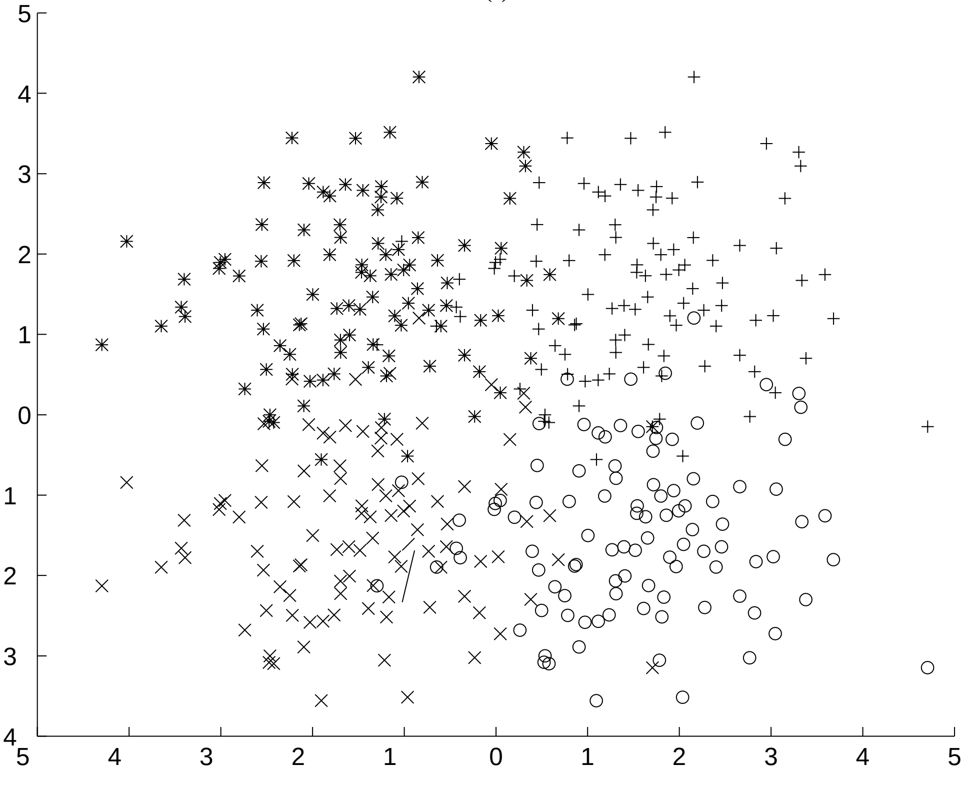
$$k^* = \arg\max_{1 \le k \le K} \mathrm{O}_k(x), \quad (53)$$

where $\mathrm{O}(x) = [\mathrm{O}_1(x), \mathrm{O}_2(x), \ldots, \mathrm{O}_K(x)]^{\mathrm{T}}$ is the output vector of an ME classifier on the input vector $\mathbf{x}$ and $k^*$ is the class label assigned to $\mathbf{x}$.

In simulations, an EM learning was terminated once the change of log-likelihood between two consecutive epochs was below a specified threshold, $L_{\mathrm{T}}$. In other words, the termination condition is $|L^{(s+1)} - L^{(s)}| \le L_{\mathrm{T}}$, where $L^{(s)}$ is the log-likelihood defined in Eq. (52) at the $s$th epoch. However, the EM algorithm was also terminated once the

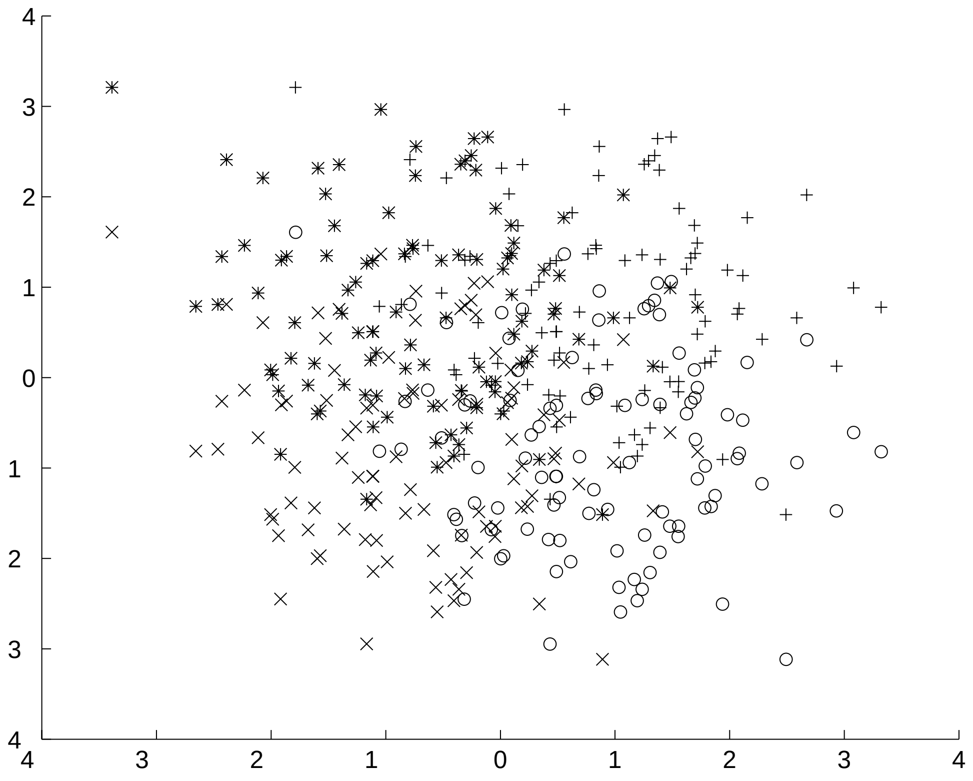


(a)

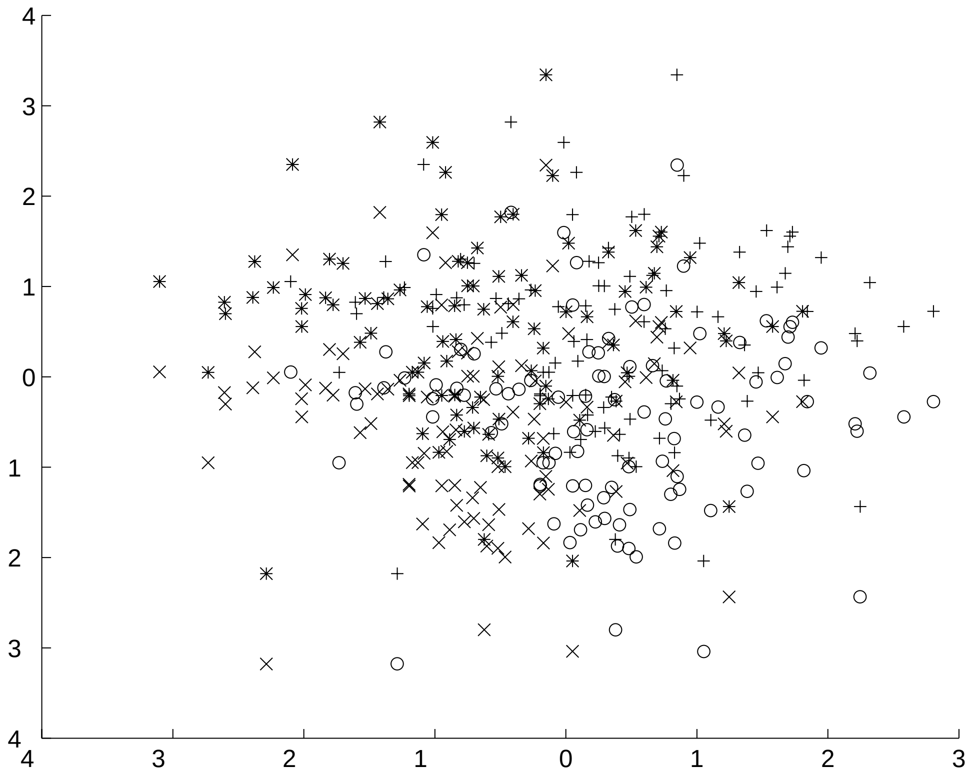


(b)

Fig. 3(a) and (b). *Caption overleaf.*

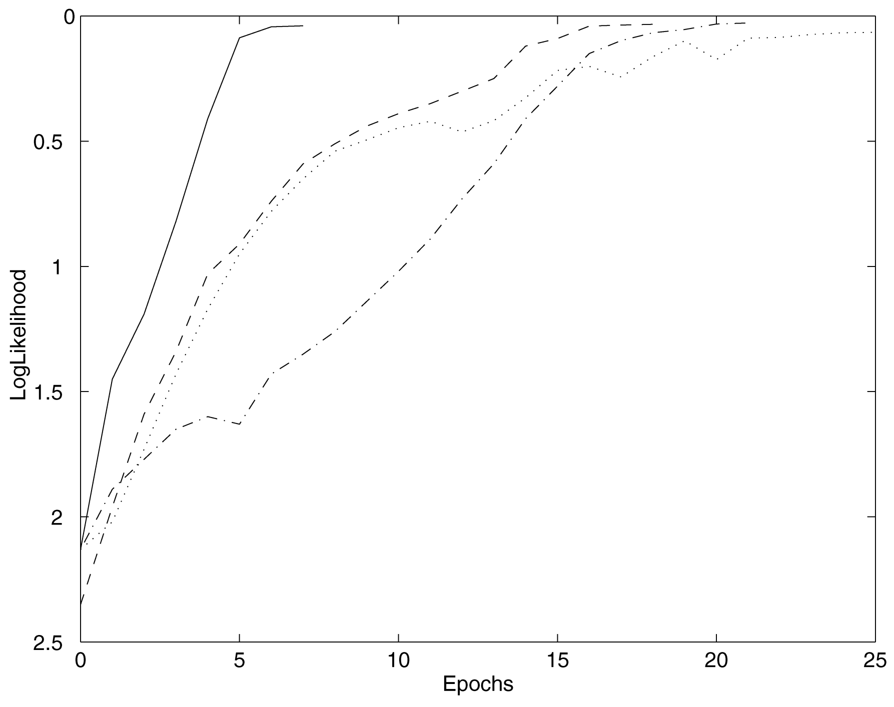


(c)

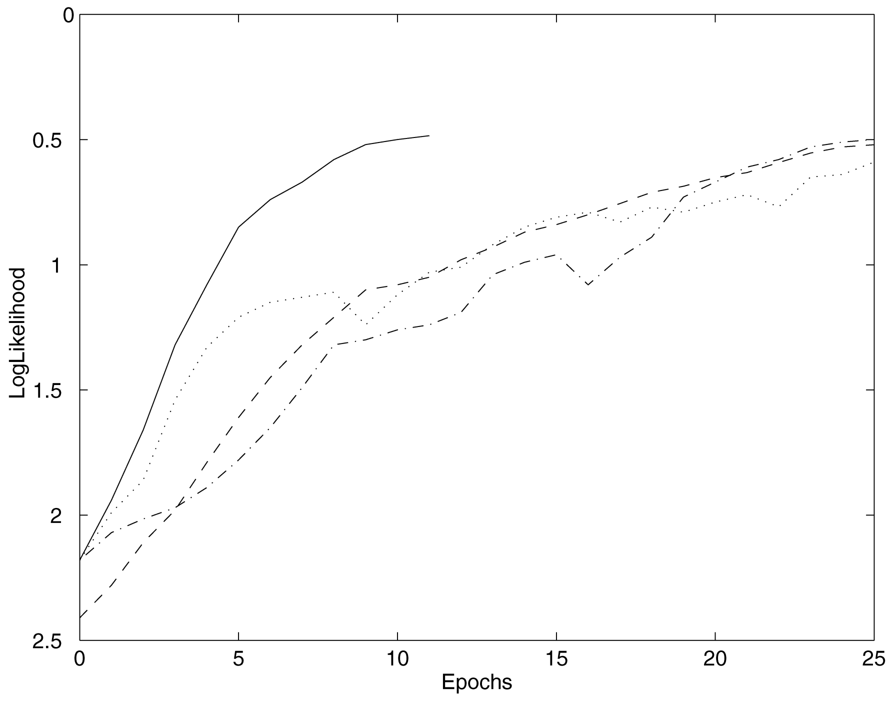


(d)

Fig. 3. The synthetic four-category problem: train sets corresponding to different $\gamma$, where $*$, $+$, $\times$, and $\bigcirc$ stand for samples belonging to different categories. (a) $\gamma = 3.0$.; (b) $\gamma = 1.5$.; (c) $\gamma = 0.8$.; (d) $\gamma = 0.5$.



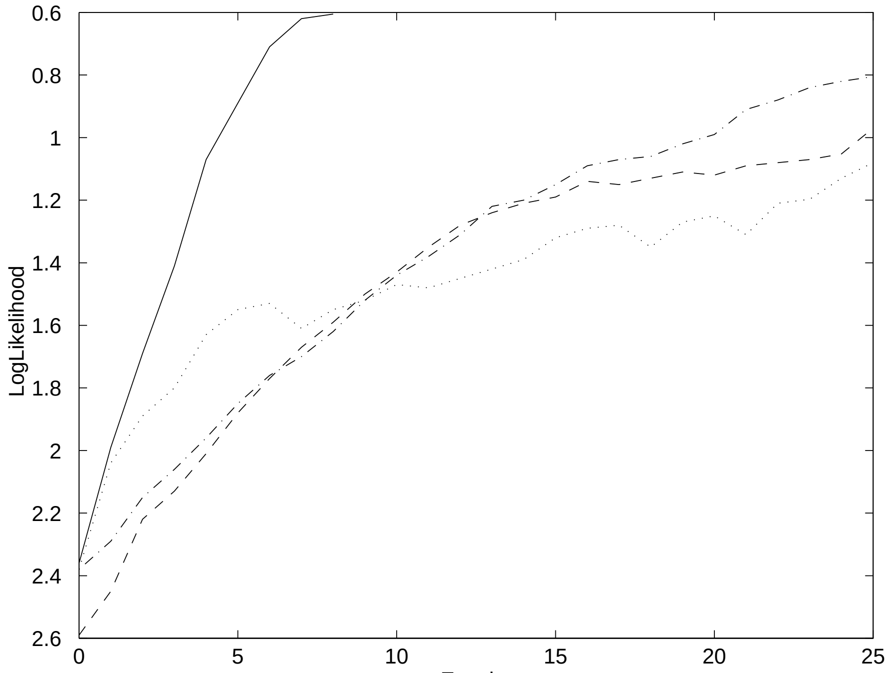
(a)



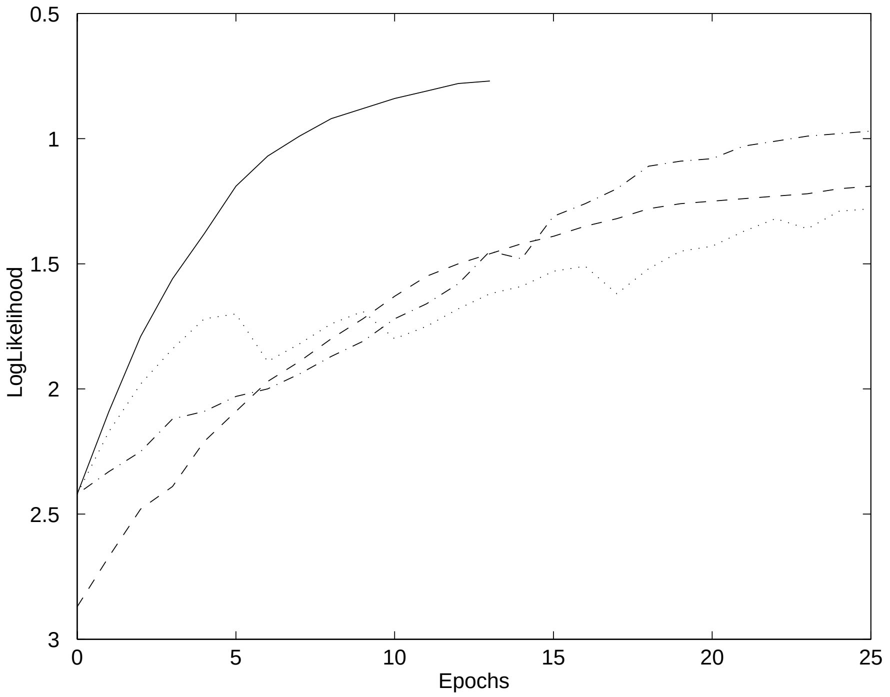
(b)

Fig. 4(a) and (b). *Caption overleaf.*

number of epochs reached a specified threshold, $E_T$, even though the change of log-likelihood was still above the value of $L_T$ in this circumstance. That is, the EM algorithm was also terminated if $s = E_T$, where $s$ is the number of epochs. As mentioned in the introduction, the IRLS algorithm often causes the log-likelihood not to reach the steady state in multiclass classification. For comparison, we also terminated the current *M*-step in simulations if an algorithm was still not convergent after the number of iterations in the inner loop was above a pre-specified threshold, $I_T$. All the results were achieved based on these thresholds. Note that our approximation algorithm is only used for training expert networks in simulations, while the gating network is always trained by the



(c)



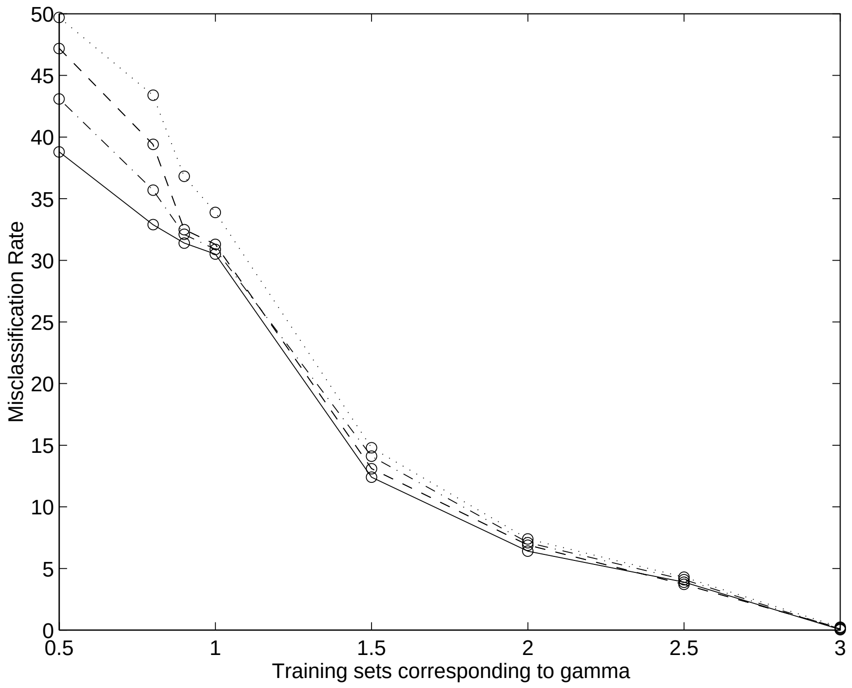
(d)

Fig. 4. The synthetic four-category problem: evolution of log-likelihood vs. epochs on training sets corresponding to different $\gamma$. Solid, dashed, dash-dot, and dotted curves denote results produced by the Newton–Raphson algorithm, our approximation algorithm, the BFGS algorithm, and the IRLS algorithm used in the inner loop of the EM algorithm, respectively. (a) $\gamma = 3.0$.; (b) $\gamma = 1.5$; (c) $\gamma = 0.8$; (d) $\gamma = 0.5$.

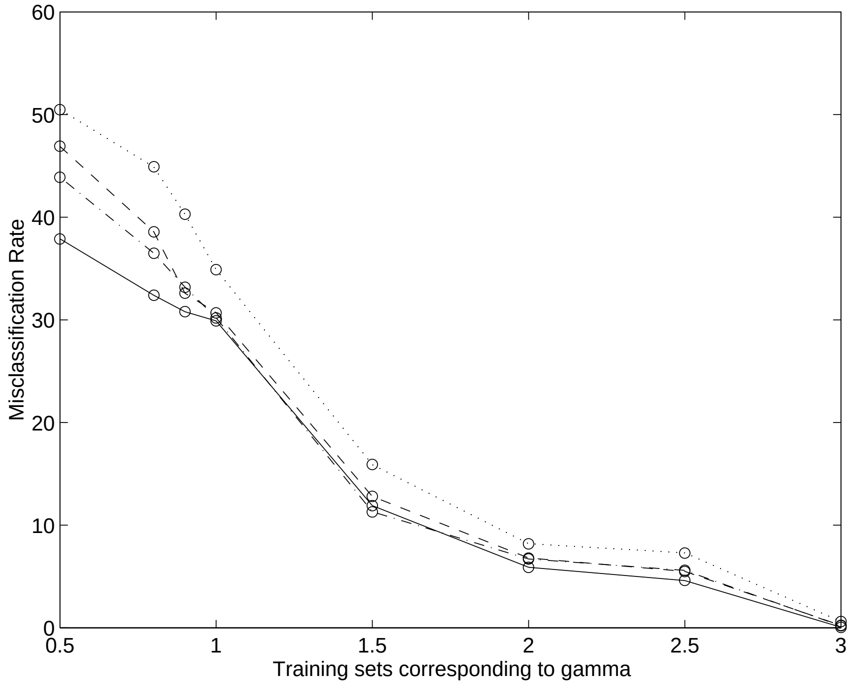
proposed Newton–Raphson algorithm when our learning algorithms are tested.

### 6.1. A four-category problem

The first problem used to test our learning algorithms is a synthetic four-category problem. It is designed not only to evaluate the performance of our learning algorithms in generalization but also to investigate the limitation of our approximation algorithm. For comparison, the IRLS algorithm and the BFGS algorithm were also applied in the inner loop of the EM algorithm, respectively, for the same problem. The



(a)



(b)

Fig. 5. The synthetic four-category problem: mean classification results produced by the ME architecture consisting of different expert networks on testing sets corresponding to different $\gamma$. The notation is the same as stated in the caption of Fig. 4. (a) Results produced by the ME classifier consisting of three expert networks. (b) Results produced by the ME classifier consisting of four expert networks.

Table 2
Synthetic four-category problem: classification results on the training set corresponding to different $\gamma$ when different learning algorithms are used in the inner loop of the EM algorithm, respectively

| | Newton–Raphson | | Approximation | | BFGS | | IRLS | |
|---|---|---|---|---|---|---|---|---|
| Data set | Correct no. | Error no. | Correct no. | Error no. | Correct no. | Error no. | Correct no. | Error no. |
| $\gamma = 3.0$ | 400 | 0 | 400 | 0 | 400 | 0 | 397 | 3 |
| $\gamma = 1.5$ | 352 | 48 | 351 | 49 | 349 | 51 | 301 | 99 |
| $\gamma = 0.8$ | 289 | 111 | 239 | 161 | 255 | 145 | 212 | 188 |
| $\gamma = 0.5$ | 269 | 131 | 219 | 181 | 228 | 172 | 194 | 206 |

Table 3
Synthetic four-category problem: mean classification results on 10 testing sets corresponding to different $\gamma$ when different algorithms are used in the inner loop of the EM algorithm, respectively

| | Newton–Raphson | | Approximation | | BFGS | | IRLS | |
|---|---|---|---|---|---|---|---|---|
| Data set | Correct no. | Error no. | Correct no. | Error no. | Correct no. | Error no. | Correct no. | Error no. |
| $\gamma = 3.0$ | 3988.4 | 11.6 | 3987.7 | 12.3 | 3987.2 | 12.8 | 3976.3 | 23.7 |
| $\gamma = 1.5$ | 3471.5 | 528.5 | 3425.3 | 574.7 | 3431.6 | 568.4 | 3201.5 | 798.5 |
| $\gamma = 0.8$ | 2887.0 | 1113.0 | 2732.1 | 1267.9 | 2756.3 | 1243.7 | 2251.8 | 1748.2 |
| $\gamma = 0.5$ | 2628.0 | 1372.0 | 2167.7 | 1832.3 | 2231.4 | 1768.6 | 1967.2 | 2032.8 |

programs were written in MATLAB. In simulations, therefore, learning time used in all the algorithms was measured by *floating-point operations* (FPO).

The synthetic four-category data are produced by four Gaussian distributed random number producers. The four bivariate Gaussian distributions are defined as

$$P(\mathbf{x};\mu_i,\Sigma) = \frac{1}{2\pi|\Sigma|^{1/2}}\exp\{-\frac{1}{2}(\mathbf{x}-\mu_i)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu_i)\};\quad i=1,2,3,4 \tag{54}$$

where

$$\mathbf{x} = [x_1, x_2]^{\mathrm{T}} \text{ and } \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

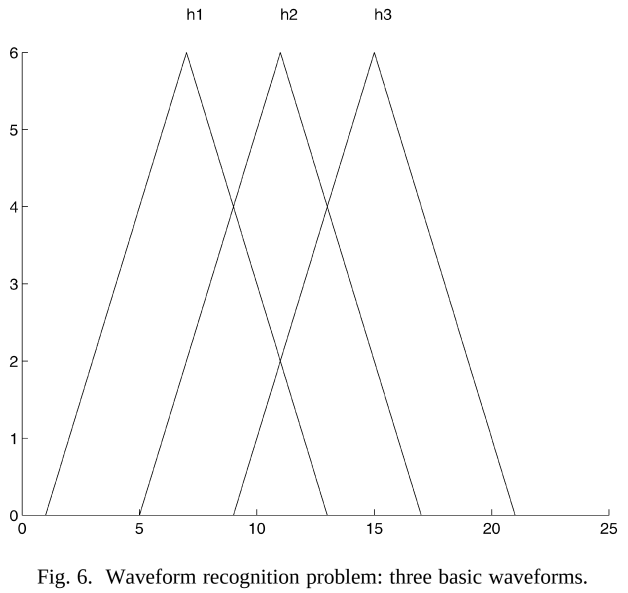


Fig. 6. Waveform recognition problem: three basic waveforms.

To produce the four-category data, we adopt different mean vectors in four Gaussian distributions, i.e. $\mu_1 = [\gamma,\gamma]^{\mathrm{T}}, \mu_2 = [-\gamma,\gamma]^{\mathrm{T}}, \mu_3 = [\gamma,-\gamma]^{\mathrm{T}}$, and $\mu_4 = [-\gamma,-\gamma]^{\mathrm{T}}$. The smaller the value of $\gamma$ is, the higher the overlaps among the data belonging to different categories are. So we can control $\gamma$ to produce several data sets for the aforementioned requirement. As a result, we have produced eight data sets with eight different $\gamma$ ranging from 3.0 to 0.5 as training sets. There are 400 samples (100 samples/category) in each training set. Similarly, we randomly produced 10 data sets (4000 samples/set and 1000 samples/category in each set) corresponding to a specific $\gamma$ for test. Due to limited space, we merely report four typical cases, i.e. $\gamma = 3.0$, 1.5, 0.8 and 0.5, in detail. Fig. 3 illustrates the two-dimensional points in the training sets corresponding to four typical cases, respectively.

We have chosen ME architecture consisting of two, three, and four expert networks, respectively, for learning the classification task. Here we merely report results produced by the ME classifier consisting of two expert networks in detail. In simulations, the learning rate, $\alpha$, was set as 0.2 in all the algorithms except for the IRLS algorithm where no learning rate is required. The thresholds used for terminating the EM learning were chosen as $L_{\mathrm{T}} = 10^{-3}, E_{\mathrm{T}} = 25$, and $I_{\mathrm{T}} = 20$, respectively. For four different training sets, evolution of the log-likelihood vs. epochs during learning is illustrated in Fig. 4, and Table 2 shows classification results on training sets corresponding to four different $\gamma$, respectively. To evaluate the generalization capability, we test the ME classifier using 10 testing sets corresponding to each training set for reliability. Table 3 lists the mean classification results on the 10 testing sets corresponding to different $\gamma$, respectively. In addition, Fig. 5 also illustrates classification results of ME classifiers consisting of three and four expert networks on the testing sets corresponding to different $\gamma$, respectively.

Table 4
Synthetic four-category problem: mean learning times (unit: FPO) on eight different training sets and the mean relative learning time on the different ME structures. The ME-N structure stands for an ME consisting of *N* expert networks

| Algorithms | ME-2 | ME-3 | ME-4 | Relative time |
|---|---|---|---|---|
| Newton–Raphson | $2.78 \times 10^{07}$ | $3.92 \times 10^{07}$ | $6.36 \times 10^{07}$ | 1.0000 |
| Approximation | $1.22 \times 10^{07}$ | $1.83 \times 10^{07}$ | $2.44 \times 10^{07}$ | 0.3983 |
| BFGS | $2.58 \times 10^{07}$ | $3.99 \times 10^{07}$ | $5.37 \times 10^{07}$ | 0.8975 |
| IRLS | $1.13 \times 10^{07}$ | $1.92 \times 10^{07}$ | $2.39 \times 10^{07}$ | 0.4065 |

For comparison, we summarize the mean learning time taken by the ME classifiers consisting of different numbers of expert networks in Table 4. We observe from simulation results that the use of the Newton–Raphson algorithm makes the ME architecture produce the best performance through it involves slow learning. In particular, this proposed learning algorithm considerably outperforms its approximation, the BFGS algorithm, and the IRLS algorithm on those data sets corresponding to $\gamma \le 0.8$, which is equivalent to highly inseparable problems, though all the algorithms except for the IRLS algorithm yield similar classification rates on the data sets corresponding to $\gamma > 0.8$. In contrast to our approximation algorithm, the BFGS algorithm yields the slightly better performance in general and, however, our approximation algorithm leads to significantly faster learning. On the other hand, the use of the IRLS algorithm results in the poor performance in comparison with other three algorithms for the problem. Furthermore, we find that the performance of our approximation algorithm is highly degraded as there are high overlaps among samples belonging to different categories, while the performance of the proposed learning algorithm is stable to a great extent. In terms of simulation results, our approximation algorithm is subject to limitation as it is applied to a highly inseparable problem and, therefore, we suggest that the Newton–Raphson algorithm should be used in this situation.

## 6.2. Benchmark problems

In order to extensively investigate the performance, we have applied four different learning algorithms in the inner loop of the EM algorithm, respectively, to train the ME architecture for two benchmark multiclass classification problems, i.e. classification of irises and waveform recognition.

### 6.2.1. Classification of irises

The classification of irises is a famous benchmark problem in pattern recognition. Since Fisher (1936) used the data set in his classic paper on discriminant analysis, the data set has become a favorite example in pattern recognition (Duda & Hart, 1973). Irises are classified into three categories: setosa, versicolor, and virginica. Each category is composed of 50 samples. In the iris data set, four attributes are associated with each sample: sepal length, sepal width, petal length, and petal width.

In simulations, we adopted two ways to produce training and testing sets from the iris data. One was to use all the data as both training and testing sets to evaluate the classification performance of a classifier, and the other was to divide the iris data set into two disjoint subsets to evaluate the generalization capability. In doing so, a subset of data were randomly chosen for learning and the remaining data were used for test. Here the generalization capability of an ME architecture was assessed by mean prediction error. For reliability, we randomly selected five subsets of data as training sets corresponding to a specific number of samples. We chose the ME architecture consisting of three expert networks as the classifier. The thresholds used for terminating the EM learning were $L_T = 10^{-3}, E_T = 25$, and $I_T = 10$, respectively. Tables 5–7 show the classification results, the log-likelihood after the learning is terminated, the learning time, and the relative learning time through the use of different algorithms in the inner loop of the EM algorithm, respectively. With respect to the problem, the performance of *multi-layered perceptron* (MLP) has been reported in the literature (Ishikawa, 1996), and the numbers of misclassified samples using the MLP are 1.2, 4.4, and 5.2, respectively, corresponding to training and testing sets as stated in the caption of Tables 5–7. It is evident from simulations that the use of either the Newton–Raphson algorithm or its approximation yields satisfactory results for classification of irises

Table 5
Classification of irises: performance of ME architecture when different learning algorithms are used in the inner loop of the EM algorithm, respectively. In this simulation, all the samples in the data set are used in training and test

| Algorithm | Error no. | Likelihood | Epochs | FPO | Relative time |
|---|---|---|---|---|---|
| Newton–Raphson | 1 | − 0.001 | 10 | $8.43 \times 10^{06}$ | 1.0000 |
| Approximation | 2 | − 0.003 | 25 | $3.56 \times 10^{06}$ | 0.4223 |
| BFGS | 2 | − 0.003 | 25 | $6.92 \times 10^{06}$ | 0.8205 |
| IRLS | 4 | − 0.022 | 25 | $3.45 \times 10^{06}$ | 0.4093 |

Table 6
Classification of irises: performance of the ME architecture when different learning algorithms are used in the inner loop of the EM algorithm, respectively. In this simulations, the number of samples in training and testing sets are 90 and 60, respectively. The averaging results on five testing sets are only shown here

| Algorithm | Error no. | Likelihood | Epochs | FPO | Relative time |
|---|---|---|---|---|---|
| Newton–Raphson | 4.0 | − 0.044 | 8.0 | $5.21 \times 10^{06}$ | 1.0000 |
| Approximation | 4.2 | − 0.049 | 19.2 | $1.61 \times 10^{06}$ | 0.3098 |
| BFGS | 4.2 | − 0.048 | 23.4 | $4.38 \times 10^{06}$ | 0.8407 |
| IRLS | 6.8 | − 0.121 | 25.0 | $1.84 \times 10^{06}$ | 0.3532 |

and, in particular, our approximation algorithm results in faster learning. In contrast, the performance of the BFGS algorithm is satisfactory, but it still does not lead to significantly faster learning. As for the IRLS algorithm, its performance is quite poor for the benchmark problem in contrast to other three algorithms.

#### 6.2.2. Waveform recognition problem

The synthetic waveform recognition problem was first introduced in Breiman, Friedman, Olshen and Stone (1984) to study the behavior of decision trees. It is a three-category problem based on the waveforms $h_1(i), h_2(i)$, and $h_3(i)$ depicted in Fig. 6. The $h_k(k = 1, 2, 3)$ are the shifted triangular waveforms: $h_1(i) = \max\{6 - |i - 11|, 0\}, h_2(i) = h_1(i - 4)$, and $h_3(i) = h_1(i + 4)$. Each class is a random convex combination of two of these waveforms. The pattern vector is obtained by sampling 21 points and adding noises. Hence, the components of the pattern vector are given as follows:

For class 1,

$$x_i = uh_1(i) + (1 - u)h_2(i) + \varepsilon_i, \quad i = 1, \ldots, 21.$$

For class 2,

$$x_i = uh_1(i) + (1 - u)h_3(i) + \varepsilon_i, \quad i = 1, \ldots, 21.$$

For class 3,

$$x_i = uh_2(i) + (1 - u)h_3(i) + \varepsilon_i, \quad i = 1, \ldots, 21.$$

Here $u$ is a uniform random variable on the interval [0,1], and $\varepsilon_1, \ldots, \varepsilon_{21}$ are independent Gaussian random variables with zero mean and unit variance. The three classes have equal prior probabilities. Breiman et al. (1984) reported that the Bayesian misclassification rate for this problem is about 14.0%, and the best performance of the MLP reported in the literature (Guo & Gelfand, 1992) is about 14.9% misclassification rate.

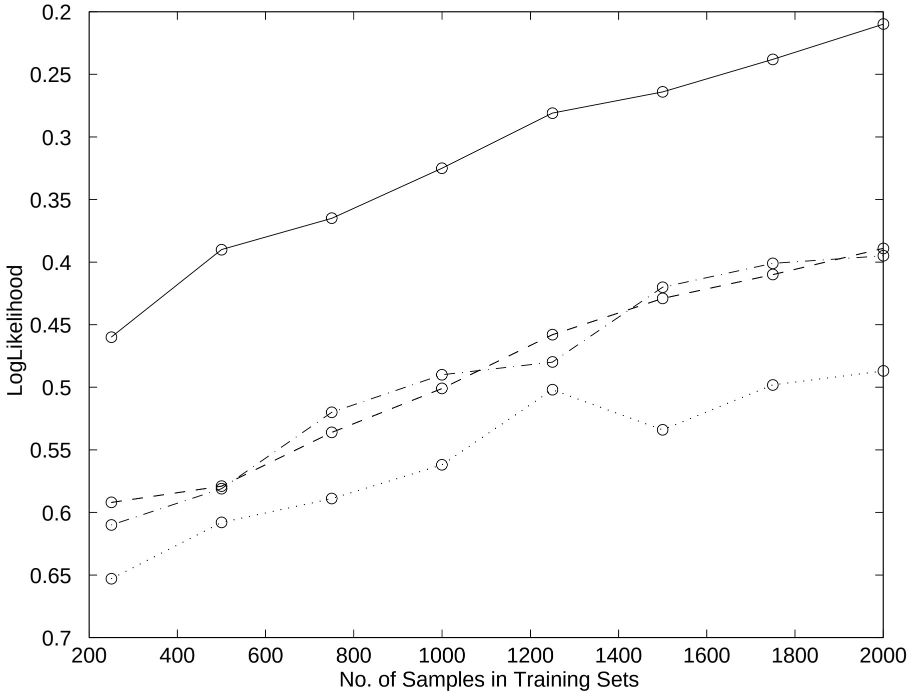
For this benchmark problem, we adopted the ME architecture consisting of 12 expert networks. The ME classifier was trained using four different learning algorithms in the inner loop of the EM algorithm on training sets with equal numbers of independent samples, ranging in size from 250 to 2000 samples, respectively. An additional set of 5000 independent samples was employed to evaluate the generalization capability. The experimental method used was the same as suggested by Guo and Gelfand (1992). In simulations, the thresholds used for terminating the EM learning were $L_T = 10^{-3}, E_T = 80$, and $I_T = 20$, respectively. Table 8 shows the mean learning time of the ME classifier trained on different training sets for different learning algorithms used in the inner loop of the EM algorithm, respectively. Fig. 7(a) depicts the log-likelihood after the ME architecture reached the steady state or satisfied the termination condition of the EM learning on different training sets and Fig. 7(b) illustrates classification results on the testing set. It is evident from the simulation results that the use of the Newton–Raphson algorithm in the EM algorithm makes the ME architecture produce the best performance, and the use of our approximation algorithm readily leads to significantly faster learning. Moreover, the ME architecture trained on the set of 2000 samples using the proposed learning algorithm achieves a classification rate close to the best one produced by the MLP. In contrast to our approximation algorithm, the BFGS method yields slightly better performance in general, but it does not lead to considerably faster learning. Our simulation results show that it needs longer time or more epochs to make the ME architecture reach the steady state. In comparison with other learning algorithms, the use of the IRLS algorithm still leads to the worst performance for the benchmark problem.

### 6.3. Speaker identification

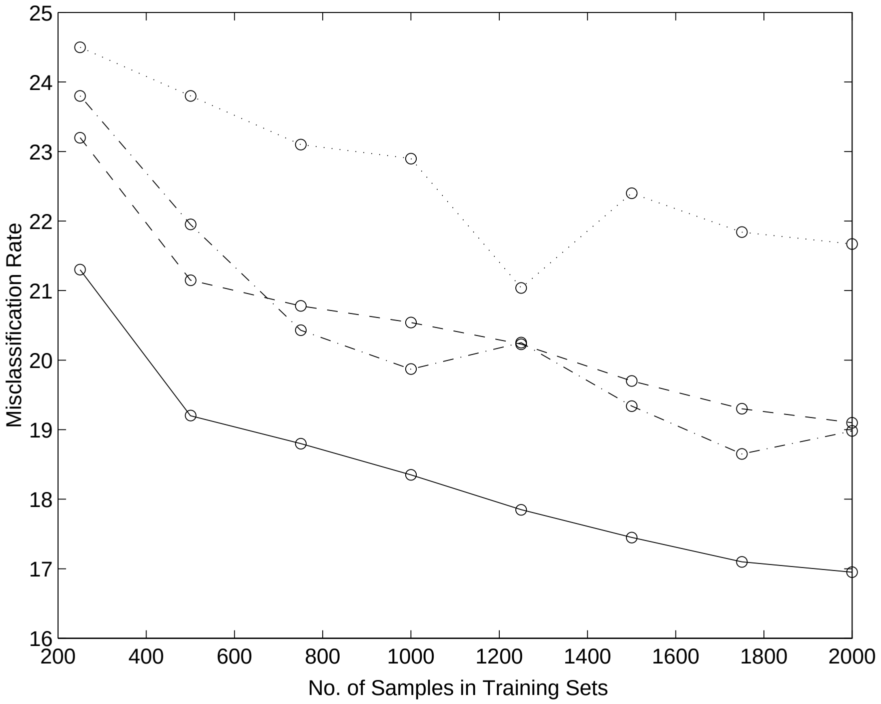
Speaker identification is a real-world problem to classify an unlabeled voice token as belonging to one of reference

Table 7
Classification of irises: performance of the ME architecture when different learning algorithms are used in the inner loop of the EM algorithm, respectively. In this simulation, the number of samples in training and testing sets are 60 and 90, respectively. The averaging results on five testing sets are only shown here

| Algorithm | Error no. | Likelihood | Epochs | FPO | Relative time |
|---|---|---|---|---|---|
| Newton–Raphson | 4.4 | − 0.090 | 9.0 | $2.95 \times 10^{06}$ | 1.0000 |
| Approximation | 5.0 | − 0.093 | 18.0 | $6.95 \times 10^{05}$ | 0.2356 |
| BFGS | 5.2 | − 0.102 | 23.2 | $2.47 \times 10^{06}$ | 0.8373 |
| IRLS | 7.4 | 0.213 | 25.0 | $5.88 \times 10^{05}$ | 0.1993 |



(a)



(b)

Fig. 7. Waveform recognition problem: performance of the ME architecture trained by different learning algorithms in the inner loop of the EM algorithm. The notation is the same as stated in the caption of Fig. 4. (a) The log-likelihood after learning is terminated vs. the number of samples in different training sets. (b) Results produced by the ME architecture consisting of 12 expert networks on the testing set.

speakers. It is a hard learning task since a person's voice often changes over time. There have been extensive studies on speaker identification (for reviews of the subject see Campbell, 1997; Furui, 1997). Recently, connectionist approaches have been introduced to speaker identification systems and lead to better performance (Bennani & Gallinari, 1994; Chen et al., 1996a; Chen, 1998). The classification in speaker identification is a typical multiclass

Table 8
Waveform recognition problem: mean learning time when different learning algorithms is used in the inner-loop of the EM algorithm, respectively

| Algorithm | Mean epochs | Mean FPO | Relative time |
|---|---|---|---|
| Newton–Raphson | 32.7 | $1.93 \times 10^{10}$ | 1.0000 |
| Approximation | 75.8 | $3.21 \times 10^{09}$ | 0.1663 |
| BFGS | 72.2 | $1.79 \times 10^{10}$ | 0.7275 |
| IRLS | 80.0 | $4.34 \times 10^{09}$ | 0.2249 |

classification task since the population of reference speakers is usually more than two. In the sequel, we utilize the real-world problem to test different learning algorithms used in the inner loop of the EM algorithm.

The database for simulations reported in this paper is a subset of a standard speech database in China. The set represents 10 male speakers of the same dialect (Mandarin). The utterances of 10 isolated digits from '0' to '9' were recorded in three different sessions. For each digit, 100 utterances (10 utterances/speaker) were recorded in each session. The technical details of preprocessing are briefly as follows, (1) 16-bit A/D-converter with 11.025 kHz sampling rate, (2) processing the data with a pre-emphasis filter $H(z) = 1 - 0.95z^{-1}$, and (3) 25.6 ms Hamming window with 12.8 ms overlapping for blocking an utterance into several feature frames for the short-time spectral analysis. In simulations, we adopted 19-order Mel-scaled cepstrum as the feature. After preprocessing, feature vectors were extracted from an utterance corresponding to one digit. In simulations, for one digit, five utterances of each speaker recorded in the first session were used as the training data, while the other five utterances of each speaker recorded in the same session were employed as the cross-validation data. All the utterances of the digit recorded in other two sessions were used for test. To facilitate the presentation, we refer as to TEST-1 and TEST-2, respectively, for testing results on utterances recorded in the second and the third sessions. In order to evaluate the performance, we apply different learning algorithms in the inner loop of the EM algorithm to train both the standard ME architecture (Jacobs et al., 1991) and a modified ME architecture proposed specially for speaker identification (Chen et al., 1996a). In simulations, the thresholds used for terminating the EM learning were chosen as $L_T = 10^{-4}, E_T = 30$, and $I_T = 20$, respectively. Due to the programs written C language, we would rather use the CPU time to measure the learning time.

### 6.3.1. Results of mixture of experts architecture

Depending upon the fixed text (10 isolated digits), 10 ME classifiers were used to handle 10 digits from '0' to '9' so that each ME classifier corresponds to one digit, respectively. We applied the two-fold cross-validation technique for model selection and investigated eight ME structures with different numbers of expert networks (Chen et al., 1995). Finally, we chose the ME architecture with six expert networks as the structure of ME classifiers. Table 9 summarizes results produced by those ME classifiers. Due to limited space, here we merely report the averaging results produced by 10 ME classifiers.

From simulation results, we observe that the use of the proposed learning algorithm yields the best identification rates in comparison with other three algorithms in general. Its approximation yields satisfactory classification results and fast learning. The ratio of mean relative training time between them is about 1:0.4425. In contrast to our approximation algorithm, the BFGS algorithm produces the similar performance, but does not lead to significantly faster learning as shown in Table 9. In addition, the proposed learning algorithm and its approximation outperform the IRLS algorithm. Although we do not show the detailed result here due to limited space, it should be mentioned that the ME classifier corresponding to the digit '2' cannot reach the steady state when the IRLS algorithm was used in the inner loop of the EM algorithm. Note that our further observation by prolonging the learning indicates that the ME architecture still cannot reach the steady state after 50 epochs.

### 6.3.2. Results of a modified mixture of experts architecture

For text-dependent speaker identification, the text in both training and testing is the same. Thus, the utterance of a fixed text naturally becomes a sequence consisting of successive feature frames after preprocessing and feature extraction. Accordingly, the problem of text-dependent speaker identification may be viewed as a specific problem of sequence recognition. For a single feature frame, it conveys the instantaneous spectral information which carries speaker related information correlated with talking behavior and physiological structure of the vocal tracts in additional to conveying phonetic information. On the other hand, successive feature frames convey the transitional information. Earlier studies have shown that both instantaneous and transitional spectral information is useful to speaker identification. Furthermore, the instantaneous and

Table 9
Speaker identification: performance of the mixture of experts trained by different learning algorithms in the inner-loop of the EM algorithm

| Algorithm | TEST-1 | TEST-2 | Log-likelihood | Epochs | CPU time (min) |
|---|---|---|---|---|---|
| Newton–Raphson | 93.2 | 91.8 | − 0.2090 | 5.8 | 10:58 |
| Approximation | 92.4 | 91.4 | − 0.2127 | 8.3 | 4:51 |
| BFGS | 92.6 | 90.4 | − 0.2172 | 12.0 | 9:39 |
| IRLS | 87.0 | 85.4 | − 0.3484 | 14.2 | 6:38 |

transitional spectral information is relatively uncorrelated, thus providing complementary information for speaker identification. To use both transitional and instantaneous spectral information, we previously proposed a modified ME architecture for text-dependent speaker identification (Chen et al., 1996a). As illustrated in Fig. 8, the original ME architecture remains in the modified ME architecture to deal with the instantaneous information based on each feature frame of an utterance and a new gating network called *S*-gating is added for use of the transitional spectral information. Accordingly, a generalized finite mixture model is defined as follows

$$P(\mathbf{y}|\mathbf{X},\mathbf{\Theta}) = \sum_{t=1}^{T} \lambda_{\mathbf{X}}(\mathbf{x}_t,\mathbf{\Phi}) \sum_{i=1}^{N} g(\mathbf{x}_t,\mathbf{v}_i) P(\mathbf{y}|\mathbf{x}_t,\mathbf{W}_i), \qquad (55)$$

where $\mathbf{\Theta}$ is the set of all the free parameters which include the expert network parameters $\mathbf{W}_\mathrm{i}$, the gating network parameters $\mathbf{v}_i$, and the *S*-gating network parameters $\mathbf{\Phi}$. The $\lambda_x(x_t)$, for $\mathbf{x}_t$, in $X$ is defined as

$$\lambda_{\mathbf{X}}(\mathbf{x}_t) = \frac{P(\mathbf{x}_t,\mathbf{\Phi})}{\sum_{s=1}^{T} P(\mathbf{x}_s,\mathbf{\Phi})}, \qquad (56)$$

where

$$P(\mathbf{x}_t,\Phi) = \frac{1}{(2\pi)^{(n/2)}|\Sigma|^{(1/2)}} \exp\{-\frac{1}{2}(\mathbf{x}_t - m)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}_t - \mathbf{m})\}$$

and $\mathbf{\Phi} = (\mathbf{m},\Sigma)$. In the architecture, all the problems of parameter estimation in the gating and expert networks are the same as the original ME architecture except that the parameter estimation in the *S*-gating network can be performed analytically (Chen et al., 1996a).

We have also applied the modified ME architecture to speaker identification. In simulations, six expert networks were used in each modified ME classifier. Table 10 summarizes the averaging results produced by 10 modified ME classifiers. For comparison, we also adopted the IRLS algorithm and the BFGS algorithm in the inner loop of the EM algorithm, respectively, to train the modified ME classifiers. Simulation results are also shown in Table 10. Simulation results show that the use of the proposed learning algorithm produces the best identification rates in general. Evidently, the use of our approximation algorithm also leads to satisfactory identification results and fast learning. The ratio of mean relative learning time used between the Newton–Raphson algorithm and its approximation is about 1:0.4026. In contrast, the performance of our approximation algorithm is similar to that of the BFGS algorithm in general, but the BFGS algorithm still does not yield significantly faster learning as shown in Table 10. Once again, the use of the IRLS algorithm still results in the poor performance. In particular, the same problem happened again; that is, when the IRLS algorithm was used in the inner loop of the EM algorithm, the modified ME classifier corresponding to the digit ‘2’ still cannot reach the steady state even though the learning was prolonged up to 50 epochs.

## 7. Discussion

As a realization of multinomial density for multiclass classification, the multinomial logit or softmax function results in mutual dependency among components of an output vector. Due to this dependency, each row of the weight matrix of a component network in the ME architecture cannot be viewed as an independent and separable parameter vector. Therefore, the exact evaluation of the Hessian matrix must be performed by considering all the parameters in the weight matrix simultaneously. As a special case of the Fisher scoring method (McCullagh & Nelder, 1983), the IRLS algorithm is a second-order optimal method in which the information of the exact Hessian matrix is still required. Unfortunately, the derivation of the IRLS algorithm used for the ME architecture is based on the assumption that row vectors of the weight matrix are independent (Jordan & Jacobs, 1994). The assumption results in an inexact evaluation of the Hessian matrix and the instability of learning for the ME architecture in multiclass classification. Since the gating network in the original ME architecture (Jacobs et al., 1991; Jordan & Jacobs, 1994) is modeled as a multinomial distribution for task allocation, the aforementioned problem always holds for the ME architecture regardless of regression and classification if the IRLS algorithm is employed in the inner loop of the EM algorithm to train the gating network. As a result, our studies in this paper have shown that the use of the incomplete Hessian matrix in the IRLS algorithm is responsible for the poor performance of the ME architecture, in particular, in multiclass classification. Our studies indicate that there are two ways to tackle the problem. One is to change the architectures of gating and expert networks

Table 10
Speaker identification: performance of the modified mixture of experts trained by different learning algorithms in the inner step of the EM algorithm

| Algorithm | TEST-1 | TEST-2 | Log-likelihood | epochs | CPU time (min) |
|---|---|---|---|---|---|
| Newton–Raphson | 94.5 | 93.8 | − 0.1734 | 7.3 | 13:42 |
| Approximation | 93.4 | 93.1 | − 0.1934 | 8.0 | 5:30 |
| BFGS | 93.5 | 91.0 | − 0.2172 | 11.4 | 12:42 |
| IRLS | 88.7 | 86.7 | − 0.2840 | 15.7 | 8:32 |

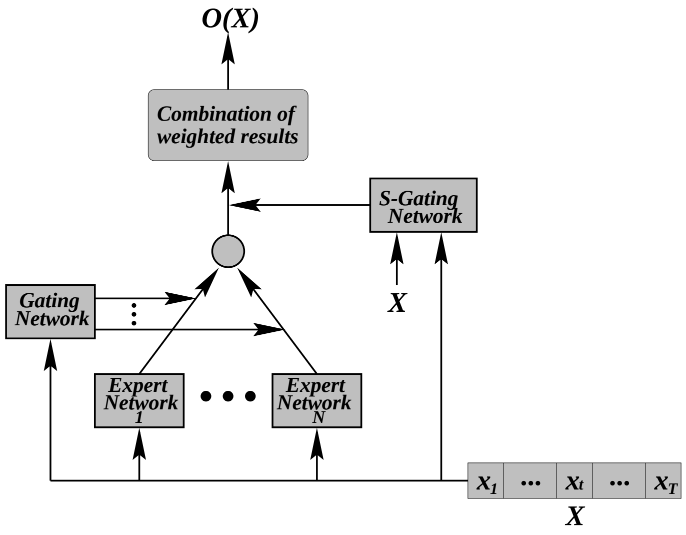


Fig. 8. The architecture of a modified mixture of experts for speaker identification.

such that parameter estimation becomes analytically solvable, which leads to the alternative ME model (Xu & Jordan, 1994; Xu et al., 1995). The other is to develop a new learning algorithm used in the inner loop of the EM algorithm instead of the IRLS algorithm. Since the original ME model with the IRLS algorithm for learning has been widely used in literature, it is necessary and nontrivial to develop such an learning algorithm to make the original ME architecture work for multiclass classification. For this purpose, we have proposed a second-order learning algorithm by means of the Newton–Raphson method in this paper, where the exact Hessian matrix is used.

In order to speed up learning, we have presented an approximation algorithm based on the proposed statistical model for expert networks in multiclass classification. As a result, the proposed Newton–Raphson algorithm is still applied to its exact form and the proposed statistical model causes the Hessian matrix to be simplified. In nonlinear programming, a class of quasi-Newton methods have also been proposed to reduce the computational burden of the Newton–Raphson method from a general perspective. As a typical quasi-Newton method, the BFGS algorithm also provides an alternative approximation to the proposed Newton–Raphson algorithm. In principle, however, the two approximation algorithms are totally distinct: The BFGS algorithms uses a trick to avoid the calculation of inverse of the Hessian matrix in general, while our method adopts an approximate statistical model to simplify the calculation of the Hessian matrix specially for the ME architecture in multiclass classification. Simulation results indicate that our approximation algorithm yields faster learning although the performance of the BFGS algorithm is similar to that of our approximation algorithm for most of problems used in this paper. On the other hand, it could be argued that the use of the incomplete Hessian matrix in the IRLS algorithm might be also regarded as an approximation to the proposed Newton–Raphson algorithm. However, both simulation results reported in literature and our studies in this paper have shown that the unreasonable independence assumption on parameter vectors causes the ME architecture to produce the poor performance in multiclass classification. In terms of the practical performance, the IRLS algorithm cannot be viewed as an effective approximation. For multinomial logistic regression, other modified Newton–Raphson algorithms (Neter, Wassermand & Kutner, 1985; Böning, 1993) have been proposed to achieve reliable maximum likelihood during parameter estimation. However, their performance is still not clear when they are applied in the inner loop of the EM algorithm to train the ME architecture.

Theoretically, the computational complexity of the Hessian matrix is $O(W^3)$ where $W$ is the number of parameters. In multiclass classification with $K$ categories, there are $(K-1)\times n$ parameters in a component network in the ME architecture where $n$ is the dimensionality of input vectors, while our approximation algorithm merely needs to calculate $K$ Hessian matrices with $n$ parameters instead of the calculation of the Hessian matrix with $(K-1)\times n$ parameters. From a large number of simulations, however, we find out that the Newton–Raphson algorithm does not yield prohibitively slow learning. Due to the elegant property of the Hessian matrix, the Cholesky decomposition method (Golub & Van Loan, 1989) can be used for achieving inverse of the Hessian matrix. Empirical studies indicate that fewer iterations in the inner loop and fewer epochs in the EM algorithm are merely needed for each component network of the ME architecture to reach the steady state. In contrast, our simulation results also indicate that the BFGS algorithm does not yield significantly faster learning for problems studied in this paper. In particular, the BFGS algorithm is often sensitive to the learning rate though the high precision linear search is not necessary theoretically (Fletcher, 1987; Minoux, 1986). Our simulation results are consistent with some theoretical arguments that the quasi-Newton methods can not be effective in practice (Fletcher, 1987; Minoux, 1986).

## 8. Concluding remarks

We have investigated the reason why the ME architecture produces the poor performance in the case of multiclass classification when the IRLS algorithm is used in the inner loop of the EM algorithm to train the ME architecture. We find out that an incorrect assumption on parameter independence is implicitly imposed for multiclass classification, which causes an incomplete Hessian matrix is used in the second-order IRLS algorithm. The use of the incomplete Hessian matrix is responsible for the poor performance of the ME architecture in multiclass classification. On the basis of the finding, we propose a learning algorithm by means of the Newton–Raphson method to replace the IRLS algorithm and an approximation algorithm to speed up learning. Simulation results by the proposed learning algorithm readily improves the performance of the ME architecture and our approximation algorithm yields significantly fast learning through it is subject to limitation for those highly inseparable problems.

## Acknowledgements

Authors are grateful to the anonymous referees for their comments that improve the presentation of this paper. This work was supported by the HK RGC Earmarked Grant CUHK 250/94E and National Science Foundation in china.

## Appendix A

In this appendix, we give the method of calculating the derivative of the softmax function $\boldsymbol{g_k}(\boldsymbol{\Theta}, \mathbf{x_t})$ defined in Eq. (20). In order to evaluate the derivatives of $l(\boldsymbol{\Theta}, \chi)$, it is necessary to consider three different cases of the derivative of $g_k\,(\boldsymbol{\Theta}, \mathbf{x}_t)$ $(1 \le k \le K)$ on a parameter vector $\boldsymbol{\beta}_\mathrm{q}$ in $\boldsymbol{\Theta}$ as follows:

*Case 1:* for $k \neq q$ and $k \neq K$,

$$\frac{\partial g_k(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \boldsymbol{\beta}_q} = \frac{\partial g_k(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \eta_{tq}} \frac{\partial \eta_{tq}}{\partial \boldsymbol{\beta}_q} = \frac{-\mathrm{e}^{\eta_{tk}}\mathrm{e}^{\eta_{tq}}}{\left(1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}\right)^2}\mathbf{x}_t$$

$$= \left(\frac{-\mathrm{e}^{\eta_{tk}}}{1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}}\right)\left(\frac{\mathrm{e}^{\eta_{tk}}}{1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}}\right)\mathbf{x}_t$$

$$= -g_k(\boldsymbol{\Theta}, \mathbf{x}_t)g_q(\boldsymbol{\Theta}, \mathbf{x}_t)\mathbf{x}_t \tag{A.1}$$

*Case 2:* $k = q$ and $q \neq K$,

$$\frac{\partial g_q(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \boldsymbol{\beta}_q} = \frac{\partial g_q(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \eta_{t_q}} \frac{\partial \eta_{tq}}{\partial \boldsymbol{\beta}_q}$$

$$= \left[\frac{\mathrm{e}^{\eta_{tq}}}{\left(1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}\right)} - \frac{\mathrm{e}^{\eta_{tq}}\mathrm{e}^{\eta_{tq}}}{\left(1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}\right)^2}\right]\mathbf{x}_t$$

$$= [g_q(\boldsymbol{\Theta}, \mathbf{x}_t) - g_q^2(\boldsymbol{\Theta}, \mathbf{x}_t)]\mathbf{x}_t$$

$$= g_q(\boldsymbol{\Theta}, \mathbf{x}_t)[1 - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)]\mathbf{x}_t \tag{A.2}$$

*Case 3:* $k = K$,

$$\frac{\partial g_K(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \boldsymbol{\beta}_q} = \frac{\partial g_K(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \eta_{tq}} \frac{\partial \eta_{tq}}{\partial \boldsymbol{\beta}_q} = \frac{-\mathrm{e}^{\eta_{tq}}}{\left(1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}\right)^2}\mathbf{x}_t$$

$$= \left(\frac{-1}{1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}}\right)\left(\frac{-\mathrm{e}^{\eta_{tq}}}{1 + \sum_{i=1}^{K-1} \mathrm{e}^{\eta_{ti}}}\right)\mathbf{x}_t$$

$$= -g_k(\boldsymbol{\Theta}, \mathbf{x}_t)g_q(\boldsymbol{\Theta}, \mathbf{x}_t)\mathbf{x}_t \tag{A.3}$$

According to all three cases, for $q = 1, \ldots, K - 1$, we obtain

$$\frac{\partial g_k(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \boldsymbol{\beta}_q} = \frac{\partial g_k(\boldsymbol{\Theta}, \mathbf{x}_t)}{\partial \boldsymbol{\beta}_q} \frac{\partial \eta_{tq}}{\partial \boldsymbol{\beta}_q}$$

$$= g_k(\boldsymbol{\Theta}, \mathbf{x}_t)[\delta_{kg} - g_q(\boldsymbol{\Theta}, \mathbf{x}_t)]\mathbf{x}_t \tag{A.4}$$

where $\delta_{kq}$ is the Kronecker delta.

## References

Bennani, Y., & Gallinari, P. (1994). Connectionist approaches for automatic speaker recognition. *Proceeding of ESCA Workshop on Automatic Speaker Recognition, Identification and Verification*, Martigny, Switzerland, pp. 95-102.

Bishop, C. (1991). A fast procedure for re-training the multilayer perceptron. *International Journal of Neural Systems*, *2* (3), 229–396.

Bishop, C. (1992). Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Computation*, *4* (4), 494–501.

Böning, D. (1993). Construction of reliable maximum likelihood algorithms with applications to logistic and Cox regression. In C. R. Rao (Ed.), *Computational statistics*, (pp. 409–422). New York: North-Holland.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*, Monterey: Wadsworth and Brooks.

Bridle, J. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman-Soulie & J. Herault (Eds.), *Neurocomputing: algorithm, architectures, and applications*, (pp. 227–236). New York: Springer.

Broyden, C. G. (1970). The convergence of a class of double rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, *6*, 76–90.

Campbell, F. P. (1997). Speaker recognition: a tutorial. *Proceedings of the IEEE*, *85* (9), 1437–1463.

Chen, K. (1998). A connectionist method for pattern classification with diverse features. *Pattern Recognition Letters*, *19* (7), 545–558.

Chen, K., Xie, D., & Chi, H. (1995). Speaker identification based on hierarchical mixture of experts. *Proceedings of World Congress on Neural Networks, Washington, DC*,, 1493–1496.

Chen, K., Xie, D., & Chi, H. (1996). A modified HME architecture for text-dependent speaker identification. *IEEE Transactions on Neural Networks*, *7*(5): 1309–1313 (for errata see *IEEE Transactions on Neural Networks*, *8*(2): 455, 1997).

Chen, K., Xie, D., & Chi, H. (1996). Speaker identification using time-delay HMEs. *International Journal of Neural Systems*, *7* (1), 29–43.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39* (1), 1–38.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*, New York: Wiley.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problem. *Annals of Eugenices*, *7*, 179–188.

Fletcher, R. (1970). A general quadratic programming algorithm. *Computer Journal*, *13*, 317–322.

Fletcher, R. (1987). *Practical methods of optimization*, New York: Wiley.

Furui, S. (1997). Recent advances in speaker recognition. *Pattern Recognition Letters*, *18* (9), 859–872.

Goldfarb, D. (1970). A family of variable metric methods derived by variational means. *Mathematics of Computation*, *24*, 23–26.

Golub, G. H., & Van Loan, G. (1989). *Matrix computations*, Baltimore: John Hopkins University Press.

Guo, H., & Gelfand, S. B. (1992). Classification trees with neural network decision trees. *IEEE Transactions on Neural Networks*, *3* (5), 923–933.

Ishikawa, M. (1996). Structural learning with forgetting. *Neural Networks*, *9* (3), 509–521.

Jacobs, R. A., Jordan, M. I., Nowlan, S., & Hinton, G. (1991). Adaptive mixture of local experts. *Neural Computation*, *3* (1), 79–87.

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, *6* (2), 181–214.

Jordan, M. I., & Xu, L. (1995). Convergence results for the EM approach to mixtures of experts. *Neural Networks*, *8* (9), 1409–1431.

McCullagh, P., & Nelder, J. A. (1983). *Generalized linear models*, London: Chapman and Hall.

Minoux, M. (1986). *Mathematical programming: theory and algorithms*, New York: Wiley.

Neter, J., Wassermand, W., & Kutner, M. H. (1985). *Applied linear statistical models*, Homewood: Richard Irwin.

Ramamurti, V., & Ghosh, J. (1996). Advances in using hierarchical mixture of experts for signal classification. *Proceedings of International Conference on Acoustic, Speech, and Signal Processing*, Atlanta, pp. 3569–3572.

Ramamurti, V., & Ghosh, J. (1997). Regularization and error bars for the mixture of experts network. *Proceedings of IEEE International Conference of Neural Networks*, Houston, pp. 221–225.

Shanno, D. F. (1970). On variable metric methods for sparse Hessians. *Mathematics of Computation*, *24*, 647–657.

Waterhouse, S.R. (1993). The application of HME with the EM algorithm to speech recognition. Master Thesis, Department of Engineering, Cambridge University.

Waterhouse, S.R. (1997). Classification and regression using mixtures of experts. Ph.D., Thesis, Department of Engineering, Cambridge University.

Xu, L. (1996). Bayesian-Kullback YING-YANG learning scheme: reviews and new results, Proceedings of International Conference on Neural Information Processing, Hong Kong, pp. 59-67.

Xu, L. (1998). *RBF nets, mixture of experts, and Bayesian Ying-Yang learning, Neurocomputing*, *19* (1-3), 223–257.

Xu, L., & Jordan, M.I. (1994). A modified gating network for the mixture of experts architecture. *Proceedings of World Congress on Neural Networks*, San Diego, pp. II405–II410.

Xu, L., & Jordan, M. I. (1996). On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, *8* (2), 129–151.

Xu, L., Jordan, M. I., & Hinton, G. E. (1995). Advances in neural information processing systems. In G. Tesauro & D. Touretzky & T. Leed (Eds.), *Advances in Neural Information Processing Systems*, (pp. 633–640). Cambridge, MA: MIT Press.

## Further reading

Bengio, Y., & Frasconi, P. (1996). Input/output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, *7* (5), 1231–1249.

Chen, K., & Chi, H. (1998). A method of combining multiple probabilistic classifiers through soft competition on different features sets. *Neurocomputing*, *20* (1-3), 227–252.

Chen, K., Xie, D., & Chi, H. (1996). Text-dependent speaker identification based on input/output HMM: an empirical study. *Neural Processing Letters*, *3* (2), 81–89.