Using the EM Algorithm to Train Neural Networks: Misconceptions and a New Algorithm for Multiclass Classification

Shu-Kay Ng and Geoffrey John McLachlan

***Abstract*—The expectation-maximization (EM) algorithm has been of considerable interest in recent years as the basis for various algorithms in application areas of neural networks such as pattern recognition. However, there exists some misconceptions concerning its application to neural networks. In this paper, we clarify these misconceptions and consider how the EM algorithm can be adopted to train multilayer perceptron (MLP) and mixture of experts (ME) networks in applications to multiclass classification. We identify some situations where the application of the EM algorithm to train MLP networks may be of limited value and discuss some ways of handling the difficulties. For ME networks, it is reported in the literature that networks trained by the EM algorithm using iteratively reweighted least squares (IRLS) algorithm in the inner loop of the M-step, often performed poorly in multiclass classification. However, we found that the convergence of the IRLS algorithm is stable and that the log likelihood is monotonic increasing when a learning rate smaller than one is adopted. Also, we propose the use of an expectation-conditional maximization (ECM) algorithm to train ME networks. Its performance is demonstrated to be superior to the IRLS algorithm on some simulated and real data sets.**

***Index Terms*—Expectation-conditional maximization (ECM) algorithm, expectation-maximization (EM) algorithm, mixture of experts, multiclass classification, multilayer perceptron (MLP), variational relaxation.**

I. Introduction

In many important application areas such as control, pattern recognition, and signal processing, nonlinear adaptive systems are needed to approximate underlying nonlinear mappings through learning from examples. Neural networks have been used as such nonlinear adaptive systems, since they can be regarded as universal function approximators of nonlinear functions that can be trained (learned) from examples of input-output data. When the data includes noise, the input–output relation for a neural network is described stochastically in terms of the conditional probability $\mathrm{pr}(\boldsymbol{y}|\boldsymbol{x})$ of the output $\boldsymbol{y}$ given the input $\boldsymbol{x}$. Some neural networks (for example, Boltzmann machines) do have explicit probabilistic components in their definition. However, even when a neural network is deterministic, it can be effective to train it as if it were a stochastic network, although it behaves deterministically in the execution model [1]. By working with a stochastic version of a deterministic network, we are able to employ statistical methodology in the learning process of the network [2], [3].

Manuscript received April 13, 2003; revised January 23, 2004. This work was supported in part by grants from the Australian Research Council.

The authors are with the Department of Mathematics, University of Queensland, Brisbane QLD 4072, Australia (e-mail: skn@maths.uq.edu.au).

Digital Object Identifier 10.1109/TNN.2004.826217

The expectation-maximization (EM) algorithm [4] has been of considerable interest in recent years in the development of algorithms in various application areas of neural networks; see for example [5]–[8]. Amari [1], [9] has looked at the application of the EM algorithm to neural networks from a geometrical perspective in terms of the information geometry. In related work, Jacobs *et al.* [10] and Jordan and Xu [11] have considered the use of the EM algorithm to train mixtures of experts models, which are weighted combinations of subnetworks, while Jordan and Jacobs [12] and Jacobs, Peng, and Tanner [13] have considered hierarchical mixtures of experts and the EM algorithm. The statistical framework of the EM algorithm allows us to treat the learning process as a maximum likelihood (ML) problem, so standard likelihood-based methodology can be used to train the neural networks in the first instance and to subsequently obtain confidence intervals for a predicted output $\boldsymbol{y}$ corresponding to an input $\boldsymbol{x}$. The EM algorithm has a number of desirable properties, including its numerically stability, simplicity of implementation, and reliable global convergence [14, Sec. 1.7].

The EM algorithm is a broadly applicable approach to the iterative computation of ML estimates, useful in a variety of incomplete-data problems. It is based on the idea of solving a succession of simpler problems that are obtained by augmenting the original observed variables (the incomplete-data) with a set of additional variables that are unobservable or unavailable to the user. These additional data are referred to as the missing data $\boldsymbol{z}$ in the EM framework. The EM algorithm is closely related to the *ad hoc* approach to estimation with missing data, where the parameters are estimated after filling in initial values for the missing data. The latter are then updated by their predicted values using these initial parameter estimates. The parameters are then reestimated, and so on, proceeding iteratively until convergence. On each iteration of the EM algorithm, there are two steps called the expectation (E) step and the maximization (M) step. The E-step involves the computation of the so-called Q-function, which is given by the conditional expectation of the complete-data log likelihood given the observed data and the current estimates. The M-step updates the estimates that maximizes the Q-function over the parameter space. A detailed account of the properties and applications of the EM algorithm and its more recent extensions such as the expectation-conditional maximization (ECM) and the expectation-conditional maximization either (ECME) algorithms are provided in the monograph [14]. There is now a whole battery of EM-related algorithms and more are still being developed. Unfortunately, there exists some misunderstanding about its applications in training a neural network.

In some instances, the conditional expectation of the complete-data log likelihood (the E-step) is effected simply by replacing the random vector of missing data $\boldsymbol{z}$ by its conditional expectation. However, this will be valid only if the complete-data log likelihood were linear in $\boldsymbol{z}$. Unfortunately, it is in general not true and this condition often seems to be neglected in the application of the EM algorithm in the training process of neural networks. In the paper, we clarify this misconception about the implementation of the EM algorithm in neural networks. We also investigate its application to train multilayer perceptron (MLP) networks and mixture of experts (ME) neural networks in applications to multiclass classification problems. We identify some situations where the application of the EM algorithm for training MLP networks may be of limited value due to complications in performing the E-step. A computationally intensive Monte Carlo EM (MCEM) algorithm may be adopted in this situation to approximate the E-step. Alternatively, a variational relaxation approach derived from a mean field approximation can be used to obtain an approximate E-step. These approaches will be described further in Section III. For the ME networks, it is reported in the literature [15] that networks trained by the EM algorithm using the IRLS algorithm in the inner loop of the M-step, often performed poorly in multiclass classification because of the incorrect assumption on parameter independence. In our study, we found that the convergence of the IRLS algorithm is stable and that the log likelihood is monotonic increasing even though the assumption of independence is incorrect. Moreover, we present in the paper a new ECM algorithm to train ME networks for multiclass classification such that the parameters in the gating and expert networks are separable. Thus, the independence assumption is not required and the parameters in both gating and expert networks can be estimated separately.

The rest of the paper is organized as follows: Section II describes briefly the EM algorithm and the misconception of its application in training neural networks. In Section III, we show how the EM algorithm can be adopted to train MLP neural networks. An integration of the methodology related to the EM training of radial basis function (RBF) networks is also presented. In Section IV, we propose a new ECM algorithm to train ME networks for multiclass classification problems. Section V reports simulation results to investigate the performance of the ECM algorithm for training the ME networks. In Section VI, the algorithm is illustrated using three real examples, and Section VII ends the paper by presenting some concluding remarks.

II. The EM Algorithm

In the sequel, we shall assume that the neural network is being used in a multiclass classification context. In the classification context, there are g populations or groups, $G_1, \ldots, G_g$ and the problem is to infer the unknown membership of an unclassified entity with feature vector of p-dimensions. This membership can be defined by a g-dimensional output vector of zero-one indicator variables, where the ith element of the output vector is one or zero, according as the entity does or does not belong to the ith group $G_i (i = 1, \ldots, g)$. We let

$$\left(\boldsymbol{x}_1^T, \boldsymbol{y}_1^T\right)^T, \ldots, \left(\boldsymbol{x}_n^T, \boldsymbol{y}_n^T\right)^T \tag{1}$$

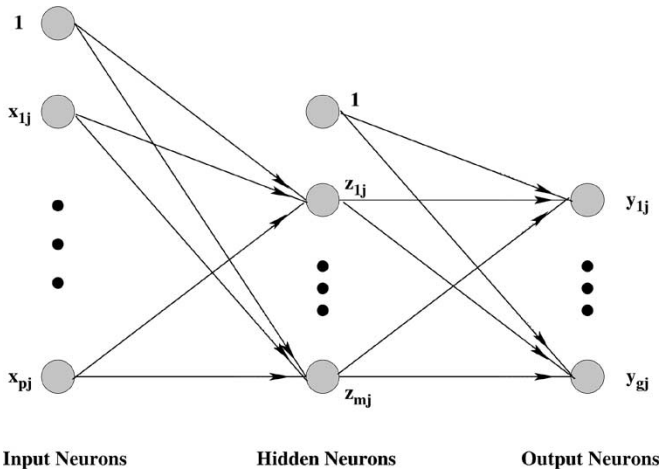


Fig. 1. MLP neural networks.

denote the n examples available for training the neural network, where the superscript T denotes vector transpose, $\boldsymbol{x}_j = (x_{1j}, \ldots, x_{pj})^T$ is an input feature vector, and $\boldsymbol{y}_j = (y_{1j}, \ldots, y_{gj})^T$ is an output vector with $j = 1, \ldots, n$. In the training process, the unknown parameters in the neural network, denoted by a vector $\boldsymbol{\Psi}$, are inferred from the observed training data given by (1). We let $\boldsymbol{x} = (\boldsymbol{x}_1^T, \ldots, \boldsymbol{x}_n^T)^T$ and $\boldsymbol{y} = (\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_n^T)^T$. In order to estimate $\boldsymbol{\Psi}$ by the statistical technique of maximum likelihood, we have to impose a statistical distribution for the observed data (1), which will allow us to form a log likelihood function, $\log L(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{x})$, for $\boldsymbol{\Psi}$. In general, we proceed conditionally on the values for the input variable $\boldsymbol{x}$; that is, we shall consider the specification of the conditional distribution of the random variable $\boldsymbol{Y}$ corresponding to the observed output $\boldsymbol{y}$ given the input $\boldsymbol{x}$ as

$$\log L(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{x}) \propto \log \mathrm{pr}(\boldsymbol{Y}|\boldsymbol{x}; \boldsymbol{\Psi}).$$

The EM algorithm is a popular tool for the iterative computation of ML estimates, useful in a variety of incomplete-data problems, where algorithms such as gradient ascent methods may turn out to be more complicated. Further details on the EM algorithm in a general context can be found in the monograph of McLachlan and Krishnan [14]. Within the EM framework, the unknown vector $\boldsymbol{\Psi}$ is estimated by consideration of the complete-data log likelihood (that is, the log likelihood function for $\boldsymbol{\Psi}$ based on both the observed and the missing data $\boldsymbol{z}$). For example, with MLP neural networks (Fig. 1), an obvious choice for the missing data is the set of hidden units whose values cannot be observed directly. The complete-data log likelihood formed on the basis of both the observed and the missing data is given by

$$\begin{aligned} \log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x}) &\propto \log \mathrm{pr}(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{x}; \boldsymbol{\Psi}) \\ &= \log \mathrm{pr}(\boldsymbol{Y}|\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\Psi}) \\ &\quad + \log \mathrm{pr}(\boldsymbol{Z}|\boldsymbol{x}; \boldsymbol{\Psi}). \end{aligned} \tag{2}$$

That is, we need to specify the distribution of the random variable $\boldsymbol{Z}$, conditional on $\boldsymbol{x}$, and the conditional distribution of $\boldsymbol{Y}$ given $\boldsymbol{x}$ and $\boldsymbol{z}$. On the $(k+1)$th iteration of the EM algorithm, the E-step computes the Q-function, which is given by

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}\{\log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x})|\boldsymbol{y}, \boldsymbol{x}\} \tag{3}$$

where $E_{\Psi^{(k)}}$ denotes the expectation operator using the current value $\Psi^{(k)}$ for Ψ. On the M-step, $\Psi^{(k)}$ is updated by taking $\Psi^{(k+1)}$ to be the value of Ψ that maximizes $Q(\Psi;\Psi^{(k)})$ over all admissible values of Ψ.

As described in Section I, in some instances, a modified form of the EM algorithm is being used unwittingly by the author(s) in that on the E-step, the conditional expectation of the complete-data log likelihood, the Q-function, is effected simply by replacing the random vector $\boldsymbol{z}$ by its conditional expectation. For example, in (3), [16], [17] is computed by the approximation

$$Q(\Psi;\Psi^{(k)}) \approx \log L_c(\Psi;\boldsymbol{y},\tilde{\boldsymbol{z}},\boldsymbol{x}) \tag{4}$$

where

$$\tilde{\boldsymbol{z}} = E_{\Psi^{(k)}}\{\boldsymbol{Z}|\boldsymbol{y},\boldsymbol{x}\}.$$

However, the approximation (4) will be valid only in special cases. It is valid if the complete-data log likelihood were linear in $\boldsymbol{z}$ as in the ME neural networks presented in Section IV [11], [12], but in general it is not. For instance, it is a nonlinear function of $\boldsymbol{z}$ for MLP neural networks (see Section III) and it is a quadratic function of $\boldsymbol{z}$ for the regression models in both the RBF network of [16] and the Sugeno-type model of [17].

III. Training Multilayer Perceptron Networks

An MLP neural network constructs a decision surface in the data space for discriminating instances with similar features by forming a boundary between them. For a MLP neural network with one hidden layer of m units (Fig. 1), we can specify a stochastic model of MLP neural network as follows. Let z_{hj} $(h = 1,\ldots,m; j = 1,\ldots,n)$ be the realization of the zero-one random variable Z_{hj} for which its conditional distribution given $\boldsymbol{x}_j$ is specified by

$$\mathrm{pr}(Z_{hj} = 1|\boldsymbol{x}_j) = \frac{\exp(\boldsymbol{w}_h^T\boldsymbol{x}_j)}{1+\exp(\boldsymbol{w}_h^T\boldsymbol{x}_j)} \tag{5}$$

where $\boldsymbol{w}_h = (w_{h0}, w_{h1},\ldots,w_{hp})^T$ is the synaptic weight vector of the hth hidden unit. The bias term w_{h0} is included in $\boldsymbol{w}_h$ by adding a constant input $x_{0j} = 1$ for all $j = 1,\ldots,n$ so that the input is now $\boldsymbol{x}_j = (x_{0j}, x_{1j},\ldots,x_{pj})^T$; that is

$$\boldsymbol{w}_h^T\boldsymbol{x}_j = \sum_{l=1}^{p} w_{hl}x_{lj} + w_{h0} = \sum_{l=0}^{p} w_{hl}x_{lj}.$$

The probability $\mathrm{pr}(Z_{hj} = 0|\boldsymbol{x}_j)$ is equal to $1 - \mathrm{pr}(Z_{hj} = 1|\boldsymbol{x}_j)$. That is, Z_{hj} has a Bernoulli distribution. The output of g-dimensional zero-one indicator variables $\boldsymbol{Y}_j$ is distributed according to a multinomial distribution consisting of one draw on g cells with probabilities

$$\mathrm{pr}(Y_{ij} = 1|\boldsymbol{x}_j,\boldsymbol{z}_j) = \frac{\exp(\boldsymbol{v}_i^T\boldsymbol{z}_j)}{\sum_{r=1}^{g}\exp(\boldsymbol{v}_r^T\boldsymbol{z}_j)} \tag{6}$$

for $i = 1,\ldots,g$, where $\boldsymbol{v}_i = (v_{i0}, v_{i1},\ldots,v_{im})^T$ is the synaptic weight vector of the ith output unit. The bias term v_{i0} is included in $\boldsymbol{v}_i$ by adding a constant hidden unit $z_{0j} = 1$ for all $j = 1,\ldots,n$ so that the hidden layer is now $\boldsymbol{z}_j = (z_{0j}, z_{1j},\ldots,z_{mj})^T$, that is

$$\boldsymbol{v}_i^T\boldsymbol{z}_j = \sum_{h=1}^{m} v_{ih}z_{hj} + v_{i0} = \sum_{h=0}^{m} v_{ih}z_{hj}.$$

In the EM framework, the missing data are then given by $\boldsymbol{z} = (\boldsymbol{z}_1^T,\ldots,\boldsymbol{z}_n^T)^T$.

The term on the right-hand side (RHS) of (6) is known as the normalized exponential, or *softmax* function [18]. This function represents a smooth version of the *winner-takes-all* activation model in which the unit with the largest input has output 1 while all other units have output 0. It can be seen from (6) that the probabilities are unchanged whenever the same additive constant is added to $\boldsymbol{v}_i^T\boldsymbol{z}_j$ $(i = 1,\ldots,g)$. For uniqueness, we therefore set $v_{gh} = 0$ for $h = 0, 1,\ldots,m$ [19]. This corresponds to a network with $(g-1)$ output neurons in Fig. 1.With the case of $g = 2$, (6) reduces to the logistic transformation. This stochastic specification of the MLP neural network, using the Bernoulli distribution for $\boldsymbol{z}_j$ and the multinomial distribution for $\boldsymbol{y}_j$, has been considered by Amari [1]. He pointed out that the EM algorithm for training stochastic MLP is more flexible with a better global convergence property than the back propagation approach. Ma, Ji, Farmer [20] considered a MLP network in which the output $\boldsymbol{y}_j$ was linear in $\boldsymbol{z}_j$ and the stochastic model was specified by assuming the conditional distribution $\mathrm{pr}(\boldsymbol{Z}|\boldsymbol{x};\Psi)$ to be Gaussian with known variance. Aitkin and Foxall [21] studied a latent variable LV-MLP model which is a special case of the ME networks to be presented in Section IV.

With reference to (5) and (6), the vector of all the unknown parameters is given by $\Psi = (\boldsymbol{w}_1^T,\ldots,\boldsymbol{w}_m^T,\boldsymbol{v}_1^T,\ldots,\boldsymbol{v}_{g-1}^T)^T$. The ML estimate of Ψ is obtained via the EM algorithm. Precisely, it follows from (5) and (6), respectively, that:

$$\mathrm{pr}(\boldsymbol{Z}|\boldsymbol{x};\Psi) = \prod_{j=1}^{n}\prod_{h=1}^{m} u_{hj}^{z_{hj}}(1-u_{hj})^{(1-z_{hj})}$$

and

$$\mathrm{pr}(\boldsymbol{Y}|\boldsymbol{x},\boldsymbol{z};\Psi) = \prod_{j=1}^{n}\prod_{i=1}^{g} o_{ij}^{y_{ij}}$$

where

$$u_{hj} = \mathrm{pr}(Z_{hj} = 1|\boldsymbol{x}_j) = \frac{\exp\left(\sum_{l=0}^{p} w_{hl}x_{lj}\right)}{1+\exp\left(\sum_{l=0}^{p} w_{hl}x_{lj}\right)} \tag{7}$$

for $h = 1,\ldots,m$

$$\begin{aligned} o_{ij} &= \mathrm{pr}(Y_{ij} = 1|\boldsymbol{x}_j,\boldsymbol{z}_j) \\ &= \frac{\exp\left(\sum_{h=0}^{m} v_{ih}z_{hj}\right)}{1+\sum_{r=1}^{g-1}\exp\left(\sum_{h=0}^{m} v_{rh}z_{hj}\right)} \end{aligned} \tag{8}$$

for $i = 1, \ldots, g-1$, and

$$o_{gj} = \mathrm{pr}(Y_{gj} = 1 | \boldsymbol{x}_j, \boldsymbol{z}_j) = \frac{1}{1 + \sum_{r=1}^{g-1} \exp\left(\sum_{h=0}^{m} v_{rh} z_{hj} \right)}. \quad (9)$$

From (2), apart from an additive function not involving $\boldsymbol{\Psi}$, the complete-data log likelihood for $\boldsymbol{\Psi}$, $\log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x})$, is given by

$$\sum_{j=1}^{n} \left\{ \sum_{h=1}^{m} [z_{hj} \log \frac{u_{hj}}{1 - u_{hj}} + \log(1 - u_{hj})] + \sum_{i=1}^{g} y_{ij} \log o_{ij} \right\}. \quad (10)$$

It follows on application of the EM algorithm in training MLP networks that on the $(k+1)$th iteration of the E-step, we calculate the expectation of $\log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x})$ conditional on the current estimate $\boldsymbol{\Psi}^{(k)}$ of the parameter and the observed input and output vectors (the Q-function). In view of (7)–(10), the Q-function can be decomposed as

$$\begin{aligned} Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) &= E_{\boldsymbol{\Psi}^{(k)}} \{ \log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x}) | \boldsymbol{y}, \boldsymbol{x} \} \\ &= \sum_{j=1}^{n} \sum_{h=1}^{m} \left[E_{\boldsymbol{\Psi}^{(k)}}(Z_{hj} | \boldsymbol{y}, \boldsymbol{x}) \times \log \frac{u_{hj}}{1 - u_{hj}} + \log(1 - u_{hj}) \right] \\ &\quad + \sum_{j=1}^{n} \sum_{i=1}^{g} y_{ij} E_{\boldsymbol{\Psi}^{(k)}}(o_{ij} | \boldsymbol{y}, \boldsymbol{x}) \\ &= Q_w + Q_v \end{aligned} \quad (11)$$

with respect to the unknown parameters $\boldsymbol{w}_h$ $(h = 1, \ldots, m)$ and $\boldsymbol{v}_i$ $(i = 1, \ldots, g-1)$, respectively.

The first term of the complete-data log likelihood (10) is linear in $\boldsymbol{z}$, so its expectation can be replaced by the expectation of $\boldsymbol{z}$ as in Q_w. The last term of (10), however, is nonlinear in $\boldsymbol{z}$. The decomposition of the Q-function implies that the estimates of $\boldsymbol{w}_h$ and $\boldsymbol{v}_i$ can be updated separately by maximizing Q_w and Q_v, respectively. On differentiation of Q_w with respect to $\boldsymbol{w}_h$ for $h = 1, \ldots, m$, it follows that $\boldsymbol{w}_h^{(k+1)}$ satisfies

$$\sum_{j=1}^{n} [E_{\boldsymbol{\Psi}^{(k)}}(Z_{hj} | \boldsymbol{y}, \boldsymbol{x}) - u_{hj}] \boldsymbol{x}_j = \boldsymbol{0} \quad (h = 1, \ldots, m) \quad (12)$$

where

$$E_{\boldsymbol{\Psi}^{(k)}}(Z_{hj} | \boldsymbol{y}, \boldsymbol{x}) = \frac{\sum_{\boldsymbol{z}_j : z_{hj} = 1} \mathrm{pr}_{\boldsymbol{\Psi}^{(k)}}(\boldsymbol{x}_j, \boldsymbol{y}_j, \boldsymbol{z}_j)}{\sum_{\boldsymbol{z}_j} \mathrm{pr}_{\boldsymbol{\Psi}^{(k)}}(\boldsymbol{x}_j, \boldsymbol{y}_j, \boldsymbol{z}_j)} \quad (13)$$

and where

$$\mathrm{pr}_{\boldsymbol{\Psi}^{(k)}}(\boldsymbol{x}_j, \boldsymbol{y}_j, \boldsymbol{z}_j) = \prod_{h=1}^{m} u_{hj}^{z_{hj}} (1 - u_{hj})^{(1 - z_{hj})} \prod_{i=1}^{g} o_{ij}^{y_{ij}}. \quad (14)$$

On differentiation of Q_v with respect to $\boldsymbol{v}_i$ for $i = 1, \ldots, g-1$, it follows that $\boldsymbol{v}_i^{(k+1)}$ satisfies

$$\sum_{j=1}^{n} \left[y_{ij} E_{\boldsymbol{\Psi}^{(k)}}(Z_{hj} | \boldsymbol{y}, \boldsymbol{x}) - \frac{\sum_{\boldsymbol{z}_j : z_{hj} = 1} o_{ij} \mathrm{pr}_{\boldsymbol{\Psi}^{(k)}}(\boldsymbol{x}_j, \boldsymbol{y}_j, \boldsymbol{z}_j)}{\sum_{\boldsymbol{z}_j} \mathrm{pr}_{\boldsymbol{\Psi}^{(k)}}(\boldsymbol{x}_j, \boldsymbol{y}_j, \boldsymbol{z}_j)} \right] = \boldsymbol{0}. \quad (15)$$

A. Intractability of the Exact E-Step for Training MLPs

It can be seen from (13) and (15) that each E-step of the EM algorithm involves summation over $\boldsymbol{z}_j$ $(j = 1, \ldots, n)$. There are 2^m tuples $(z_{1j}, \ldots, z_{mj})$ with $z_{hj} = 0$ or 1 $(h = 1, \ldots, m)$. Hence, the computational complexity grows exponentially with m. The EM algorithm may provide an efficient training algorithm if the number of hidden units m is small. For example, Lai and Wong [22] adopted a similar E-step procedure in fitting neural networks to time series data. When m is large, say $m > 10$, Monte Carlo (MC) approach may be used to implement the E-step [14, Ch. 6]. This MCEM algorithm iterates between a MC estimate of the conditional expectation of the complete-data log likelihood (MC E-step) and a maximization of this expectation over the relevant parameters (M-step) [23]. In the implementation of the MCEM algorithm, the MC error incurred in obtaining the MC sample in the E-step is monitored so as to adjust the number of MC replications when the algorithm progresses [24], [25].

An alternative variational relaxation approach derived from a mean field approximation has been proposed to circumvent the difficulty of the intractable E-step in training MLP [26]. The basic idea involves approximating the intractable posterior distribution $\mathrm{pr}(Z|Y, X)$ with a family of factorial distributions, in which the troublesome coupling in the exact EM algorithm in (13) and (14) are approximated with a factorization assumption for each value of the observed data. A best approximation in the family that is "closest" to the posterior distribution is obtained by minimizing the Kullback-Leibler (KL) divergence between them. By replacing the E-step with an approximate E-step, this variational EM algorithm guarantees a global convergence on a lower bound on the log likelihood [26].

B. An Integration of the Methodology Related to EM Training of RBF Networks

For RBF neural networks, the EM algorithm has been used for the unsupervised or supervised modes of the training process [16], [27]. In the training of RBF networks, the hidden variable z_{hj} corresponding to the input value $\boldsymbol{x}_j$ in a RBF neural network has the form

$$z_{hj} = \phi(\boldsymbol{x}_j; \boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) \quad (h = 1, \ldots, m)$$

where $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m$ denote the radial basis centres and $\boldsymbol{\Sigma}_h$ is a covariance matrix, usually taken to be spherical (that is, $\boldsymbol{\Sigma}_h = \sigma_h^2 I_p$, where I_p is the $p \times p$ identity matrix). The function $\phi(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the p-variate density with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

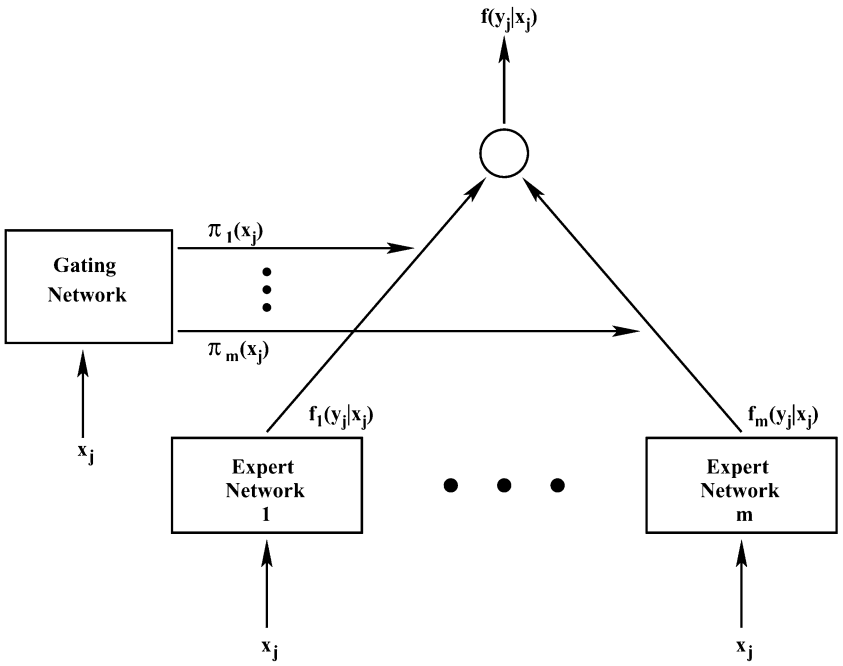


Fig. 2. Mixture of experts.

Typically, with RBF neural networks, these centres are found by using a clustering algorithm such as k-means in an unsupervised mode; that is, the clustering algorithm is applied to just the input values $\boldsymbol{x}_j$, ignoring their known classification labels. However, attention has been given to using a normal mixture model to find suitable values for the $\boldsymbol{\mu}_h$ and the σ_h or the $\boldsymbol{\Sigma}_h$ before the second stage of finding the weights by conventional neural network training procedures. Detailed description of unsupervised learning of normal mixtures model via the EM algorithm can be obtained in [28, Ch. 3]. Thus, in contrast to the training of MLP networks where a boundary between different groups of instances is sought, training of RBF networks forms clusters in the data space with a centre $\boldsymbol{\mu}_h$ for each cluster $(h = 1, \ldots, m)$. These clusters are then used to classify different groups of data instances. This implies that the training of RBF networks can be substantially faster than the methods used for MLP networks by separately training the basis functions by some unsupervised method and the weights by some linear optimum approaches. On the other hand, Ungar *et al.* [29] introduced a statistical interpretation of RBF's as a normal mixture and showed how the EM algorithm could be used to estimate the centres and covariance matrices of the basis functions and the weights simultaneously. They called this method to train fully RBF neural networks the EMRBF.

IV. ECM Algorithm for Training Mixture of Experts

In Section III, we have identified some situations where the application of the EM algorithm for training MLP networks may be of limited value due to complications in performing the E-step. In this section, we show that the E-step for training ME neural networks is easy to implement. In ME neural networks (Fig. 2), there are several modules, referred to as expert networks. These expert networks approximate the distribution of $\boldsymbol{y}_j$ within each region of the input space. The expert network maps its input $\boldsymbol{x}_j$ to an output, the density $f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)$, where $\boldsymbol{\theta}_h$ is a vector of unknown parameters for the hth expert network. It is assumed that different experts are appropriate in different regions of the input space. The gating network provides a set of scalar coefficients $\pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha})$ that weight the contributions of the various experts, where $\boldsymbol{\alpha}$ is a vector of unknown parameters in the gating network. Therefore, the final output of the ME neural network is a weighted sum of all the output vectors produced by expert networks

$$f(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\Psi}) = \sum_{h=1}^{m} \pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h) \tag{16}$$

where $\boldsymbol{\Psi}$ is the vector of all the unknown parameters. For multiclass classification, the local output density $f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)$ is modeled by a multinomial distribution consisting of one draw on g categories.

To apply the EM algorithm to the ME networks, we introduce the indicator variables z_{hj}, where z_{hj} is one or zero according to whether $\boldsymbol{y}_j$ belongs or does not delong to the hth expert [28, Sec. 5.13]. That is, we let the missing data $\boldsymbol{z}$ be the vector containing all these indicator variables. The probability that Z_{hj} is one, given the input $\boldsymbol{x}_j$, is

$$\pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) = \mathrm{pr}\{Z_{hj} = 1|\boldsymbol{x}_j\}.$$

The complete-data log likelihood for $\boldsymbol{\Psi}$, $\log L_c(\boldsymbol{\Psi};\boldsymbol{y},\boldsymbol{z},\boldsymbol{x})$, is given by

$$\sum_{j=1}^{n}\sum_{h=1}^{m} z_{hj}\{\log \pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) + \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)\}. \tag{17}$$

It follows on application of the EM algorithm in training ME networks that on the $(k+1)$th iteration, the E-step calculates the Q-function as

$$\begin{aligned} Q(\Psi;\Psi^{(k)}) &= E_{\Psi^{(k)}}\{\log L_c(\Psi;\boldsymbol{y},\boldsymbol{z},\boldsymbol{x})|\boldsymbol{y},\boldsymbol{x}\} \\ &= \sum_{j=1}^{n}\sum_{h=1}^{m} E_{\Psi^{(k)}}(Z_{hj}|\boldsymbol{y},\boldsymbol{x}) \\ &\quad \times \left\{\log \pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) + \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)\right\} \\ &= Q_\alpha + Q_\theta. \end{aligned} \tag{18}$$

It can be seen that the complete-data log likelihood (17) is linear in $\boldsymbol{z}$. Thus, the E-step just replaces z_{hj} in (17) by its current conditional expectation $\tau_{hj}^{(k)}$ given $\boldsymbol{y}_j$, $\boldsymbol{x}_j$, and the current estimate $\Psi^{(k)}$ for Ψ, where

$$\begin{aligned} \tau_{hj}^{(k)} &= \mathrm{pr}_{\Psi^{(k)}}\{Z_{hj}=1|\boldsymbol{y}_j,\boldsymbol{x}_j\} \\ &= \frac{\pi_h\left(\boldsymbol{x}_j;\boldsymbol{\alpha}^{(k)}\right) f_h\left(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h^{(k)}\right)}{\sum_{r=1}^{m}\pi_r\left(\boldsymbol{x}_j;\boldsymbol{\alpha}^{(k)}\right) f_r\left(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_r^{(k)}\right)} \end{aligned}$$

for $h=1,\ldots,m$. In addition, the Q-function can be decomposed into two terms with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}_h$ $(h=1,\ldots,m)$, respectively.

Hence, the M-step consists of two separate maximization problems. The updated estimate of $\boldsymbol{\alpha}^{(k+1)}$ is obtained by solving

$$\sum_{j=1}^{n}\sum_{h=1}^{m}\tau_{hj}^{(k)}\frac{\partial \log \pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \boldsymbol{0}. \tag{19}$$

The updated estimate of $\boldsymbol{\theta}_h^{(k+1)}$ is obtained by solving

$$\sum_{j=1}^{n}\tau_{hj}^{(k)}\frac{\partial \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_h} = \boldsymbol{0} \tag{20}$$

for each h $(h=1,\ldots,m)$. Both (19) and (20) require iterative methods. Jordon and Jacobs [12] proposed an iterative reweighted least squares (IRLS) algorithm for all the generalized linear models (GLM) [30] used in the ME networks.

The output of the gating network is usually modeled by the multinomial logit (or softmax) function as

$$\pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) = \frac{\exp\left(\boldsymbol{v}_h^T\boldsymbol{x}_j\right)}{1+\sum_{l=1}^{m-1}\exp\left(\boldsymbol{v}_l^T\boldsymbol{x}_j\right)} \tag{21}$$

where $\pi_m(\boldsymbol{x}_j;\boldsymbol{\alpha}) = 1/\left(1+\sum_{l=1}^{m-1}\exp\left(\boldsymbol{v}_l^T\boldsymbol{x}_j\right)\right)$ for $h=1,\ldots,m-1$. Here $\boldsymbol{\alpha}$ contains the elements in $\boldsymbol{v}_h$ $(h=1,\ldots,m-1)$. Equation (19) becomes

$$\sum_{j=1}^{n}\left(\tau_{hj}^{(k)} - \frac{\exp(\boldsymbol{v}_h^T\boldsymbol{x}_j)}{1+\sum_{l=1}^{m-1}\exp(\boldsymbol{v}_l^T\boldsymbol{x}_j)}\right)\boldsymbol{x}_j = 0 \tag{22}$$

for $h=1,\ldots,m-1$, which is a set of nonlinear equations with $(m-1)p$ unknown parameters.

For multiclass classification problems, the hth expert is taken to be the multinomial consisting of one draw on g categories. The local output of the hth expert $(h=1,\ldots,m)$ is thus modeled as

$$\begin{aligned} f_h(\boldsymbol{y}_j|\boldsymbol{x}_j,\boldsymbol{\theta}_h) = \prod_{i=1}^{g-1}&\left(\frac{\exp(\boldsymbol{w}_{hi}^T\boldsymbol{x}_j)}{1+\sum_{r=1}^{g-1}\exp(\boldsymbol{w}_{hr}^T\boldsymbol{x}_j)}\right)^{y_{ij}} \\ &\cdot\left(\frac{1}{1+\sum_{r=1}^{g-1}\exp(\boldsymbol{w}_{hr}^T\boldsymbol{x}_j)}\right)^{y_{gj}} \end{aligned}$$

where $\boldsymbol{\theta}_h$ contains the elements in $\boldsymbol{w}_{hi}$ $(i=1,\ldots,g-1)$. Equation (20) becomes

$$\sum_{j=1}^{n}\tau_{hj}^{(k)}\left(y_{ij} - \frac{\exp(\boldsymbol{w}_{hi}^T\boldsymbol{x}_j)}{1+\sum_{r=1}^{g-1}\exp(\boldsymbol{w}_{hr}^T\boldsymbol{x}_j)}\right)\boldsymbol{x}_j = \boldsymbol{0} \tag{23}$$

for $h=1,\ldots,m$ and $i=1,\ldots,g-1$, which are m sets of nonlinear equations each with $(g-1)p$ unknown parameters.

It can be seen from (22) that the nonlinear equation for the hth expert depends not only on the parameter vector $\boldsymbol{v}_h$, but also on other parameter vectors $\boldsymbol{v}_l$ $(l=1,\ldots,m-1)$. In other words, each parameter vector $\boldsymbol{v}_h$ cannot be updated independently. With the IRLS algorithm presented in [12], the independence assumption on these parameter vectors was used implicitly and each parameter vector was updated independently and in parallel as

$$\boldsymbol{v}_h^{(s+1)} = \boldsymbol{v}_h^{(s)} + \gamma_\alpha\left(\frac{\partial^2 Q_\alpha}{\partial \boldsymbol{v}_h\boldsymbol{v}_h^T}\right)^{-1}\frac{\partial Q_\alpha}{\partial \boldsymbol{v}_h} \tag{24}$$

for $h=1,\ldots,m-1$, where $\gamma_\alpha \le 1$ is the learning rate [11]. That is, there are $m-1$ sets of nonlinear equations each with p variables instead of a set of nonlinear equations with $(m-1)p$ variables. In [12], the iteration (24) is referred to as the inner loop of the EM algorithm. This inner loop is terminated when the algorithm has converged or the algorithm has still not converged after some prespecified number of iterations.

Similarly, each parameter vector $\boldsymbol{w}_{hi}$ for $h=1,\ldots,m$ was updated independently as

$$\boldsymbol{w}_{hi}^{(s+1)} = \boldsymbol{w}_{hi}^{(s)} + \gamma_\theta\left(\frac{\partial^2 Q_\theta}{\partial \boldsymbol{w}_{hi}\boldsymbol{w}_{hi}^T}\right)^{-1}\frac{\partial Q_\theta}{\partial \boldsymbol{w}_{hi}} \tag{25}$$

for $i=1,\ldots,g-1$, where $\gamma_\theta \le 1$ is the learning rate for $\boldsymbol{w}_{hi}$. In the simulation experiment (Section IV) and real example illustration (Section V) to follow, we set $\gamma_\alpha = 1$ and $\gamma_\theta = 0.1$. Here, a smaller learning rate is adopted for $\boldsymbol{w}_{hi}$ to ensure better convergence as y_{ij} in (23) is binary zero or one; see the discussion in [1, Sec.8].

With reference to (24) and (25), the independence assumption on the parameter vectors is equivalent to the adoption of an incomplete Hessian matrix of the Q-function. Chen *et al.*[15]

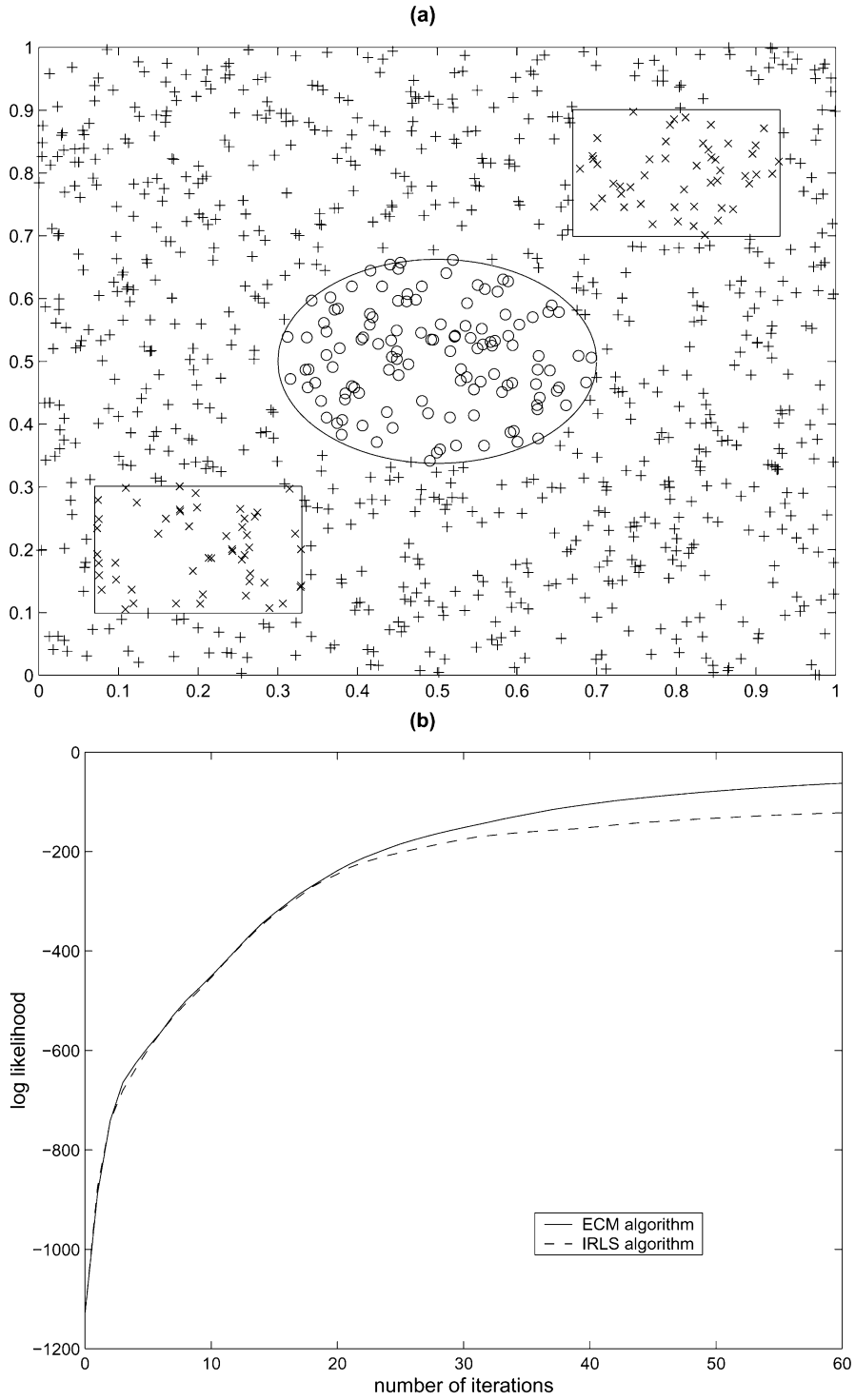


Fig. 3. Simulation experiment (a) the training set: x, o, and $+$ stand for samples belonging to G_1, G_2, and G_3, respectively. (b) Log likelihood against the number of iterations.

proposed a learning algorithm based on the Newton-Raphson method for use in the inner loop of the EM algorithm. In particular, they pointed out that the parameter vectors cannot be updated separately due to the incorrect independence assumption. Rather, they adopted the exact Hessian matrix in the inner loop of the EM algorithm. However, the use of the exact Hessian matrix results in expensive computation during learning. To this end, they proposed a modified algorithm whereby an approximate statistical model called the generalized Bernoulli density is introduced for expert networks in multiclass classification. This approximation simplifies the Newton–Raphson algorithm for multiclass classification in that all off-diagonal block matrices in the Hessian matrix are zero matrices and so the parameter vectors $\boldsymbol{w}_{hi}$ $(i = 1, \ldots, g-1)$ are separable. With this approximation, the learning time is decreased, but the error rate is reported to be increased [15].

In this paper, we propose an ECM algorithm for which both parameter vectors $\boldsymbol{v}_h$ and $\boldsymbol{w}_{hi}$ are separable for $h = 1, \ldots, m-1$ and $i = 1, \ldots, g-1$, respectively. That is, the parameters in the expert and gating networks can be estimated separately. With the ECM algorithm of Meng and Rubin [31], the M-step is replaced by several computationally simpler conditional-maximization (CM) steps. For example, the parameter vector $\boldsymbol{\alpha}$ is partitioned as $(\boldsymbol{v}_1^T, \ldots, \boldsymbol{v}_{m-1}^T)^T$. On the $(k+1)$th iteration of the ECM algorithm, the E-step is the same as given above for the EM algorithm, but the M-step of the latter is replaced by $(m-1)$ CM-steps, as follows:

- **CM-step 1**: Calculate $\boldsymbol{v}_1^{(k+1)}$ by maximizing Q_α with $\boldsymbol{v}_l$ $(l = 2, \ldots, m-1)$ fixed at $\boldsymbol{v}_l^{(k)}$.
- **CM-step 2**: Calculate $\boldsymbol{v}_2^{(k+1)}$ by maximizing Q_α with $\boldsymbol{v}_1$ fixed at $\boldsymbol{v}_1^{(k+1)}$ and $\boldsymbol{v}_l$ $(l = 3, \ldots, m-1)$ fixed at $\boldsymbol{v}_l^{(k)}$.
- $\vdots$
- **CM-step** $(\boldsymbol{m-1})$: Calculate $\boldsymbol{v}_{(m-1)}^{(k+1)}$ by maximizing Q_α with $\boldsymbol{v}_l$ $(l = 1, \ldots, m-2)$ fixed at $\boldsymbol{v}_l^{(k+1)}$.

As the CM maximizations are over smaller dimensional parameter space, they are often simpler and more stable than the corresponding full maximization called for on the M-step of the EM algorithm, especially when iteration is required. More importantly, each CM-step above corresponds to a separable set of the parameters in $\boldsymbol{v}_h$ for $h = 1, \ldots, m-1$, and can be obtained using the IRLS approach. Moreover, the ECM algorithm preserves the appealing convergence properties of the EM algorithm, such as its monotone increasing of likelihood after each iteration [14, Ch. 5], [32]. To see this, it is noted that each of the above CM-steps maximizes the Q_α function found in the preceding E-step subject to constraints on $\boldsymbol{\alpha}$. Thus, we have

$$\begin{aligned} Q_\alpha^{k+1} &\geq Q_\alpha^{k+(m-2)/(m-1)} \geq \cdots \\ &\geq Q_\alpha^{k+l/(m-1)} \geq \cdots \geq Q_\alpha^{k} \end{aligned} \tag{26}$$

where $Q_\alpha^{k+l/(m-1)}$ is the Q_α function after the lth CM-step on the $(k+1)$th iteration $(l = 0, \ldots, m-1)$. The inequality (26) is a sufficient condition for

$$\log L(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{y}, \boldsymbol{x}) \geq \log L(\boldsymbol{\Psi}^{(k)}; \boldsymbol{y}, \boldsymbol{x})$$

to hold. That is, the ECM algorithm monotonically increases the log likelihood after each CM-step, and hence, after each iteration. This desirable property of reliable convergence ensures a better result for multiclass classification problems.

V. Simulation Experiment

Here, we report the results of a simulation experiment performed to compare the relative performance of the IRLS algorithm and the ECM algorithm for training the ME networks. Our simulation experiment is similar to that described in [15], where the IRLS algorithm and the Newton–Raphson algorithm with exact Hessian matrix were compared. In their study, they set both γ_α and γ_θ to be one for the IRLS algorithm and reported that the log likelihood obtained by the IRLS algorithm was oscillatory and unstable and that the error rate was higher.

TABLE I
Simulation Results for the Three-Group Data

	IRLS Algorithm	ECM Algorithm
Training Set		
No. correctly classified	907	937
No. misclassified	43	13
Test Set		
No. correctly classified	2352	2414
No. misclassified	148	86
Log likelihood	-122.0	-62.4

TABLE II
Classification Results for the *Iris* Data

	IRLS Algorithm		ECM Algorithm
	$(\gamma_\theta = 1)$	$(\gamma_\theta = 0.1)$	
Training Set			
No. correctly classified	141	147	148
No. misclassified	9	3	2
Log likelihood	-20.1	-14.6	-7.1

In the simulated data set, there are three $(g = 3)$ groups. As illustrated in Fig. 3(a), two small rectangles, an ellipse, and other regions in the large square constitute the three groups denoted G_1, G_2, and G_3, respectively. A training set of $(n = 950)$ points are produced by a uniformly distributed random number producer. Among these points, 100, 122, and 728 points belong to G_i $(i = 1, 2, 3)$, respectively. As in [15], the ME networks consists of $m = 12$ experts. For comparative purpose, the learning rates are set to be $\gamma_\alpha = 1$ and $\gamma_\theta = 0.1$ for both IRLS and ECM algorithms. We also run both algorithms for 60 iterations, where each iteration was composed of a complete E-step and M-step of the EM algorithm. For evaluating the generalization capability, a test set of 2500 points uniformly distributed in the large square was generated. Table I shows the classification results on both the training and test sets. In addition, we plot in Fig. 3(b) the log likelihood against the number of iterations.

From Fig. 3(b), it can be seen that with both algorithms the log likelihood is increased monotonically after each iteration. The unstable behavior of the IRLS algorithm described in [15] did not occur in our simulation experiment (compared to [15, Fig. 2(b)]) because a learning rate of $\gamma_\theta = 0.1$ is adopted in our study. From Table I, it can be seen that the ECM algorithm outperforms the IRLS algorithm in this simulated experiment, in terms of the misclassified rate for both the training and test sets. Moreover, the ECM algorithm converges to a larger log likelihood value compared to that using the IRLS algorithm [Fig. 3(b)].

VI. Real Examples

Here, we illustrate the ECM algorithm using three real data sets: the *Leptograpsus* crab data of Campbell and Mahon [33], the well-known set of *Iris* data firstly analyzed by Fisher [34],

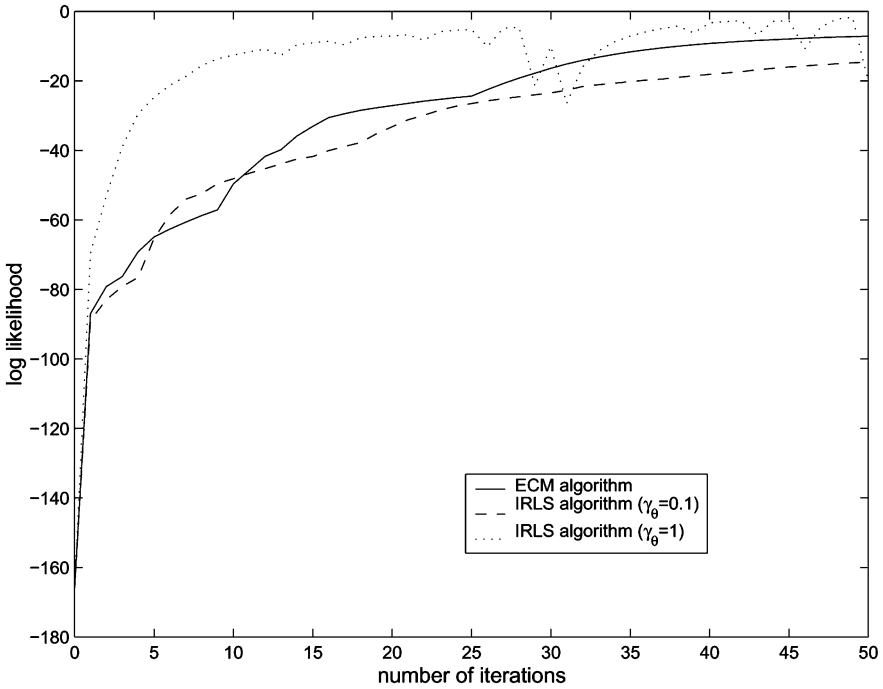


Fig. 4. Log likelihood against the number of iterations (the *Iris* data).

and a thyroid data "ann-thyorid.dat." The last two data sets are available from the UCI Repository of machine learning databases [35]. With the *Iris* data, we compare the ECM algorithm to the IRLS algorithm with learning rates of $\gamma_\theta = 1$ and $\gamma_\theta = 0.1$. The data consist of measurements of the length and width of both sepals and petals of 50 plants for each of three types of *Iris* species *setosa*, *versicolor*, and *virginica*. Here, the ME networks consists of $m = 3$ experts and we run the algorithms for 50 iterations. Table II shows the classification results for the IRLS and the ECM algorithms. It can be seen that the performance of the IRLS algorithm improves when a learning rate smaller than one is adopted. This result indicates that the comparison between the Newton-Raphson-type algorithms and the IRLS algorithm performed in [15] was unfair and misleading as the learning rate $\gamma_\theta = 1$ was adopted for the IRLS algorithm but a smaller value was adopted for the Newton–Raphson algorithms.

In Fig. 4, we plot the log likelihood against the number of iterations. It can be seen that the log likelihood obtained by the IRLS algorithm with $\gamma_\theta = 1$ is unstable, but that obtained by the ECM algorithm and the IRLS algorithm with $\gamma_\theta = 0.1$ are both increased monotonically after each iteration.

With the *Leptograpsus* crab data, one species has been split into two new species, previously grouped by color form, orange and blue. Data are available on 50 specimens of each sex of each species. Each specimen has measurements on the width of the frontal lip, the rear width, length along the midline and the maximum width of the carapace and the body depth. Ripley [3] had studied this data using various approaches. Here, we apply the IRLS algorithm and the ECM algorithm. As in [3], the 200 examples were divided as 20 in each class for a training set and the remaining 120 as a test set. The ME networks consists of $m = 2$ experts. For comparative purposes, the learning rates are set to be $\gamma_\alpha = 1$ and $\gamma_\theta = 0.1$ for both IRLS and ECM algorithms. We run both algorithms for 50 iterations. Table III shows the classification results on both the training and test sets.

TABLE III
CLASSIFICATION RESULTS FOR THE *LEPTOGRAPSUS* CRAB DATA

	IRLS Algorithm	ECM Algorithm
Training Set		
No. correctly classified	77	78
No. misclassified	3	2
Test Set		
No. correctly classified	110	113
No. misclassified	10	7
Log likelihood	-8.9	-8.0

TABLE IV
CLASSIFICATION RESULTS FOR THE THYROID DATA

	IRLS Algorithm	ECM Algorithm
Training Set		
No. correctly classified	3702	3730
No. misclassified	70	42
Test Set		
No. correctly classified	3327	3347
No. misclassified	101	81
Log likelihood	-462.4	-311.1

The thyroid data are measurements of the thyroid gland. Each measurement consists of 21 variables — 15 binary and

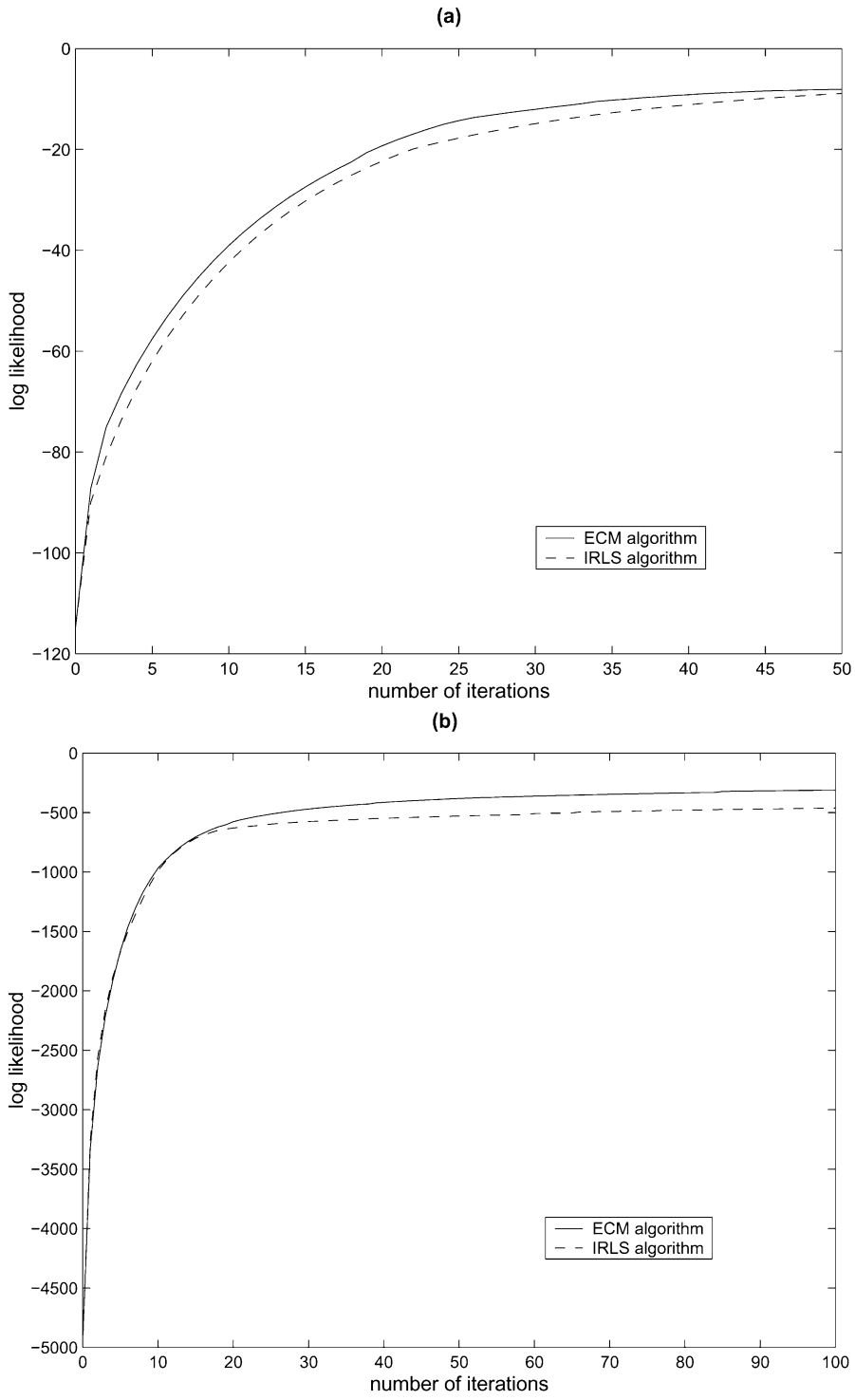


Fig. 5. Log likelihood against the number of iterations (a) the *Leptograpsus* crab data. (b) The thyroid data.

6 continuous variables. Three classes are assigned to each of the measurement which correspond to the hyper-, hypo-, and normal function of the thyroid gland. The training set consists of 3772 measurements and again 3428 measurements are available for the testing set. In the analysis, the six continuous variables are each standardized to have zero mean and unit variance. The ME networks consists of $m = 8$ experts. The learning rates are set to be $\gamma_\alpha = 1$ and $\gamma_\theta = 0.1$ for both IRLS and ECM algorithms. We run both algorithms for 100 iterations. Table IV shows the classification results on both the training and test sets.

From Tables III and IV, it can be seen that the ECM algorithm outperforms the IRLS algorithm in both real data sets, in terms of the misclassified rate for both the training and test sets. In Fig. 5, the log likelihood is plotted against the number of iterations for these two data sets. It can be seen that the ECM algo-

rithm converges to a larger local maximum of the log likelihood compared to that using the IRLS algorithm.

VII. Conclusion

The EM algorithm has been of much interest in recent years in its application to the training of neural networks. In this paper, we have clarified some misunderstandings that have arisen in training neural networks. This is very important for further development of algorithms and their applications to neural networks. We have demonstrated in Sections III and IV how EM-based algorithms can be adopted to train MLP and ME neural networks, respectively, for multiclass classification problems. In particular, we showed that the application of EM algorithm to train MLP networks may be of limited value due to complications in performing the E-step when the number of hidden units is large. A computationally intensive Monte Carlo E-step may be adopted in this situation.

For training ME networks with applications to multiclass classification problems, we found that the convergence of the IRLS algorithm [12] was stable and that the log likelihood increased monotonically when a learning rate $\gamma_\theta < 1$ was adopted. This result is somewhat different from that presented in [15] where the IRLS algorithm showed unstable behavior when the learning rate $\gamma_\theta = 1$ was used. Thus, the result presented in this paper indicates that the convergence of the IRLS algorithm can be stable even though the independence assumption on parameter vectors is invalid for multiclass classification problems [15].

In Section IV, we described how the ECM algorithm can be adopted to train ME networks such that both the parameter vectors $\boldsymbol{v}_h$ and $\boldsymbol{w}_{hi}$ are separable for $h = 1, \ldots, m$ and $i = 1, \ldots, g-1$, respectively. The conditional maximizations on the M-step thus involve a smaller dimensional parameter space, which implies that the proposed ECM algorithm is in general more stable. These appealing properties are demonstrated by the simulation experiemnt (Section V) and the real examples (Section VI). Although we have concentrated on how the ECM algorithm can be used to train ME networks, the ECM algorithm can also be applied in general to train the hierarchical mixtures of experts (HME) network model of Jordan and Jacobs [12].

Acknowledgment

The authors wish to thank the Associate Editor and the reviewers for helpful comments on the paper.

References

[1] S. Amari, "Information geometry of the EM algorithm and em algorithms for neural networks," *Neural Networks* , vol. 8, pp. 1379–1408, 1995.

[2] B. Cheng and D. M. Titterington, "Neural networks: A review from a statistical perspective (with discussion)," *Statist. Sci.*, vol. 9, pp. 2–54, 1994.

[3] B. D. Ripley, "Statistical aspects of neural networks," in *Networks and Chaos—Statistical and Probabilistic Aspects*, O.E. Barndorff-Nielsen, J. L. Jensen, and W. S. Kendall, Eds. London, U.K.: Chapman Hall, 1993, pp. 40–123.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm (with discussion)," *J. Roy. Statist. Soc. B*, vol. 39, pp. 1–38, 1977.

[5] M. I. B. Chai, A. Chai, and P. Sullivan, "Boundary detection of retinoblastoma tumors with neural networks," *Comput. Med. Imaging Graph.*, vol. 25, pp. 257–264, 2001.

[6] S. Ma and C. Ji, "Fast training of recurrent networks based on the EM algorithm," *IEEE Trans. Neural Networks*, vol. 9, pp. 11–26, 1998.

[7] V. Tresp, T. Briegel, and J. Moody, "Neural-network models for the blood glucose metabolism of a diabetic," *IEEE Trans. Neural Networks*, vol. 10, pp. 1204–1213, Sept. 1999.

[8] Y. Yamamoto and P. N. Nikiforuk, "A new supervised learning algorithm for multilayered and interconnected neural networks," *IEEE Trans. Neural Networks*, vol. 11, pp. 36–46, Jan. 2000.

[9] S. Amari, "Information geometry of neural networks—An overview," in *Mathematics of Neural Networks Models, Algorithms, and Applications*, S. W. Ellacott, J. C. Mason, and I. J. Anderson, Eds. Boston, MA: Kluwer, 1997, pp. 15–23.

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, pp. 79–87, 1991.

[11] M. I. Jordan and L. Xu, "Convergence results for the EM approach to mixtures of experts architectures," *Neural Networks*, vol. 8, pp. 1409–1431, 1995.

[12] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Comput.*, vol. 6, pp. 181–214, 1994.

[13] R. A. Jacobs, F. Peng, and M. A. Tanner, "A bayesian approach to model selection in hierarchical mixtures-of-experts architectures," *Neural Networks*, vol. 10, pp. 231–241, 1997.

[14] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. New York: Wiley, 1997.

[15] K. Chen, L. Xu, and H. Chi, "Improved learning algorithms for mixture of experts in multiclass classification," *Neural Networks*, vol. 12, pp. 1229–1252, 1999.

[16] R. Langari, L. Wang, and J. Yen, "Radial basis function networks, regression weights, and the expectation-maximization algorithm," *IEEE Trans. Syst., Man, Cybern. A*, vol. 27, pp. 613–623, 1997.

[17] L. Wang and R. Langari, "Sugeno model, fuzzy discretization, and the EM algorithm," *Fuzzy Sets Syst.*, vol. 82, pp. 279–288, 1996.

[18] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neuro-Computing: Algorithms, Architectures and Applications*, F. F. Soulié and J. Hérault, Eds. Berlin, Germany: Springer, 1990, pp. 227–236.

[19] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, MA: Cambridge Univ. Press, 1996, ch. 5.

[20] S. Ma, C. Ji, and J. Farmer, "An efficient EM-based training algorithm for feedforward neural networks," *Neural Networks*, vol. 10, pp. 243–256, 1997.

[21] M. Aitkin and R. Foxall, "Statistical modeling of artificial neural networks using the multi-layer perceptron," *Statist. Comput.*, vol. 13, pp. 227–239, 2003.

[22] T. L. Lai and S. P. S. Wong, "Stochastic neural networks with applications to nonlinear time series," *J. Amer. Statist. Assoc.*, vol. 96, pp. 968–981, 2001.

[23] G. C. G. Wei and M. A. Tanner, "A monte carlo implementation of the EM algorithm and the poor man's data augmentation algorithm," *J. Amer. Statist. Assoc.*, vol. 85, pp. 699–704, 1990.

[24] R. A. Levine and G. Casella, "Implementations of the monte carlo EM algorithm," *J. Comput. Graph. Statist.*, vol. 10, pp. 422–439, 2001.

[25] M. A. Tanner, *Tools for Statistical Inference*, 3 ed. New York: Springer-Verlag, 1996, sec. 4.5.

[26] L. K. Saul and M. I. Jordan, "Attractor dynamics in feedforward neural networks," *Neural Comput.*, vol. 12, pp. 1313–1335, 2000.

[27] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood training of probabilistic neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 764–783, Sept. 1994.

[28] G. J. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.

[29] L. H. Ungar, T. Johnson, and R. D. De Veaux, "Radial basis function neural networks for process control," in *CIMPRO Proceedings*, 1994, http://www.cis.upenn.edu/~ungar/papers/cimpro.ps, pp. 357–364.

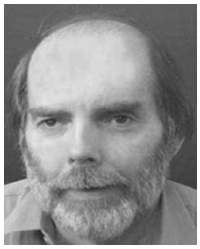
[30] P. A. McCullagh and J. Nelder, *Generalized Linear Models*, 2 ed. London, U.K.: Chapman Hall, 1989.

[31] X. L. Meng and D. B. Rubin, "Maximum likelihood estimation via the ECM algorithm: A general framework," *Biometrika*, vol. 80, pp. 267–278, 1993.

[32] X. L. Meng, "On the rate of convergence of the ECM algorithm," *Ann. Statist.*, vol. 22, pp. 326–339, 1994.
[33] N. A. Campbell and R. J. Mahon, "A multivariate study of variation in two species of rock crab of genus leptograpsus," *Austral. J. Zool.*, vol. 22, pp. 417–425, 1974.
[34] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, pp. 179–188, 1936.
[35] C. L. Blake and C. J. Merz. (1998) UCI Repository of Machine Learning Databases. Univ. Calif., Dept. Inform. Comput. Sci., Irvine, CA. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

Shu-Kay Ng received the B.Sc. degree in civil engineering from the University of Hong Kong, Hong Kong, in 1986, and the Ph.D. degree in statistics from the Department of Mathematics, University of Queensland, Brisbane, Australia, in 1999.

He is currently an Australian Research Council (ARC) Postdoctoral Research Fellow with the University of Queensland. His research interests include machine learning, neural networks, pattern recognition, statistical inference, and survival analysis.



Geoffrey John McLachlan received the B.Sc. (Hons.), Ph.D., and D.Sc. (on the basis of his publications in the scientific literature) degrees, from the University of Queensland, Brisbane, Australia, in 1969, 1973, and 1994, respectively,

In 1974, he was appointed to a Lectureship at the University of New England. Since 1975, he has been with the Department of Mathematics, University of Queensland, where he was promoted to a Readership in 1984, and subsequently given a personal chair. He has published more than 145 research articles, including four monographs. The last three monographs, which are volumes in the *Wiley Series in Probability and Statistics*, are on the topics of discriminant analysis, the EM algorithm, and finite mixture models. His research interests have been concentrated in the related fields of classification, cluster and discriminant analyzes, image analysis, machine learning, neural networks, pattern recognition, and data mining, and in the field of statistical inference. The focus in the latter field has been on the theory and applications of finite mixture models and on estimation via the EM algorithm. More recently, he has become actively involved in the field of bioinformatics with the focus on the statistical analysis of microarray gene-expression data.

Dr. McLachlan is a Fellow of the American Statistical Association, the Royal Statistical Society, and the Australian Mathematical Society. He is on the editorial board of several international journals and has served on the program committee for many international conferences.