Self-Organizing Mixture Networks for Probability Density Estimation

Hujun Yin and Nigel M. Allinson

***Abstract*—A self-organizing mixture network (SOMN) is derived for learning arbitrary density functions. The network minimizes the Kullback–Leibler information metric by means of stochastic approximation methods. The density functions are modeled as mixtures of parametric distributions A mixture needs not to be homogenous, i.e., it can have different density profiles. The first layer of the network is similar to Kohonen's self-organizing map (SOM), but with the parameters of the component densities as the learning weights. The winning mechanism is based on maximum posterior probability, and updating of the weights is limited to a small neighborhood around the winner. The second layer accumulates the responses of these local nodes, weighted by the learned mixing parameters. The network possesses a simple structure and computational form, yet yields fast and robust convergence. The network has a generalization ability due to the relative entropy criterion used. Applications to density profile estimation and pattern classification are presented. The SOMN can also provide an insight to the role of neighborhood function used in the SOM.**

***Index Terms*—Density estimation, expectation-maximization (EM) algorithm, maximum likelihood, mixture distribution, self-organizing maps, unsupervised learning.**

I. Introduction

IN PRACTICAL pattern recognition, a sample distribution can rarely be represented by a single parametric form. When there is little or no prior knowledge about data properties except for the data samples themselves, the sample's joint distribution can often be modeled as a mixture of simple parametric forms such as Gaussian, Cauchy, or Laplace [1], [2]. This provides a tradeoff between the simple and limited *parametric* approaches and the computationally intensive *nonparametric* approaches. The parameters for each component density in the mixture, however, have to be derived or learned solely from the data samples. The maximum likelihood (ML) approach to this problem, when all parameters are unknown, results in a set of implicit equations, which require numerical methods to solve [3].

Xu and Jordan [4] applied the expectation-maximization (EM) method [5] to the Gaussian mixture problem and showed its advantages over other algorithms. The resulting algorithm agrees with Duda and Hart's earlier suggestion of an indirectly solvable ML approach [3]. The EM algorithm and standard ML based algorithms maximize the sample joint-likelihood, i.e. the pseudo-likelihood, in batch operation. This process can over-emphasize or favor these observations, so may lead to over-fitting problems, especially when the sample size is small. The sample likelihood will approach the true likelihood only when the number of the sample points tends to infinity. In addition, due to its use of deterministic gradient descent and batch operation nature, the EM algorithm has a high possibility of being trapped in local optima and is also slow to converge [6]–[8]. Some modifications, such as the stochastic EM algorithm, maximum penalized likelihood method, and ensemble averaging method have since been proposed to overcome these problems (e.g., [7]). The penalized likelihood approach offers a balance between goodness-of-fit and smoothness (i.e., generalization ability) [2]. The authors have recently proposed a Bayesian self-organizing map (BSOM) for solving Gaussian mixture problems, and have shown additional advantages (e.g., fewer local optima and faster convergence speed) over the EM algorithm [8], [9]. Wang *et al.* have proposed a probabilistic SOM (PSOM) [10] for segmentation of brain tissue images, whose histograms are modeled as mixtures of Gaussians. Their algorithm optimizes the learning density by referencing to pixel histograms. Its stochastic version is similar to the BSOM. However, in the BSOM the updating rules for the mixing parameters are more generalized; and learning is limited to a neighborhood around the winner. Other similar algorithms include [11], [12]. In [11], a homoscedastic Gaussian mixture model is used for the data; equal mixing is assumed; and a single Gaussian smoothing prior is used to express variation of centroids in the neuron space. The SOM can then be related to the elastic net algorithm [13]. In [12], the generative topographic mapping (GTM) is proposed as a principled dimensional reduction method. The GTM also uses homoscedastic Gaussian mixtures and equal mixing weights. The GTM is used mainly to visualize and model high-dimensional data in a latent variable space of a low dimension.

In this paper, we further extend and generalize the BSOM to general mixture distributions. The resulting network, the self-organizing mixture network (SOMN), combines the Kullback–Leibler information metric [14], the stochastic approximation method [15], and the SOM structure [16]. Since the SOMN is derived for density estimation, it places the neural grid in the same dimension as the input. The Kullback–Leibler metric is the expectation of log likelihood, thus has a natural generalization ability over the raw sample likelihood criterion (or batch likelihood methods). Furthermore, the SOMN limits the learning in a small neighborhood of the winner like the SOM. The algorithm requires only scalar and local calculations, and hence is computational efficient. The SOMN can be applied not only to homogenous mixtures and/or homoscedastic components, but also to inhomogenuous mixtures and/or heteroscedastic components.

II. The Mixture Distribution and Unsupervised Learning

A. Mixture Distributions

Mixture models have been employed in many pattern classification applications (e.g., [1], [10], and [17]). In a mixture distri-

Manuscript received January 4, 1999; revised October 14, 1999 and November 22, 2000. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under Grant GR/M 12889.

The authors are with the Department of Electrical Engineering and Electronics, University of Manchester Institute of Science and Technology, Manchester, M60 1QD, U.K.

Publisher Item Identifier S 1045-9227(01)02055-0.

bution, each sample observation, $\mathbf{x}$, from a d-dimensional input space, $\Omega \in \mathbf{R}^d$, is assigned to one of K pattern components, each of which has a prior probability P_i and a prescribed distribution. The joint probability density of data sample is given by

$$p(\mathbf{x}|\boldsymbol{\Theta}) = \sum_{i=1}^{K} p_i(\mathbf{x}|\boldsymbol{\theta}_i)P_i \tag{1}$$

where

$p_i(\mathbf{x}\|\boldsymbol{\theta}_i)$	ith component-conditional density;
$\boldsymbol{\theta}_i$	sufficient statistics or parameter vector for the ith conditional density, $i = 1, 2, \ldots, K$ and $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K)^T$;
P_i	prior probability of the ith component and is also called the mixing parameter or mixing weight.

For Gaussian and Cauchy mixtures, the conditional densities have the following forms, respectively:

$$p_i(\mathbf{x}|\boldsymbol{\theta}_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}}\, e^{-(1/2)(\mathbf{x}-\mathbf{m}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)} \tag{2}$$

$$p_i(\mathbf{x}|\boldsymbol{\theta}_i) = \frac{1}{\pi|\Sigma_i|^{1/2}[1+(\mathbf{x}-\mathbf{m}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)]} \tag{3}$$

where $\boldsymbol{\theta}_i = \{\theta_{i1}, \theta_{i2}\} = \{\mathbf{m}_i, \Sigma_i\}$ are the mean vector and covariance matrix of the Gaussian or Cauchy density function, respectively.

Commonly used mixture models are homogenous, i.e., all components in a mixture are assumed to possess the same density profile. When a single variance is used for all components, the model is termed homoscedastic. Most existing approaches use either homogenous mixtures or homoscedastic components in order to derive a workable algorithm. This will reduce the accuracy of density matching and applicability of the algorithms. The SOMN, however, does not have these assumptions or simplifications, thus is more suitable for general densities.

B. Maximum Likelihood Estimation and the EM Algorithm

For most unsupervised learning applications, only the forms of the component-conditional densities are known. The model parameters, however, have to be estimated from a set of N unlabeled sample observations, $\mathbf{X} = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(N)\} \in \Omega$. In these cases, the joint-likelihood of all observations is often approximated by

$$p(\mathbf{X}|\Theta) = \prod_{n=1}^{N} p(\mathbf{x}(n)|\Theta),$$

or

$$\hat{l}(\mathbf{x}) = \log[p(\mathbf{X}|\Theta)] = \sum_{n=1}^{N} \log\{p[\mathbf{x}(n)|\Theta]\} \tag{4}$$

assuming independence of the observations. This makes the sample likelihood an approximate (or time average) to the true likelihood, $\int \log[p(\mathbf{x})]p(\mathbf{x})\,d\mathbf{x}$. *Maximizing* this sample *likelihood* (ML) may lead to a singular solution (for example, placing components in data points and letting the variance tend toward zero). When restricted to the largest finite maximum and Gaussian components, it results in implicit equations for these parameters [3], which are only solvable numerically.

The EM algorithm is an iterative ML procedure for parameter estimation under incomplete data or missing data situations [5]. Many problems can be viewed as instances of such situations. For example, in the unsupervised learning for the mixture distribution, the input samples are incomplete and the missing data are the component-labels or indicator functions for each sample. By using the EM procedure, the marginal, or incomplete-data, likelihood is obtained by the average or expectation of the complete-data likelihood with respect to the missing data using the current parameter estimates (E-step), then the new parameter estimates are obtained by maximizing the marginal likelihood (M-step). The EM algorithm has been shown to be an iterative gradient ascent algorithm, in which the likelihood function exhibits no decrease after each iteration.

The EM method has been applied to the estimation of Gaussian mixtures by Xu and Jordan [4], [18]. It is an extended and generalized k-means algorithm with a consideration of component distributions and priors, thus will generally result in much improved clustering than the k-means algorithm. Xu and Jordan [18], [19] have also shown that this EM algorithm is a variable metric gradient ascent algorithm with first-order convergence. They have also acknowledged the slow convergence of the algorithm, especially when the mixture components are not well separated, but found that faster methods such as superlinear and Hessian gradient generally perform poorly for this kind of ill-conditioned problem. The EM algorithm provides a feasible solution to this kind of unsupervised learning problem, but its slow convergence and high computational costs need to be addressed for practical applications. Besides, in solving the likelihood, (4), the entire data set is needed during each iteration of the algorithm. Batch EM operation optimizes the sample likelihood only, so will have poor generalization when the sample size is small.

III. The Self-Organizing Mixture Network

A. The SOMN Structure

Based on the mixture distribution model, i.e., (1), the SOMN structure can be illustrated as in Fig. 1. For a mixture of finite components, the network Ξ places M nodes (M does not need to be equal to K, if K is not known *a priori*) in the input space, Ω. The kernel parameters, e.g., mean vectors, $\mathbf{m}_i$ and covariance matrices, Σ_i, are the learning weights. The output of a kernel is the conditional density of that component in the mixture. The upper layer, or the network output, sums the responses of these kernel weighted by the prior probabilities or mixing parameters, P_i, which are also the learning weights. At each time step, n, a sample point, denoted by $\mathbf{x}(n)$, is randomly taken from Ω or from a finite data set Φ. A winner is chosen according to its kernel output multiplied by its mixing parameter, i.e., estimated posterior probability

$$\hat{P}(i|\mathbf{x}) = \frac{\hat{P}_i\hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i)}{\hat{p}(\mathbf{x}|\hat{\Theta})}. \tag{5}$$

Within a neighborhood of the winner, $\boldsymbol{\eta}_c$, the weights are updated. Thus the SOMN is similar to the SOM in the sense of local learning properties.

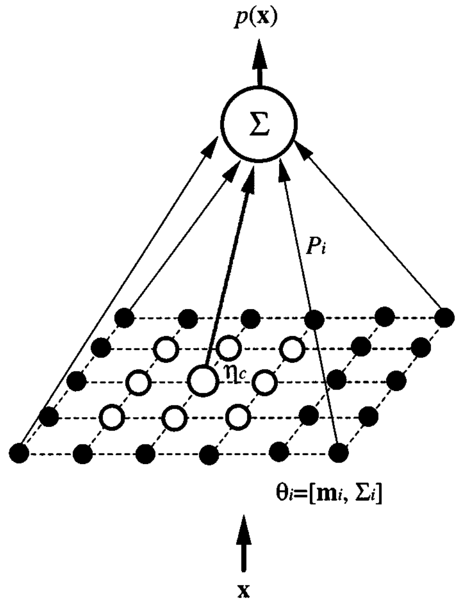

Fig. 1. The structure of the self-organizing mixture network (SOMN).

The number of nodes, however, needs not to be known *a priori*, but should be equal to or larger than the number of underlying components in the mixture to avoid the under-represented problem. That is, one can always use a large number of nodes to learn the mixture, and only the significant ones will remain. Such a number can be a subjective factor in the learning. For a smooth estimation of an arbitrary density, one can use a large number of nodes. For mixtures with a known class number or where only a number of major or principal components need to be traced, a SOMN with this number of nodes can be used to interpret interesting subdensities.

B. The SOMN Updating Algorithm

Suppose that the true environmental data density function and the estimated one are $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$, respectively. The Kullback–Leibler information metric [14] measures the divergence between these two, and is defined as

$$\mathbf{I} = -\int \log \frac{\hat{p}(\mathbf{x})}{p(\mathbf{x})} p(\mathbf{x})\, d\mathbf{x}. \tag{6}$$

It is also referred to as *relative entropy*, and is an expectation of the negative log-likelihood in the limit of an infinite number of data points subtracting a bias which is known as the entropy of the data. It measures the average information remaining in each data point by the estimator. It is always a positive number, and will be zero if and only if the estimated density equals the true one. The sample likelihood approximates the Kullback–Leibler measure by the time average of sample probabilities (without input entropy, a constant for a stationary source). This can be easily seen when comparing (4) and (6). The sample likelihood approach will equal the Kullback–Leibler metric only when the size of the sample tends to infinite and the input is an ergodic process. The Kullback–Leibler measure is arguably more suitable measure for density estimation or unsupervised learning [1], [20], [21]. It is a natural integration of accumulative log likelihood.

When the estimated density is modeled as a mixture distribution, i.e., a function of various subdensities and their parameters, one can seek the optimal estimate of these parameters by minimizing $\mathbf{I}$ via its partial differentials in respect to every model parameter, i.e.,

$$\begin{aligned}\frac{\partial \mathbf{I}}{\partial \theta_{ij}} &= -\int \left[\frac{1}{\hat{p}(\mathbf{x}|\hat{\Theta})} \frac{\partial \hat{p}(\mathbf{x}|\hat{\Theta})}{\partial \theta_{ij}}\right] p(\mathbf{x})\, d\mathbf{x} \equiv 0,\\ &\qquad i = 1, 2, \ldots, K, \text{ and } j = 1, 2\end{aligned} \tag{7}$$

$$\begin{aligned}\frac{\partial \mathbf{I}}{\partial P_i} &= -\int \left[\frac{1}{\hat{p}(\mathbf{x}|\hat{\Theta})} \frac{\partial \hat{p}(\mathbf{x}|\hat{\Theta})}{\partial \hat{P}_i}\right] p(\mathbf{x})\, d\mathbf{x}\\ &\quad + \lambda \frac{\partial}{\partial \hat{P}_i}\left[\sum_{j=1}^{K} \hat{P}_j - 1\right]\\ &= -\frac{1}{\hat{P}_i} \int \left[\frac{\hat{P}_i \hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i)}{\hat{p}(\mathbf{x}|\hat{\Theta})} - \lambda \hat{P}_i\right] p(\mathbf{x})\, d\mathbf{x} \equiv 0,\\ &\qquad i = 1, 2, \ldots, K\end{aligned} \tag{8}$$

where θ_{ij} represents the jth parameter of the ith conditional density, e.g., the mean vector and covariance matrix for $j = 1$ and 2, respectively. In (8), the method of Lagrange multipliers with a constraint parameter λ is used to ensure a valid probability, i.e., $\sum_{i=1}^{K} \hat{P}_i = 1$.

As the true data density is not known, the Robbins–Monro stochastic approximation method [15] can be used for solving these nondirectly solvable equations. This results in the following adaptive updating algorithm with the constraint parameter λ set to one:

$$\begin{aligned}\hat{\theta}_{ij}(n+1) &= \hat{\theta}_{ij}(n) + \alpha(n)\left[\frac{1}{\hat{p}(\mathbf{x}|\hat{\Theta})} \frac{\partial \hat{p}(\mathbf{x}|\hat{\Theta})}{\partial \theta_{ij}(n)}\right]\\ &= \hat{\theta}_{ij}(n) + \alpha(n)\left[\frac{\hat{P}_i(n)}{\hat{p}(\mathbf{x}|\hat{\Theta})} \frac{\partial \hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i)}{\partial \theta_{ij}(n)}\right]\end{aligned} \tag{9}$$

$$\begin{aligned}\hat{P}_i(n+1) &= \hat{P}_i(n) + \alpha(n)\left[\frac{\hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i)\hat{P}_i(n)}{\hat{p}(\mathbf{x}|\hat{\Theta})} - \hat{P}_i(n)\right]\\ &= \hat{P}_i(n) - \alpha(n)[\hat{P}(i|\mathbf{x}) - \hat{P}_i(n)]\end{aligned} \tag{10}$$

where $\alpha(n)$ is the learning rate at time step n, and $0 < \alpha(n) < 1$ and decreases monotonically.

The updating of the above parameters can be limited to a small neighborhood of the winning node, which has the largest response or posterior probability, i.e., $\hat{P}(i|\mathbf{x})$, due to the diminishing spreading properties of the most conditional densities. That is, the density can be approximated by a mixture of a small number of nodes at one time, i.e.,

$$\hat{p}(\mathbf{x}|\Theta) \approx \sum_{i \in \boldsymbol{\eta}_c} \hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i)\hat{P}_i \tag{11}$$

where c is the winning node index and $\boldsymbol{\eta}_c$ is a neighborhood of the winner.

It is straightforward to calculate the corresponding partial differential terms in (9) for a specified conditional model, e.g., (2)

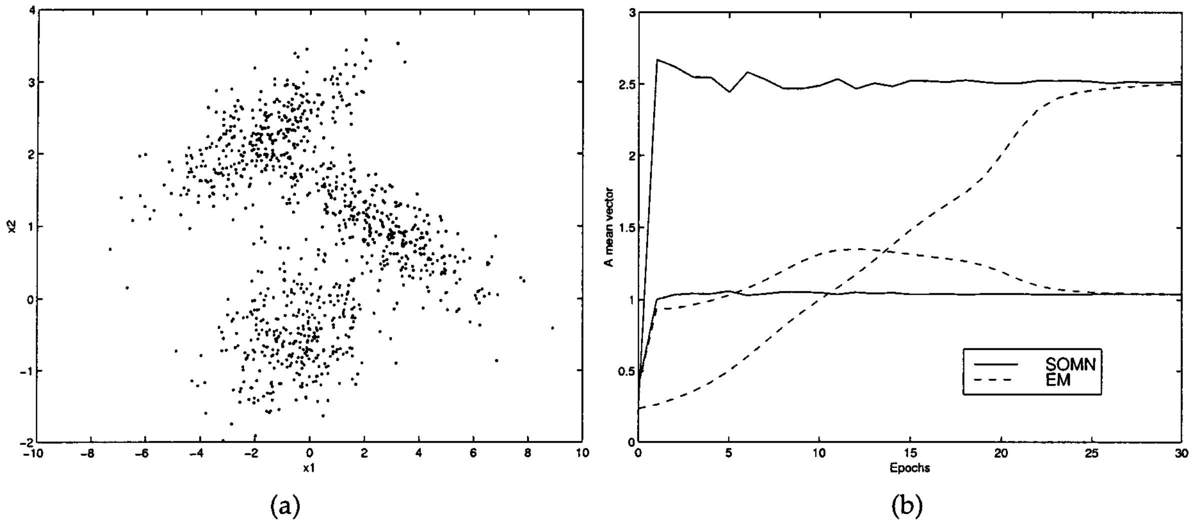

Fig. 2. (a) Data scatter. (b) The Gaussian SOMN versus the EM algorithm.

or (3). That is, for a Gaussian mixture or a Gaussian component in a mixture, the learning rules for the mean vector and covariance matrix are

$$\Delta\hat{\mathbf{m}}_i = \alpha(n)\hat{P}(i|\mathbf{x})[\mathbf{x}(n) - \hat{\mathbf{m}}_i(n)] \tag{12}$$

$$\Delta\hat{\Sigma}_i = \alpha(n)\hat{P}(i|\mathbf{x})\{[\mathbf{x}(n)-\hat{\mathbf{m}}_i(n)][\mathbf{x}(n)-\hat{\mathbf{m}}_i(n)]^T - \hat{\Sigma}_i(n)\}. \tag{13}$$

For Cauchy mixture or a Cauchy component in a mixture, they are

$$\Delta\hat{\mathbf{m}}_i = \alpha(n)\hat{P}(i|\mathbf{x})\hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i)[\mathbf{x}(n) - \hat{\mathbf{m}}_i(n)] \tag{14}$$

$$\Delta\hat{\Sigma}_i = \alpha(n)\hat{P}(i|\mathbf{x})\hat{p}_i(\mathbf{x}|\hat{\boldsymbol{\theta}}_i) \cdot \{[\mathbf{x}(n) - \hat{\mathbf{m}}_i(n)][\mathbf{x}(n) - \hat{\mathbf{m}}_i(n)]^T - \hat{\Sigma}_i(n)\}. \tag{15}$$

These forms, (12) in particular, resemble the SOM updating algorithms. This shows that the SOM is a simplified Gaussian mixture estimator and uses the homoscedastic Gaussian mixture model for the data, when a Gaussian neighborhood function is used as it is often the case (the posterior, $\hat{P}(i|\mathbf{x})$, is also Gaussian for Gaussian mixtures). Only the mean vectors are learning, the mixing parameters are assumed equal, and single variance is implied by the size of the neighborhood. A large neighborhood at the beginning means a large variance of the Gaussians or a broad overlapping among them. It also means a high mobility for the neurons. This is helpful in finding the global or, at least, a better local optimum, especially at the beginning of the learning. Small neighborhood sizes mean small variances for the Gaussians and little overlapping among them, thus a low mobility. Shrinking the neighborhood during the process provides a subjective and *ad hoc* adjustment to the variance of the Gaussians in order to aid the fitting and to allow them to settle to the data distribution as the learning progresses.

IV. Experiments and Results

A. Classification Problems

For the following unsupervised classification example, the data used are the same to those in [4]. There are three classes. The scatter of the data vectors is shown in Fig. 2(a). As a comparison, convergence curves for one of the two mean vectors of both SOMN and EM algorithms under the same random initial conditions are given in Fig. 2(b). In this example, the initial mean vectors of a three-node Gaussian SOMN were set to small random vectors around $[0,\ 0]^T$, the initial mixing priors were set equally to 1/3, and the initial variances were set equally to a large diagonal matrices (comparable to the raw sample variance), e.g., $\mathrm{diag}(10,\ 10)$. Learning rates in the SOMN were linearly decreasing from 0.5 and 0.05 for the mean vectors and for the covariances and mixing priors, respectively.

Although both algorithms can achieve similar final results and classification rates (around 96%), it is found that the SOMN algorithm has three advantages over the EM algorithm. Firstly, the stochastic-gradient-based SOMN algorithm converges much faster than the deterministic gradient-based EM (batch) algorithm. This can be seen from Fig. 2(b). Generally, stochastic learning algorithms use large learning rates. While the batch EM uses $1/n$ as learning rate (it can be easily seen when the EM algorithm is written in an on-line fashion), which becomes very small after only a few iterations. This results in weakened power in learning. Detailed quantitative analysis has been given in [22]. Second, the initial conditions have a greater influence on the convergent results of the EM algorithm. The SOMN, however, is more robust when random initial values are used. In our numerous trials for this example with random initials, the SOMN converged successfully in all cases, while the EM did not converge to the correct results in almost half of them. Third, the EM algorithm is easily trapped in local optima as it is an exact gradient ascent/descent algorithm. The SOMN, being a stochastic gradient based, can easily escape shallow local minima.

When a stochastic version of EM algorithm is used, two algorithms may become comparable. However, in SOMN, the learning is limited to a small neighborhood of the winner only rather than to the entire network as the EM does—this is computationally advantageous especially when the size of the network is large.

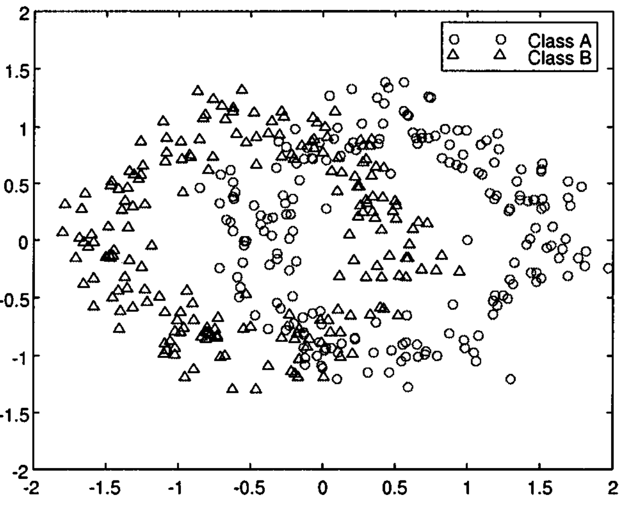

Fig. 3. Data scatters (two pattern classes).

An accurate estimation of pattern class-conditional densities is vital for an optimal classifier. The next experiment uses the data similar to that used by Ormoneit and Tresp [7]. In total, 200 observations were generated for each class, of which half were randomly chosen for training and the other half for testing. There are two pattern classes, the scatters of the entire data are shown in Fig. 3. Details of how these data sets are generated are given in [7]. Basically a Gaussian distributed data point is first generated and mapped onto the unit circle then summed with a further Gaussian dispersion and an additional offset. Such a density requires a larger number (if not an infinity) of Gaussians to approximate.

Each class-conditional distribution was estimated by a 10×10 SOMN, which was initially placed regularly into the data region (roughly, say $[-2, 2]\times[-2, 2]$ square). Isotropic Gaussian nodes were used with initial variances set equally. A fairly uniform (flat) distribution was assumed before learning as no prior information about the density was available. Hence, the initial variance was chosen approximately proportional to the (minimum) inter-distance between the nodes, so that the initial map would evenly cover the entire region. The exact value of this initial variance, however, is not critical. At each iteration, one data point was randomly taken from the 100-point training set (sample with replacement). Update was limited to the winner and its second-order nearest neighbors. The learning rates for means and for variances and mixing priors were decreasing from 0.2 and 0.02, respectively. Typical results, i.e., estimated densities, after just 2000 iterations (i.e., 20 epochs), are given in Fig. 4. Note, only 100 data points were used in learning, a very small data set for such a complicated density. Based on these probability distribution, an optimal classifier can then be constructed according to the Bayes rule (with class priors in proportional to the percentages of the class samples, 0.5 for each class in this example). The average classification rates after 30 independent simulations are of 88.6% and 85.1% for training and testing, respectively. The EM algorithm was also applied to this problem under same initial conditions. It is found that it needs at least 40 epochs to converge and each epoch takes much longer than the SOMN (as the SOMN only requires 25% computation of the EM for each iteration for this example). The averaged classification rates of the EM, also after 30 runs, are lower, 85.2% and 82.9% for training and testing, respectively.

B. *PDF and Function Profiles*

The profile of an x-ray rocking curve (i.e., the profile of a X-ray diffraction peak as a function of photon energy) is shown in Fig. 5 (dash line). Though in reality a spectrum, the experimental sample (200 data points in total) were resampled randomly to provide an effective density histogram for network training (19 512 points were generated). The network, initially assigned five Cauchy components as the true number of peaks was assumed unknown, has successfully learned the two main peaks. The estimated density (solid line) is after five epochs. The resulting weights of five components are list in Table I. Two components (nodes 4 and 5) are active, others are too small to contribute meaningfully. A Gaussian mixture would have required many more nodes for a smooth interpretation. This illustrates the advantage of choosing a parametric form that agrees with the suspected form of the data or the underlying physical generation of the data. The estimated relative entropy, (4) (without data entropy), can be used to validate the choice of component forms.

In the final example, a SOMN with five Gaussian nodes was used to approximate a noisy snapshot image from a capillary electrophoresis system [23]. The data is shown in Fig. 6 as a dashed line. The solid line shows the result after only five learning epochs. Two major peaks have been accurately located, and other peaks have also been revealed. An important feature of the SOMN is that it can simultaneously provide the width information about each peak, which in this application is important as they relate directly to the diffusion coefficients of the chemical analyzes. The resulting parameters are listed in Table II.

In the last two examples, only the winning node and its two neighboring nodes (one on each side) were updated at each iteration. The learning rates for the means and for the variances and mixing priors were linearly decreasing from 0.2 and 0.02, respectively. To prevent the variances becoming singular, negative definite or varying drastically between consecutive updates, it is better to use small learning rates for both variances and mixing priors. However, it has been found that the network converges over a wide range for these rates.

V. Conclusion and Discussion

An unsupervised learning structure, based on minimizing the Kullback–Leibler information entropy using stochastic approximation and a self-organizing principle, is proposed for estimating general mixture distributions. As it has been shown that the SOMN yields robust and accurate estimations even for small data sets and converges fast. It outperforms the batch EM algorithm in various aspects. It is computationally efficient, using only a fraction of the computation cost of the EM (batch and stochastic) algorithm. It uses heteroscedatic components in a mixture and can also use more general inhomogenous mixture models, thus is more applicable. It can be regarded as a more generalized and adaptive version of the EM algorithm or ML approach for unsupervised learning and data density modeling,

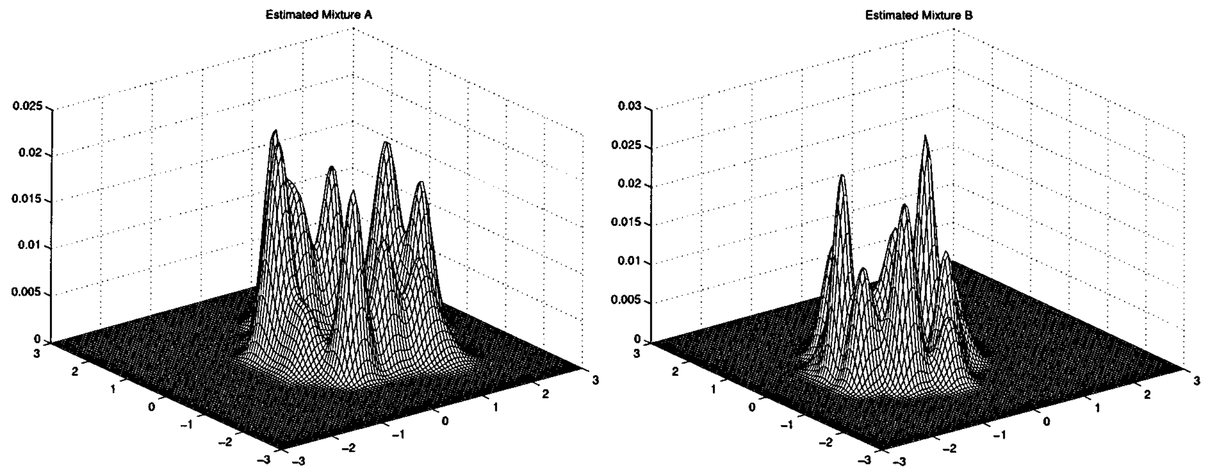

Fig. 4. Estimated mixtures by two SOMNs from half of the data points in Fig. 3.

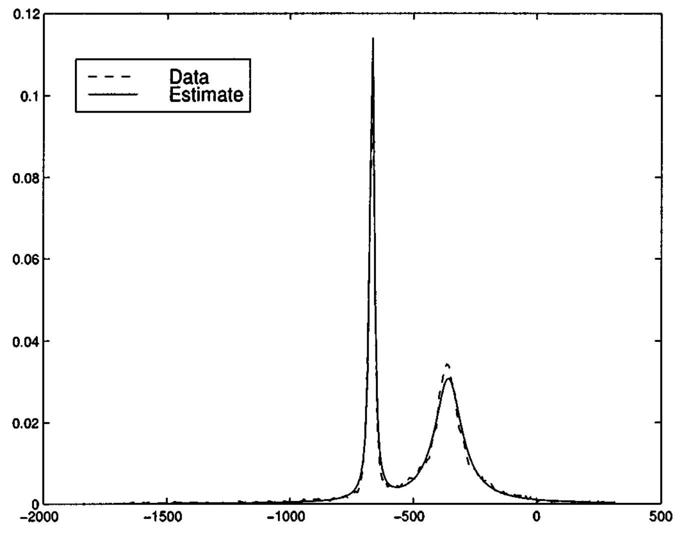

Fig. 5. Mixture estimation using a Cauchy SOMN for a X-ray rocking curve profile.

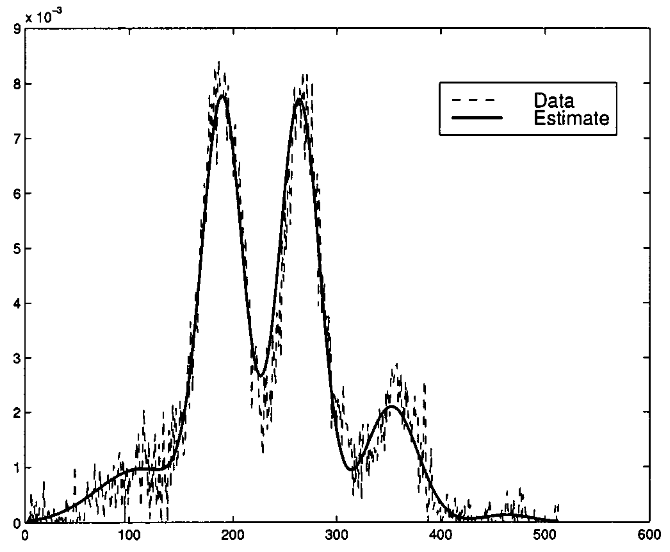

Fig. 6. Mixture estimation using a Gaussian SOMN for capillery electrophoresis data.

TABLE I
RESULTS FOR INDIVIDUAL NODES IN THE MIXTURE OF FIG. 4

node	mixing weight	mean	variance (σ)
1	6.42×10^{-322}	819.69	20.10
2	6.37×10^{-322}	1302.90	20.10
3	3.82×10^{-33}	30.27	119.03
4	**0.3711**	**-667.79**	**10.29**
5	**0.6289**	**-357.93**	**67.03**

TABLE II
ESTIMATED COMPONENT PARAMETERS OF THE MIXTURE IN FIG. 5

node	mixing weight	mean	variance (σ)
1	0.1014	112.2	41.9
2	0.3710	189.6	19.7
3	0.3857	266.6	19.8
4	0.1319	351.8	25.2
5	0.0099	463.7	23.6

since it optimizes the expected entropy. The EM algorithm optimizes the sample likelihood, which is only an approximation (by simple averaging) to the entropy. The PSOM, close to Gaussian SOMN, optimizes the sample entropy, which uses the sample histograms as an estimate of the true distribution. The advantage of a good estimator is that it will reduce the burden of a complicated regularization procedure or duplicated training processes as in "assembling" or "bagging" committee methods.

Like the SOM algorithm, the SOMN is a computationally simple algorithm and is easy to implement. Its neighborhood conscience learning and locality of the kernel function provide the network with efficient local computation. It also reveals the mixing role of the neighborhood function in the SOM, thus provides some insight to the self-organization process. In the SOM, a homogenous and homoscedastic Gaussian mixture model is implied. Variances are prejudged and controlled by the radius of the updating neighborhood. Beginning with a large neighborhood, it assumes large variances or broad overlapping among the active fields of neurons. This assumes a flat or unshaped distribution to the neurons thus provides them a needed learnability. Shrinking the neighborhood provides adjustments to the variance thus the mixing range in order to fit the map to the data more closely and to limit the activity of the neurons as they gradually learn from the data. The SOMN shows the relationships between likelihood and entropy approaches and relations between the EM algorithm and the SOM. Although the SOM will finally minimize the mean-squared-error (if the neighborhood disappears), the SOMN shows that the self-organizing process of the SOM is also an entropy-related learning process.

Acknowledgment

The authors greatly acknowledge the provision of X-ray data by Prof. K. E. Singer (UMIST), and capillary electrophoresis data by Dr. B. Pokric (UMIST) and Dr. E. Bergstrom (University of York). We are also grateful to anonymous reviewers for their helpful comments.

References

[1] D. M. Titterington, A. F. M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. New York: Wiley, 1985.
[2] B. W. Silverman, *Density Estimation: For Statistics and Data Analysis*. London: Chapman and Hall, 1986.
[3] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[4] L. Xu and M. I. Jordan, "Unsupervised learning by EM algorithm based on finite mixture of Gaussians," in *Proc. World Congr. Neural Networks (II)*, 1993, pp. 431–434.
[5] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete date via the EM algorithm," *J. Roy. Statist. Soc. B*, vol. 39, pp. 1–38, 1977.
[6] R. A. Redener and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Rev.*, vol. 26, no. 2, pp. 195–239, 1984.
[7] D. Ormoneit and V. Tresp, "Averaging, maximum penalised likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates," *IEEE Trans. Neural Networks*, vol. 9, pp. 639–650, 1998.
[8] H. Yin and N. M. Allinson, "Comparison of a Bayesian SOM with the EM algorithm for Gaussian mixtures," in *Proc. Workshop Self-Organizing Maps*, 1997, pp. 304–305.
[9] H. Yin and N. M. Allinson, "Bayesian learning for self-organizing maps," *Electron. Lett.*, vol. 33, pp. 304–305, 1997.
[10] Y. Wang, T. Adali, S.-Y. Kung, and Z. Szabo, "Quantification and segmentation of brain tissues from MR images: A probabilistic neural network approach," *IEEE Trans. Image Processing*, vol. 7, Aug. 1998.
[11] A. Utsugi, "Hyperparameter selection for self-organizing maps," *Neural Comput.*, vol. 9, pp. 637–648, 1997.
[12] C. M. Bishop, M. Svensen, and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Comput*, vol. 10, pp. 215–234, 1998.
[13] R. Durbin, R. Szeliski, and A. Yuille, "An analysis of the elastic net approach to the travelling salesman problem," *Neural Comput.*, vol. 1, pp. 348–358, 1989.
[14] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, pp. 79–86, 1951.
[15] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.
[16] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, pp. 56–69, 1982.
[17] 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.
[18] L. Xu and M. I. Jordan, "On convergence properties of the EM algorithm for Gaussian mixtures," *Neural Comput.*, vol. 8, pp. 129–151, 1996.
[19] M. I. Jordan and L. Xu, "Convergence results for the EM approach to mixture of experts architectures," *Neural Networks*, vol. 8, pp. 1409–1431, 1995.
[20] M. Benaim and L. Tomasini, "Competitive and self-organizing algorithms based on the minimization of an information criterion," in *Proc. Int. Conf. Artificial Neural Networks*, 1991, pp. 391–396.
[21] H. White, "Learning in artificial neural networks," *Neural Comput.*, vol. 1, pp. 425–464, 1989.
[22] H. Yin and N. M. Allinson, "A Bayesian self-organizing map for Gaussian mixtures," in Proc. Inst. Elect. Eng. Vision, Image, and Signal Processing, 2001, to be published.
[23] B. Pokric, N. M. Allinson, E. T. Bergstrom, and D. M. Goodall, "Dynamic analysis of Capillary electrophoresis data using real-time neural networks," *J. Chromatography A*, pp. 231–244, 1999.