Model-Based Clustering by Probabilistic Self-Organizing Maps

Shih-Sian Cheng, Hsin-Chia Fu, *Member, IEEE*, and Hsin-Min Wang, *Senior Member, IEEE*

***Abstract*—In this paper, we consider the learning process of a probabilistic self-organizing map (PbSOM) as a model-based data clustering procedure that preserves the topological relationships between data clusters in a neural network. Based on this concept, we develop a coupling-likelihood mixture model for the PbSOM that extends the reference vectors in Kohonen's self-organizing map (SOM) to multivariate Gaussian distributions. We also derive three expectation–maximization (EM)-type algorithms, called the SOCEM, SOEM, and SODAEM algorithms, for learning the model (PbSOM) based on the maximum-likelihood criterion. SOCEM is derived by using the classification EM (CEM) algorithm to maximize the classification likelihood; SOEM is derived by using the EM algorithm to maximize the mixture likelihood; and SODAEM is a deterministic annealing (DA) variant of SOCEM and SOEM. Moreover, by shrinking the neighborhood size, SOCEM and SOEM can be interpreted, respectively, as DA variants of the CEM and EM algorithms for Gaussian model-based clustering. The experimental results show that the proposed PbSOM learning algorithms achieve comparable data clustering performance to that of the deterministic annealing EM (DAEM) approach, while maintaining the topology-preserving property.**

***Index Terms*—Classification expectation–maximization (CEM) algorithm, deterministic annealing expectation–maximization (DAEM) algorithm, expectation–maximization (EM) algorithm, model-based clustering, probabilistic self-organizing map (PbSOM), self-organizing map (SOM).**

I. INTRODUCTION

IN model-based clustering, data samples are grouped by learning a mixture model (usually a Gaussian mixture model) in which each mixture component represents a group or cluster. There are two major learning methods for model-based clustering: the *mixture-likelihood approach*, where the likelihood of each data sample is a mixture of all the component likelihoods of the data sample, and the *classification-likelihood approach*, where the likelihood of each data sample is generated by its winning component only [1]–[9]. In both approaches, when the globally optimal estimation of the model parameters cannot be obtained analytically, iterative learning algorithms that only guarantee obtaining locally optimal solutions are usually employed. The expectation–maximization (EM) algorithm for mixture-likelihood learning [10], [11] and the classification EM (CEM) algorithm for classification-likelihood learning [8] are two such algorithms. However, a critical aspect of the EM and CEM algorithms is that their learning performance is very sensitive to the initial conditions of the model's parameters. To address this issue, Ueda and Nakano [12] proposed a deterministic annealing EM (DAEM) algorithm that tackles the initialization issue via a deterministic annealing process, which performs robust optimization based on an analogy to the cooling of a system in statistical physics. Some heuristic-like learning algorithms have also been proposed. For example, in [13], the authors propose an algorithm that finds the appropriate initial conditions for EM learning by using split and merge operations. Another method, proposed in [14], overcomes the initialization issue of EM by iteratively splitting the mixture components using the Bayesian information criterion as the splitting validity measure.

In addition to the initialization issue of the learning algorithms, conventional model-based clustering suffers from another limitation in that it cannot preserve the topological relationships among clusters after the clustering procedure. To overcome this shortcoming, the clustering task can be performed by using Kohonen's self-organizing map (SOM) [15], [16], a well-known neural network model for data clustering and visualization. After the clustering procedure, the topological relationships among data clusters can be preserved (or visualized) on the network, which is usually a 2-D lattice. Kohonen's sequential and batch SOM learning algorithms have proved successful in many practical applications [15], [16]. However, they also suffer from some shortcomings, such as the lack of an objective (cost) function, a general proof of convergence, and a probabilistic framework [17]. The following are some related works that have addressed these issues. In [18] and [19], the behavior of Kohonen's sequential learning algorithm was studied in terms of energy functions, based on which, Cheng [20] proposed an energy function for SOM whose parameters can be learned by a K-means-type algorithm. Luttrell [21], [22] proposed a noisy vector quantization model called the topographic vector quantizer (TVQ), whose training process coincides with the learning process of SOM. The cost function of TVQ represents the topographic distortion between the input data and the output code vectors in terms of Euclidean distance. Graepel *et al.* [23], [24] derived a soft topographic vector quantization (STVQ) algorithm by applying a deterministic annealing process to the optimization of TVQ's cost function. Based on the topographic distortion concept,

Manuscript received November 01, 2007; revised November 30, 2008; accepted January 02, 2009. First published March 31, 2009; current version published May 01, 2009. This work was supported in part by the National Science Council of Taiwan under Grants NSC96-2628-E-001-024-MY3 and NSC96-3113-H-001-012.

S.-S. Cheng is with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan and also with the Institute of Information Science, Academia Sinica, Taipei, Taiwan (e-mail: sscheng@iis.sinica.edu.tw).

H.-C. Fu is with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan (e-mail: hcfu@csie.nctu.edu.tw).

H.-M. Wang is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan (e-mail: whm@iis.sinica.edu.tw).

Digital Object Identifier 10.1109/TNN.2009.2013708

Heskes [25] applied a different DA implementation from that of STVQ, and obtained an algorithm identical to STVQ when the quantization error is expressed in terms of Euclidean distance. In [26], Chow and Wu proposed an online algorithm for STVQ; later, motivated by STVQ, they proposed a data visualization method that integrates SOM and multidimensional scaling [27]. Based on the Bayesian analysis of SOMs in [28], Anouar *et al.* [29] proposed a probabilistic formalism for SOM, where the parameters are learned by a K-means-type algorithm. To help users select the correct model complexity for SOM by probabilistic assessment, Lampinen and Kostiainen [30] developed a generative model in which the SOM is trained by Kohonen's algorithm. Meanwhile, Van Hulle [31] developed a kernel-based topographic formation in which the parameters are adjusted to maximize the joint entropy of the kernel outputs. He subsequently developed a new algorithm with heteroscedastic Gaussian mixtures that allows for a unified account of vector quantization, log likelihood, and Kullback–Leibler divergence [32]. Another probabilistic formulation is proposed in [33], whereby a normalized neighborhood function of SOM is used as the posterior distribution in the *E-step* of the EM algorithm for a mixture model to enforce the self-organizing of the mixture components. Sum *et al.* [34] interpreted Kohonen's sequential learning algorithm in terms of maximizing the local correlations (coupling energies) between neurons and their neighborhoods for the given input data. They then proposed an energy function for SOM that reveals the correlations, and a gradient-ascent learning algorithm for the energy function.

In Kohonen's SOM architecture, neurons in the network associate with reference vectors in the data space. This contrasts with a SOM whose neurons associate with reference models that present probability distributions, such as the isotropic Gaussians used in [33] and the heteroscedastic Gaussians used in [29] and [32]. In this paper, we call the latter a probabilistic SOM (PbSOM). Motivated by the coupling energy concept in Sum *et al.*'s work [34], we develop a coupling-likelihood mixture model for the PbSOM that uses multivariate Gaussian distributions as the reference models. In the proposed model, local coupling energies between neurons and their neighborhoods are expressed in terms of probabilistic likelihoods; and each mixture component expresses the local coupling-likelihood between one neuron and its neighborhood. Based on this model, we develop CEM, EM, and DAEM algorithms for learning PbSOMs, namely, the SOCEM, SOEM, and SODAEM algorithms, respectively. Because they inherit the properties of the CEM and EM algorithms, the proposed algorithms are characterized by reliable convergence, low cost per iteration, economy of storage, and ease of programming. From our experiments on the organizing property, we observe that SOEM is less sensitive to the initialization of the parameters than SOCEM when using a small-fixed neighborhood, while SODAEM overcomes the initialization problem of SOCEM and SOEM through an annealing process. Furthermore, we show that SOCEM and SOEM can be interpreted, respectively, as deterministic annealing variants of the CEM and EM algorithms for Gaussian model-based clustering, where the neighborhood shrinking is interpreted as an annealing process. We conducted experiments on data sets from the University of California at Irvine (UCI) Machine Learning Database Repository [35]. The experiment results show that the proposed PbSOM learning algorithms achieve comparable data clustering performance to the DAEM algorithm, while maintaining the topology-preserving property.

The remainder of this paper is organized as follows. In Section II, we review the EM, CEM, and DAEM algorithms for model-based clustering. Then, the proposed coupling-likelihood mixture model, and the SOCEM, SOEM, and SODAEM algorithms are described in Section III. The experimental results are detailed in Section IV. The differences and relations between the proposed algorithms and other ones are discussed in Section V. We then present our conclusions in Section VI.

II. The EM, CEM, and DAEM Algorithms for Model-Based Clustering

A. The Mixture-Likelihood Approach and EM Algorithm

In the mixture-likelihood approach for model-based clustering, it is assumed that the given data set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \Re^d$ is generated by a set of independently and identically distributed (i.i.d.) random vectors from a mixture model

$$p(\mathbf{x}_i;\boldsymbol{\Theta}) = \sum_{k=1}^{G} w(k) p(\mathbf{x}_i;\boldsymbol{\theta}_k) \tag{1}$$

where $w(k)$ is the mixing weight of the mixture component $p(\mathbf{x}_i;\boldsymbol{\theta}_k)$, subject to $0 \le w(k) \le 1$ for $k = 1, 2, \ldots, G$; $\sum_{k=1}^{G} w(k) = 1$; and $\boldsymbol{\theta}_k$ denotes the parameter set of $p(\mathbf{x}_i;\boldsymbol{\theta}_k)$.

The maximum-likelihood estimate of the parameter set of the mixture model $\hat{\boldsymbol{\Theta}} = \{\hat{w}(1), \hat{w}(2), \ldots, \hat{w}(G), \hat{\boldsymbol{\theta}}_1, \hat{\boldsymbol{\theta}}_2, \ldots, \hat{\boldsymbol{\theta}}_G\}$ can be obtained by maximizing the following log-likelihood function:

$$\begin{aligned} L(\boldsymbol{\Theta};\mathcal{X}) &= \log \prod_{i=1}^{N} p(\mathbf{x}_i;\boldsymbol{\Theta}) \\ &= \sum_{i=1}^{N} \log \left(\sum_{k=1}^{G} w(k) p(\mathbf{x}_i;\boldsymbol{\theta}_k) \right). \end{aligned} \tag{2}$$

This is usually achieved by using the EM algorithm [10], [11]. After learning the mixture model, we derive a partition of $\mathcal{X}$, $\hat{\mathcal{P}} = \{\hat{\mathcal{P}}_1, \hat{\mathcal{P}}_2, \ldots, \hat{\mathcal{P}}_G\}$, by assigning each $\mathbf{x}_i \in \mathcal{X}$ to the mixture component that has the largest posterior probability for $\mathbf{x}_i$, i.e., $\mathbf{x}_i \in \hat{\mathcal{P}}_j$ if $j = \arg\max_k p(\hat{\boldsymbol{\theta}}_k \mid \mathbf{x}_i;\hat{\boldsymbol{\Theta}})$.

1) The EM Algorithm for Mixture Models: If the maximum-likelihood estimation of the parameters cannot be accomplished analytically, the EM algorithm is normally used as an alternative approach when the given data is incomplete or contains hidden information.

In the case of the mixture model, suppose that $\boldsymbol{\Theta}^{(t)}$ denotes the current estimate of the parameter set, and k is the hidden variable that indicates the mixture component from which the observation is generated. The *E-step* of the EM algorithm then computes the following so-called *auxiliary function*:

$$Q(\boldsymbol{\Theta};\boldsymbol{\Theta}^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{G} p(k \mid \mathbf{x}_i;\boldsymbol{\Theta}^{(t)}) \log p(\mathbf{x}_i, k;\boldsymbol{\Theta}) \tag{3}$$

where

$$p(\mathbf{x}_i, k; \mathbf{\Theta}) = w(k)p(\mathbf{x}_i; \boldsymbol{\theta}_k) \tag{4}$$

and

$$p(k \mid \mathbf{x}_i; \mathbf{\Theta}^{(t)}) = \frac{w(k)^{(t)}p(\mathbf{x}_i; \boldsymbol{\theta}_k^{(t)})}{\sum_{j=1}^{G} w(j)^{(t)}p(\mathbf{x}_i; \boldsymbol{\theta}_j^{(t)})} \tag{5}$$

denotes the posterior probability of the kth mixture component for $\mathbf{x}_i$ with the given $\mathbf{\Theta}^{(t)}$. Then, in the following *M-step*, the $\mathbf{\Theta}^{(t+1)}$ that satisfies

$$Q(\mathbf{\Theta}^{(t+1)}; \mathbf{\Theta}^{(t)}) = \max_{\mathbf{\Theta}} Q(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) \tag{6}$$

is chosen as the new estimate of the parameter set. By iteratively creating the auxiliary function in (3) and performing the subsequent maximization step, the EM algorithm guarantees to converge to a local maximum of the log-likelihood function in (2).

When $Q(\mathbf{\Theta}; \mathbf{\Theta}^{(t)})$ cannot be maximized analytically, the *M-step* is modified to find some $\mathbf{\Theta}^{(t+1)}$ such that $Q(\mathbf{\Theta}^{(t+1)}; \mathbf{\Theta}^{(t)}) > Q(\mathbf{\Theta}^{(t)}; \mathbf{\Theta}^{(t)})$. This type of algorithm, called generalized EM (GEM), is also guaranteed to converge to a local maximum [10], [11].

B. The Classification-Likelihood Approach and CEM Algorithm

In the classification-likelihood approach for model-based clustering [6]–[8], instead of maximizing the log-likelihood function of the mixture model in (2), the objective is to find the partition $\hat{\mathcal{P}} = \{\hat{\mathcal{P}}_1, \hat{\mathcal{P}}_2, \dots, \hat{\mathcal{P}}_G\}$ of $\mathcal{X}$ and the model parameters that maximize

$$C_1(\mathcal{P}, \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_G\}; \mathcal{X}) = \sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \mathcal{P}_k} \log p(\mathbf{x}_i; \boldsymbol{\theta}_k) \tag{7}$$

or

$$C_2(\mathcal{P}, \mathbf{\Theta}; \mathcal{X}) = \sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \mathcal{P}_k} \log(w(k)p(\mathbf{x}_i; \boldsymbol{\theta}_k)). \tag{8}$$

The relation between C_1 and C_2 is

$$C_2(\mathcal{P}, \mathbf{\Theta}; \mathcal{X}) = C_1(\mathcal{P}, \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_G\}; \mathcal{X}) + \sum_{k=1}^{G} |\mathcal{P}_k| \log w(k) \tag{9}$$

where $|\mathcal{P}_k|$ denotes the number of samples in $\mathcal{P}_k$. If all the mixture components are equally weighted, $\sum_{k=1}^{G} |\mathcal{P}_k| \log w(k)$ becomes a constant, such that C_1 and C_2 are equivalent.

1) The CEM Algorithm for Mixture Models: Celeux and Govaert [8] proposed the classification EM (CEM) algorithm for estimating the parameter set $\mathbf{\Theta}$ and partition $\mathcal{P}$. Like the EM algorithm, the CEM algorithm is also an iterative learning approach. In each iteration, CEM inserts a *classification* step (*C-step*) between the *E-step* and *M-step* of the EM algorithm. In the *E-step*, the posterior probability of each mixture component is calculated for each data sample. In the *C-step*, to obtain the partition $\hat{\mathcal{P}}$ of the data samples, each sample is assigned to the mixture component that yields the largest posterior probability for that sample. In the *M-step*, the maximization process is applied to $\hat{\mathcal{P}}_k$ individually for $k = 1, 2, \dots, G$. For example, if a multivariate Gaussian is used as the mixture component, the reestimated mean vector and covariance matrix are the mean vector and the covariance matrix of the data samples in $\hat{\mathcal{P}}_k$, respectively, while the reestimated mixture weight is $|\hat{\mathcal{P}}_k|/N$. From a practical point of view, CEM is a K-means-type algorithm that represents the prototypes with probability distributions [8].

C. The DAEM Algorithm

In the DAEM algorithm for learning a mixture model [12], the objective is to minimize the following system energy function during the annealing process:

$$F_\beta(\mathbf{\Theta}; \mathcal{X}) = -\frac{1}{\beta} \sum_{i=1}^{N} \log \left(\sum_{k=1}^{G} (w(k)p(\mathbf{x}_i; \boldsymbol{\theta}_k))^\beta \right) \tag{10}$$

where $1/\beta$ corresponds to the *temperature* that controls the annealing process. The auxiliary function in this case is

$$U_\beta(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) = -\sum_{i=1}^{N} \sum_{k=1}^{G} f(k \mid \mathbf{x}_i; \mathbf{\Theta}^{(t)}) \log p(\mathbf{x}_i, k; \mathbf{\Theta}) \tag{11}$$

where

$$f(k \mid \mathbf{x}_i; \mathbf{\Theta}^{(t)}) = \frac{(w(k)^{(t)}p(\mathbf{x}_i; \boldsymbol{\theta}_k^{(t)}))^\beta}{\sum_{j=1}^{G} (w(j)^{(t)}p(\mathbf{x}_i; \boldsymbol{\theta}_j^{(t)}))^\beta} \tag{12}$$

is the posterior probability derived by using the maximum entropy principle.

Ueda and Nakano [12] showed that $F_\beta(\mathbf{\Theta}; \mathcal{X})$ can be iteratively minimized by iteratively minimizing $U_\beta(\mathbf{\Theta}; \mathbf{\Theta}^{(t)})$. When using DAEM to learn a mixture model, β is initialized with a small value (less than 1) such that the energy function itself is simple enough to be optimized. Then, the value of β is gradually increased to 1. During the learning process, the parameters learned in the current learning phase are used as the initial parameters of the next phase.[1] In the case of $\beta = 1$, $F_\beta(\mathbf{\Theta}; \mathcal{X})$ and $U_\beta(\mathbf{\Theta}; \mathbf{\Theta}^{(t)})$ are the negatives of the log-likelihood function in (2) and the Q-function in (3), respectively, thus, minimizing $F_\beta(\mathbf{\Theta}; \mathcal{X})$ is equivalent to maximizing the log-likelihood function.

According to [12], (10) can be rewritten as

$$F_\beta(\mathbf{\Theta}; \mathcal{X}) = U_\beta(\mathbf{\Theta}) - \frac{1}{\beta} S_\beta(\mathbf{\Theta}) \tag{13}$$

where

$$U_\beta(\mathbf{\Theta}) = -\sum_{i=1}^{N} \sum_{k=1}^{G} f(k \mid \mathbf{x}_i; \mathbf{\Theta}) \log p(\mathbf{x}_i, k; \mathbf{\Theta}) \tag{14}$$

[1]Each β value corresponds to a learning phase. The algorithm proceeds to the next phase after it converges in the current phase.

and

$$S_\beta(\boldsymbol{\Theta}) = -\sum_{i=1}^{N}\sum_{k=1}^{G} f(k \mid \mathbf{x}_i;\boldsymbol{\Theta}) \log f(k \mid \mathbf{x}_i;\boldsymbol{\Theta}) \tag{15}$$

is the entropy of the posterior distribution. When $\beta \to \infty$, the rational function $f(k \mid \mathbf{x}_i;\boldsymbol{\Theta})$ approximates to a zero-one function; thus, the entropy term $S_\beta(\boldsymbol{\Theta}) \to 0$. In this case, $F_\beta(\boldsymbol{\Theta};\mathcal{X})$ is equivalent to the negative of the objective function for CEM in (8). Therefore, DAEM can be viewed as a DA variant of CEM.

III. The EM-Type Algorithms for Learning PbSOMs

A. Formulation of the Coupling-Likelihood Mixture Model

In this paper, we define a PbSOM as a SOM that consists of G neurons $\mathcal{R} = \{r_1, r_2, \ldots, r_G\}$ in a network with a neighborhood function h_{kl} that defines the strength of lateral interaction between two neurons r_k and r_l, for $k, l \in \{1, 2, \ldots, G\}$; and each neuron r_k associates with a reference model $\boldsymbol{\theta}_k$ that represents some probability distribution in the data space.

Sum *et al.* [34] interpreted Kohonen's sequential SOM learning algorithm in terms of maximizing the local correlations (coupling energies) between the neurons and their neighborhoods with the given input data. Given a data sample $\mathbf{x}_i \in \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, the local coupling energy between r_k and its neighborhood is defined as

$$\begin{aligned} E_{\mathbf{x}_i|k} &= \sum_{l=1}^{G} h_{kl} r_k(\mathbf{x}_i;\boldsymbol{\theta}_k) r_l(\mathbf{x}_i;\boldsymbol{\theta}_l) \\ &= r_k(\mathbf{x}_i;\boldsymbol{\theta}_k) \sum_{l=1}^{G} h_{kl} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l) \end{aligned} \tag{16}$$

where $r_k(\mathbf{x}_i;\boldsymbol{\theta}_k)$ denotes the response of neuron r_k to $\mathbf{x}_i$, which is modeled by an isotropic Gaussian density. Then, the coupling energy over the network for $\mathbf{x}_i$ is defined as

$$E_{\mathbf{x}_i} = \sum_{k=1}^{G} E_{\mathbf{x}_i|k} \tag{17}$$

and the energy function to be maximized is

$$C = \sum_{i=1}^{N} \log E_{\mathbf{x}_i}. \tag{18}$$

In (16), the term $\sum_{l=1}^{G} h_{kl} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)$ can be considered as the neighborhood response of r_k, where the conjunction between the neuron responses is implemented using the *summing* operation.

In this study, we express the neuron response $r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)$ as a multivariate Gaussian distribution as follows:

$$r_l(\mathbf{x}_i;\boldsymbol{\theta}_l) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_l|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_l)^T \boldsymbol{\Sigma}_l^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_l)\right) \tag{19}$$

for $l = 1, 2, \ldots, G$, and formulate the neighborhood response of r_k as

$$\prod_{l \neq k} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)^{h_{kl}} \tag{20}$$

where the conjunction between the neuron responses in the neighborhood of r_k is implemented using the *multiplicative* operation. Then, for a given $\mathbf{x}_i$, we define the local coupling energy between r_k and its neighborhood as the following coupling likelihood:

$$\begin{aligned} p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h) &= r_k(\mathbf{x}_i;\boldsymbol{\theta}_k)^{h_{kk}} \prod_{l \neq k} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)^{h_{kl}} \\ &= \prod_{l=1}^{G} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)^{h_{kl}} \\ &= \exp\left(\sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)\right) \end{aligned} \tag{21}$$

where $\boldsymbol{\Theta}$ is the set of reference models, and h denotes the given neighborhood function.[2] Then, we define the coupling likelihood of $\mathbf{x}_i$ over the network as the following (unnormalized) mixture likelihood:

$$p_s(\mathbf{x}_i;\boldsymbol{\Theta},h) = \sum_{k=1}^{G} w_s(k) p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h) \tag{22}$$

where $w_s(k)$ for $k = 1, 2, \ldots, G$ is fixed at $1/G$. Note that, theoretically, the mixture weights can be learned automatically. When maximizing the local coupling likelihood $p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h)$ for each neuron r_k, $k = 1, 2, \ldots, G$, the topological order between neuron r_k and its neighborhood for the given data sample $\mathbf{x}_i$ is learned in the learning process; therefore, we use equal mixture weights in the mixture model to take account of the topological order learning induced by the neurons faithfully (with equal prior importance). In fact, this is important for learning an ordered map. From our experimental analysis, if the mixture weights are updated in the learning process, the learning of topological order is frequently dominated by some particular mixture components, which makes it difficult to obtain an ordered map. For details, one can refer to the Appendix.

Comparing the network structure of the proposed coupling-likelihood mixture model in (22) with that of the Gaussian mixture model (GMM), as shown in Fig. 1, the proposed model inserts a coupling-likelihood layer between the Gaussian-likelihood layer and the mixture-likelihood layer to take account of the coupling between the neurons and their neighborhoods. When the neighborhood size is reduced to zero (i.e., $h_{kl} = \delta_{kl}$), the coupling-likelihood mixture model becomes a GMM with equal mixture weights.

Note that other probability distributions are possible for $r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)$ in the formulation of the coupling-likelihood mixture model, although we use the multivariate Gaussian distribution in this paper.

[2]From (21), it is obvious that, in our formulation, the coupling between r_k and its neighboring neurons is considered jointly, whereas Sum *et al.*'s formulation considers it in a pairwise manner, as shown in (16). Note that we use the term "coupling likelihood" instead of "coupling energy" for two reasons: 1) (21) is a coupling of Gaussian likelihoods; and 2) using "coupling-likelihood" can help describe the link between our proposed approaches and model-based clustering.

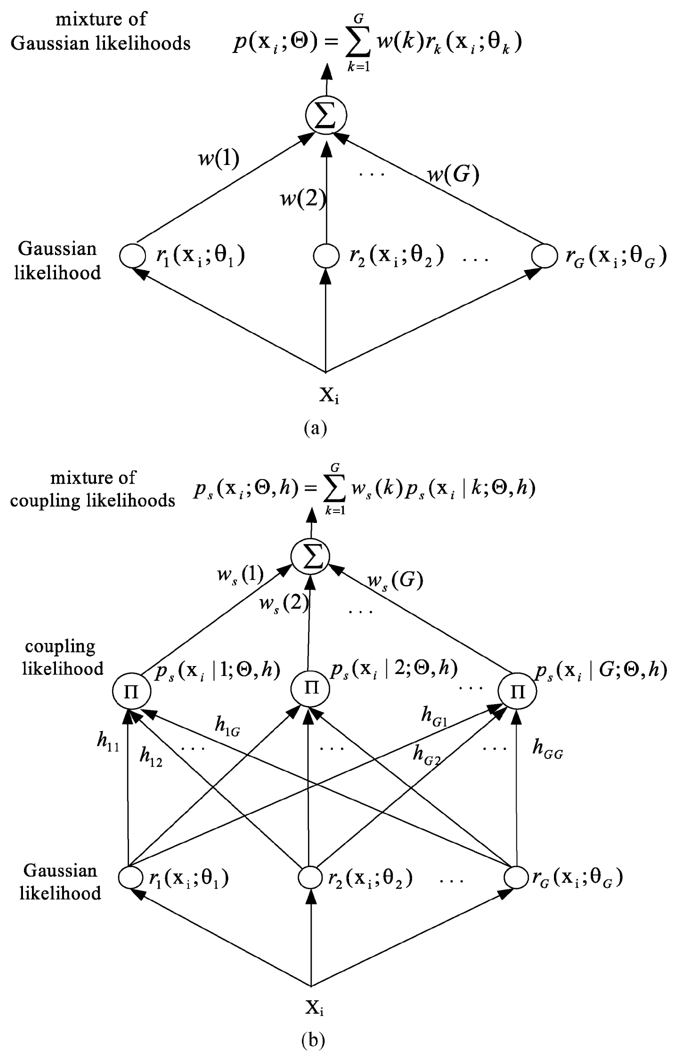
mixture of Gaussian likelihoods
$p(\mathrm{x}_i;\Theta)=\sum_{k=1}^{G} w(k) r_k(\mathrm{x}_i;\theta_k)$
$w(1)$
$w(2)$
$w(G)$
Gaussian likelihood
$r_1(\mathrm{x}_i;\theta_1)$
$r_2(\mathrm{x}_i;\theta_2)$
$r_G(\mathrm{x}_i;\theta_G)$
x_i
(a)
mixture of coupling likelihoods
$p_s(\mathrm{x}_i;\Theta,h)=\sum_{k=1}^{G} w_s(k) p_s(\mathrm{x}_i\,|\,k;\Theta,h)$
$w_s(1)$
$w_s(2)$
$w_s(G)$
coupling likelihood
$p_s(\mathrm{x}_i\,|\,1;\Theta,h)$
$p_s(\mathrm{x}_i\,|\,2;\Theta,h)$
$p_s(\mathrm{x}_i\,|\,G;\Theta,h)$
h_{11}
h_{12}
h_{1G}
h_{G1}
h_{G2}
h_{GG}
Gaussian likelihood
$r_1(\mathrm{x}_i;\theta_1)$
$r_2(\mathrm{x}_i;\theta_2)$
$r_G(\mathrm{x}_i;\theta_G)$
x_i
(b)

Fig. 1. (a) Network structure of a Gaussian mixture model, and (b) the proposed coupling-likelihood mixture model. Here, $r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)$ denotes the multivariate Gaussian distribution described in (19).

B. The SOCEM Algorithm

The self-organizing process of PbSOM can be described as a model-based data clustering procedure that preserves the spatial relationships between the clusters in a network. Based on the classification-likelihood criterion for data clustering [8], the computation of the coupling likelihood of a data sample is restricted to its winning neuron. Thus, the goal is to estimate the partition of $\mathcal{X}$, $\hat{\mathcal{P}}=\{\hat{\mathcal{P}}_1,\hat{\mathcal{P}}_2,\ldots,\hat{\mathcal{P}}_G\}$, and the set of reference models $\hat{\boldsymbol{\Theta}}$, so as to maximize the accumulated classification log likelihood over all the data samples as follows:

$$\begin{aligned}
&C_s(\mathcal{P},\boldsymbol{\Theta};\mathcal{X},h)\\
&=\sum_{k=1}^{G}\sum_{\mathbf{x}_i\in\mathcal{P}_k}\log(w_s(k)p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h))\\
&=\sum_{k=1}^{G}\sum_{\mathbf{x}_i\in\mathcal{P}_k}\log\left(w_s(k)\exp\left(\sum_{l=1}^{G}h_{kl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)\right)\right).
\end{aligned}\tag{23}$$

As $w_s(k)$ for $k=1,2,\ldots,G$ is fixed at $1/G$, the objective function can be rewritten as

$$C_s(\mathcal{P},\boldsymbol{\Theta};\mathcal{X},h)=\sum_{k=1}^{G}\sum_{\mathbf{x}_i\in\mathcal{P}_k}\sum_{l=1}^{G}h_{kl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)+\text{Const.}\tag{24}$$

Similar to the derivation of the CEM algorithm for model-based clustering in [8], the CEM algorithm for the proposed PbSOM, i.e., the SOCEM algorithm, is derived as follows.

— *E-step:* Given the current reference model set $\boldsymbol{\Theta}^{(t)}$, compute the posterior probability of each mixture component of $p_s(\mathbf{x}_i;\boldsymbol{\Theta}^{(t)},h)$ for each $\mathbf{x}_i$ as follows:

$$\begin{aligned}
\gamma_{k|i}^{(t)}&=p_s(k|\mathbf{x}_i;\boldsymbol{\Theta}^{(t)},h)=\frac{p_s(\mathbf{x}_i,k;\boldsymbol{\Theta}^{(t)},h)}{p_s(\mathbf{x}_i;\boldsymbol{\Theta}^{(t)},h)}\\
&=\frac{\exp\left(\sum\limits_{l=1}^{G}h_{kl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)})\right)}{\sum\limits_{j=1}^{G}\exp\left(\sum\limits_{l=1}^{G}h_{jl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)})\right)}
\end{aligned}\tag{25}$$

for $k=1,2,\ldots,G$, and $i=1,2,\ldots,N$.

— *C-step:* Assign each $\mathbf{x}_i$ to the cluster whose corresponding mixture component has the largest posterior probability for $\mathbf{x}_i$, i.e., $\mathbf{x}_i\in\hat{\mathcal{P}}_j^{(t)}$ if $j=\arg\max_k\gamma_{k|i}^{(t)}$.

— *M-step:* After the *C-step*, the partition of $\mathcal{X}$ (i.e., $\hat{\mathcal{P}}^{(t)}$) is formed, and the objective function C_s defined in (24) becomes

$$\begin{aligned}
&C_s(\boldsymbol{\Theta};\hat{\mathcal{P}}^{(t)},\mathcal{X},h)\\
&=\sum_{l=1}^{G}\sum_{k=1}^{G}\sum_{\mathbf{x}_i\in\hat{\mathcal{P}}_k^{(t)}}h_{kl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)+\text{Const.}
\end{aligned}\tag{26}$$

Similar to the derivation of the *M-step* of the EM algorithm for learning a Gaussian mixture model [10], we can obtain the reestimation formulas for the mean vectors and covariance matrices by substituting (19) into (26), taking the derivative of C_s with respect to individual parameters, and then setting it to zero. The reestimation formulas are as follows:

$$\boldsymbol{\mu}_l^{(t+1)}=\frac{\sum\limits_{k=1}^{G}\sum\limits_{\mathbf{x}_i\in\hat{\mathcal{P}}_k^{(t)}}h_{kl}\mathbf{x}_i}{\sum\limits_{k=1}^{G}|\hat{\mathcal{P}}_k^{(t)}|h_{kl}}\tag{27}$$

$$\boldsymbol{\Sigma}_l^{(t+1)}=\frac{\sum\limits_{k=1}^{G}\sum\limits_{\mathbf{x}_i\in\hat{\mathcal{P}}_k^{(t)}}h_{kl}(\mathbf{x}_i-\boldsymbol{\mu}_l^{(t+1)})(\mathbf{x}_i-\boldsymbol{\mu}_l^{(t+1)})^T}{\sum\limits_{k=1}^{G}|\hat{\mathcal{P}}_k^{(t)}|h_{kl}}\tag{28}$$

for $l=1,2,\ldots,G$. When the neighborhood size is reduced to zero (i.e., $h_{kl}=\delta_{kl}$), SOCEM reduces to the

TABLE I
DAEM ALGORITHM FOR LEARNING GMMs WITH EQUAL MIXTURE WEIGHTS AND SOCEM ALGORITHM

Algorithm	DAEM	SOCEM
Objective function	$F_\beta(\boldsymbol{\Theta}; \mathcal{X})$ in (13) where $p(\mathbf{x}_i, l; \boldsymbol{\Theta}) = \frac{1}{G} r_l(\mathbf{x}_i; \boldsymbol{\theta}_l)$	$\sum_{i=1}^{N} \sum_{l=1}^{G} h_{\text{win}_i l} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l) + \text{Const.}$
Posterior distribution	$f(l\|\mathbf{x}_i; \boldsymbol{\Theta}^{(t)}) = \frac{r_l(\mathbf{x}_i; \boldsymbol{\theta}_l^{(t)})^\beta}{\sum_{j=1}^{G} r_j(\mathbf{x}_i; \boldsymbol{\theta}_j^{(t)})^\beta}$ $l = 1, 2, \cdots, G$	$h_{\text{win}_i^{(t)} l} = \exp(-\frac{\\|r_{\text{win}_i^{(t)}} - r_l\\|^2}{2\sigma^2})$ $l = 1, 2, \cdots, G$
Temperature	$1/\beta$	σ
Re-estimation formulae	$\boldsymbol{\mu}_l^{(t+1)} = \frac{\sum_{i=1}^{N} f(l\|\mathbf{x}_i; \boldsymbol{\Theta}^{(t)}) \mathbf{x}_i}{\sum_{i=1}^{N} f(l\|\mathbf{x}_i; \boldsymbol{\Theta}^{(t)})}$ $l = 1, 2, \cdots, G$	$\boldsymbol{\mu}_l^{(t+1)} = \frac{\sum_{i=1}^{N} h_{\text{win}_i^{(t)} l} \mathbf{x}_i}{\sum_{i=1}^{N} h_{\text{win}_i^{(t)} l}}$ $l = 1, 2, \cdots, G$

CEM algorithm for learning GMMs with equal mixture weights.

1) SOCEM—A DA Variant of CEM for GMM: Similar to Kohonen's sequential or batch algorithm, the SOCEM algorithm is applied in two stages. First, it is applied to a large neighborhood to form an ordered map near the center of the data samples. Then, the reference models are adapted to fit the distribution of the data samples by gradually shrinking the neighborhood.

Without loss of generality, we suppose the neighborhood function is the widely adopted (unnormalized) Gaussian kernel

$$h_{kl} = \exp\left(-\frac{\|r_k - r_l\|^2}{2\sigma^2}\right) \tag{29}$$

where $\|r_k - r_l\|$ is the Euclidean distance between two neurons r_k and r_l in the SOM network. Initially, SOCEM is applied with a large σ value, which is reduced after the algorithm converges. Then, we use the new σ value and the learned parameters as the initial condition of the next learning phase. This process is repeated until the value of σ is reduced to the predefined minimum value $\sigma_{\min}$. The above shrinking of the neighborhood (reduction of the σ value) can be interpreted as an annealing process, where a large σ value corresponds to a high temperature. Table I lists the learning rules of the DAEM algorithm for learning GMMs with equal mixture weights [12] and the SOCEM algorithm. To facilitate the interpretation, we rewrite the objective function and reestimation formulae of SOCEM in (24) and (27)-(28), respectively, with the new variable win_i, which denotes the index of the winning neuron of $\mathbf{x}_i$. For simplicity, we only list the reestimation formulas of the mean vectors of the Gaussian components.

By analyzing these two algorithms carefully, one may view $h_{\text{win}_i^{(t)} l}$ as a kind of posterior probability of $\boldsymbol{\theta}_l^{(t)}$ for $\mathbf{x}_i$ in the network domain. More precisely, $\mathbf{x}_i$ is initially projected into $r_{\text{win}_i^{(t)}}$ in the network domain; then, $r_{\text{win}_i^{(t)}}$ is applied to (29) as an observation of the Gaussian kernel centered at r_l to obtain the value of $h_{\text{win}_i^{(t)} l}$. In both the DAEM and SOCEM algorithms, when the temperature ($1/\beta$ or σ) is high, the posterior distribution becomes almost uniform; hence, all the reference models will be moved to locations near the center of the data samples in this learning phase. By gradually reducing the temperature, the influence of each $\mathbf{x}_i$ becomes more localized, and the reference models gradually spread out to fit the distribution of the data samples. When the temperature approaches zero, the probabilistic assignment strategy for the data samples becomes the winners-take-all strategy, and the objective functions and learning rules of DAEM and SOCEM are equivalent to those of CEM. The major difference between DAEM and SOCEM seems to be that the posterior distribution in SOCEM is constrained by the network topology, but DAEM does not have this property.

To visualize the transition of the objective function, we show a simulation on a simple one-dimension, two-component Gaussian mixture problem in Fig. 2.[3] The training data contains 200 observations drawn from

$$\begin{aligned} p(x; \{m_1, v_1\}, \{m_2, v_2\}) &= \frac{0.3}{v_1\sqrt{2\pi}} \exp\left(\frac{-(x-m_1)^2}{2v_1^2}\right) \\ &\quad + \frac{0.7}{v_2\sqrt{2\pi}} \exp\left(\frac{-(x-m_2)^2}{2v_2^2}\right) \end{aligned} \tag{30}$$

where the Gaussian means are $(m_1, m_2) = (-5, 5)$, and the Gaussian variances are $(v_1^2, v_2^2) = (1, 1)$.[4] The PbSOM network structure is a 1×2 lattice in $[0, 1]$. The two reference models are $\boldsymbol{\theta}_1 = \{\mu_1, \Sigma_1\}$ and $\boldsymbol{\theta}_2 = \{\mu_2, \Sigma_2\}$, where $\Sigma_1 = \Sigma_2 = 1$. The objective function in (23) is calculated with different setups for (μ_1, μ_2) to form the log-likelihood surface. From Fig. 2, we observe that a larger σ for h_{kl} yields a simpler objective function for optimization. The log-likelihood surface is symmetric along $\mu_1 = \mu_2$ because of the symmetric lattice structure and equal weighting of the reference models. For the case of $\sigma = 0.6$, the log-likelihood value is close to the global maximum of the surface when both μ_1 and μ_2 are close to the center of the data (2.39 in this case). With the reduction in the value of σ, the location of (μ_1, μ_2) for the global maximum moves toward (m_1, m_2) and (m_2, m_1).

2) Relation to Kohonen's Batch Algorithm: There are two differences between the SOCEM algorithm and Kohonen's batch algorithm. First, SOCEM considers the neighborhood information when selecting the winning neuron, but Kohonen's algorithm does not. Second, SOCEM extends the reference vectors in Kohonen's algorithm with multivariate Gaussians. In other words, if we set $\gamma_{k|i}^{(t)}$ in SOCEM to $r_k(\mathbf{x}_i; \boldsymbol{\theta}_k^{(t)}) / \sum_{j=1}^{G} r_j(\mathbf{x}_i; \boldsymbol{\theta}_j^{(t)})$, instead of the setting in (25),

[3]Visualization of how deterministic annealing EM/CEM works for function optimization is illustrated in detail in [12].

[4]The data is generated using the function gmmsamp.m in Netlab software from http://www.ncrg.aston.ac.uk/netlab/

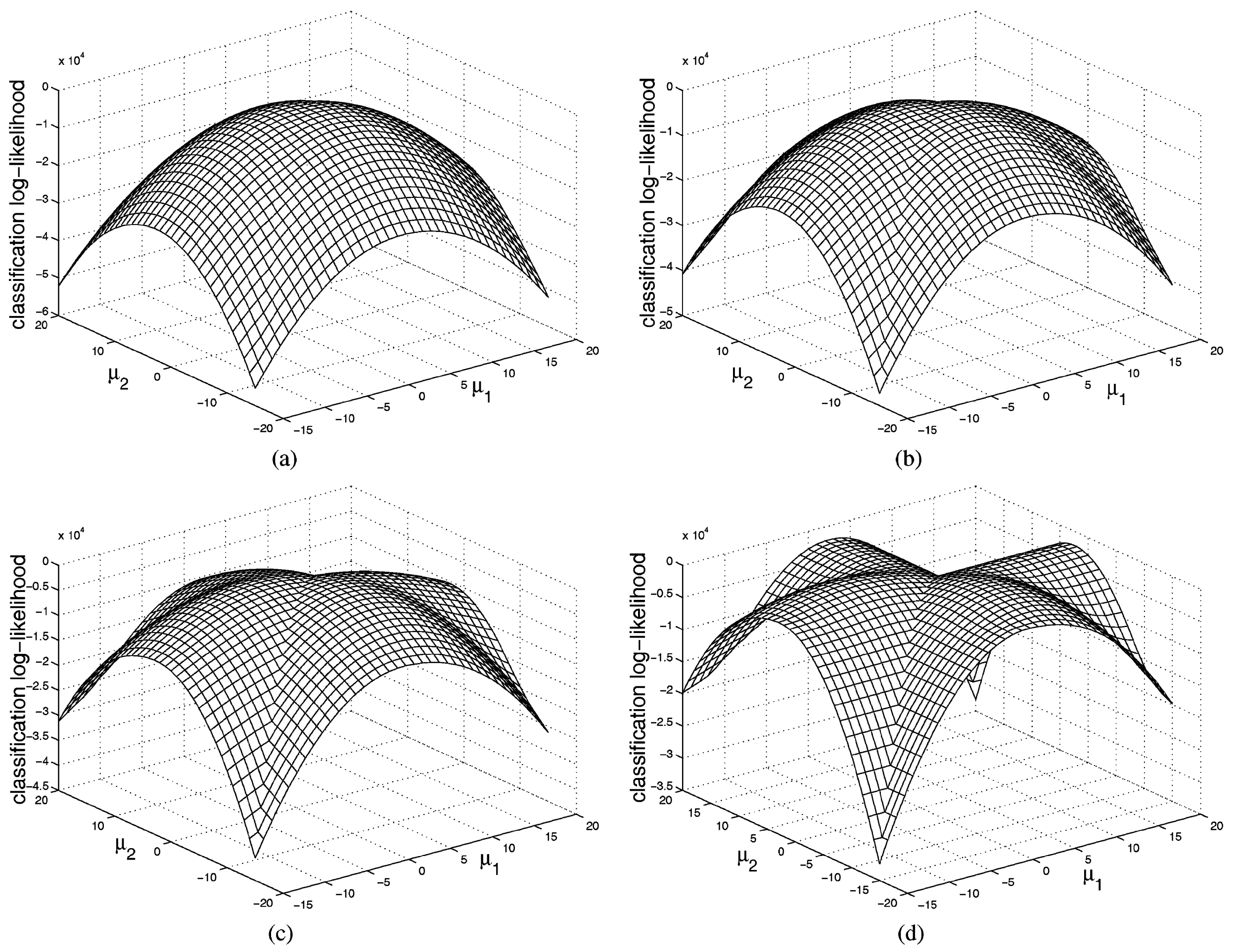

Fig. 2. SOCEM's objective function becomes more complex with the reduction of neighborhood size (σ in h_{kl}). (a) $\sigma = 0.6$. (b) $\sigma = 0.4$. (c) $\sigma = 0.3$. (d) $\sigma = 0$ (i.e., $h_{kl} = \delta_{kl}$).

we obtain a probabilistic variant of Kohonen's batch algorithm (denoted as KohonenGaussian), where Kohonen's winner selection strategy is applied and the reference vectors are replaced with multivariate Gaussians. Thus, we may view KohonenGaussian as an approximate implementation of SOCEM that optimizes SOCEM's objective function. Moreover, if we set the covariance matrices in KohonenGaussian to be diagonal with small, identical variances, KohonenGaussian is equivalent to Kohonen's batch algorithm. Therefore, we can interpret the neighborhood shrinking of Kohonen's algorithms as a deterministic annealing process, and thereby explain why they need to start with a large neighborhood size.

Recently, Zhong and Ghosh [3] interpreted the neighborhood size of the SOM algorithms that apply Kohonen's winner selection strategy as a temperature parameter in a deterministic annealing process. However, their interpretations were not based on the optimization of an objective function, which is the essential part of DA-based optimization. In contrast, in SOCEM, the neighborhood shrinking leads to the transition of the objective function from a simpler one to a more complex one, as illustrated in Fig. 2.

3) Computational Cost: It is clear from Table I that the computational cost of DAEM is $O(GNM)$, where G, N, and M are the numbers of reference models, data samples, and learning iterations, respectively. Compared to DAEM, SOCEM needs additional $O(G^2N)$ multiplication and addition operations for winner selection in each iteration, while KohonenGaussian needs additional $O(GN)$ multiplications and additions.

C. The SOEM Algorithm

As is obvious from (23), in the formulation of the objective function of the SOCEM algorithm, only the local coupling likelihoods associated with the winning neurons are considered. Alternatively, we can compute the coupling likelihood of $\mathbf{x}_i$ using the mixture likelihood defined in (22) and apply the EM algorithm to maximize the objective log-likelihood function

$$L_s(\boldsymbol{\Theta}; \mathcal{X}, h) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{G} w_s(k) p_s(\mathbf{x}_i | k; \boldsymbol{\Theta}, h) \right). \quad (31)$$

The steps of the EM algorithm for the proposed PbSOM, i.e., the SOEM algorithm, are as follows.

— *E-step:* With the mixture model in (22), we form the auxiliary function as

$$Q_s(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)}) = \sum_{i=1}^{N} \sum_{k=1}^{G} \gamma_{k|i}^{(t)} \log p_s(\mathbf{x}_i, k; \boldsymbol{\Theta}, h) \quad (32)$$

where $\gamma_{k|i}^{(t)}$ is the same as (25). Since $p_s(\mathbf{x}_i, k; \mathbf{\Theta}, h) = w_s(k)p_s(\mathbf{x}_i|k; \mathbf{\Theta}, h)$, (32) can be rewritten as

$$Q_s(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) = \sum_{i=1}^{N}\sum_{k=1}^{G} \gamma_{k|i}^{(t)} \log(w_s(k)p_s(\mathbf{x}_i|k; \mathbf{\Theta}, h)). \tag{33}$$

As $w_s(k)$ for $k = 1, 2, \ldots, G$ is fixed at $1/G$, by substituting (21) into (33), the auxiliary function can be rewritten as

$$\begin{aligned} Q_s(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) &= \sum_{i=1}^{N}\sum_{k=1}^{G} \gamma_{k|i}^{(t)} \sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l) + \text{Const.} \\ &= \sum_{l=1}^{G}\sum_{i=1}^{N}\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l) + \text{Const.} \end{aligned} \tag{34}$$

— *M-step:* By replacing the response $r_l(\mathbf{x}_i; \boldsymbol{\theta}_l)$ in (34) with the multivariate Gaussian density in (19) and setting the derivative of Q_s with respect to individual mean vectors and covariance matrices to zero, we obtain the following reestimation formulas:

$$\boldsymbol{\mu}_l^{(t+1)} = \frac{\sum_{i=1}^{N}\left(\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}\right)\mathbf{x}_i}{\sum_{i=1}^{N}\left(\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}\right)} \tag{35}$$

$$\mathbf{\Sigma}_l^{(t+1)} = \frac{\sum_{i=1}^{N}\left(\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}\right)(\mathbf{x}_i - \boldsymbol{\mu}_l^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_l^{(t+1)})^T}{\sum_{i=1}^{N}\left(\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}\right)} \tag{36}$$

for $l = 1, 2, \ldots, G$. When the neighborhood size is reduced to zero (i.e., $h_{kl} = \delta_{kl}$), SOEM reduces to the EM algorithm for learning GMMs with equal mixture weights.

There are two major differences between the SOCEM and SOEM algorithms. First, they learn maps based on the classification-likelihood criterion and the mixture-likelihood criterion, respectively. Second, SOEM adapts the reference models in a more global way than SOCEM. To explain this perspective, we can consider the learning of SOCEM and SOEM in the sense of sequential learning. As illustrated in Fig. 3, in the SOCEM algorithm [cf., (27) and (28)], each data sample $\mathbf{x}_i$ only contributes to the adaptation of the winning reference model and its neighborhood (i.e., $\mathbf{x}_i$ only contributes to the learning of the topological order between the winning reference model and its neighborhood). However, in the SOEM algorithm [cf.. (35) and (36)], each data sample $\mathbf{x}_i$ contributes proportionally to the adaptation of each reference model and its neighborhood according to the posterior probabilities $\gamma_{k|i}^{(t)}$ for $k = 1, 2, \ldots, G$.

1) SOEM—A DA Variant of EM for GMM: As with the SOCEM algorithm, we can apply SOEM to a large neighborhood and obtain different map configurations by gradually reducing the neighborhood size. The term $\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}$ in (35) and (36) can be considered as a kind of posterior probability $\pi(l|\mathbf{x}_i; \mathbf{\Theta}^{(t)}, h)$ of the reference model $\boldsymbol{\theta}_l^{(t)}$ for $\mathbf{x}_i$, which is also constrained by the neighborhood function. With a large σ value in h_{kl} [see (29)], $\pi(l|\mathbf{x}_i; \mathbf{\Theta}^{(t)}, h)$ for $l = 1, 2, \ldots, G$, will be nearly a uniform distribution due to the small variation in the values of $\gamma_{k|i}^{(t)}$ for $k = 1, 2, \ldots, G$, and the small variation in the values of h_{kl} for $k = 1, 2, \ldots, G$, for each case of l. Hence, all the reference models will be moved to locations near the center of the data samples. When the neighborhood size is reduced to zero (i.e., $h_{kl} = \delta_{kl}$), the SOEM algorithm becomes the EM algorithm for learning GMMs with equal mixture weights. As with the annealing interpretation of SOCEM, SOEM can be viewed as a topology-constrained deterministic annealing variant of the EM algorithm for learning GMMs with equal mixture weights.[5]

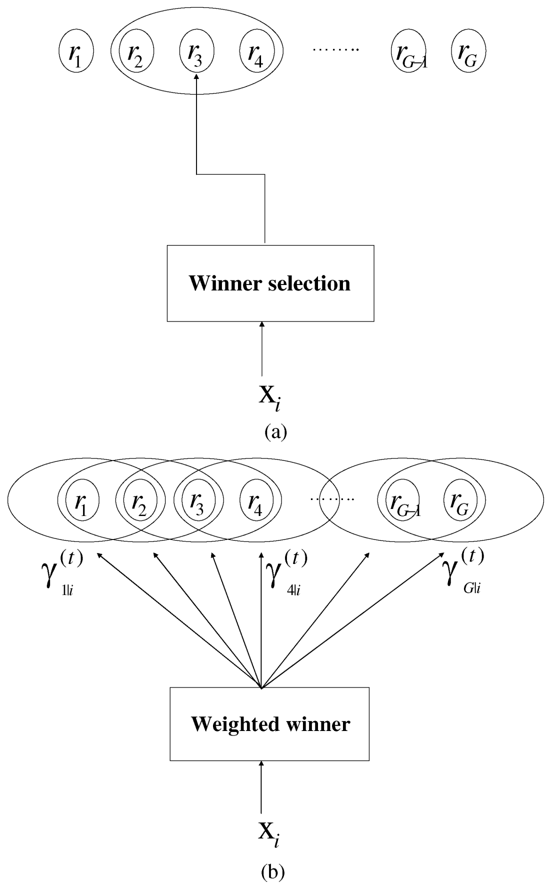

Fig. 3. For each data sample $\mathbf{x}_i$, the adaptation of the reference models in SOCEM is restricted to the winning reference model and its neighborhood. However, in SOEM, the winner is relaxed to the weighted winners by the posterior probabilities $\gamma_{k|i}^{(t)}$, for $k = 1, 2, \ldots, G$. Each data sample $\mathbf{x}_i$ contributes proportionally to the adaptation of each reference model and its neighborhood according to the posterior probabilities. (a) SOCEM. (b) SOEM.

2) Computational Cost: Comparing (34)–(36) to (26)–(28), we can see that, in each learning iteration, SOEM and SOCEM have a similar computational cost in the *E-step*, but the former needs additional $O(GN)$ multiplication and addition operations for updating the model parameters in the *M-step*.

[5]SOEM yielded a similar result on the one-dimension, two-component Gaussian mixture problem in Fig. 2; however, we do not present it here to avoid redundancy.

D. The SODAEM Algorithm

Similar to the derivation of the DAEM algorithm for learning GMMs [12], we developed a DAEM algorithm for the proposed PbSOM, called the SODAEM algorithm. With the mixture likelihood defined in (22), DAEM first derives the posterior density in the *E-step* using the principle of maximum entropy. Following the derivation of the posterior probability in [12] with the current model's parameter set $\mathbf{\Theta}^{(t)}$, we obtain the posterior probability of the kth mixture component for $\mathbf{x}_i$ as follows:

$$\tau_{k|i}^{(t)} = \frac{p_s(\mathbf{x}_i|k;\mathbf{\Theta}^{(t)},h)^\beta}{\sum_{j=1}^{G} p_s(\mathbf{x}_i|j;\mathbf{\Theta}^{(t)},h)^\beta} = \frac{\exp\left(\beta\sum_{l=1}^{G} h_{kl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)})\right)}{\sum_{j=1}^{G}\exp\left(\beta\sum_{l=1}^{G} h_{jl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)})\right)}. \tag{37}$$

Then, the auxiliary function to be minimized is

$$U_{s\beta}(\mathbf{\Theta};\mathbf{\Theta}^{(t)}) = -\sum_{i=1}^{N}\sum_{k=1}^{G}\tau_{k|i}^{(t)}\log p_s(\mathbf{x}_i,k;\mathbf{\Theta},h) \tag{38}$$

and the reestimation formulas for the mean vectors and covariance matrices are

$$\boldsymbol{\mu}_l^{(t+1)} = \frac{\sum_{i=1}^{N}\left(\sum_{k=1}^{G}\tau_{k|i}^{(t)}h_{kl}\right)\mathbf{x}_i}{\sum_{i=1}^{N}\left(\sum_{k=1}^{G}\tau_{k|i}^{(t)}h_{kl}\right)} \tag{39}$$

$$\mathbf{\Sigma}_l^{(t+1)} = \frac{\sum_{i=1}^{N}\left(\sum_{k=1}^{G}\tau_{k|i}^{(t)}h_{kl}\right)(\mathbf{x}_i-\boldsymbol{\mu}_l^{(t+1)})(\mathbf{x}_i-\boldsymbol{\mu}_l^{(t+1)})^T}{\sum_{i=1}^{N}\left(\sum_{k=1}^{G}\tau_{k|i}^{(t)}h_{kl}\right)} \tag{40}$$

for $l = 1,2,\ldots,G$.

Note that the reestimation formulas for SODAEM are the same as those for SOEM, except that $\gamma_{k|i}^{(t)}$ is replaced by $\tau_{k|i}^{(t)}$. $1/\beta$ corresponds to the temperature that controls the annealing process, in which a high temperature is applied initially. Then, the system is cooled down by gradually reducing the temperature. When $1/\beta \to 1$, the SODAEM algorithm becomes the SOEM algorithm; however, when $1/\beta \to 0$, it is equivalent to the SOCEM algorithm. In other words, SODAEM can be viewed as a deterministic annealing variant of SOEM and SOCEM.

By considering certain cases and approximations of SODAEM, SOEM, and SOCEM, we summarize the family of EM-based approaches for Gaussian model-based clustering discussed in this section in Fig. 4. Both EM under the mixture-likelihood criterion and CEM under the classification-likelihood criterion are widely used model-based data clustering methods. SOEM (SOCEM) can be applied instead of EM (CEM) in model-based clustering if we want to preserve the spatial relationships between the resulting data clusters on a network. Since SODAEM is a DA variant of SOEM and SOCEM, it can be applied in model-based data clustering under both mixture-likelihood and classification-likelihood criteria.

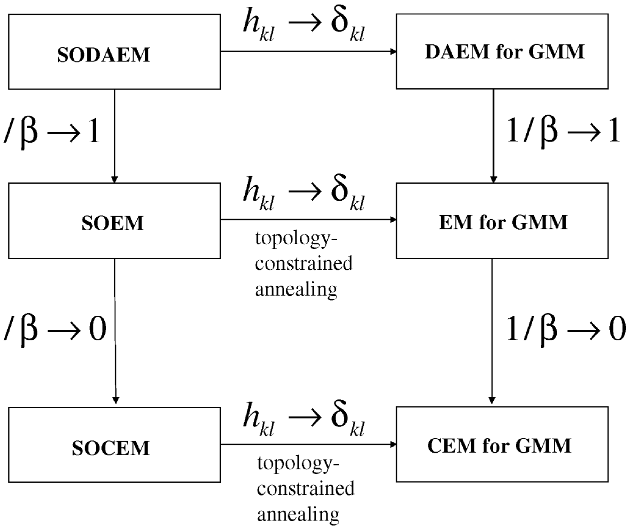

Fig. 4. Family of Gaussian model-based clustering algorithms derived from the SODAEM, SOEM, and SOCEM algorithms. $\delta_{kl} = 1$ if $k = l$; otherwise, $\delta_{kl} = 0$.

1) Computational Cost: Comparing (39)-(40) to (35)-(36), we can see that SODAEM and SOEM have similar computational costs in each learning iteration.

IV. Experimental Results

A. Experiments on the Organizing Property

Data Set Description: We conducted experiments on two types of data: a synthetic data set and a real-world data set. The synthetic data set consisted of 500 points uniformly distributed in a unit square. For the real-world data set, we used the training set of class "0" in the "Pen-Based Recognition of Handwritten Digits" database (denoted as PenRecDigits_C0) in the UCI Machine Learning Database Repository [35]. The data set consists of 802 16-dimensional vectors. To demonstrate the map-learning process, we used the first two dimensions of the feature vectors as data for simulations. As a preprocessing step, we scaled down each element of the vectors in PenRecDigits_C0 to 1/100 of its original value to avoid numerical traps.

Experimental Setup: In the experiments, an 8×8 equally spaced square lattice in a unit square was used as the structure of the SOM network. For the neighborhood function, we used the Gaussian kernel h_{kl} in (29).

We evaluated SOCEM, SOEM, SODAEM, and Kohonen-Gaussian (Kohonen's batch algorithm that uses Gaussian reference models) in 20 independent random initialization trials and two setups for σ in h_{kl}. For each trial, data samples were randomly selected from the data set as the initial mean vectors, $\mu'_1, \mu'_2, \ldots, \mu'_G$, of the reference models, which were multivariate Gaussians with full covariance matrices. The initial covariance matrix Σ'_l was set as $\rho_l\mathbf{I}$, where $\rho_l = \min_{k\neq l}\{\|\mu'_l - \mu'_k\|\}$, for $l = 1,2,\ldots,G$. To avoid the singularity problem, we applied the variance limiting step to the covariance matrices during the learning process. If the value of any element of the covariance matrix was less than 0.001, it was set at 0.001.

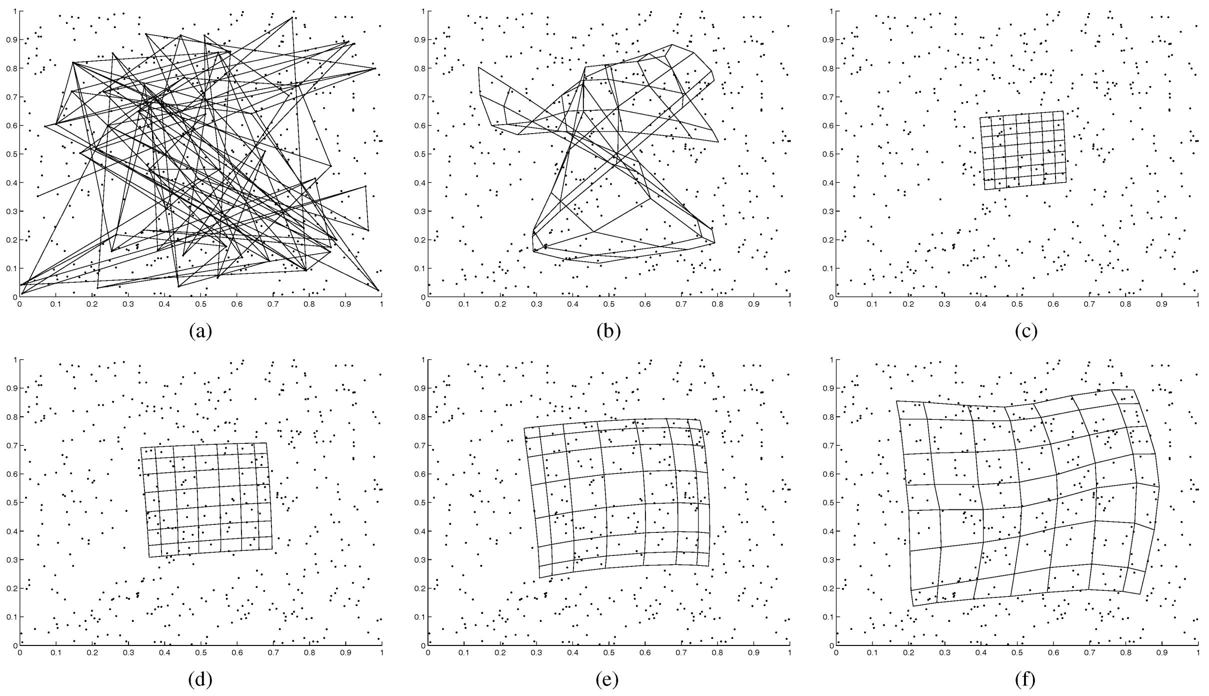
Fig. 5. Map-learning process obtained by running the SOCEM algorithm on the synthetic data. (a) and (b) Simulation 1: When SOCEM is run with the random initialization in (a) and $\sigma = 0.15$, it converges to the unordered map in (b). (a) and (c)-(f) Simulation 2: SOCEM starts with $\sigma = 0.6$ and the random initialization in (a). Then, the value of σ is reduced to 0.15 in 0.15 decrements. (a) Random initialization. (b) $\sigma = 0.15$ with random initialization. (c) $\sigma = 0.6$ with random initialization. (d) $\sigma = 0.45$. (e) $\sigma = 0.3$. (f) $\sigma = 0.15$.

1) Results Using the Synthetic Data: We first demonstrate the map-learning processes of SOCEM, SOEM, and SODAEM using one of the 20 random initializations by showing the configurations of the Gaussian means on the maps, and then summarize the overall results of all the initializations.

Simulations Using SOCEM: Fig. 5 shows two simulations using the SOCEM algorithm. In the first simulation, SOCEM is run with the random initialization in Fig. 5(a) and a fixed σ of 0.15 in h_{kl}. As shown in Fig. 5(b), the algorithm's learning converges to an unordered map. In the second simulation, SOCEM starts with the same random initialization as that in Fig. 5(a), but with a larger σ of 0.6. When it converges at the current σ value, σ is reduced by 0.15. Then, the algorithm is applied again with the new σ value and the reference models obtained in the previous phase. This process continues until SOCEM converges at $\sigma = 0.15$. Fig. 5(c)–(f) depicts the maps obtained when $\sigma = 0.6, 0.45, 0.3,$ and 0.15, respectively. We can explain the second simulation in terms of *annealing* (cf., Section III-B1). When using SOCEM, we start with a larger σ value (a higher temperature) so that the objective function is simple enough to be optimized. Then, we obtain the target map configuration by gradually reducing the value of σ (the temperature). Though the reduction in σ produces a more complex objective function for optimization, SOCEM can still learn well because the reference models obtained at the larger σ value provide a sound initialization for the next learning phase at the smaller σ value.

Simulations Using SOEM: We conducted two similar simulations using the SOEM algorithm. In the first simulation, SOEM was run with the random initialization in Fig. 6(a) [the same as that in Fig. 5(a)] and a fixed σ of 0.15. As shown in Fig. 6(b), the learning of SOEM converged to an unordered map. In the second simulation, SOEM started with the random initialization in Fig. 6(a) and a larger σ of 0.6. Then, the value of σ was gradually reduced to 0.15 in 0.15 decrements. Fig. 6(c)–(f) depicts the maps obtained when SOEM converges at $\sigma = 0.6, 0.45, 0.3,$ and 0.15, respectively. Similar to SOCEM, we can interpret the reduction of σ in SOEM as an annealing process (cf., Section III-C1), which overcomes the initialization issue. Comparing Fig. 6(c)-(d) to Fig. 5(c)-(d), we observe that the map obtained by SOEM is more concentrated than that obtained by SOCEM for the same σ value. This may be because SOEM learns the map in a more global manner than SOCEM, as noted in Section III-C. In other words, each data sample contributes to all the neurons in a more global manner in SOEM than in SOCEM.

Simulations Using SODAEM: Fig. 7 depicts the simulations using the SODAEM algorithm with the same random initialization as that in Figs. 5(a) and 6(a). The value of σ is also fixed at 0.15, and the initial value of β is set to 0.16. When SODAEM converges at a β value, it is applied again with $\beta^{\text{new}} = \beta \times 1.6$ and the reference models obtained in the previous phase. We stop the learning process at $\beta = 17.592$. In our experience, it is appropriate to set the maximum value of β within the range 10–20 for practical applications. When $\beta = 0.16$, the temperature is high enough to ensure a smooth objective function. Therefore, according to the parameter update rules of SODAEM, the reference models form a compact ordered map via lateral interactions near the center of the data samples, even though the neighborhood size is small ($\sigma = 0.15$ in this case). When $\beta = 1.04$ and 17.592, SODAEM is almost equivalent to SOEM and SOCEM, respectively. In these two cases, SODAEM converges to the ordered maps in Fig. 7(f) and (i), re-

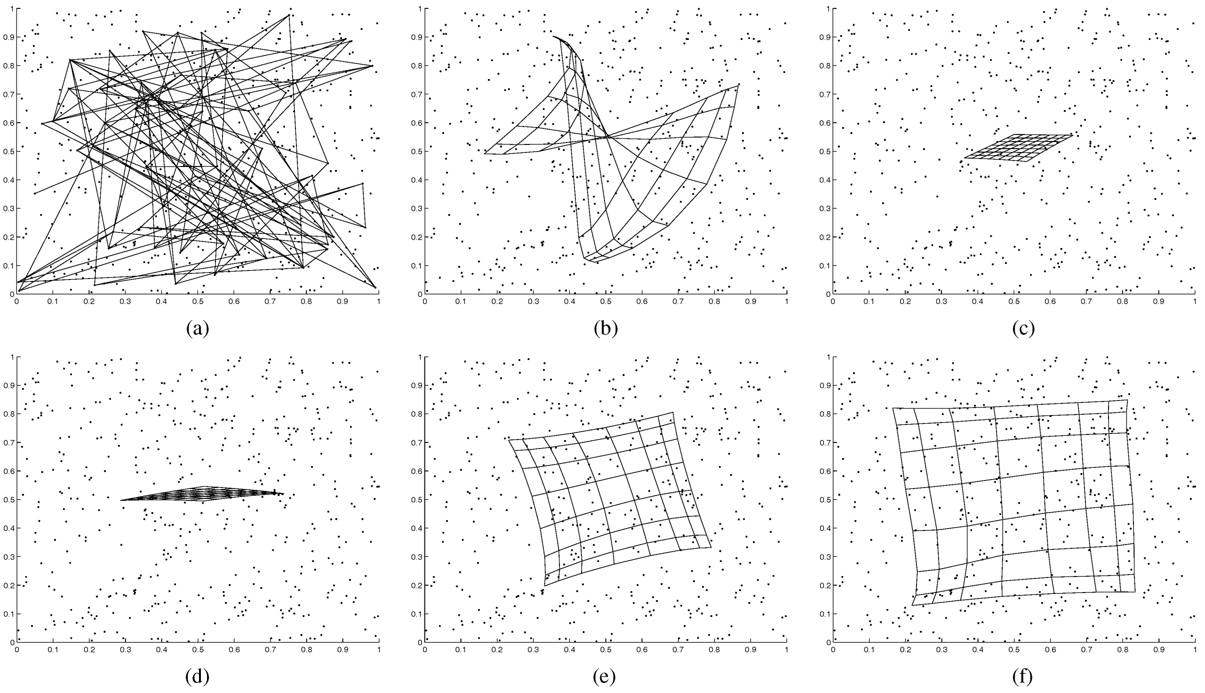
Fig. 6. Map-learning process obtained by running the SOEM algorithm on the synthetic data. (a) and (b) Simulation 1: When SOEM is run with the random initialization in (a) and $\sigma = 0.15$, it converges to the unordered map in (b). (a) and (c)-(f) Simulation 2: SOEM starts with $\sigma = 0.6$ and the random initialization in (a). Then, the value of σ is reduced to 0.15 in 0.15 decrements. (a) Random initialization. (b) $\sigma = 0.15$ with random initialization. (c) $\sigma = 0.6$ with random initialization. (d) $\sigma = 0.45$. (e) $\sigma = 0.3$. (f) $\sigma = 0.15$.

TABLE II
RESULTS OF SIMULATIONS USING KOHONENGAUSSIAN, SOCEM, SOEM, AND SODAEM IN 20 INDEPENDENT RANDOM INITIALIZATION TRIALS ON THE SYNTHETIC DATA. THE ALGORITHMS WERE RUN WITH TWO SETUPS FOR σ IN h_{kl}. WHEN $\sigma = 0.15$, KOHONENGAUSSIAN SUCCEEDED IN CONVERGING TO AN ORDERED MAP IN ONE RANDOM INITIALIZATION CASE (S:1), BUT FAILED IN THE REMAINING CASES (F:19)

Setup for σ	$\sigma = 0.15$	$\sigma = 0.6$ initially, and is reduced to 0.15 in 0.15 decrements
KohonenGaussian	S:1	S:20
	F:19	F:0
SOCEM	S:1	S:20
	F:19	F:0
SOEM	S:15	S:20
	F:5	F:0
SODAEM	S:20	-
	F:0	-

spectively. However, as shown in Figs. 5(a)-(b) and 6(a)-(b), SOCEM and SOEM do not converge to an ordered map when $\sigma = 0.15$, which demonstrates that the annealing process of SODAEM overcomes the initialization problem of SOCEM and SOEM when $\sigma = 0.15$. Note that SODAEM may not be able to obtain any ordered map during the annealing process if the value of σ is too small to form an ordered map at a small β value.

Discussion: The experiment results obtained by the three proposed algorithms and KohonenGaussian for the 20 random initializations are summarized in Table II. Several conclusions can be drawn from the results. First, SOEM often converges to an ordered map even at a small, fixed σ value ($\sigma = 0.15$ in the experiments), but KohonenGaussian and SOCEM seldom do so. This may be because SOEM learns the map in a more global way, as noted in Section III-C; hence, it is less sensitive to the initialization of the parameters when σ is small. The results for KohonenGaussian and SOCEM are similar. This may be because they only differ in the winner selection strategy. Second, the initialization issue of KohonenGaussian, SOCEM, and SOEM can be overcome by using a larger σ value (0.6 in the experiments) initially, and then gradually reducing the value to the target σ value (0.15 in the experiments). The reduction of σ can be interpreted as an annealing process (cf., Sections III-B1, III-B2, and III-C1). Third, the experiment results show that SODAEM overcomes the initialization issue of SOCEM and SOEM at a small σ value (0.15 in the experiments) using the annealing process, which is controlled by the temperature parameter β.

2) Results Using PenRecDigits_C0: We also conducted experiments on real-world data using the setups for the neighborhood function described in Section IV-A1. Table III summarizes the results obtained by the four PbSOM learning algorithms. From the results, we can draw the same conclusions as those made for the experiment results on the synthetic data. Figs. 8–10 demonstrate, respectively, the map-learning processes of SOCEM, SOEM, and SODAEM using one of the 20 random initializations. Comparing Figs. 8–10, we observe that these three algorithms obtain rather different results. SOCEM and SOEM usually obtain different maps because they learn the maps based on different clustering criteria (classification likelihood versus mixture likelihood). SODAEM and SOEM (or SOCEM) usually obtain different results because SODAEM's annealing is achieved by increasing the β value, while SOEM's (or SOCEM's) annealing is achieved by decreasing the σ value. Comparing Figs. 9(f) and 10(f), although SODAEM becomes equivalent to SOEM when the value of β is increased to 1.04,

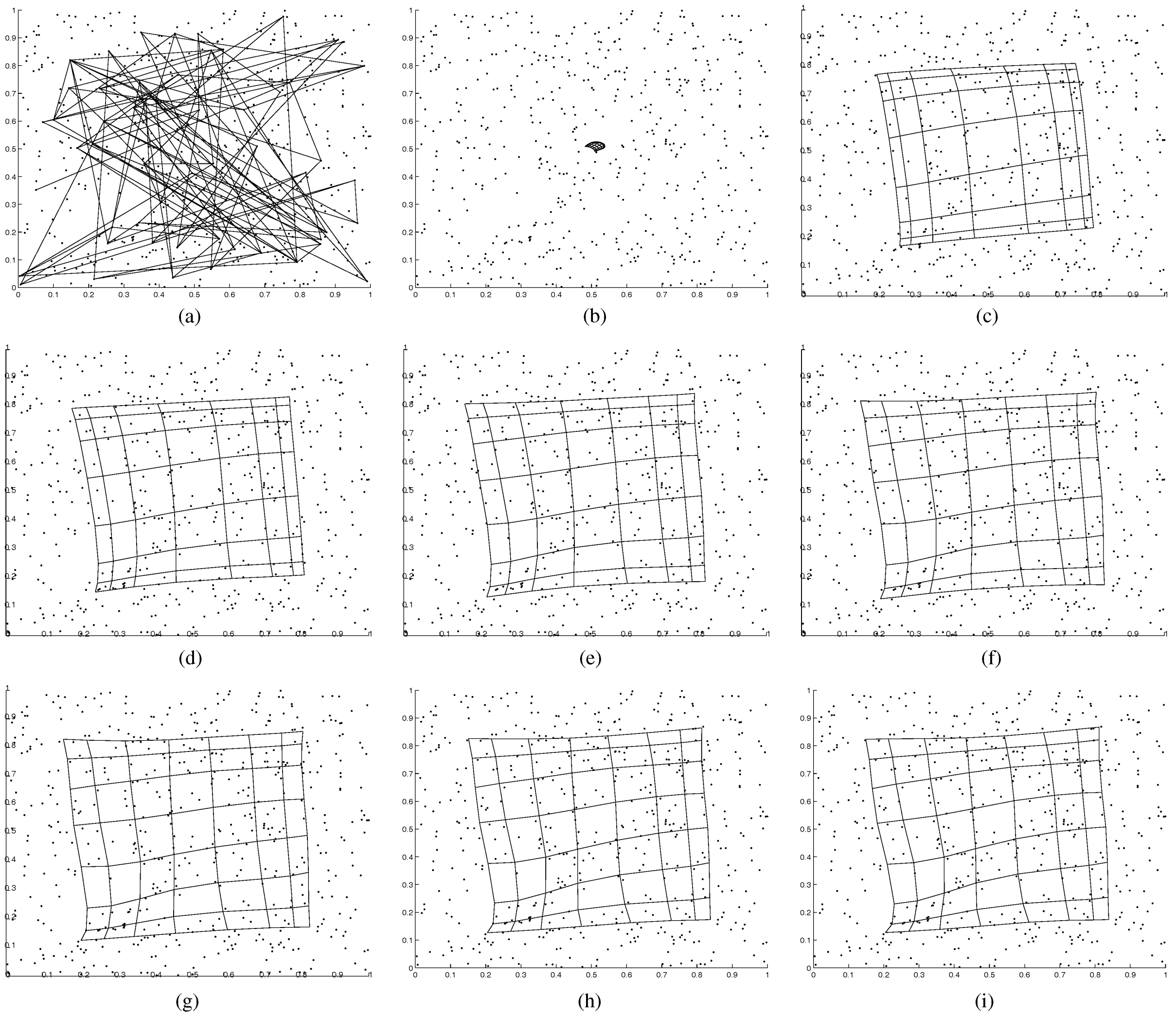

Fig. 7. Map-learning process obtained by running the SODAEM algorithm on the synthetic data. The value of σ is fixed at 0.15, while value of β is initialized at 0.16 and increased in multiples of 1.6 up to 17.592. (a) Random initialization. (b) $\sigma = 0.15, \beta = 0.16$. (c) $\sigma = 0.15, \beta = 0.256$. (d) $\sigma = 0.15, \beta = 0.409$. (e) $\sigma = 0.15, \beta = 0.655$. (f) $\sigma = 0.15, \beta = 1.04$. (g) $\sigma = 0.15, \beta = 2.68$. (h) $\sigma = 0.15, \beta = 6.871$. (i) $\sigma = 0.15, \beta = 17.592$.

their search paths on the objective function surface are different because they have rather different seed models [Fig. 10(e) versus Fig. 9(e)]. Therefore, they converge to different local maxima of the objective function and obtain different maps. Likewise, although SODAEM becomes equivalent to SOCEM when the value of β is increased to 17.592, they converge to different local maxima of the objective function and obtain different maps [Fig. 10(i) versus Fig. 8(f)].

B. Experiments to Evaluate the Performance of Data Clustering and Visualization

Data Set Description: In this section, we evaluate the performance of data clustering and visualization of the proposed algorithms on two data sets from the UCI Machine Learning Database Repository [35]: the test set of the "image segmentation" database (denoted as ImgSeg), which consists of 2100 19-dimensional feature vectors, and the Ecoli data set (denoted as Ecoli), which consists of 336 8-dimensional feature vectors. Here, we used the full vector, rather than only two dimensions, in the experiments. As a preprocessing step, we scaled down each element of the data vectors in ImgSeg to 1/100 of its original value to avoid numerical traps.

Experimental Setup: To avoid the singularity problem that often occurs when using CEM or EM to learn full covariance GMMs, we used diagonal covariance Gaussians in the experiments. We also applied the variance limiting step, in which the minimum value for a variance was set at 0.01.

For the PbSOM learning algorithms, we used five configurations for the network structure; they are 3×3, 4×4, 5×5, 6×6, and 7×7 lattices equally spaced in a unit square. We used the Gaussian kernel h_{kl} in (29) as the neighborhood function.

To avoid ambiguity, when the DAEM and SODAEM algorithms are applied in data clustering based on the classification-likelihood criterion, they are denoted as DAEM_C and SODAEM_C; and they are denoted as DAEM_M and SODAEM_M when applied in data clustering based on the mixture-likelihood criterion.

All the algorithms discussed here were run with random initializations generated in the same way described in Section IV-A.

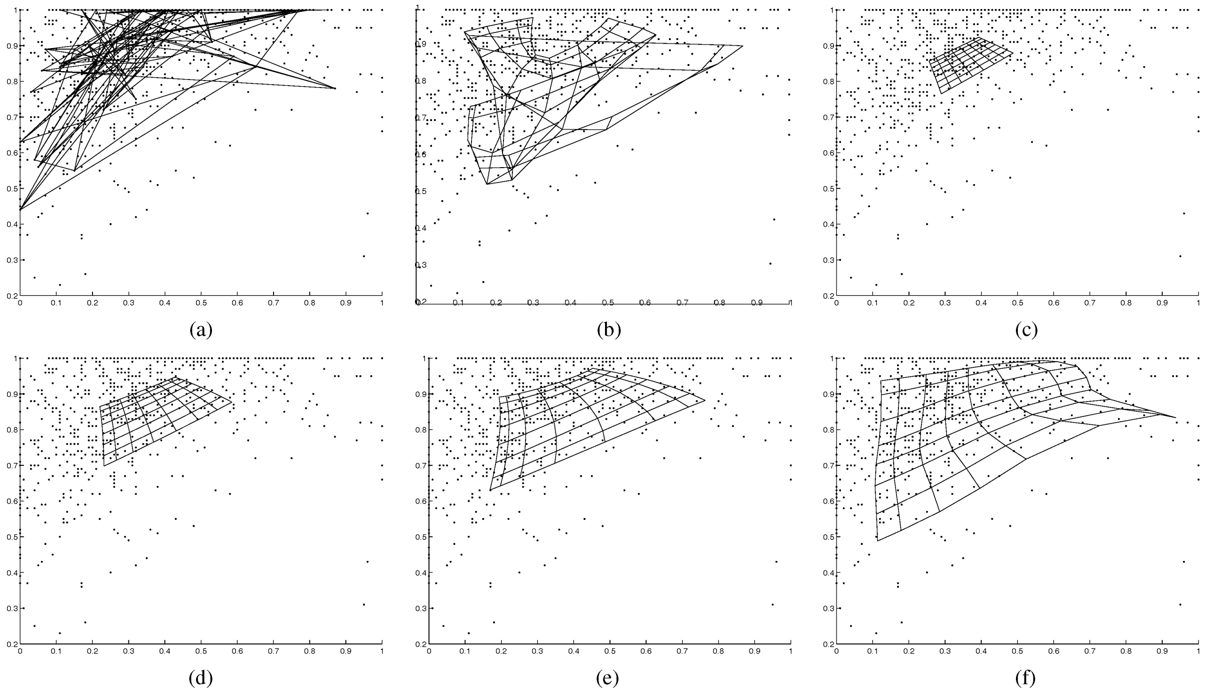

Fig. 8. Map-learning process obtained by running the SOCEM algorithm on PenRecDigits_C0. (a) and (b) Simulation 1: When SOCEM is run with the random initialization in (a) and $\sigma = 0.15$, it converges to the unordered map in (b). (a) and (c)-(f) Simulation 2: SOCEM starts with $\sigma = 0.6$ and the random initialization in (a). Then, the value of σ is reduced to 0.15 in 0.15 decrements. (a) Random initialization. (b) $\sigma = 0.15$ with random initialization. (c) $\sigma = 0.6$ with random initialization. (d) $\sigma = 0.45$. (e) $\sigma = 0.3$. (f) $\sigma = 0.15$.

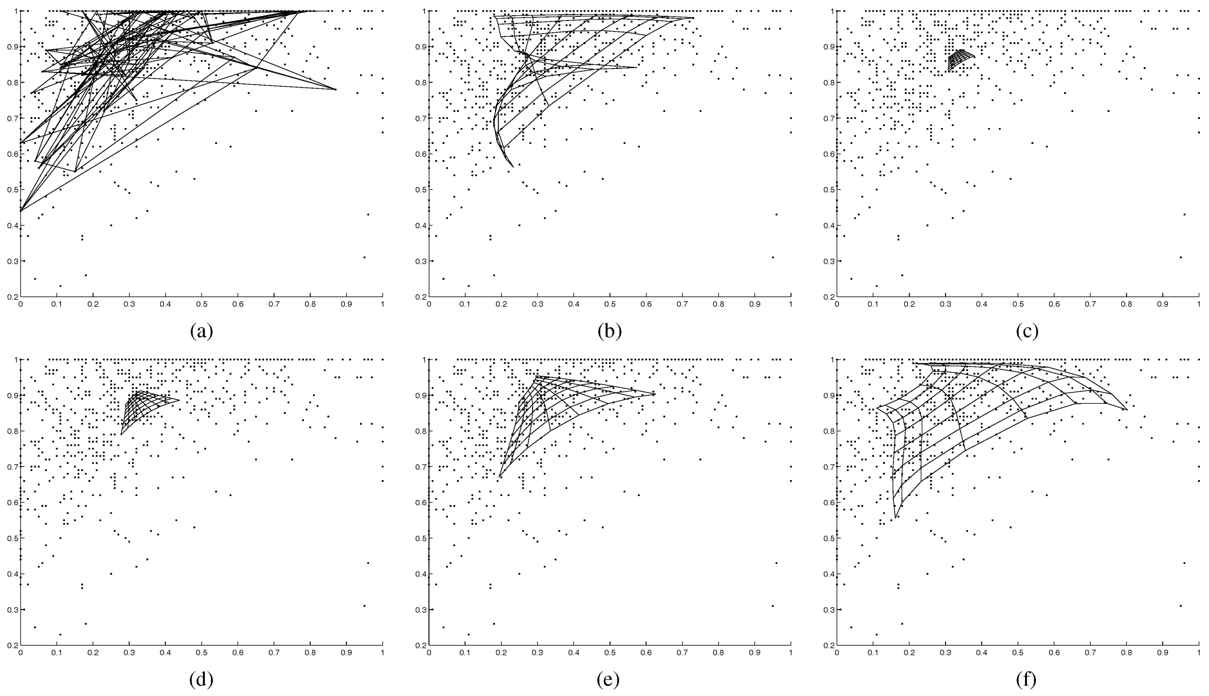

Fig. 9. Map-learning process obtained by running the SOEM algorithm on PenRecDigits_C0. (a) and (b) Simulation 1: When SOEM is run with the random initialization in (a) and $\sigma = 0.15$, it converges to the unordered map in (b). (a) and (c)-(f) Simulation 2: SOEM starts with $\sigma = 0.6$ and the random initialization in (a). Then, the value of σ is reduced to 0.15 in 0.15 decrements. (a) Random initialization. (b) $\sigma = 0.15$ with random initialization. (c) $\sigma = 0.6$ with random initialization. (d) $\sigma = 0.45$. (e) $\sigma = 0.3$. (f) $\sigma = 0.15$.

1) Experiments on ImgSeg by Using SOCEM and SODAEM_C: First, we evaluated the data clustering performance of KohonenGaussian, SOCEM, and SODAEM_C in terms of the classification log likelihood defined in (7). The performance was compared with that of CEM and DAEM_C. The setting for each algorithm was as follows.

- DAEM_C: The value of β was set at 0.2 initially, and increased to 10 by the formula $\beta^{\text{new}} = \beta \times 1.2$.

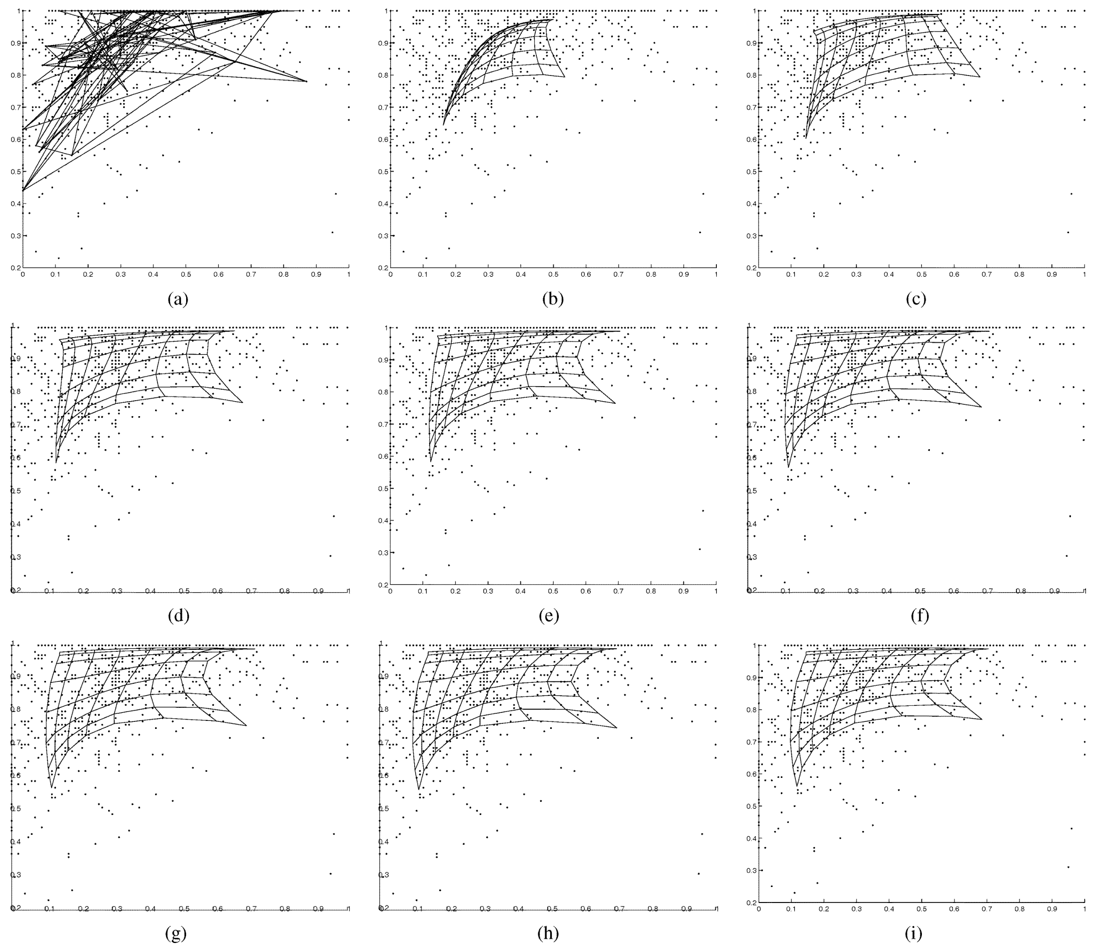

Fig. 10. Map-learning process obtained by running the SODAEM algorithm on PenRecDigits_C0. The value of σ is fixed at 0.15, while value of β is initialized at 0.16 and increased in multiples of 1.6 up to 17.592. (a) Random initialization. (b) $\sigma = 0.15, \beta = 0.16$. (c) $\sigma = 0.15, \beta = 0.256$. (d) $\sigma = 0.15, \beta = 0.409$. (e) $\sigma = 0.15, \beta = 0.655$. (f) $\sigma = 0.15, \beta = 1.04$. (g) $\sigma = 0.15, \beta = 2.68$. (h) $\sigma = 0.15, \beta = 6.871$. (i) $\sigma = 0.15, \beta = 17.592$.

- SOCEM: The value of σ in h_{kl} was set at 0.7 initially, and reduced to 0 (i.e., $h_{kl} = \delta_{kl}$) in 0.02 decrements.
- SODAEM_C: Both the values of β and σ in h_{kl} were set at 0.2 initially. To perform data clustering using the classification-likelihood criterion, the value of β was increased to 10 by the formula $\beta^{\text{new}} = \beta \times 1.2$ first; then, the value of σ was reduced to 0 in 0.02 decrements.
- KohonenGaussian: The value of σ in h_{kl} was set at 0.7 initially, and reduced to 0 in 0.02 decrements every 30 learning iterations.[6]

We ran the algorithms except CEM with 20 independent trials using 9, 16, 25, 36, and 49 Gaussian components. To conduct a fair comparison of CEM and the proposed approaches, we ran CEM many trials until the accumulated execution time was close to that of one SOCEM trial. The mean and standard deviations (error bars) of the classification log-likelihood values over the trials for each algorithm and the best results of CEM (denoted as CEM-best) are shown in Fig. 11. Note that, in the figure, we slightly separate the results associated with a specific Gaussian component number in order to distinguish between

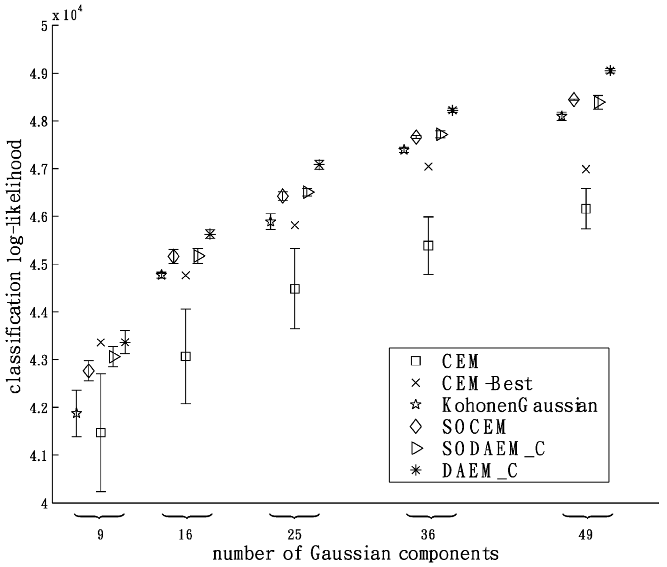

Fig. 11. Data clustering performance of CEM, DAEM_C, SOCEM, SODAEM_C, and KohonenGaussian on ImgSeg in terms of the classification log likelihood.

them. From the figure, we observe that the clustering performance of SOCEM, SODAEM_C, and KohonenGaussian is close to that of DAEM_C. Moreover, they obtain larger and

[6]In our implementation for SOCEM, SOEM, and SODAEM, the phase transition occurs when the likelihood increase is below a threshold or the number of learning iterations exceeds 30 in the current phase. However, KohonenGaussian does not have the convergence property; thus, we ran 30 iterations for each phase of the algorithm.

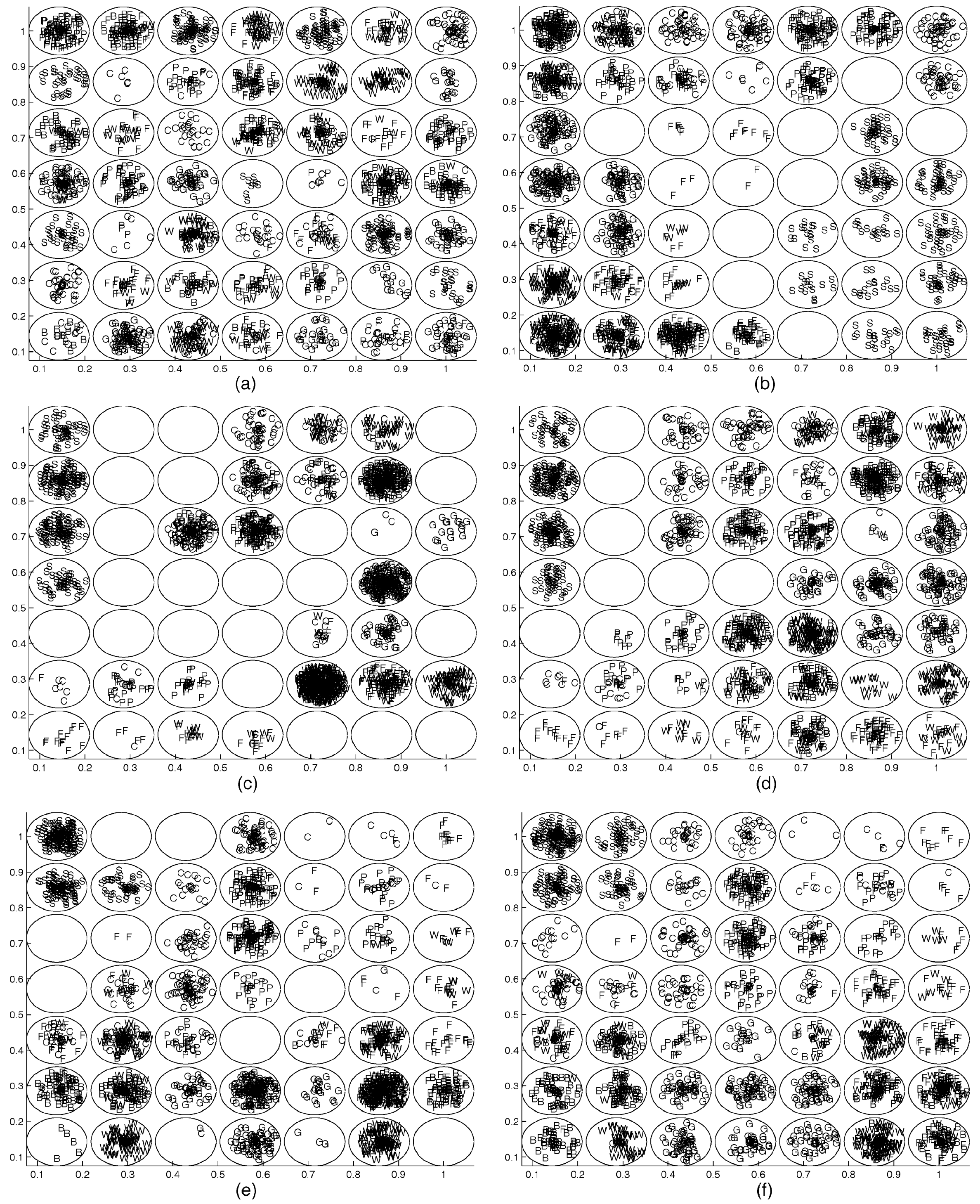

Fig. 12. Data visualization for ImgSeg by running KohonenGaussian (b), SOCEM (c) and (d), and SODAEM_C (e) and (f) with the random initialization in (a). The network structure is a 7 × 7 equally spaced square lattice in a unit square. (a) Random initialization. (b) KohonenGaussian $(\sigma = 0)$. (c) SOCEM $(\sigma = 0.06)$. (d) SOCEM $(\sigma = 0)$. (e) SODAEM_C $(\beta = 10, \sigma = 0.14)$. (f) SODAEM_C $(\beta = 10, \sigma = 0)$.

more stable classification log likelihoods than CEM. These results are rational since SOCEM is a topology-constrained DA variant of the CEM algorithm, and SODAEM_C is an annealing variant of SOCEM with the settings for β and σ here.

Next, we evaluated the data visualization ability of KohonenGaussian, SOCEM, and SODAEM_C. To visualize the data clusters on the network, each data sample was assigned to its winning reference model, and then randomly plotted within the neuron that associates to the reference model [36]. Here, the winner selection strategy for SODAEM_C was the same as that of SOCEM (i.e., the *C-step* of SOCEM). Fig. 12 shows the projections of the data samples on 7×7 lattices obtained by different algorithms. The ImgSeg data set comprises seven classes, namely, brickface: B, sky: S, foliage: F, cement: C, window: W, path: P, and grass: G; each class consists of 300 data samples. Fig. 12(a) depicts the initial mapping of the data

TABLE III
RESULTS OF SIMULATIONS USING KOHONENGAUSSIAN, SOCEM, SOEM, AND SODAEM IN 20 INDEPENDENT RANDOM INITIALIZATION TRIALS ON PENRECDIGITS_C0. THE ALGORITHMS WERE RUN WITH TWO SETUPS FOR σ IN h_{kl}. WHEN $\sigma = 0.15$, KOHONENGAUSSIAN SUCCEEDED IN CONVERGING TO AN ORDERED MAP IN ONE RANDOM INITIALIZATION CASE (S:1), BUT FAILED IN THE REMAINING CASES (F:19)

Setup for σ	$\sigma = 0.15$	$\sigma = 0.6$ initially, and is reduced to 0.15 in 0.15 decrements
KohonenGaussian	S:1	S:20
	F:19	F:0
SOCEM	S:2	S:20
	F:18	F:0
SOEM	S:14	S:20
	F:6	F:0
SODAEM	S:20	-
	F:0	-

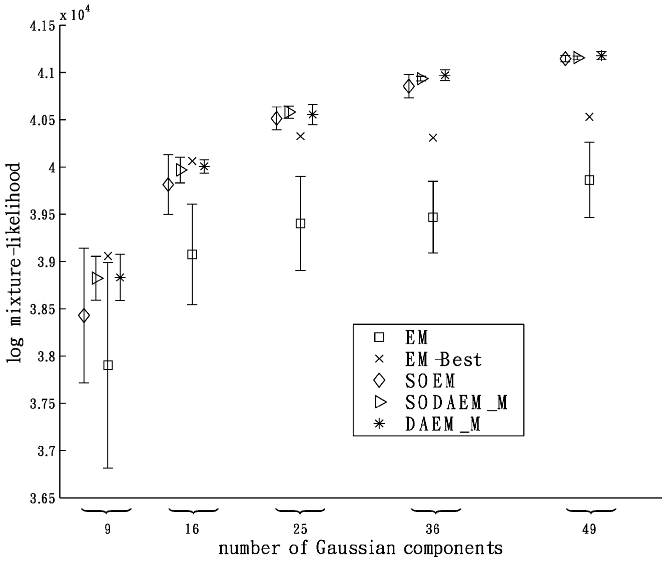

Fig. 13. Learning a Gaussian mixture model by applying EM, DAEM_M, SOEM, and SODAEM_M to ImgSeg.

obtained with a random initialization for the reference models. As we can see from the figure, the data clusters are randomly projected to the neurons (lattice nodes) and the network does not preserve the topological (spatial) relationships among the clusters. Fig. 12(b)–(f) shows the results of the three PbSOM learning algorithms obtained with the random initialization in Fig. 12(a). We see that they can preserve the topological relationships among the data clusters on the network. Moreover, it seems that the data samples of classes "S," "G," "P," and "C" are more distinguishable and well grouped on the network than those of the other classes. In particular, from Fig. 12(b)–(d), we see that only class "S" is separated from the other classes with empty nodes; thus, we may infer that the separability between "S" and the other classes is higher than that between the remaining classes.

For SOCEM, as shown in Fig. 12(c) and (d), the network contains less empty nodes at $\sigma = 0$ than at $\sigma = 0.06$. This may be because in the former case the lateral interactions have vanished, and thus the reference models are adapted to better fit the data distribution than the latter case. Comparing Fig. 12(b) to Fig. 12(d), we see that the data projection results of Kohonen-Gaussian and SOCEM are rather different although they obtain similar classification log likelihoods in Fig. 11. However, we can draw similar observations from the two figures. For example, the data samples of class "S" are more close to those of class "C" and "P" than those of class "G." Fig. 12(e) and (f) shows the results obtained by SODAEM_C. We see that the result in Fig. 12(f) is rather different from that in Fig. 12(d) although SODAEM_C has become equivalent to SOCEM when $\sigma = 0.2$. This may be because these two approaches search on the objective function surface along different paths and converge to different local maxima, as the explanation for the difference of Figs. 9(f) and 10(f) shows in Section IV-A2.

2) Experiments on ImgSeg by Using SOEM and SODAEM_M: First, we evaluated the performance of SOEM and SODAEM_M in learning a Gaussian mixture model with equal mixture weights. The objective function was the log mixture-likelihood function in (2) with equal mixture weights. We compared the performance with that of EM and DAEM_M. The setting for each algorithm was as follows.

- DAEM_M: The value of β was set at 0.2 initially, and increased to 1 by the formula $\beta^{\text{new}} = \beta \times 1.2$.
- SOEM: The value of σ in h_{kl} was set at 0.6 initially, and reduced to 0 (i.e., $h_{kl} = \delta_{kl}$) in 0.02 decrements.
- SODAEM_M: Both the values of β and σ in h_{kl} were set at 0.2 initially. To perform data clustering using the mixture-likelihood criterion, the value of β was increased to 1 by the formula $\beta^{\text{new}} = \beta \times 1.2$ first; then, the value of σ was reduced to 0 in 0.02 decrements.

We ran DAEM_M, SOEM, and SODAEM_M with 20 independent random initialization trials. Similar to the experiments on CEM, we ran EM many trials until the accumulated execution time was close to that of one SOEM trial. The mean and standard deviations (error bars) of the log mixture-likelihood values over the trials for each algorithm and the best results of EM (denoted as EM-best) are shown in Fig. 13. From the figure, it is clear that DAEM_M, SOEM, and SODAEM_M achieve similar performance. Moreover, they obtain larger and more stable log mixture-likelihoods than EM. The results are rational since SOEM is a topology-constrained DA variant of the EM algorithm, and SODAEM_M is an annealing variant of SOEM with the settings for β and σ here.

Next, we evaluated the data visualization ability of SOEM and SODAEM_M. We ran these two algorithms with a 7×7 lattice and the initial reference models used in Section IV-B1 for evaluating SOCEM; therefore, the initial projection of the data was the same as that shown in Fig. 12(a). When clustering the data samples, each sample was assigned to its winning reference model using SOCEM's winner selection strategy. From Fig. 14, we observe that these two algorithms can preserve topological relationships among data clusters (samples). Similar to the results revealed by Fig. 12, data samples of classes "S," "G," "P," and "C" are more distinguishable than those of the other classes.

Comparing Fig. 14(b) to Fig. 12(b) and (d), it is clear that SOEM produces less empty nodes than KohonenGaussian and SOCEM when the value of σ is reduced to zero. It may be explained as follows. For KohonenGaussian and SOCEM, in the case of $\sigma = 0$, they become the CEM (K-means-type) algorithm where each data sample only adapts its winner. However, when $\sigma = 0$, SOEM becomes the EM algorithm where each

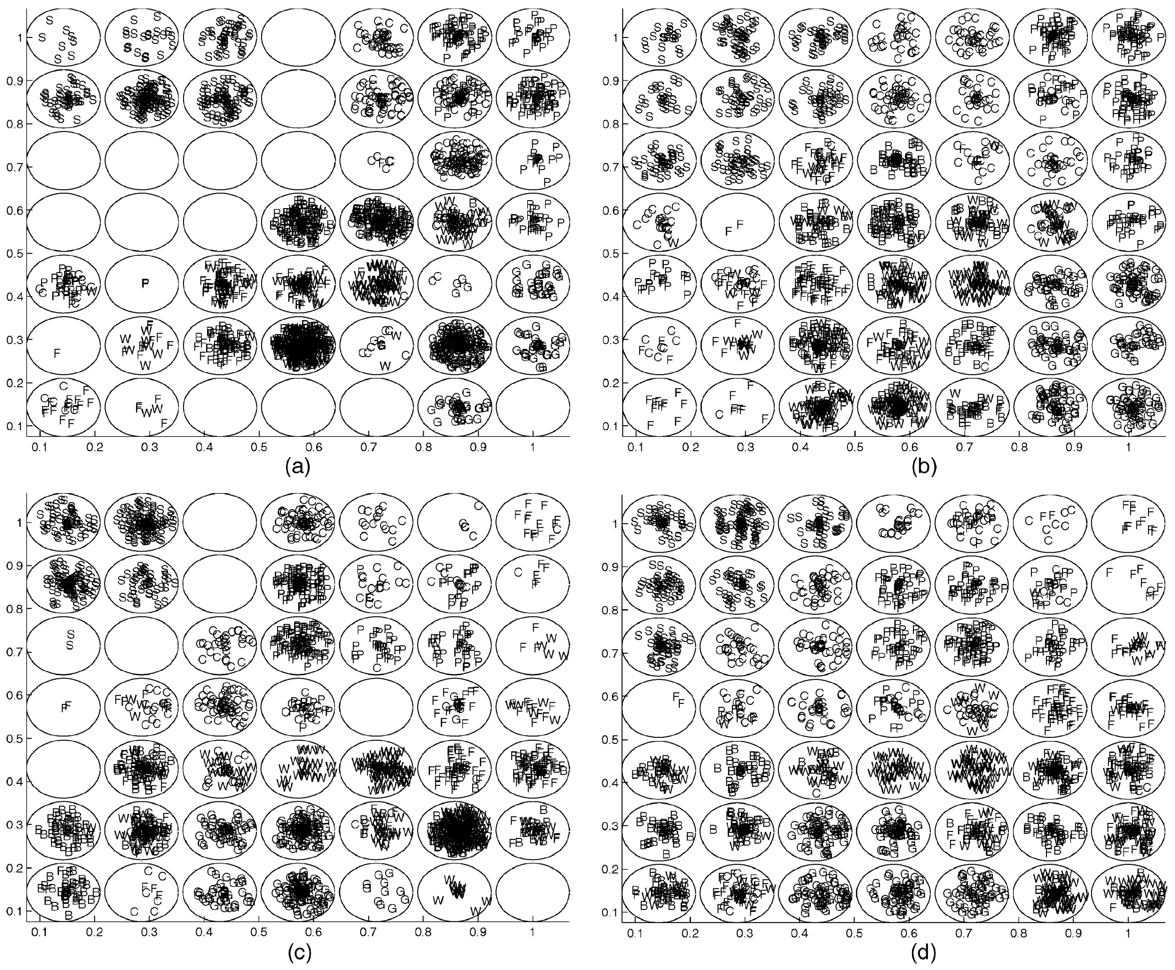

Fig. 14. Data visualization for ImgSeg by running SOEM (a) and (b) and SODAEM_M (c) and (d) with the random initialization in Fig. 12(a). The network structure is a 7×7 equally spaced square lattice in a unit square. (a) SOEM $(\sigma = 0.06)$. (b) SOEM $(\sigma = 0)$. (c) SODAEM_M $(\beta = 1, \sigma = 0.14)$. (d) SODAEM_M $(\beta = 1, \sigma = 0)$.

data sample adapts all the reference models according to their posterior probabilities; thus, the models are more adapted to fit the data than the models of the other two algorithms.

3) Experiments on Ecoli: We conducted experiments on Ecoli using the algorithms applied to ImgSeg in Sections IV-B1 and IV-B2. Fig. 15(a) and (b) shows the data clustering performance of each algorithm in terms of the classification log likelihood and the log mixture likelihood, respectively. Similar to the results on ImgSeg, the PbSOM learning algorithms also achieve decent data clustering performance on Ecoli.

In Fig. 16, for each algorithm, we show the result at the σ value that the class separability can be best visualized on the network. The Ecoli data set comprises eight classes, namely, cp: C, im: I, pp: P, imU: U, om: O, omL: M, imL: L, and imS: S. The numbers of data samples are 143, 77, 52, 35, 20, 5, 2, and 2, respectively. From the figure, we can see that topological relationships among data clusters are preserved well and data classes can be roughly separated on the network.

V. Relation to Other Algorithms

In this section, we explore the differences and relations between the proposed algorithms and other related algorithms.

A. For SOCEM

In [37], Ambroise and Govaert proposed a topology preserving EM (TPEM) algorithm that introduces topological constraints in the CEM algorithm. If Kohonen's winner selection strategy is applied, SOCEM is equivalent to TPEM whose mixture weights are equally fixed. In SOCEM, the covariance matrix of a Gaussian component $\mathbf{\Sigma}_l$ can have different parameterizations for different geometric interpretations [1]. When $\mathbf{\Sigma}_l = \lambda \mathbf{I}$ for $l = 1, 2, \ldots, G$ (where λ is a small positive constant and $\mathbf{I}$ denotes the identity matrix), the clusters are spherical and of equal volume. In this case, the SOCEM algorithm is equivalent to the TVQ algorithm in [22], which was developed for noisy vector quantization. It is also equivalent to the batch SOM learning algorithm described in [20], which employs an energy function in the learning phase of a SOM. However, SOCEM was developed from a different perspective. We consider the learning of a PbSOM as a model-based clustering process. By this perspective, a coupling-likelihood mixture model is developed first, and an objective function is then formulated based on the classification-likelihood criterion. Moreover, the connection between the coupling-likelihood mixture model and the Gaussian mixture model helps interpret

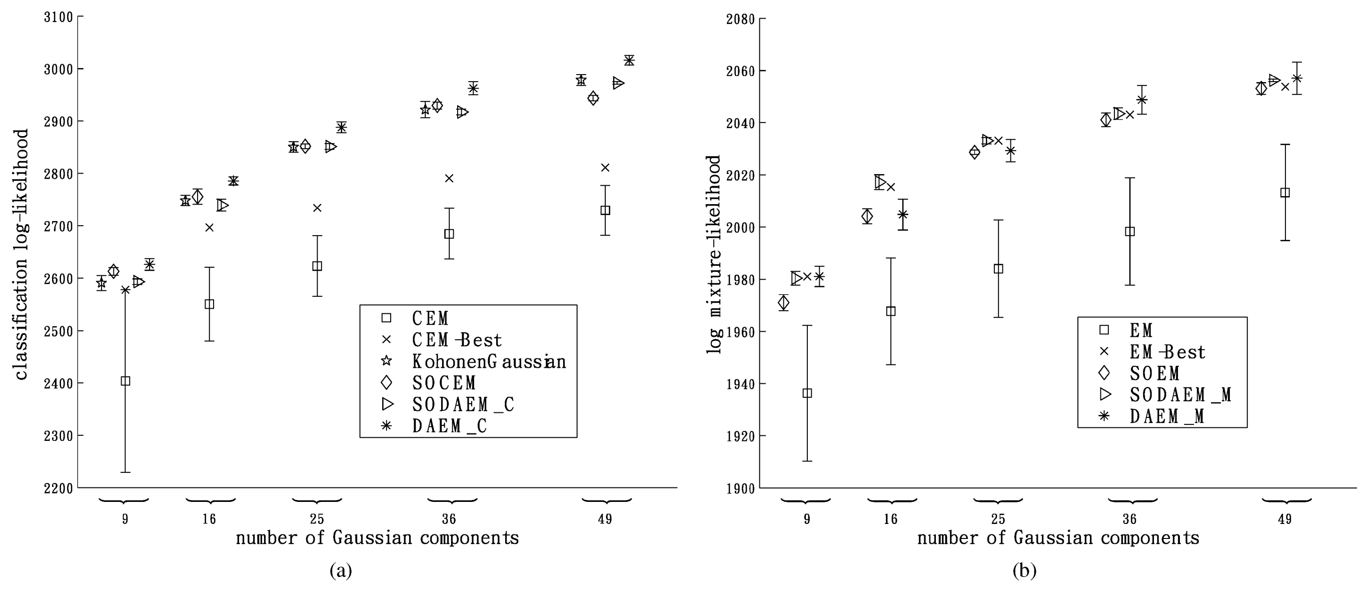

Fig. 15. Data clustering performance on Ecoli in terms of (a) the classification log likelihood and (b) the log mixture likelihood.

SOCEM as a topology-constrained DA variant of the CEM algorithm for GMM.

B. For SOEM and SODAEM

In SODAEM, when $\mathbf{\Sigma}_l = \lambda \mathbf{I}$ for $l = 1, 2, \ldots, G$, SODAEM is equivalent to the STVQ algorithm [23], which learns the parameters by maximizing their density function predicted by the maximum entropy principle. In STVQ, the inverse temperature β is the Lagrange multiplier introduced for the constrained optimization induced by the maximum entropy principle. Heskes [25] extends TVQ's cost function to an expected quantization error. Then, an objective function is obtained by weighting the quantization error with the inverse temperature β and pulsing it to an entropy term that introduces the annealing process. With the resulting objective function, Heskes obtained an algorithm identical to STVQ. The implementations for deterministic annealing in STVQ and Heskes' algorithm can also be found in [38] and [39], where the DA is applied for vector quantization.

SODAEM differs from Graepel *et al.*'s STVQ and Heskes' algorithm in the following ways. First, the deterministic annealing processes are implemented differently. SODAEM is a DAEM algorithm developed to learn the mixture models with a deterministic annealing process, which is implemented based on predicting the posterior distribution in the *E-step* using the maximum entropy principle. Second, the case of $\beta = 1$ was not well addressed in Graepel *et al.*'s and Heskes' papers. This may be because their original goal was to develop a DA learning for TVQ. When β is fixed at 1, however, SODAEM becomes the SOEM algorithm. Moreover, the connection between the proposed coupling-likelihood mixture model and the Gaussian mixture model helps interpret SOEM as a topology-constrained DA variant of the EM algorithm for GMM.

VI. Conclusion

Considering the learning of a PbSOM as a model-based clustering process, we develop a coupling-likelihood mixture model for PbSOM, and derive three EM-type learning algorithms, namely, the SOCEM, SOEM, and SODAEM algorithms, for learning the model (PbSOM). The proposed algorithms improve Kohonen's learning algorithms by including a cost function, an EM-based convergence property, and a probabilistic framework.

In addition, the proposed algorithms provide some insights into the choice of neighborhood size that would ensure topographic ordering. From the experiment results, we observe that the learning performance of SOCEM is very sensitive to the initial setting of the reference models when the neighborhood is small. Conversely, it is not sensitive to the initial condition when the neighborhood is sufficiently large. To deal with the initialization problem, we first run SOCEM with a large neighborhood, and then gradually reduce the neighborhood size until the learning converges to the desired map. When using a small neighborhood, SOEM is less sensitive to the initialization than SOCEM. However, to learn an ordered map, SOEM still needs to start with a large neighborhood. In both SOCEM and SOEM, the neighborhood shrinking can be interpreted as an annealing process that overcomes the initialization issue. Alternatively, we can apply SODAEM, which is a deterministic annealing variant of SOCEM and SOEM, to learn a map. In our experiments, SODAEM overcomes the initialization issue of SOCEM and SOEM via the annealing process controlled by the temperature parameter. Moreover, through the comparison of SOCEM and Kohonen's batch algorithm, we can also apply the DA interpretation of neighborhood shrinking to Kohonen's algorithms to explain why they need to start with a large neighborhood size.

We have also shown that the SOCEM and SOEM algorithms can be interpreted, respectively, as topology-constrained deterministic annealing variants of the CEM and EM algorithms for Gaussian model-based clustering. The experimental results show that our proposed PbSOM learning algorithms achieve an effective data clustering performance, while maintaining the topology-preserving property.

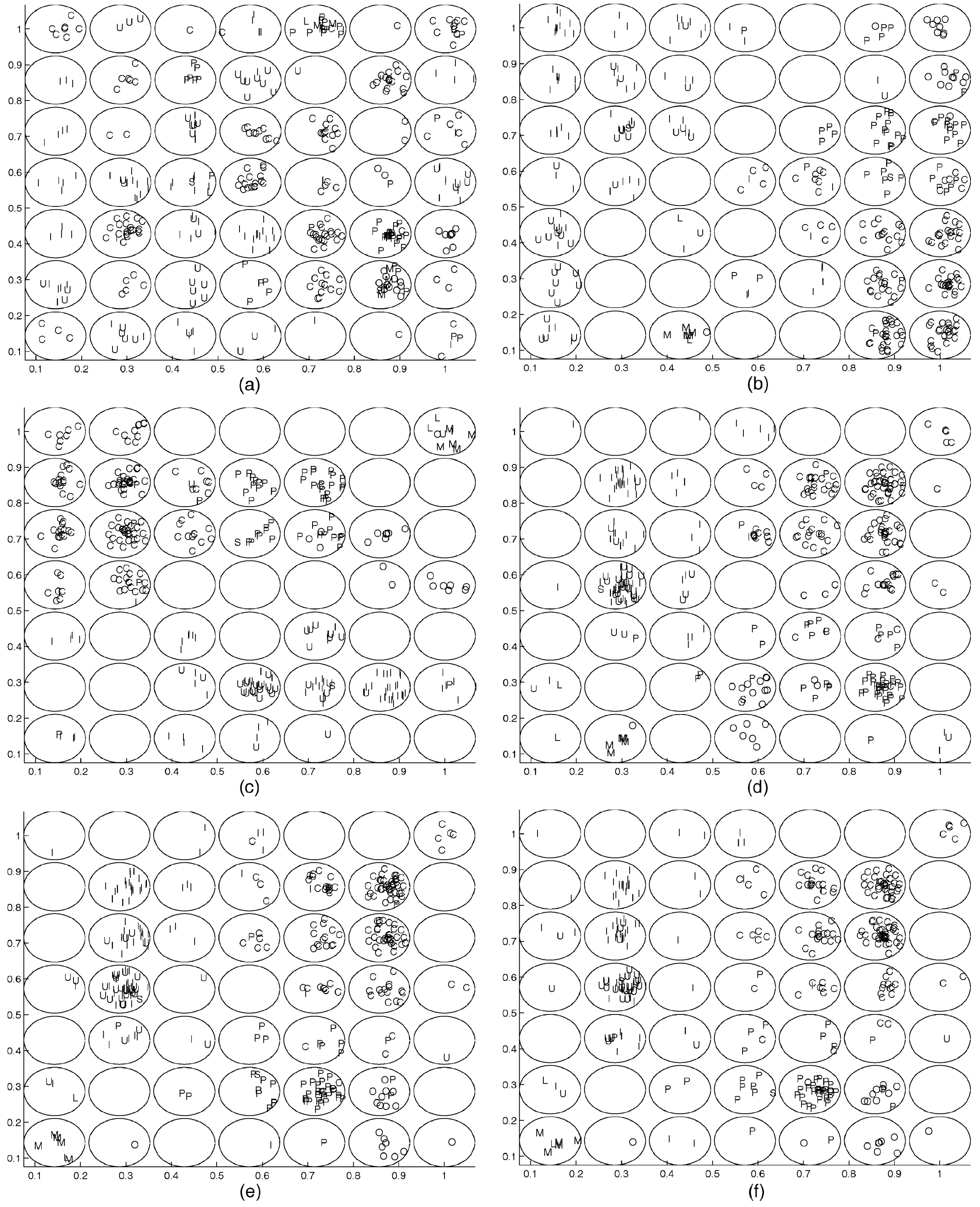
Fig. 16. Data visualization for Ecoli by running (b) KohonenGaussian, (c) SOCEM, (d) SOEM, (e) SODAEM_C, and (f) SODAEM_M with the random initialization in (a). The network structure is a 7×7 equally spaced square lattice in a unit square. (a) Random initialization. (b) KohonenGaussian ($\sigma = 0.06$). (c) SOCEM ($\sigma = 0.06$). (d) SOEM ($\sigma = 0.08$). (e) SODAEM_C ($\beta = 10, \sigma = 0.2$). (f) SODAEM_M ($\beta = 1, \sigma = 0.2$).

Appendix

Theoretically, the mixture weights of the coupling-likelihood mixture model in (22) can be learned automatically. Following the derivations of the SOCEM, SOEM, and SODAEM algorithms in Sections III-B–III-D, the learning rules for the mixture weights are derived as follows.

- **Posterior distribution:**

— for SOCEM and SOEM

$$\gamma_{k|i}^{(t)} = \frac{w_s(k)^{(t)} \exp\left(\sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l^{(t)})\right)}{\sum_{j=1}^{G} w_s(j)^{(t)} \exp\left(\sum_{l=1}^{G} h_{jl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l^{(t)})\right)}; \quad (41)$$

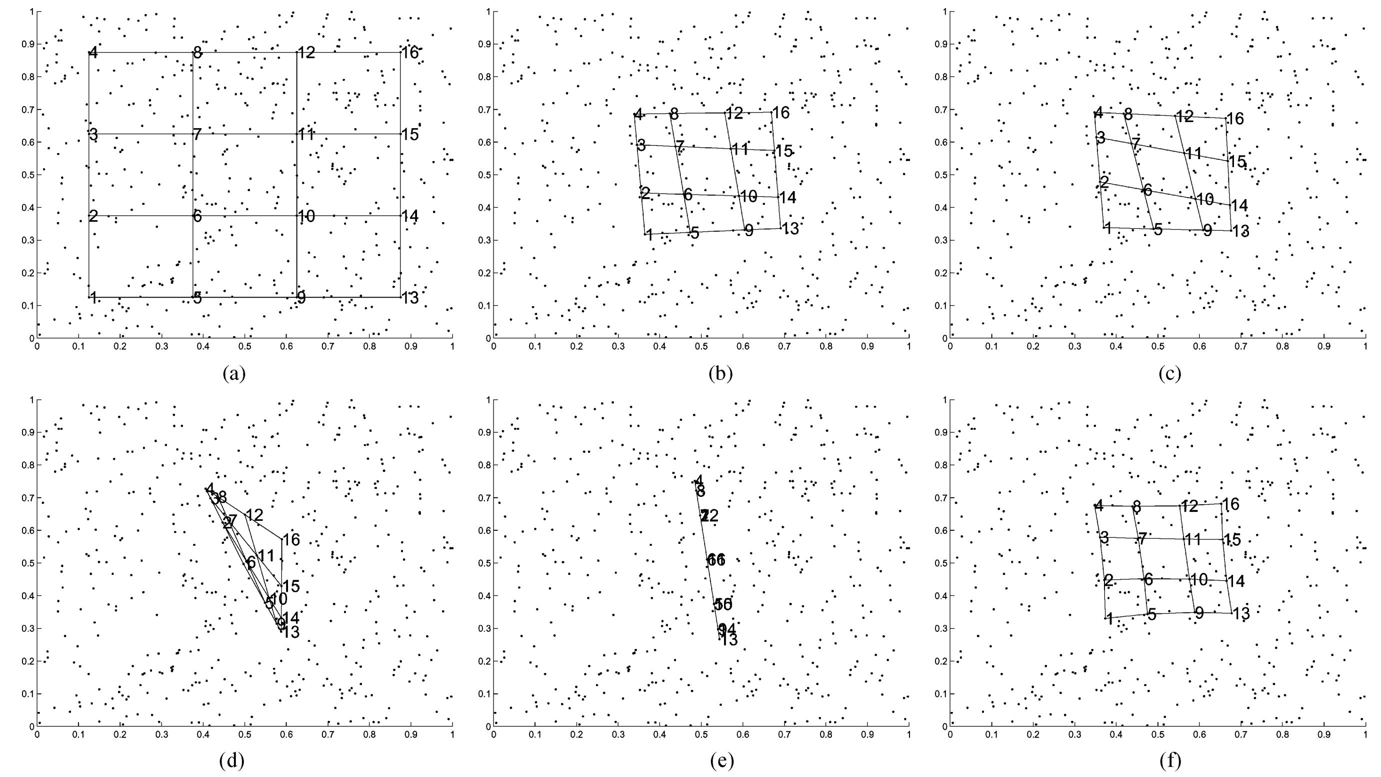

Fig. 17. Map-learning process obtained by running the SOCEM algorithm on the synthetic data with an ordered initialization in (a). (a)–(e) Simulation 1: The mixture weights are initialized at 1/16, and updated in the learning process; the algorithm starts with the initialization in (a) and converges to the unordered map in (e). (a) and (f) Simulation 2: SOCEM is performed with equal mixture weights throughout the learning process; the algorithm starts with the initialization in (a) and converges to the map in (f). The network structure is a 4 × 4 square lattice; the value of σ is set at 0.4. (a) Initialization. (b) Weights are updated, $\mathrm{iter} = 5$. (c) Weights are updated, $\mathrm{iter} = 10$. (d) Weights are updated, $\mathrm{iter} = 16$. (e) Weights are updated, $\mathrm{iter} = 18$. (f) Fixed equal weights.

TABLE IV
THE MIXTURE WEIGHTS LEARNED BY SOCEM WITH THE INITIALIZATION IN FIG. 17(A). THE MIXTURE WEIGHTS ARE INITIALIZED AT 1/16

weight index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Initial	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
iter=5	0.202	0	0	0.304	0	0	0	0	0	0	0	0	0.286	0	0	0.208
iter=10	0.144	0	0	0.354	0	0	0	0	0	0	0	0	0.344	0	0	0.158
iter=16	0	0	0	0.454	0	0	0	0	0	0	0	0	0.494	0	0	0.052
iter=18	0	0	0	0.504	0	0	0	0	0	0	0	0	0.496	0	0	0

— for SODAEM

$$\tau_{k|i}^{(t)} = \frac{\left(w_s(k)^{(t)} \exp\left(\sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l^{(t)})\right)\right)^{\beta}}{\sum_{j=1}^{G}\left(w_s(j)^{(t)} \exp\left(\sum_{l=1}^{G} h_{jl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l^{(t)})\right)\right)^{\beta}}. \quad (42)$$

- **Reestimation formulas:**

— for SOCEM

$$w_s(k)^{(t+1)} = \frac{1}{N}|\hat{\mathcal{P}}_k^{(t)}|; \quad (43)$$

— for SOEM

$$w_s(k)^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N}\gamma_{k|i}^{(t)}; \quad (44)$$

— for SODAEM

$$w_s(k)^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N}\tau_{k|i}^{(t)}. \quad (45)$$

The mean vectors and covariance matrices in SOCEM, SOEM, and SODAEM algorithms are updated using (27)-(28), (35)-(36), and (39)-(40), respectively, where $\gamma_{k|i}^{(t)}$ and $\tau_{k|i}^{(t)}$ are computed by (41) and (42), respectively.

However, in our experience, if the mixture weights are learned in the three algorithms, the learning of topological order is frequently dominated by some particular mixture components, which makes it difficult to obtain an ordered map. As an example, we applied SOCEM to the synthetic data set, which consisted of 500 points uniformly distributed in a unit square. The network structure was a 4×4 equally spaced square lattice in a unit square. All the mixture weights were set at 1/16 initially. The value of σ in the neighborhood function [i.e., (29)] was set at 0.4. The results are shown in Fig. 17(a)–(e). From the figures, we observe that the map shrinks to near a line after the algorithm converges (with 18 iterations). This phenomenon can be verified by inspecting the values of mixture weights during the learning process. As shown in Table IV, after the algorithm converges, most of the mixture weights become zero and the learning only maximizes the local coupling likelihoods of neurons 4 and 13, whose mixture weights are 0.504 and 0.496, respectively. In contrast, as shown in Fig. 17(f), if the mixture weights are equally fixed at 1/16 throughout the learning process, SOCEM converges to an ordered map. For SOEM and SODAEM, we obtained the similar results.

REFERENCES

[1] C. Fraley and A. E. Raftery, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *Comput. J.*, vol. 41, pp. 578–588, 1998.

[2] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *J. Amer. Statist. Assoc.*, vol. 97, no. 458, pp. 611–631, 2002.

[3] S. Zhong and J. Ghosh, "A unified framework for model-based clustering," *J. Mach. Learn. Res.*, vol. 4, no. 6, pp. 1001–1037, 2003.

[4] C. Fraley and A. E. Raftery, "Bayesian regularization for normal mixture estimation and model-based clustering," *J. Classification*, vol. 24, no. 2, pp. 155–181, 2007.

[5] M. S. Oh and A. E. Raftery, "Model-based clustering with dissimilarities: A Bayesian approach," *J. Comput. Graph. Statist.*, vol. 16, no. 3, pp. 559–585, 2007.

[6] M. J. Symons, "Clustering criteria and multivariate normal mixture," *Biometrics*, vol. 37, pp. 35–43, 1981.

[7] S. Ganesalingam, "Classification and mixture approach to clustering via maximum likelihood," *Appl. Statist.*, vol. 38, no. 3, pp. 455–466, 1989.

[8] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Comput. Statist. Data Anal.*, vol. 14, no. 3, pp. 315–332, 1992.

[9] J. D. Banfield and A. E. Raftery, "Model-based Gaussian and non-Gaussian clustering," *Biometrics*, vol. 49, no. 3, pp. 803–821, 1993.

[10] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," Int. Comput. Sci. Inst., Berkeley, CA, Tech. Rep. TR-97-021, 1998.

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

[12] N. Ueda and R. Nakano, "Deterministic annealing EM algorithm," *Neural Netw.*, vol. 11, no. 2, pp. 271–282, 1998.

[13] N. Ueda, R. Nakano, Z. Ghahramani, and G. Hinton, "SMEM algorithm for mixture models," *Neural Comput.*, vol. 12, no. 9, pp. 2109–2128, 2000.

[14] S. S. Cheng, H. M. Wang, and H. C. Fu, "A model-selection-based self-splitting Gaussian mixture learning with application to speaker identification," *EURASIP J. Appl. Signal Process.*, vol. 2004, no. 17, pp. 2626–2639, 2004.

[15] T. Kohonen*, Self-Organizing Maps*. New York: Springer-Verlag, 2001.

[16] T. Kohonen, "The self-organizing maps," *Neurocomputing*, vol. 21, pp. 1–6, 1998.

[17] C. Bishop, M. Svensén, and C. Williams, "The generative topographic mapping," *Neural Comput.*, vol. 10, no. 1, pp. 215–234, 1998.

[18] V. V. Tolat, "An analysis of Kohonen's self-organizing maps using a system of energy functions," *Biol. Cybern.*, vol. 64, no. 2, pp. 155–164, 1990.

[19] E. Erwin, K. Obermayer, and K. Schulten, "Self-organizing maps: Ordering, convergence properties and energy functions," *Biol. Cybern.*, vol. 67, no. 1, pp. 47–55, 1992.

[20] Y. Cheng, "Convergence and ordering of Kohonen's batch map," *Neural Comput.*, vol. 9, no. 8, pp. 1667–1676, 1997.

[21] S. P. Luttrell, "Self-organization: A derivation from fist principles of a class of learning algorithm," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 1989, pp. II-495–II-498.

[22] S. P. Luttrell, "Code vector density in topographic mappings: Scalar case," *IEEE Trans. Neural Netw.*, vol. 2, no. 4, pp. 427–436, Jul. 1991.

[23] T. Graepel, M. Burger, and K. Obermayer, "Phase transitions in stochastic self-organization maps," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 56, no. 4, pp. 3876–3890, 1997.

[24] T. Graepel, M. Burger, and K. Obermayer, "Self-organizing maps: Generalizations and new optimization techniques," *Neurocomputing*, vol. 21, pp. 173–190, 1998.

[25] T. Heskes, "Self-organizing maps, vector quantization, and mixture modeling," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1299–1305, Nov. 2001.

[26] T. W. S. Chow and S. Wu, "An online cellular probabilistic self-organizing map for static and dynamical data sets," *IEEE Trans. Circuit Syst. I, Reg. Papers*, vol. 51, no. 4, pp. 732–747, Apr. 2004.

[27] S. Wu and T. W. S. Chow, "PRSOM: A new visualization method by hybridizing multidimensional scaling and self-organizing map," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1362–1380, Nov. 2005.

[28] S. P. Luttrell, "A Bayesian analysis of self-organizing maps," *Neural Comput.*, vol. 6, no. 5, pp. 767–794, 1994.

[29] F. Anouar, F. Badran, and S. Thiria, "Probabilistic self-organizing map and radial basis function networks," *Neurocomputing*, vol. 20, pp. 93–96, 1998.

[30] J. Lampinen and T. Kostiainen, "Generative probability density model in the self-organizing map," in *Self-Organizing Neural Networks: Recent Advances and Applications*, U. Seiffert and L. Jain, Eds. Berlin, Germany: Physica Verlag, 2002, pp. 75–94.

[31] M. M. Van Hulle, "Joint entropy maximization in kernel-based topographic maps," *Neural Comput.*, vol. 14, no. 8, pp. 1887–1906, 2002.

[32] M. M. Van Hulle, "Maximum likelihood topographic map formation," *Neural Comput.*, vol. 17, no. 3, pp. 503–513, 2005.

[33] J. J. Verbeek, N. Vlassis, and B. J. A. Kröse, "Self-organizing mixture models," *Neurocomputing*, vol. 63, pp. 99–123, 2005.

[34] J. Sum, C. S. Leung, L. W. Chan, and L. Xu, "Yet another algorithm which can generate topography map," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 1204–1207, Sep. 1997.

[35] Univ. California Irvine, UCI Machine Learning Repository, Irvine, CA [Online]. Available: http://archive.ics.uci.edu/ml/

[36] T. Hastie, R. Tibshirani, and J. Friedman*, The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.

[37] C. Ambroise and G. Govaert, "Constrained clustering and Kohonen self-organizing maps," *J. Classification*, vol. 13, no. 2, pp. 299–313, 1996.

[38] K. Rose, E. Gurewitz, and G. C. Fox, "Vector quantization by deterministic annealing," *IEEE Trans. Inf. Theory*, vol. 38, no. 4, pp. 1249–1257, Jul. 1992.

[39] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proc. IEEE*, vol. 86, no. 11, pp. 2210–2239, Nov. 1998.

Shih-Sian Cheng received the B.S. degree in mathematics from National Kaohsiung Normal University, Kaohsiung, Taiwan, in 1999 and the M.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2002. He is currently working towards the Ph.D. degree at the Department of Computer Science, National Chiao Tung University.

In 2002, he joined the Spoken Language Group, Chinese Information Processing Laboratory, Institute of Information Science, Academia Sinica, Taipei, Taiwan, as a Research Assistant. His research interests include machine learning, pattern recognition, speech processing, and neural networks.

Hsin-Chia Fu (S'79–M'80) received the B.S. degree in electrical and communication engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1972 and the M.S. and Ph.D. degrees in electrical and computer engineering from New Mexico State University, Las Cruces, in 1975 and 1981, respectively.

From 1981 to 1983, he was a Member of the Technical Staff at Bell Laboratories. Since 1983, he has been on the faculty of the Department of Computer Science and Information Engineering, National Chiao Tung University. Since 2003, he has also been the Taiwan representative of TEI Consortium. Form 1987 to 1988, he served as the Director of the Department of Information Management, Research Development and Evaluation Commission, Executive Yuan, Taiwan. From 1988 to 1989, he was a Visiting Scholar at Princeton University, Princeton, NJ. From 1989 to 1991, he served as the Chairman of the Department of Computer Science and Information Engineering. From September to December 1994, he was a Visiting Scientist at Fraunhofer Institute for Production Systems and Design Technology (IPK), Berlin, Germany. He has authored more than 100 technical papers, and two textbooks *PC/XT BIOS Analysis* (Taipei, Taiwan: Sun-Kung Book, 1986) and *Introduction to Neural Networks* (Taipei, Taiwan: Third Wave, 1991). His research interests include digital signal/image processing, multimedia information processing, and neural networks.

Dr. Fu was the corecipient of the 1992 and 1993 Long-Term Best Thesis Award with Koun Tem Sun and Cheng Chin Chiang, and the recipient of the 1996 Xerox OA paper Award. He has served as a founding member, Program Co-Chair (1993), and General Co-Chair (1995) of the International Symposium on Artificial Neural Networks. He has been the Technical Committee on Neural Networks for Signal Processing of the IEEE Signal Processing Society from 1997 to 2000. He is a member of the IEEE Signal Processing and Computer Societies, Phi Tau Phi, and the Eta Kappa Nu Electrical Engineering Honor Society.

Hsin-Min Wang (S'92–M'95–SM'04) received the B.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1989 and 1995, respectively.

In October 1995, he joined the Institute of Information Science, Academia Sinica, Taipei, Taiwan, as a Postdoctoral Fellow. He was promoted to Assistant Research Fellow and then Associate Research Fellow in 1996 and 2002, respectively. He was an Adjunct Associate Professor at the National Taipei University of Technology and the National Chengchi University. His major research interests include speech processing, natural language processing, spoken dialogue processing, multimedia information retrieval, and pattern recognition.

Dr. Wang was a board member and chair of academic council of The Association for Computational Linguistics and Chinese Language Processing (ACLCLP). He currently serves as secretary-general of ACLCLP and as an editorial board member of *International Journal of Computational Linguistics* and *Chinese Language Processing*. He was a recipient of the Chinese Institute of Engineers (CIE) Technical Paper Award in 1995. He is a life member of ACLCLP and Institute of Information and Computing Machinery (IICM) and a member of International Speech Communication Association (ISCA).