CEM, EM, AND DAEM ALGORITHMS FOR LEARNING SELF-ORGANIZING MAPS

Shih-Sian Cheng[1,2], *Hsin-Chia Fu*[1], *and Hsin-Min Wang*[2]

[1]Department of Computer Science, National Chiao Tung University, Hsin-Chu, Taiwan, ROC
hcfu@csie.nctu.edu.tw
[2]Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC
{sscheng, whm}@iis.sinica.edu.tw

ABSTRACT

In this paper, we propose a generative model for self-organizing maps (SOM). Based on this model, we derive three EM-type algorithms for learning SOM, namely, the SOCEM, SOEM, and SODAEM algorithms. SOCEM is derived by using the *classification EM* (CEM) algorithm to learn the classification likelihood; SOEM is derived by using the *EM* algorithm to learn the mixture likelihood; and SODAEM is a *deterministic annealing* variant of SOCEM and SOEM. From our experiments on the organizing property of SOM, we observe that SOEM is less sensitive to the initialization of the parameters when using a small-fixed neighborhood than SOCEM, while SODAEM can overcome the initialization problem of SOCEM and SOEM through an annealing process.

1. INTRODUCTION

The self-organizing map (SOM) [1] is a neural network model for data visualization and clustering. The sequential and batch SOM learning algorithms proposed by Kohonen have proved successful in many practical applications. However, they also suffer from some shortcomings, such as the lack of an objective (cost) function, a general proof of convergence, and a probability framework [2]. Some alternative SOM learning algorithms that addressed these issues have been proposed as follows.

In [3], the behavior of Kohonen's sequential learning algorithm was studied in terms of energy functions, based on which, Cheng [4] proposed an energy function for SOM whose parameters can be learned by the K-means type algorithm. Luttrell [5] proposed a noisy vector quantization model called the topographic vector quantizer (TVQ), whose training process coincides with the learning 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.* [6] applied the idea of deterministic annealing to the optimization of TVQ's cost function, and developed an algorithm for noisy vector quantization which was called soft topographic vector quantizer (STVQ). On the basis of topographic distortion, Heskes [7] developed an algorithm identical to STVQ by applying another implementation for deterministic annealing. To enable choosing the correct model complexity for SOM by probabilistic assessment, Lampinen and Kostiainen [8] developed a generative model for which the SOM trained by Kohonen's algorithm or TVQ gives the maximum likelihood estimate. Van Hulle developed a kernel-based topographic formation in [9], where the parameters are adjust to maximize the joint entropy of the kernel outputs. Later, he developed an new algorithm with heteroscedastic Gaussian mixtures that allows for a unified account of vector quantization, log-likelihood, and Kullback-Leibler divergence [10]. Another probabilistic formulation can be found in [11], where a normalized neighborhood function of SOM is adopted as the posterior distribution in E-step of EM algorithm used for learning a mixture model to enforce the self-organizing of the mixture components.

Sum *et al.* [12] interpreted Kohonen's sequential learning algorithm as maximizing the local correlations (coupling energies) between neurons and their neighborhoods for the given input data. Thus, they proposed an energy function for SOM that reveals the correlations, and a gradient ascent learning algorithm for the energy function. Motivated by the work of Sum *et al.*, we propose a generative model for SOM that expresses the local coupling energies over the network with probabilistic likelihoods. Based on the proposed model, we also develop three EM-type algorithms for learning SOM, namely, the SOCEM, SOEM, and SODAEM algorithms. SOCEM is derived by using the classification EM (CEM) algorithm [13] to learn the classification likelihood; SOEM is derived by using the EM algorithm to learn the mixture likelihood; and SODAEM is a deterministic annealing variant of SOCEM and SOEM. Because they inherit the properties of the CEM and EM algorithms, all three algorithms include the features of reliable convergence, low cost per iteration, economy of storage, and ease of programming. From our experiments on the organizing property of SOM, we observe that SOEM is less sensitive to the initialization of the parameters when using a small-fixed neighborhood than SOCEM, while SODAEM can overcome the initialization problem of SOCEM and SOEM through an annealing process.

The remainder of this paper is organized as follows. We first describe the formulation of the generative model in Section 2. Then, the derivations of the SOCEM, SOEM, and SODAEM algorithms are given in Section 3. The experimental results are presented in Section 4. We then present our conclusions in Section 5.

2. FORMULATION OF THE GENERATIVE MODEL FOR SOM

The SOM model [1] consists of G neurons in a network $\mathcal{R} = \{r_1, r_2, \cdots, r_G\}$ with a neighborhood function h_{kl} that defines the strength of lateral interaction between two neurons, r_k and r_l, for $k, l \in \{1, 2, \cdots, G\}$. Each neuron, r_k, associates with a reference model $\boldsymbol{\theta}_k$ in the input data space.

Sum *et al.* [12] interpreted Kohonen's sequential SOM learning algorithm as maximizing the local correlations (coupling energies) between the neurons and their neighborhoods with the given

input data. Given a data sample $\mathbf{x}_i \in \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, the coupling energy between r_k and its neighborhood is defined as

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

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

$$E_{\mathbf{x}_i} = \sum_{k=1}^{G} E_{\mathbf{x}_i|k}, \quad (2)$$

and the energy function to be maximized is

$$C = \sum_{i=1}^{N} \log E_{\mathbf{x}_i}. \quad (3)$$

In Eq. (1), 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:

$$\begin{aligned} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l) &= \frac{1}{(2\pi)^{(d/2)}|\boldsymbol{\Sigma}_l|^{d/2}} \\ &\cdot \exp(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_l)\boldsymbol{\Sigma}_l^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_l)^T) \end{aligned} \quad (4)$$

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

$$\prod_{l \neq k} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)^{h_{kl}}, \quad (5)$$

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 coupling energy between r_k and its neighborhood in terms of probabilistic likelihood as follows:

$$\begin{aligned} p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h) &= r_k(\mathbf{x}_i;\boldsymbol{\theta}_k)^{h_{kk}} \prod_{l \neq k} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)^{h_{kl}} \\ &= \prod_{l=1}^{G} r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)^{h_{kl}} \\ &= \exp(\sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l)), \end{aligned} \quad (6)$$

where $\boldsymbol{\Theta}$ is the set of reference models, h denotes the given neighborhood function[1]. Then, we define the coupling likelihood of $\mathbf{x}_i$ over the network as the following mixture likelihood:

$$p_s(\mathbf{x}_i;\boldsymbol{\Theta},h) = \sum_{k=1}^{G} w_s(k) p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h), \quad (7)$$

where $w_s(k)$, for $k = 1, 2, \cdots, G$, is fixed at $1/G$. When the neighborhood size is reduced to zero (i.e., h_{kl}=δ_{kl}), $p_s(\mathbf{x}_i;\boldsymbol{\Theta},h)$ becomes a Gaussian mixture model with equal mixture weights.

[1]From another perspective, the coupling between r_k and its neighboring neurons is jointly considered in our formulation, rather than being considered in a pairwise manner as in Eq. (1).

3. THE EM-TYPE ALGORITHMS FOR SOM

3.1. The CEM algorithm for SOM (SOCEM)

The self-organizing process of SOM can be described as a data clustering procedure that preserves the spatial relationships between the clusters in a network. Based on the classification likelihood criterion for data clustering [13], the computation of the coupling likelihood of a data sample is restricted to its winning neuron. The goal is to estimate the partition of $\mathcal{X}$, $\hat{\mathcal{P}} = \{\hat{\mathcal{P}}_1, \hat{\mathcal{P}}_2, \cdots, \hat{\mathcal{P}}_G\}$, and the set of reference models, $\hat{\boldsymbol{\Theta}}$, so as to maximize the following objective function:

$$\begin{aligned} & C_s(\mathcal{P},\boldsymbol{\Theta};\mathcal{X},h) \\ &= \sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \mathcal{P}_k} \log(w_s(k) p_s(\mathbf{x}_i|k;\boldsymbol{\Theta},h)) \\ &= \sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \mathcal{P}_k} \sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l) + Constant. \end{aligned} \quad (8)$$

Similar to the derivations of the classification EM (CEM) algorithm for clustering [13], the CEM algorithm for learning SOM (SOCEM) 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:

$$\begin{aligned} \gamma_{k|i}^{(t)} &= p_s(k|\mathbf{x}_i;\boldsymbol{\Theta}^{(t)},h) \\ &= \frac{p_s(\mathbf{x}_i,k;\boldsymbol{\Theta}^{(t)},h)}{p_s(\mathbf{x}_i;\boldsymbol{\Theta}^{(t)},h)} \\ &= \frac{\exp(\sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)}))}{\sum_{j=1}^{G} \exp(\sum_{l=1}^{G} h_{jl} \log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)}))}, \end{aligned} \quad (9)$$

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

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 Eq. (8) becomes

$$\begin{aligned} C_s(\boldsymbol{\Theta};\hat{\mathcal{P}}^{(t)},\mathcal{X},h) &= \sum_{l=1}^{G} \sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \hat{\mathcal{P}}_k^{(t)}} h_{kl} \log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l) \\ &+ Constant. \end{aligned} \quad (10)$$

By substituting Eq. (4) into Eq. (10) and taking the derivative of C_s with respect to individual parameters and setting it to zero, we obtain the following re-estimation formulae for the mean vectors and covariance matrices;

$$\begin{aligned} & \boldsymbol{\mu}_l^{(t+1)} = \frac{\sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \hat{\mathcal{P}}_k^{(t)}} h_{kl} \mathbf{x}_i}{\sum_{k=1}^{G} |\hat{\mathcal{P}}_k^{(t)}| h_{kl}}, \\ & \boldsymbol{\Sigma}_l^{(t+1)} = \\ & \frac{\sum_{k=1}^{G} \sum_{\mathbf{x}_i \in \hat{\mathcal{P}}_k^{(t)}} h_{kl} (\mathbf{x}_i - \boldsymbol{\mu}_l^{(t+1)})(\mathbf{x}_i - \boldsymbol{\mu}_l^{(t+1)})^T}{\sum_{k=1}^{G} |\hat{\mathcal{P}}_k^{(t)}| h_{kl}} \end{aligned} \quad (11)$$

for $l = 1, 2, \cdots, G$. When the neighborhood size is reduced to zero (i.e., h_{kl}=δ_{kl}), SOCEM reduces to the CEM algorithm for a Gaussian mixture model with equal mixture weights.

When $\mathbf{\Sigma}_l = \lambda \mathbf{I}$ for $l = 1, 2, \cdots, G$, the clusters are spherical and of equal volume. In this case, the SOCEM algorithm is equivalent to the TVQ algorithm in [5] and the batch SOM learning algorithm in [4]; however, they are developed from different perspectives. The major difference between the SOCEM algorithm and Kohonen's batch learning algorithm is that the former considers the neighborhood information when selecting the winning neuron, but the latter does not.

3.2. The EM algorithm for SOM (SOEM)

As is obvious from Eq. (10), in the formulation of the objective function of the SOCEM algorithm, only the coupling likelihoods associated with the winning neurons are considered. In this part, the coupling likelihood over the entire network (i.e., Eq. (7)) is considered. Consider Eq. (7) as a mixture model, where k indicates the mixture component from which the coupling likelihood is generated, then SOM can be learned in the sense of conventional maximum likelihood estimation. This can be performed by the EM algorithm [14]. The EM algorithm for learning SOM (SOEM) is as follows.

E-step: Given the mixture likelihood in Eq. (7), we form the auxiliary function as follows:

$$Q_s(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) = \sum_{i=1}^{N} \sum_{k=1}^{G} \gamma_{k|i}^{(t)} \log p_s(\mathbf{x}_i, k; \mathbf{\Theta}, h), \quad (12)$$

where $\gamma_{k|i}^{(t)}$ is the same as Eq. (9). Since

$$p_s(\mathbf{x}_i, k; \mathbf{\Theta}, h) = w_s(k) p_s(\mathbf{x}_i | k; \mathbf{\Theta}, h), \quad (13)$$

Eq. (12) can be rewritten as

$$Q_s(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) = \sum_{i=1}^{N} \sum_{k=1}^{G} \gamma_{k|i}^{(t)} \log(w_s(k) p_s(\mathbf{x}_i | k; \mathbf{\Theta}, h)). \quad (14)$$

As $w_s(k)$, for $k = 1, 2, \cdots, G$, is fixed at $1/G$, by substituting Eq. (6) into Eq. (14), we obtain

$$\begin{aligned} & Q_s(\mathbf{\Theta}; \mathbf{\Theta}^{(t)}) \quad (15) \\ &= \sum_{i=1}^{N} \sum_{k=1}^{G} \gamma_{k|i}^{(t)} \sum_{l=1}^{G} h_{kl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l) + Constant, \\ &= \sum_{l=1}^{G} \sum_{i=1}^{N} \sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl} \log r_l(\mathbf{x}_i; \boldsymbol{\theta}_l) + Constant. \end{aligned}$$

M-step: By replacing the response $r_l(\mathbf{x}_i; \boldsymbol{\theta}_l)$ in Eq. (15) with the multivariate Gaussian density in Eq. (4) and setting the derivative of Q_s with respect to individual mean vectors and covariance matrices to zero, we obtain the re-estimation formulae as follows:

$$\begin{aligned} \boldsymbol{\mu}_l^{(t+1)} &= \frac{\sum_{i=1}^{N} (\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}) \mathbf{x}_i}{\sum_{i=1}^{N} (\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl})}, \\ \mathbf{\Sigma}_l^{(t+1)} &= \quad (16) \\ & \frac{\sum_{i=1}^{N} (\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl}) (\mathbf{x}_i - \boldsymbol{\mu}_l^{(t+1)}) (\mathbf{x}_i - \boldsymbol{\mu}_l^{(t+1)})^T}{\sum_{i=1}^{N} (\sum_{k=1}^{G} \gamma_{k|i}^{(t)} h_{kl})} \end{aligned}$$

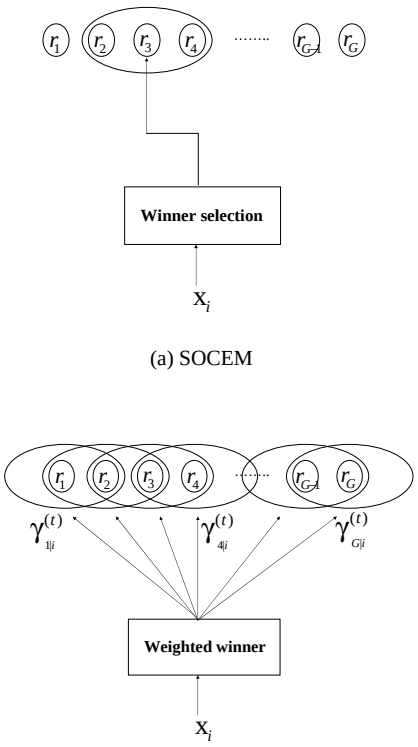


Fig. 1. For each data sample $\mathbf{x}_i$, the adaptation of the reference models in SOCEM is restricted to the winning reference model and its neighborhood. However, in SOEM, the winner is relaxed to the weighted winners by the posterior probabilities $\gamma_{k|i}^{(t)}$, for $k = 1, 2, \cdots, G$.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

for $l = 1, 2, \cdots, G$. When the neighborhood size is reduced to zero (i.e., h_{kl}=δ_{kl}), SOEM reduces to the EM algorithm for a Gaussian mixture model with equal mixture weights.

SOEM adapts the reference modes 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. 1, in the SOCEM algorithm (*cf.* Eq. (11)), 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.* Eq. (16)), 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 $k = 1, 2, \cdots, G$.

3.3. The DAEM algorithm for SOM (SODAEM)

Similar to the deterministic annealing EM (DAEM) algorithm for learning GMM [15], we can also develop a DAEM algorithm for

the proposed generative model to learn SOM. With the mixture likelihood defined in Eq. (7), the first step of DAEM derives the posterior density in the *E-step* using the principle of maximum entropy. Following the derivation for posterior probability in [15] 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:

$$\begin{aligned}\tau_{k|i}^{(t)} &= \frac{p_s(\mathbf{x}_i|k;\mathbf{\Theta}^{(t)},h)^\beta}{\sum_{j=1}^G p_s(\mathbf{x}_i|j;\mathbf{\Theta}^{(t)},h)^\beta} \\ &= \frac{\exp(\beta\sum_{l=1}^G h_{kl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)}))}{\sum_{j=1}^G \exp(\beta\sum_{l=1}^G h_{jl}\log r_l(\mathbf{x}_i;\boldsymbol{\theta}_l^{(t)}))}. \quad (17)\end{aligned}$$

Then, the auxiliary function to be minimized is $U_{s\beta}(\mathbf{\Theta};\mathbf{\Theta}^{(t)}) = -\sum_{i=1}^N\sum_{k=1}^G \tau_{k|i}^{(t)}\log p_s(\mathbf{x}_i,k;\mathbf{\Theta},h)$, and the respective re-estimation formulae for the mean vectors and covariance matrices are as follows:

$$\begin{aligned}\boldsymbol{\mu}_l^{(t+1)} &= \frac{\sum_{i=1}^N(\sum_{k=1}^G \tau_{k|i}^{(t)}h_{kl})\mathbf{x}_i}{\sum_{i=1}^N(\sum_{k=1}^G \tau_{k|i}^{(t)}h_{kl})}, \\ \mathbf{\Sigma}_l^{(t+1)} &= \\ &\frac{\sum_{i=1}^N(\sum_{k=1}^G \tau_{k|i}^{(t)}h_{kl})(\mathbf{x}_i-\boldsymbol{\mu}_l^{(t+1)})(\mathbf{x}_i-\boldsymbol{\mu}_l^{(t+1)})^T}{\sum_{i=1}^N(\sum_{k=1}^G \tau_{k|i}^{(t)}h_{kl})} \quad (18)\end{aligned}$$

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

Note that the re-estimation formulae for SODAEM are 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, during 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.

In the case of $\mathbf{\Sigma}_l = \lambda\mathbf{I}$ for $l = 1, 2, \cdots, G$, SODAEM is equivalent to the STVQ algorithm [6], which learns the parameters by maximizing their density function predicted by the maximum entropy principle. This is equivalent to the EM algorithm for SOM in [7], which is derived by another implementation for DA. However, the case of $\beta = 1$ was not well addressed in [6] and [7]. This may be because their original idea was to develop a DA learning for TVQ. When β is fixed at 1, however, SODAEM becomes the SOEM algorithm.

4. EXPERIMENTS

4.1. Data set and experimental setup

We conducted experiments on a real-world data set: training set of class '0' in the "Pen-Based Recognition of Handwritten Digits" database (PenRecDigits_C0) from the UCI Machine Learning Database Repository [16]. 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 of SOM. As a pre-processing step, we scaled down each element of the vectors in PenRecDigits_C0 to 1/100 of its original value to avoid numerical traps. In the experiments, an 8 $\times$ 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 $h_{kl} = \exp(-\frac{\|r_k - r_l\|^2}{2\sigma^2})$.

We evaluated Kohonen's algorithm, SOCEM, SOEM, and SODAEM with 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, $\mu_1^{'}$, $\mu_2^{'}$, $\cdots$, $\mu_G^{'}$, of the reference models, which were multivariate Gaussians with full covariance matrices. The initial covariance matrix $\Sigma_l^{'}$ was set as $\rho_l I$, where ρ_l=$\min_{k\neq l}\{\|\mu_l^{'} - \mu_k^{'}\|\}$, for $l = 1, 2, \cdots, G$. 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 is less than 0.001, it was set at 0.001.

4.2. Results

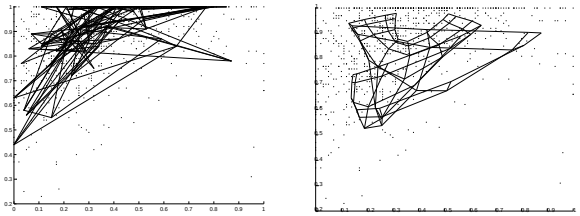
We first demonstrate the map-learning processes of SOCEM, SOEM, and SODAEM using one of the 20 random initializations, and then summarize the overall results of all the initializations.

Simulations by SOCEM: Fig. 2 shows two simulations using the SOCEM algorithm. In the first simulation, SOCEM is run with the random initialization in Fig. 2 (a) and a fixed σ of 0.15 in h_{kl}. As shown in Fig. 2 (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. 2 (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 $\sigma = 0.15$. Figs. 2 (c), (d), (e), and (f) depict the maps obtained when σ =0.6, 0.45, 0.3, and 0.15, respectively.

Simulations by SOEM: We conducted two similar simulations using the SOEM algorithm. In the first, SOEM is run with the random initialization in Fig. 3 (a) (the same as that in Fig. 2 (a)) and a fixed σ of 0.15. As shown in Fig. 3 (b), the learning of SOEM converges to an unordered map. In the second simulation, SOEM starts with the random initialization in Fig. 3 (a) and a larger σ of 0.6. Then, the value of σ is gradually reduced to 0.15 in 0.15 decrements. Figs. 3 (c), (d), (e), and (f) depict the maps obtained when SOEM converges at σ =0.6, 0.45, 0.3, and 0.15, respectively.

Simulations by SODAEM: Fig. 4 depicts the simulations using the SODAEM algorithm with the same random initialization as that in Fig. 2 (a) and Fig. 3 (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 β^{new}=$\beta \times 1.6$ and the reference models obtained in the previous phase. The learning process finally converges at $\beta = 17.592$. When $\beta = 1.04$ and $\beta = 17.592$, SODAEM is almost equivalent to SOEM and SOCEM, respectively. In these two cases, SODAEM converges to the ordered maps in Fig. 4 (e) and Fig. 4 (f), respectively. However, as shown in Figs. 2 (a)-(b) and Figs. 3 (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 $\sigma = 0.15$. It is worth mentioning that, if the value of σ is too small to form an ordered map at a small β value, SODAEM may not be able to obtain any ordered map during the annealing process.

Discussion: All the experiment results obtained by the three algorithms for the 20 random initializations are summarized in Ta-



(a) random ini.

(b) $\sigma = 0.15$ with rand. ini. (converge)

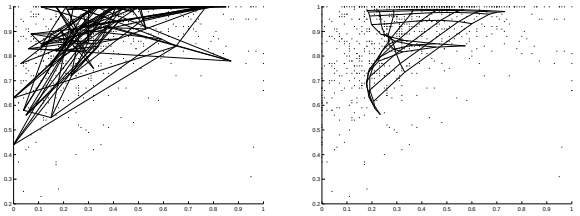
(c) $\sigma = 0.6$ with rand. ini. (converge)

(d) $\sigma = 0.45$ (converge)

(e) $\sigma = 0.3$ (converge)

(f) $\sigma = 0.15$ (converge)

Fig. 2. Simulations of SOM using the SOCEM algorithm on PenRecDigits_C0. Simulation 1 ((a)-(b)): When SOCEM is run with the random initialization in (a) and $\sigma = 0.15$, it converges to the unordered map in (b). Simulation 2 ((a) and (c)-(f)): 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 ini.

(b) $\sigma = 0.15$ with rand. ini. (converge)

(c) $\sigma = 0.6$ with rand. ini. (converge)

(d) $\sigma = 0.45$ (converge)

(e) $\sigma = 0.3$ (converge)

(f) $\sigma = 0.15$ (converge)

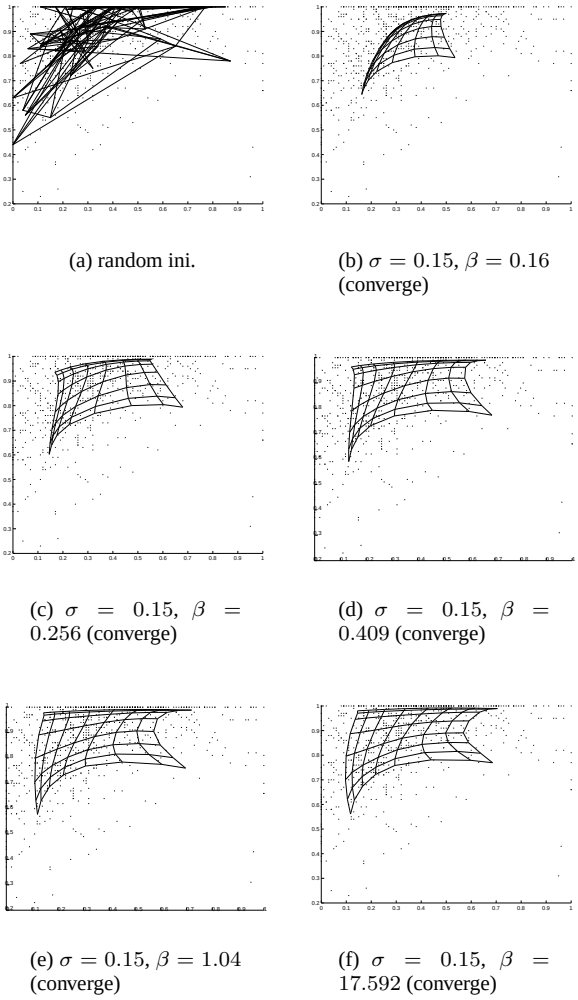
Fig. 3. Simulations of SOM using the SOEM algorithm on PenRecDigits_C0. Simulation 1 ((a)-(b)): When SOEM is run with the random initialization in (a) and $\sigma = 0.15$, it converges to the unordered map in (b). Simulation 2 ((a) and (c)-(f)): SOEM starts with $\sigma = 0.6$ and the random initialization in (a). Then, the value of σ is reduced to 0.15 in 0.15 decrements.

ble 1. Several conclusions can be drawn from the table. First, SOEM often converged to an ordered map at a small, fixed σ value ($\sigma = 0.15$ in the experiments); but SOCEM seldom did so. This is because SOEM learns the map in a more global way, as stated in Sec. 3.2, so that it is less sensitive to the initialization of the parameters when σ is small. The results for Kohonen's algorithm and SOCEM are similar. This is because they only differ in the winner selection strategy. Second, the initialization issue of Kohonen's algorithm, SOCEM and SOEM can be overcome by using a larger σ value ($\sigma = 0.6$ in the experiments) initially, and then gradually reducing the value to the target σ value (0.15 in the experiments). 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.

5. CONCLUSIONS

We have proposed a generative model for SOM and developed three EM-type learning algorithms, namely, the SOCEM, SOEM, and SODAEM algorithms. The proposed algorithms improve Kohonen's learning algorithms by including a cost function, a general proof of convergence, and a probabilistic framework.

In addition, the proposed learning 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. However, 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,



(a) random ini.

(b) $\sigma = 0.15$, $\beta = 0.16$ (converge)

(c) $\sigma = 0.15$, $\beta = 0.256$ (converge)

(d) $\sigma = 0.15$, $\beta = 0.409$ (converge)

(e) $\sigma = 0.15$, $\beta = 1.04$ (converge)

(f) $\sigma = 0.15$, $\beta = 17.592$ (converge)

Fig. 4. Simulations of SOM using the SODAEM algorithm on PenRecDigits_C0. The value of σ is fixed at 0.15. The value of β is initialized at 0.16 and increased in multiples of 1.6 up to 17.592.

SOEM is less sensitive to the initialization than SOCEM. However, to learn an ordered map, SOEM still needs to start with a large neighborhood. 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.

Table 1. Results of simulations by Kohonen's algorithm, SOCEM, SOEM, and SODAEM using 20 independent random initialization trials on PenRecDigits_C0. The algorithms are run with two setups for σ in h_{kl}. When $\sigma = 0.15$, Kohonen's algorithm succeeds in converging to an ordered map in one random initialization cases (S:1), but fails 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
Kohonen's algorithm	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	-

6. REFERENCES

[1] T. Kohonen, "The self-organing maps," *Neurocomputing*, vol. 21, pp. 1V6, 1998.

[2] C. Bishop, M. Svensén, and C. Williams, "The Generative Topographic Mapping," *Neural Computation* vol. 10, no. 1, pp. 215-234, 1998.

[3] E. Erwin, K. Obermayer, and K. Schulten,"Self-organizing maps: ordering, convergence properties and energy functions," *Biological Cybernetics*, vol. 67, pp. 47-55, 1992.

[4] Y. Cheng, "Convergence and ordering of Kohonen's batch map," *Neural Computation*, vol. 9, pp. 1667-1676, 1997.

[5] S. P. Luttrell, "Code vector density in topographic mappings: Scalar case," *IEEE Tran. Neural Networks*, vol. 2, pp. 427-436, 1991.

[6] T. Graepel, M. Burger, and K. Obermayer, "Phase transitions in stochastic self-organization maps," *Physical Review E*, vol. 56, no. 4, pp. 3876-3890, 1997.

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

[8] J. Lampinen and T. Kostiainen, "Generative probability density model in the Self-Organizing Map," U. Seiffert and L. Jain, editors, *Self-organizing neural networks: Recent advances and applications*, pp. 75-94, Physica Verlag.

[9] M. M. Van Hulle, "Joint Entropy Maximization in Kernel-based Topographic Maps," *Neural Computation*, vol. 14, pp. 1887-1906, 2002.

[10] M. M. Van Hulle, "Maximum Likelihood Topographic Map Formation," *Neural Computation*, vol. 17, pp. 503-513, 2005.

[11] J. J. Verbeek, N. Vlassis, and B. J. A. Kröse, "Self-Organizing Mixture Models," *Neurocomputing*, vol. 63, pp. 99-123, 2005.

[12] J. Sum, C. S. Leung, L. W. Chan, and L. Xu, "Yet another algorithm which can generate topology map," *IEEE Trans. Neural Network*, vol. 8, no 5, pp. 1204-1207, 1997.

[13] G. Celeux, and G. Govaert, "A Classification EM algorithm for clustering and two stochastic versions," *Computatoinal Statistics & Data Analysis*, 14 , pp. 315-332, 1992.

[14] Jeff A. Bilmes, "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models," International Computer Science Insitute Technical Reports, TR-97-021, April 1998.

[15] N. Ueda and R. Nakano, "Deterministic annealing EM algorithm," *Neural Network*, vol. 11, pp. 271-282, 1998.

[16] D. Newman, S. Hettich, C. Blake, and C. Merz, UCI repository of machine learning databases, 1998.