On the Emulation of Kohonen's Self-Organization via Single-Map Metropolis-Hastings Algorithms

Jorge Muruzábal

Statistics and Decision Sciences Group
University Rey Juan Carlos, 28936 Móstoles, Spain
j.muruzabal@escet.urjc.es

Abstract. As data sets get larger and larger, the need for exploratory methods that allow some visualization of the overall structure in the data is becoming more important. The self-organizing map (SOM) introduced by Kohonen is a powerful tool for precisely this purpose. In recent years, SOM-based methodology has been refined and deployed with success in various high-dimensional problems. Still, our understanding of the properties of SOMs fitted by Kohonen's original algorithm is not complete, and several statistical models and alternative fitting algorithms have been devised in the literature. This paper presents a new Metropolis-Hastings Markov chain Monte Carlo algorithm designed for SOM fitting. The method stems from both the previous success of bayesian machinery in neural models and the uprise of computer-intensive, simulation-based algorithms in bayesian inference. Experimental results suggest the feasibility as well as the limitations of the approach in its current form. Since the method is based on a few extremely simple chain transition kernels, the framework may well accommodate the more sophisticated constructs needed for a full emulation of the self-organization treat.

1 Introduction

Kohonen's self-organizing map (SOM) [10] provides a fast, scalable and easy-to-interpret visualization tool. Complemented with Sammon's mapping [11] and several diagnostic statistics [1], it has proved useful for several data analysis tasks, see e. g. [12,13]. While this applied success is stimulating, the relative theoretical opacity of the original fitting *algorithm* has made it hard to define the particular state of affairs that the SOM *structure* should converge to [4]. One of the frequently cited problems of the standard fitting algorithm is that it is not supported by any statistical model when perhaps it should, for it attempts after all to carry out a density estimation task.

Several bayesian models have been proposed to bear on this issue. Utsugi's prior [16] places a direct smoothing constraint on the set $\mathbf{w}$ of SOM pointers. The generative topographic mapping (GTM) approach [2] introduces a latent covariate and tries to hard-wire the SOM's smoothness by means of a non-linear map from latent to data space. This map effectively links the set of pointers beyond the SOM topology. As a result of this modelling effort, several EM-like techniques are now available as alternative fitting algorithms for the SOM

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 346–355, 2001.

structure. However, the practical usefulness of these algorithms is to be fully demonstrated yet [9]. Meanwhile, there clearly remains room for analysis of further algorithms that, on the basis of these sensible models, pursue the emulation of the self-organizing ability.

In recent years, analysis of more complex bayesian models has been made possible thanks to fundamental advances in Markov chain Monte Carlo (MCMC) theory, see e.g. [5,15]. MCMC methods are based on iterative *sampling* algorithms: given a target posterior distribution π on some space of structures of interest, they provide a collection of structures that can be approximately taken as *iid* draws from π. Estimates of a variety of posterior features, together with their corresponding standard errors, can be built in principle from these values. In the context of neural networks, the most general models involve network architecture parameters γ, network weights or pointers $\mathbf{w} = \mathbf{w}(\gamma)$ and other objects like prior hyperparameters α and likelihood parameters β (collectively denoted by $h = \{\alpha, \beta\}$). The most sophisticated MCMC methods rely on *reversible jump* MCMC theory [6] to explore full posteriors of the form $\pi = \pi(\gamma, \mathbf{w}, h/\mathbf{x})$, where $\mathbf{x}$ is the training data. For example, Rios Insua and Müller [14] derive an approximation to the posterior of the number of units in the hidden layer of a standard feed-forward network. Similar contributions in other neural models have been hindered by the difficulty in elliciting sensible prior distributions over the huge space of possible $(\gamma, \mathbf{w}, h)$. For example, in the SOM model the map size would become a variable parameter. Neither Utsugi [16,17] nor Bishop and coworkers [2,3] consider indeed MCMC methods for the SOM model.

The first MCMC method in the SOM context has been proposed in [18]. Utsugi essentially formulates a *Gibbs* sampler [5,15] for posteriors of the form $\pi = \pi(\mathbf{w}, h/\gamma, \mathbf{x})$. In practice, the SOM is typically planar, and we know that the choices of topology, shape and size are not crucial to obtain useful results. Hence, conditioning on γ substantially reduces the complexity of the problem while imposing only a mild limitation on the overall scope. In this paper I consider an even simpler MCMC method (developed independently of [18]) based on posteriors of the form $\pi(\mathbf{w}/h, \gamma, \mathbf{x})$, where α and β are scalar quantities. Since the choice of conditioning h may modify the posterior landscape substantially, the approach involves some pretesting with various α and β and is thus quite exploratory in its present form. Still, the class of *Metropolis-Hastings* (MH) algorithms [5,15] reviewed below is rather flexible and permits to explore how far can one go by replacing the conditional distributions in Gibbs samplers with simple transition kernels inspired by the SOM's original fitting algorithm and bayesian assumptions. Note also that the two MCMC algorithms just discussed always maintain a single SOM in memory. Such algorithms are thus markedly different from *multiple* MCMC samplers, see e.g. [8]; these maintain a population of networks from which transition proposals based on several networks can be made.

The organization is as follows. The basic notation and assumptions in bayesian SOM modelling are provided in section 2. Section 3 summarizes some relevant MCMC theory. Section 4 presents the new class of MH algorithms and

section 5 reports on their performance in some data sets. Section 6 summarizes and provides directions for future research.

2 Bayesian Self-Organizing Maps

The self-organizing map is a biologically-inspired network of interconnected neurons or units s, each endowed with an associated pointer $w_s \in \mathbb{R}^m$ [10]. Let us focus for simplicity on the 2-D case and consider squared SOMs with $r = k^2$ units and standard connectivity. A data matrix $\mathbf{x}$ containing n exchangeable vectors $x^{(l)} \in \mathbb{R}^m$ is used for training. A trained SOM (fitted by the standard algorithm or otherwise) should satisfy two key desiderata: (i) the "density" of the pointer cloud should resemble the underlying distribution of the data; and (ii) pointers should exhibit topological order or self-organization, a notion unfortunately hard to pin down precisely [4]. The standard fitting procedure tends to achieve these two goals at a reasonable computational cost. For inspection of trained SOMs, we usually project the map $\mathbf{w} = \{w_s, s = 1, .., r\}$ onto 2-D space via Sammon's mapping [11]. Since the set of pointers "inherits" the connectivity pattern, pointers can be linked to its immediate neighbours on these images and we can evaluate informally the amount of organization in the fitted SOM, see Figures 1 and 2 below.

Statistical models recently introduced for the SOM agree to set a Gaussian mixture sampling (or *generative*) model $P(\mathbf{x}/\mathbf{w}, \beta) = \prod_{l=1}^{n} \sum_{s=1}^{r} \frac{1}{r} f(x^{(l)}/w_s, \beta)$, where $f(x/w_s, \beta) = ()\beta 2\pi^{\frac{m}{2}} \exp\{-\frac{\beta}{2} \|x - w_s\|^2\}$ and $\beta > 0$ controls the dispersion of the data "generated" by any given unit [2,16]. As regards the prior $P(\mathbf{w}/\alpha)$, it is customary to assume independent coordinates. A general choice for $P(\mathbf{w}/\alpha)$ is the Gaussian process prior $P(\mathbf{w}/\alpha) = ()12\pi^{\frac{rm}{2}} |\alpha|^{-\frac{m}{2}} \prod_{j=1}^{m} \exp\{-\frac{1}{2} w_{(j)}^T \alpha^{-1} w_{(j)}\}$, where α is, in principle, a dispersion matrix expressing the desired bias towards smoothness in some way and $w_{(j)} \in \mathbb{R}^r$ collects the j-th coordinates from all pointers. The full model is completed by the second-stage prior $P(\alpha, \beta)$, from which the key joint distribution $P(\mathbf{x}, \mathbf{w}, \alpha, \beta) = P(\mathbf{x}/\mathbf{w}, \beta) P(\mathbf{w}/\alpha) P(\alpha, \beta)$ follows.

In their GTM model [2], Bishop, Svensén and Williams extend the previous formulation. They assume a latent variable $z \in \mathbb{R}^L$, $L \ll m$, and a prior density on latent space $P(z)$ which is discrete uniform on some fixed $\mathbf{z} = \{z_s, s = 1, ..., r\}$ (typically an evenly-spaced, rather arbitrary collection of latent vectors). A nonlinear, one-to-many map τ, parametrized by a $m \times M$ matrix Λ, formally links the latent and data spaces, so that each pointer w_s becomes $w_s = \tau(z_s, \Lambda) = \Lambda \Phi(z_s)$ for a set Φ of M fixed basis functions defined on $\mathbb{R}^L$. As shown by the authors, the emerging log-likelihood $L(\Lambda, \beta; \mathbf{x}) = \sum_{l=1}^{n} \log \left\{ \frac{1}{r} \sum_{s=1}^{r} f(x^{(l)}/\tau(z_s, \Lambda), \beta) \right\}$ can be maximized by a variant of the EM algorithm. Note that each z_s plays the role of a single neuron and each $\tau(z_s, \Lambda)$ plays the role of a single pointer, so that the number of degrees of freedom of the map $\mathbf{w}$ is reduced substantially. Further,

since τ is continuous and smooth, the $\tau(z_s, \Lambda)$ centroids will be automatically organized, see [2] for details on Φ, M and L.

Returning to our previous bayesian model, in [3] GTM's latent structure is used to reduce the complexity in $P(\mathbf{w}/\alpha)$ by taking $\alpha_{st} \exp\{-\frac{\lambda}{2} \|z_s - z_t\|^2\}$, where z_s and z_t are the latent vectors generating map units s and t respectively and $\lambda > 0$ is a single scalar expressing correlation decay that naturally plays the role of α in $P(\mathbf{x}, \mathbf{w}, \alpha, \beta)$. Utsugi [18] dismisses this idea and prefers to achieve a similar level of simplicity via the alternative choice $\alpha = (\lambda D^T D + \mu E^T E)^{-1}$, where D and E are fixed matrices and λ and μ are (scalar) weight factors. Utsugi assigns the main role to the D matrix (E is just included to guarantee a proper prior). If we set $\mu = 0$ for simplicity, we are led to $P(\mathbf{w}/\alpha) = P(\mathbf{w}/\lambda) = \prod_{j=1}^{m} \left(\frac{\lambda}{2\pi}\right)^{\frac{R}{2}} \sqrt{\Delta} \exp\left\{-\frac{\lambda}{2} \left\|Dw_{(j)}\right\|^2\right\}$, where R and Δ are the rank and product of positive eigenvalues of $D^T D$ respectively. This is the prior used below for the usual D smoothing matrix, the so-called five-point star approximation to the Laplacian operator in 2-D [16]. Specifically, D has $R = (k-2)^2$ rows (one for each interior unit), and the row addressing pointer (u, v) presents a -4 at location (u, v), ones at its four neighbours' locations $\{(u-1, v), (u+1, v), (u, v-1), (u, v+1)\}$ and zeros elsewhere. The resulting conditional log posterior function is $\log P(\mathbf{w}/\mathbf{x}, \lambda, \beta) = \sum_{l=1}^{n} \log \sum_{s=1}^{r} \exp\left\{-\frac{\beta}{2} \left\|x^{(l)} - w_s\right\|^2\right\} - \frac{\lambda}{2} \sum_{j=1}^{m} \left\|Dw_{(j)}\right\|^2$. The following MCMC computations are based on $P(\mathbf{w}/\mathbf{x}, \lambda, \beta)$ playing the role of $\pi(\mathbf{w}/h, \gamma, \mathbf{x})$ as discussed in the Introduction (it will simply be written $\pi(\mathbf{w})$ below). I sometimes refer to the two summands in this expression as the fit and smoothness components respectively.

3 Some MCMC Background

We now briefly review the basic aspects of the Metropolis-Hastings (MH) class of algorithms [5,15]. The key result states that an invariant distribution π of a time-homogeneous Markov chain G with transition kernel Γ is also its limiting distribution provided (G, Γ) is aperiodic and irreducible. Intuitively, for the chain to be irreducible any state should be reachable from any other state. Likewise, an aperiodic chain is not forced to visit certain (subsets of) states in any systematic way. The main idea in bayesian analysis is to simulate a suitable chain (G, Γ) having the posterior of interest π as invariant distribution. The limiting (long-run) behaviour of the chain is then taken as an approximation to iid sampling from π.

The class of MH algorithms easily yields kernels Γ guaranteing the stationarity of any given π as follows. Let $q(a, b)$ denote a proposal density configuring the MH transition kernel Γ. Then, given that the chain is at state a, a random proposal b is made according to $q(a, \cdot)$ and accepted with probability $\psi(a, b) = \min\{1, \frac{\pi(b)q(b,a)}{\pi(a)q(a,b)}\}$; if the proposal is not accepted, then the chain stays at state a and a new b is drawn from $q(a, \cdot)$, etc. This procedure guarantees the stationarity

of π. In practice, q is to be devised to ensure that the chain is also aperiodic and irreducible.

An easy way to do this is to decompose q as a (finite) mixture of several densities q_θ (with respective activation probabilities p_θ), all of which have π as stationary distribution and one of which, say θ_0, does possess the desired properties of aperiodicity and irreducibility in a trivial way. The mixture chain defined by $q = \sum_\theta p_\theta q_\theta$ first selects some θ and then makes a proposal according to $q_\theta(a, \cdot)$. As long as the corresponding p_{θ_0} is strictly positive, this mixture q inherits the target properties and hence provides a means to simulate the posterior π according to the MH strategy. The basis density θ_0 is usually quite simple; it is the remaining q_θ in the mixture which provide adequate scope for the strategy.

In the case of SOMs $\mathbf{w} \in \mathbb{R}^{rm}$, it is straightforward to see that the role of θ_0 can be played by either a joint spherical Gaussian random walk $\tilde{\mathbf{w}} \sim N_{rm}(\mathbf{w}, \sigma_1^2 \mathrm{I})$ (updating all pointers w_s at once) or else a uniform mixture density made up by the lower-dimensional, single-pointer spherical Gaussian random walks $N_m(w_s, \sigma_s^2 \mathrm{I})$ (updating a single w_s at a time). Here typically the σ_s^2 are all equal to some σ_2^2. These "background" processes are referred to below as (B1) and (B2) respectively. Note that, in either case, $q(\mathbf{w}, \tilde{\mathbf{w}}) = q(\tilde{\mathbf{w}}, \mathbf{w})$, so $\psi(\mathbf{w}, \tilde{\mathbf{w}})$ boils down to $\min\{1, \frac{\pi(\tilde{\mathbf{w}})}{\pi(\mathbf{w})}\}$.

Consider now the case of more general block transitions q_θ in our SOM context. Now index θ refers to (possibly overlapping) subsets of coordinates of $\mathbf{w}$, for example, those associated to one or several pointers w_s. The activation probabilities p_θ correspond to random drawing among all possible θ. If we decompose, with an obvious notation, $\mathbf{w} = \{\mathbf{w}_\theta, \mathbf{w}_{(\theta)}\}$, we typically use $q_\theta(\mathbf{w}, \tilde{\mathbf{w}}) = q(\mathbf{w}, \tilde{\mathbf{w}}_\theta)$ for all θ, that is, at each transition step proposals are made to update the θ portion only (but these proposals are always made in the same way as in the case of (B2) above). It follows that $\mathbf{w}_{(\theta)} = \tilde{\mathbf{w}}_{(\theta)}$ and hence $\psi(\mathbf{w}, \tilde{\mathbf{w}}) = \min\{1, \frac{\pi(\tilde{\mathbf{w}}_\theta / \tilde{\mathbf{w}}_{(\theta)}) q(\tilde{\mathbf{w}}, \mathbf{w}_\theta)}{\pi(\mathbf{w}_\theta / \mathbf{w}_{(\theta)}) q(\mathbf{w}, \tilde{\mathbf{w}}_\theta)}\}$. An important particular case occurs then when $q(\mathbf{w}, \tilde{\mathbf{w}}_\theta) = q(\mathbf{w}_{(\theta)}, \tilde{\mathbf{w}}_\theta)$ equals the conditional posterior density $\pi(\tilde{\mathbf{w}}_\theta / \mathbf{w}_{(\theta)})$, the so-called Gibbs sampler. In this case, $\psi(\mathbf{w}, \tilde{\mathbf{w}}) \equiv 1$, that is, all proposed transitions are automatically carried out. Of course, depending on the complexity of θ and π, it may not be always straightforward to find the conditionals $\pi(\cdot / \mathbf{w}_{(\theta)})$ required for sampling.

Utsugi's [18] Gibbs sampler involves chains that have *entire* SOMs $\mathbf{w}$ together with hyperparameters λ, β and membership dummies $\mathbf{y}$ as state space. Thus, his sampler alternates between $\pi(\mathbf{y} / \mathbf{x}, \mathbf{w}, \beta)$, $\pi(\mathbf{w} / \mathbf{x}, \mathbf{y}, \lambda)$, $\pi(\lambda / \mathbf{x}, \mathbf{y}, \mathbf{w})$ and $\pi(\beta / \mathbf{x}, \mathbf{y}, \mathbf{w})$. Here, in contrast, we are envisaging a sampler oriented to SOM portions θ (for fixed choice of hyperparameters and using no latent structure). In the next section, we consider MH algorithms based on simple proposal densities $q(\mathbf{w}, \tilde{\mathbf{w}}_\theta)$ for certain types of subsets θ. This simplicity comes of course at the price of having to evaluate the posterior ratios $\frac{\pi(\tilde{\mathbf{w}})}{\pi(\mathbf{w})}$ at each tentative $\tilde{\mathbf{w}}_\theta$.

4 SOM Fitting via MH Samplers

It is clear that the traditional SOM algorithm, although Markovian in nature, is far from the MH family. For example, in the standard sequential implementation of this algorithm all pointers in some neighbourhood are linearly shifted at each time step towards the current data item $x^{(l)}$, the neighbourhood width being a user-input decreasing function. This shifting inspires nonetheless one of the proposal densities $q(\mathbf{w}, \tilde{\mathbf{w}}_\theta)$ discussed below. The other stems from the smoothing character of the assumed prior $P(\mathbf{w}/\lambda)$.

Four MH algorithms for SOM fitting are explored. The first two, (B1) and (B2) (given above), are meant as simple reference algorithms of scant value on their own; they include a single tunable scalar each. The other two algorithms, (A1) and (A2), are mixture chains based on (B1) or (B2) as background processes that incorporate smoothing and block-shift transitions. Let us describe these kernels first.

Smoothing transitions apply to individual interior units s only. They exploit the SOM topology in order to provide a simple smoothing counterpart to the conditional density discussed in the previous section. Specifically, let ν_s denote the standard 8-unit neighbourhood of unit s (excluding s itself) and write ∂w_s for the usual average (mean) vector of w_s, $s \in \nu_s$. A transition is then proposed from w_s to some $\tilde{w}_s$ drawn from a $N_m(\partial w_s, \tau_1^2 \mathrm{I})$ distribution. The associated ratio $\frac{q(\tilde{\mathbf{w}}_{(s)}, w_s)}{q(\mathbf{w}_{(s)}, \tilde{w}_s)}$ becomes $\exp\{-\frac{1}{2\tau_1^2}(\|w_s - \partial w_s\|^2 - \|\tilde{w}_s - \partial w_s\|^2)\}$, so that $\psi(\mathbf{w}, \tilde{\mathbf{w}}) = \exp\{\min[0, \Psi(\mathbf{w}, \tilde{\mathbf{w}})]\}$ with $\Psi(\mathbf{w}, \tilde{\mathbf{w}}) = [\log \pi(\tilde{\mathbf{w}}) - \log \pi(\mathbf{w})] - \frac{1}{2\tau_1^2}(\|w_s - \partial w_s\|^2 - \|\tilde{w}_s - \partial w_s\|^2)$. Note that the intended smoothing effect occurs only if τ_1^2 is relatively small, in which case the last term in $\Psi(\mathbf{w}, \tilde{\mathbf{w}})$ may downplay any log posterior improvement introduced by the new $\tilde{w}_s$. Thus, care is needed when setting the value of the shift variance τ_1^2.

Block-shift transitions act on edges of the net. These transitions are intended to facilitate the higher mobility that such units are expected to need. Specifically, given a unit s on some edge of the network, let ν_s denote now unit s together with its 5 immediate neighbours (3 in the case of corners). A single δ is drawn from a $N_m(0, \tau_2^2 \mathrm{I})$ distribution, and $\tilde{w}_s = w_s + \delta$ for all $s \in \nu_s$. Clearly, in this case $\psi(\mathbf{w}, \tilde{\mathbf{w}})$ simplifies again as in (B1) or (B2). Note the difference with respect to the standard training algorithm whereby δ is different for each s.

We can now describe the remaining MH algorithms; they implement two possible combinations of the above ideas and are defined as follows. Algorithm (A1): at each time step a coin with probability of heads χ is tossed. If heads, a global transition (B1) is attempted. If tails, a unit is randomly selected from the network. If interior, then a smoothing transition is proposed, otherwise a local (B2) transition is attempted. Algorithm (A2): at each time step a similar coin is tossed. If heads, a local (B2) transition is attempted. If tails, a unit is selected as before. If interior, a smoothing transition is proposed, otherwise a block-shift is attempted. Note that both (A1) and (A2) present four tunable scalars each.

5 Experimental Results

In this paper I concentrate on the issue of how effective the proposed kernels are with regard to the goal of emulating the standard fitting algorithm [11]. SOMs obtained by these algorithms are contrasted in several toy problems for various choices of hyperparameters λ and β. Relatively small SOMs of 6×6 and 7×7 neurons are used (so we have approximately the same number of interior and edge units). Before we actually dwell into the experiments, a couple of remarks are in order.

It is a fact that MH samplers tend to struggle with multimodality. In practice, this entails that convergence to a suboptimal posterior mode is expected in nearly all cases. This is not so critical though, for there are clearly many useful modes in our SOM context. Hence, the proposed algorithms are evaluated on the basis of the individual SOMs obtained after a fixed number of iterations. Of course, random allocation of pointers is likely to require exceedingly long runs, and alternative initialization procedures are almost compulsory. A simple idea (similar to initialization based on principal components [11] and used below) is to place all initial SOM pointers regularly on a random hyperplane (going through m randomly chosen rows of $\mathbf{x}$). The result typically will not fit the data well, yet it is a flat structure that should free samplers from the ackward phase of early organization.

As regards selection of tunable scalars, the heads probability χ was simply set to $\frac{1}{2}$ throughout. As usual, all sampling variances were tuned on the basis of the acceptance rate ξ of their proposed transitions. This selection process is somewhat tricky since acceptance rates typically decrease along the run and not much is known about optimal values in general. The following simple strategy was based on the anticipated run length. Background processes (B1) and (B2) were first run separately in order to select values for σ_1^2 and σ_2^2 leading to ξ's around 30% after a few thousand trials. These selected values were then maintained in (A1) and (A2), and the remaining τ_1^2 and τ_2^2 were tuned so that the overall ξ's remained between 15 and 25% after 10,000 trials.

Two artificial test cases are considered: a four-cluster Y-shaped data set ($n = 100$, $m = 3$) and a cigar-shaped data set ($n = 100$, $m = 10$). The four-cluster data were generated as a balanced mixture of four Gaussians with small spherical spread and centers located at the corners of a folded Y, see Figure 2. Dimensionality is kept low in this case to allow for direct inspection; the 3-D folding guarantees an interesting structure for the 2-D SOM to capture. The cigar-shaped data (see Figure 1) consists of an elongated bulk with point-mass contamination. The bulk (80%) was derived from a Gaussian distribution with zero mean and equicorrelation dispersion matrix with correlation .9. The remaining 20% were generated by another Gaussian with small spherical spread and mean far away from the bulk.

Let us begin with the cigar-shaped data. The fit is done under $\lambda = 100$ and $\beta = 1$ (thus placing a strong emphasis on the smoothness of the final map). Algorithm (A1) was executed 5 times for 10,000 trials under standard deviations $\sigma_1 = .005$, $\sigma_2 = .045$ and $\tau_1 = .025$; these led consistently to acceptance rates

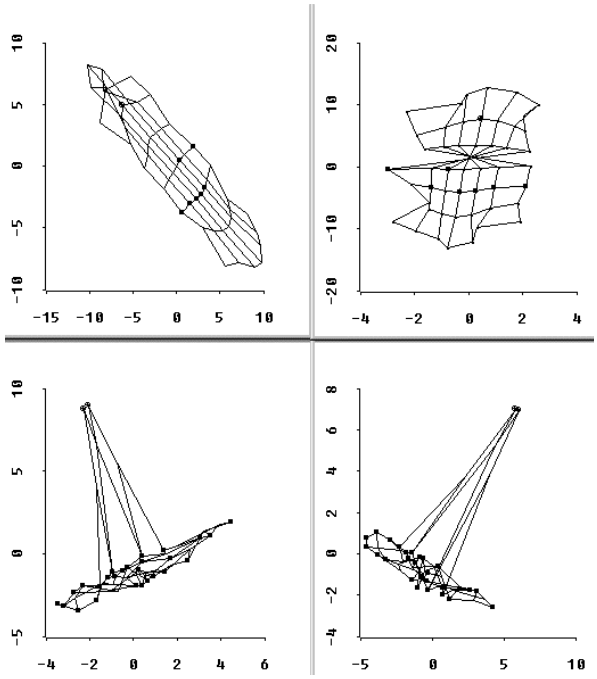

Fig. 1. Projections (via Sammon's map) of SOMs fitted to the cigar-shaped data by MH algorithm (A1) (top row) and by the standard algorithm (bottom row). In all cases solid squares cover the elongated bulk and circles cover the outliers (dots are empty).

by their associated proposals of about 15%, 3% and 58%, with a combined rate of 23%. Figure 1 portrays two SOMs obtained by this MH sampler (second and worst with regard to log posterior values) together with two SOMs fitted by the standard algorithm. As expected, the larger value of λ translates into rather flat structures by (A1), yet we can recover the structure in the data by focusing on the pattern of nonempty units. On the other hand, the standard algorithm arranges pointers more faithfully according to the underlying data density. While not fully organized, these SOMs undoubtedly provide a more accurate description of the data.

Consider next the four-cluster data. We now examine performance by (A2) under $\lambda = 10$, $\beta = 1,000$ (thus priming heavily the fit component) and $\sigma_2 = .2$, $\tau_1 = .15$ and $\tau_2 = .35$ (leading respectively to partial acceptance rates of 35%, 10% and 5%, with an overall rate of about 21%). Five runs were again conducted, and this time the median log posterior SOM was selected for comparison. Figure 2 shows this SOM together with another map fitted by the standard algorithm. The differences are again outstanding as regards visual appearance and log density values. Specifically, the standard algorithm scores about $-6,100$ and -530 in the smoothness and fit log density scale respectively, whereas (A2) yields -50 and $-1,710$ respectively. Hence, the standard algorithm is clearly willing to sacrifice a good deal of smoothness in order to arrange pointers closer to the data.

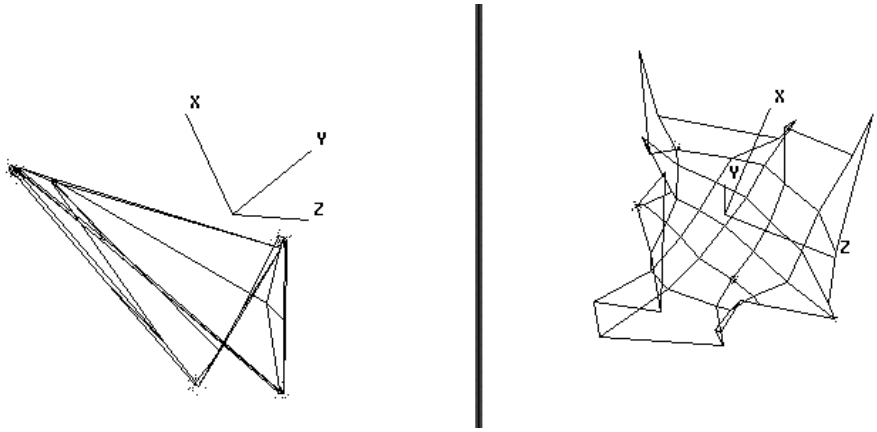

Fig. 2. 3-D rotating plots for the four-cluster data: SOMs fitted by the standard algorithm (left) and by the MH algorithm (A2). Data are superimposed for reference. Axes are included to highlight the different viewpoints adopted in each case.

We conclude that, even under the favorable β used in this run, the MH algorithm can not produce the clusters of pointers needed to improve the fit beyond the smoothness requirement.

6 Summary and Concluding Remarks

A new class of MH algorithms for SOM fitting has been presented and some preliminary experiments reported. Following Utsugi's [16] model choice for smoothing prior and gaussian mixture likelihood, it has been shown that it is relatively easy to emulate the smoothness property of the trained SOM via MH algorithms. A useful analysis may then proceed on the basis of the pattern of non-empty units on the network. However, it is the SOM's density estimating goal which remains ellusive and may require the design of additional transition kernels. Specifically, kernels that somehow home in on detected concentrations of the data should be most useful here. Furthermore, alternative prior distributions should be devised in order to set up a more flexible sampling scheme; the prior used here may be too strict in penalizing slight but useful departures from very smooth arrangements of pointers. Overall, it is hoped that the reviewed MH approach proves useful for the future development of new kinds of SOM samplers including the forementioned reversible jump samplers [6,14], multiple-map samplers [8] and Gibbs samplers [18]. In addition, the class of *adaptive* samplers presented in [7] may be useful to cope with the issue of kernel variance tuning.

Acknowledgement. The author is supported by grants HID98-0379-C02-01 and TIC98-0272-C02-01 from the spanish CICYT agency.

References

1. Bauer, H.-U., Herrmann, M., and Villmann, T. (1999). Neural Maps and Topographic Vector Quantization. *Neural Networks*, Vol. 12, 659–676.
2. Bishop, C. M., Svensén, M., and Williams, K. I. W. (1998). GTM: The Generative Topographic Mapping. *Neural Computation*, Vol. 10, No. 1, 215–235.
3. Bishop, C. M., Svensén, M., and Williams, K. I. W. (1998). Developments of the Generative Topographic Mapping. *Neurocomputing*, Vol. 21, 203–224.
4. Cottrell, M., Fort, J. C., and Pagès, G. (1998). Theoretical aspects of the SOM algorithm. *Neurocomputing*, Vol. 21, 119–138.
5. Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice.* Chapman and Hall.
6. Green, P. J. (1995). Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination. *Biometrika*, Vol. 82, 711–732.
7. Haario, H., Saksman, E., and Tamminen, J. (1999). Adaptive Proposal Distribution for Random Walk Metropolis Algorithm. *Computational Statistics* Vol. 14, No. 3, 375–395.
8. Holmes, C. C. and Mallick, B. K. (1998). Parallel Markov chain Monte Carlo Sampling: an Evolutionary Based Approach. Manuscript available from the MCMC Preprint Service, see `http://www.statslab.cam.ac.uk/~mcmc/`
9. Kiviluoto, K. and Oja, E. (1998). S-map: A Network with a Simple Self-Organization Algorithm for Generative Topographic Mappings. In *Advances in Neural Information Processing Systems* (M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds.), vol. 10, 549–555. MIT Press.
10. Kohonen, T. (1997). *Self-Organizing Maps* (2^{nd} Ed.). Springer-Verlag.
11. Kohonen, T., Hynninen, J., Kangas, L., and Laaksonen, J. (1995). SOM_PAK. The Self-Organizing Map Program Package. Technical Report, Helsinki University of Technology, Finland.
12. Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Paatero, V., and Saarela, A. (2000). Self Organization of a Massive Document Collection. *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, 574–585.
13. Muruzábal, J. and Muñoz, A. (1997). On the Visualization of Outliers via Self-Organizing Maps. *Journal of Computational and Graphical Statistics*, Vol. 6, No. 4, 355–382.
14. Rios Insua, D. and Müller, P. (1998). Feed-Forward Neural Networks for Nonparametric Regression. In *Practical Nonparametric and Semiparametric Bayesian Statistics* (D. Dey, P. Müller and D. Sinha, Eds.), 181–193. Springer-Verlag.
15. Tierney, L. (1994). Markov Chains for Exploring Posterior Distributions. *Annals of Statistics*, Vol. 22, 1701–1762.
16. Utsugi, A. (1997). Hyperparameter Selection for Self-Organizing Maps. *Neural Computation*, Vol. 9, No. 3, 623–635.
17. Utsugi, A. (1998). Density estimation by mixture models with smoothing priors. *Neural Computation*, Vol. 10, No. 8, 2115–2135.
18. Utsugi, A. (2000). Bayesian Sampling and Ensemble Learning in Generative Topographic Mapping. *Neural Processing Letters*, Vol. 12, No. 3, 277–290.