5

Topology-Preserving Mappings for Data Visualisation

Marian Peña, Wesam Barbakh, and Colin Fyfe

Applied Computational Intelligence Research Unit,
The University of Paisley, Scotland,
{marian.pena,wesam.barbakh,colin.fyfe}@paisley.ac.uk

Summary. We present a family of topology preserving mappings similar to the Self-Organizing Map (SOM) and the Generative Topographic Map (GTM) . These techniques can be considered as a non-linear projection from input or data space to the output or latent space (usually 2D or 3D), plus a clustering technique, that updates the centres. A common frame based on the GTM structure can be used with different clustering techniques, giving new properties to the algorithms.

Thus we have the topographic product of experts (ToPoE) with the Product of Experts substituting the Mixture of Experts of the GTM, two versions of the Harmonic Topographic Mapping (HaToM) that utilise the K-Harmonic Means (KHM) clustering, and the faster Topographic Neural Gas (ToNeGas), with the inclusion of Neural Gas in the inner loop. We also present the Inverse-weighted K-means Topology-Preserving Map (IKToM), based on the same structure for non-linear projection, that makes use of a new clustering technique called The Inverse Weighted K-Means. We apply all the algorithms to a high dimensional dataset, and compare it as well with the Self-Organizing Map, in terms of visualisation, clustering and topology preservation.

5.1 Introduction

Topographic mappings are a class of dimensionality reduction techniques that seek to preserve some of the structure of the data in the geometric structure of the mapping. The term "geometric structure" refers to the relationships between distances in data space and the distances in the projection to the topographic map. In some cases all distance relationships between data points are important, which implies a desire for global isometry between the data space and the map space. Alternatively, it may only be considered important that local neighbourhood relationships are maintained, which is referred to as topological ordering [19]. When the topology is preserved, if the projections of two points are close, it is because, in the original high dimensional space, the two points were close. The closeness criterion is usually the Euclidean distance between the data patterns.

One clear example of a topographic mapping is a Mercator projection of the spherical earth into two dimensions; the visualisation is improved, but some of the distances in certain areas are distorted. These projections imply a loss of some of the information which inevitably gives some inaccuracy but they are an invaluable tool for visualisation and data analysis, e.g. for cluster detection. Two previous works in this area have been the Self-Organizing Map (SOM) [13] and the Generative Topographic Map (GTM) [4].

Kohonen's SOM is a neural network which creates a topology-preserving map because there is a topological structure imposed on the nodes in the network. It takes into consideration the physical arrangement of the nodes. Nodes that are "close" together are going to interact differently than nodes that are "far" apart. The GTM was developed by Bishop et al. as a probabilistic version of the SOM, in order to overcome some of the problems of this map, especially the lack of objective function.

Without taking into account the probabilistic aspects of the GTM algorithm, this can be considered as a projection from latent space to dataspace to adapt the nodes to the datapoints (in this chapter called GTM structure), using K-Means with soft responsibilities as clustering technique to update the prototypes in dataspace.

In this chapter we review several topology-preserving maps that make use of the general structure of the GTM. We first review four clustering techniques used in our algorithms in section 5.2. Then we define the common structure based on the GTM in section 5.3.1, and develop the four topology preserving mappings in section 5.3. Finally we compare all the algorithms with the SOM in the experimental section.

5.2 Clustering Techniques

5.2.1 *K*-Means

K-Means clustering is an algorithm to divide or to group samples $\mathbf{x}_i$ based on attributes/features into K groups. K is a positive integer number that has to be given in advance. The grouping is done by minimizing the sum of squares of distances between data and the corresponding prototypes $\mathbf{m}_k$.

The performance function for K-Means may be written as

$$J = \sum_{i=1}^{N} \min_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^2 \,, \tag{5.1}$$

which we wish to minimise by moving the prototypes to the appropriate positions. Note that (5.1) detects only the prototypes closest to data points and then distributes them to give the minimum performance which determines the clustering. Any prototype which is still far from data is not utilised and does

not enter any calculation to determine minimum performance, which may result in dead prototypes, which are never appropriate for any cluster. Thus initializing prototypes appropriately can play a big effect in K-Means.

The algorithm has the following steps:

- Step 1. Begin with a decision on the value of K = number of clusters.
- Step 2. Put any initial partition that divides the data into K clusters randomly.
- Step 3. Take each sample in sequence and compute its distance from the prototypes of each of the clusters. If a sample is not currently in the cluster with the closest prototype, switch this sample to that cluster and update the prototype of the cluster gaining the new sample and the cluster losing the sample.
- Step 4. Repeat step 3 until convergence is achieved, that is until a pass through the training samples causes no new assignments.

Considering a general formula for the updating of the prototypes in clustering techniques we may write a general formula

$$\mathbf{m}_k \leftarrow \frac{\sum_{i=1}^{N} mem(\mathbf{m}_k/\mathbf{x}_i) * weight(\mathbf{x}_i) * \mathbf{x}_i}{\sum_{i=1}^{N} mem(\mathbf{m}_k/\mathbf{x}_i) * weight(\mathbf{x}_i)} \ , \tag{5.2}$$

where

- $weight(\mathbf{x}_i) > 0$ is the weighting function that defines how much influence a data point $\mathbf{x}_i$ has in recomputing the prototype parameters $\mathbf{m}_k$ in the next iteration.
- $mem(\mathbf{m}_k/\mathbf{x}_i) \geq 0$ with $\sum_{k=1}^{K} mem(\mathbf{m}_k/\mathbf{x}_i) = 1$ the membership function that decides the portion of $weight(\mathbf{x}_i) * \mathbf{x}_i$ associated with $\mathbf{m}_k$.

The membership and weight functions for KM are:

$$\begin{aligned} &mem_{KM}(\mathbf{m}_l/\mathbf{x}_i) = \begin{cases} 1 \ , \ if \ \ l = \min_k \|\mathbf{x}_i - \mathbf{m}_k\| \ , \\ 0 \ , \ \text{otherwise} \ ; \end{cases} \\ &weight_{KM}(\mathbf{x}_i) = 1 \ . \end{aligned} \tag{5.3}$$

The main problem with the K-Means algorithm is that, as with the GTM, the initialisation of the parameters can lead to a local minimum. Also the number of prototypes K has to be pre-determined by the user, although this is really one of the objectives of clustering.

5.2.2 K-Harmonic Means

Harmonic Means or Harmonic Averages are defined for spaces of derivatives. For example, if you travel $\frac{1}{2}$ of a journey at 10 km/hour and the other $\frac{1}{2}$ at 20 km/hour, your total time taken is $\frac{d}{10} + \frac{d}{20}$ and so the average speed is

$\frac{2d}{\frac{d}{10}+\frac{d}{20}} = \frac{2}{\frac{1}{10}+\frac{1}{20}}$. In general, the Harmonic Average of K values, $a_1, ..., a_K$, is defined as

$$HA(\{a_i, i = 1, \cdots, K\}) = \frac{K}{\sum_{k=1}^{K} \frac{1}{a_k}} \ . \tag{5.4}$$

Harmonic Means were applied to the K-Means algorithm in [22] to make K-Means a more robust algorithm. The recursive formula to update the prototypes is

$$J = \sum_{i=1}^{N} \frac{K}{\sum_{k=1}^{K} \frac{1}{d(\mathbf{x}_i, \mathbf{m}_k)^2}} \ ; \tag{5.5}$$

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} \frac{1}{d_{ik}^4 (\sum_{l=1}^{K} \frac{1}{d_{il}^2})^2} \mathbf{x}_i}{\sum_{i=1}^{N} \frac{1}{d_{ik}^4 (\sum_{l=1}^{K} \frac{1}{d_{il}^2})^2}} \ , \tag{5.6}$$

where d_{ik} is the Euclidean distance between the i^{th} data point and the k^{th} prototype so that $d(\mathbf{x}_i, \mathbf{m}_k) = \|\mathbf{x}_i - \mathbf{m}_k\|$.

In [22] extensive simulations show that this algorithm converges to a better solution (less prone to finding a local minimum because of poor initialisation) than both standard K-Means or a mixture of experts trained using the EM algorithm.

Zhang subsequently developed a generalised version of the algorithm [20, 21] that includes the p^{th} power of the L^2 distance which creates a "dynamic weighting function" that determines how data points participate in the next iteration in the calculation of the new prototypes $\mathbf{m}_k$. The weight is bigger for data points further away from the prototypes, so that their participation is boosted in the next iteration. This makes the algorithm insensitive to initialisation and also prevents one cluster from taking more than one prototype.

The aim of K-Harmonic Means was to improve the winner-takes-all partitioning strategy of K-Means that gives a very strong relation between each datapoint and its closest prototype, so that the change in membership is not allowed until another prototype is closer. The transition of prototypes between areas of high density is more continuous in K- Harmonic Means due to the distribution of associations between prototypes and datapoints.

The soft membership[1] in the generalised K-Harmonic Means is

$$mem(\mathbf{m}_k/\mathbf{x}_i) = \frac{\|\mathbf{x}_i - \mathbf{m}_k\|^{-p-2}}{\sum_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^{-p-2}} \tag{5.7}$$

allows the data points to belong partly to all prototypes.

[1] Soft membership means that each datapoint can belong to more than one prototype.

The boosting properties for the generalised version of K-Harmonic Means ($p > 2$) are given by the weighting function [9]:

$$weight(\mathbf{x}_i) = \frac{\sum_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^{-p-2}}{(\sum_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^{-p})^2} , \tag{5.8}$$

where the dynamic function gives a variable influence to data in clustering in a similar way to boosting [6] since the effect of any particular data point on the re-calculation of a prototype is $O(\|\mathbf{x}_i - \mathbf{m}_k\|^{2p-p-2})$, which for $p > 2$ has greatest effect for larger distances.

5.2.3 Neural Gas

Neural Gas (NG) [14] is a vector quantization technique with soft competition between the units; it is called the Neural Gas algorithm because the prototypes of the clusters move around in the data space similar to the Brownian movement of gas molecules in a closed container. In each training step, the squared Euclidean distances

$$d_{ik} = \|\mathbf{x}_i - \mathbf{m}_k\|^2 = (\mathbf{x}_i - \mathbf{m}_k)^T * (\mathbf{x}_i - \mathbf{m}_k) \tag{5.9}$$

between a randomly selected input vector $\mathbf{x}_i$ from the training set and all prototypes $\mathbf{m}_k$ are computed; the vector of these distances is $\mathbf{d}$. Each prototype k is assigned a rank $r_k(d) = 0, ..., K-1$, where a rank of 0 indicates the closest and a rank of K-1 the most distant prototype to $\mathbf{x}$. The learning rule is then

$$\mathbf{m}_k = \mathbf{m}_k + \varepsilon * h_\rho[r_k(\mathbf{d})] * (\mathbf{x} - \mathbf{m}_k) . \tag{5.10}$$

The function

$$h_\rho(r) = e^{(-r/\rho)} \tag{5.11}$$

is a monotonically decreasing function of the ranking that adapts not only the closest prototype, but all the prototypes, with a factor exponentially decreasing with their rank. The width of this influence is determined by the neighborhood range ρ. The learning rule is also affected by a global learning rate ε. The values of ρ and ε decrease exponentially from an initial positive value ($\rho(0)$, $\varepsilon(0)$) to a smaller final positive value ($\rho(T)$, $\varepsilon(T)$) according to

$$\rho(t) = \rho(0) * [\rho(T)/\rho(0)]^{(t/T)} \tag{5.12}$$

and

$$\varepsilon(t) = \varepsilon(0) * [\varepsilon(T)/\varepsilon(0)]^{(t/T)} , \tag{5.13}$$

where t is the time step and T the total number of training steps, forcing more local changes with time.

5.2.4 Weighted *K*-Means

This clustering technique was introduced in [3, 2]. We might consider the following performance function:

$$J_A = \sum_{i=1}^{N} \sum_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^2 \ , \tag{5.14}$$

which provides a relationship between all the data points and prototypes, but it doesn't provide useful clustering at minimum performance since

$$\frac{\partial J_A}{\partial \mathbf{m}_k} = 0 \Longrightarrow \mathbf{m}_k = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i, \forall k \ . \tag{5.15}$$

Minimizing the performance function groups all the prototypes to the centre of the data set regardless of the initial position of the prototypes which is useless for identification of clusters.

We wish to form a performance function with following properties:

- Minimum performance gives an intuitively 'good' clustering.
- It creates a relationship between all data points and all prototypes.

(5.14) provides an attempt to reduce the sensitivity to prototypes' initialization by making a relationship between all data points and all prototypes while (5.1) provides an attempt to cluster data points at the minimum of the performance function. Therefore it may seem that what we want is to combine features of (5.1) and (5.14) to make a performance function such as:

$$J_1 = \sum_{i=1}^{N} \left[\sum_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\| \right] \min_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^2 \ . \tag{5.16}$$

As pointed out by a reviewer, there is a potential problem with using $\|\mathbf{x}_i - \mathbf{m}_k\|$ rather than its square in the performance function but in practice, this has not been found to be a problem. We derive the clustering algorithm associated with this performance function by calculating the partial derivatives of (5.16) with respect to the prototypes. We call the resulting algorithm Weighted K-Means (though recognising that other weighted versions of K-Means have been developed in the literature). The partial derivatives are calculated as

$$\frac{\partial J_{1,i}}{\partial \mathbf{m}_r} = -(\mathbf{x}_i - \mathbf{m}_r)\{\|\mathbf{x}_i - \mathbf{m}_r\| + 2\sum_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|\} = -(\mathbf{x}_i - \mathbf{m}_r)a_{ir} \ , \tag{5.17}$$

when $\mathbf{m}_r$ is the closest prototype to $\mathbf{x}_i$ and

$$\frac{\partial J_{1,i}}{\partial \mathbf{m}_k} = -(\mathbf{x}_i - \mathbf{m}_k)\frac{\|\mathbf{x}_i - \mathbf{m}_r\|^2}{\|\mathbf{x}_i - \mathbf{m}_k\|} = -(\mathbf{x}_i - \mathbf{m}_k)b_{ik} \ , \tag{5.18}$$

otherwise.

We then solve this by summing over the whole data set and finding the fixed point solution of

$$\frac{\partial J_1}{\partial \mathbf{m}_r} = \sum_{i=1}^{N} \frac{\partial J_{1,i}}{\partial \mathbf{m}_r} = 0 \tag{5.19}$$

which gives a solution of

$$\mathbf{m}_r = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}} \ . \tag{5.20}$$

We have given extensive analysis and simulations in [3, 2] showing that this algorithm will cluster the data with the prototypes which are closest to the data points being positioned in such a way that the clusters can be identified. However there are some potential prototypes which are not sufficiently responsive to the data and so never move to identify a cluster. In fact, these points move to (a weighted) prototype of the data set. This may be an advantage in some cases in that we can easily identify redundancy in the prototypes however it does waste computational resources unnecessarily.

5.2.5 The Inverse Weighted *K*-Means

Consider the performance algorithm

$$J_2 = \sum_{i=1}^{N} \left[\sum_{k=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \right] \min_{k=1}^{K} \|\mathbf{x}_i - \mathbf{m}_k\|^n \ . \tag{5.21}$$

Let $\mathbf{m}_r$ be the closest prototype to $\mathbf{x}_i$. Then

$$\begin{aligned} J_2(\mathbf{x}_i) &= \left[\sum_{k=1}^{K} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \right] \|\mathbf{x}_i - \mathbf{m}_r\|^n \\ &= \|\mathbf{x}_i - \mathbf{m}_r\|^{n-p} + \sum_{j \neq r} \frac{\|\mathbf{x}_i - \mathbf{m}_r\|^n}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \ . \end{aligned} \tag{5.22}$$

Therefore

$$\begin{aligned} \frac{\partial J_2(\mathbf{x}_i)}{\partial \mathbf{m}_r} &= -(n-p)(\mathbf{x}_i - \mathbf{m}_r)\|\mathbf{x}_i - \mathbf{m}_r\|^{n-p-2} \\ &\quad -n(\mathbf{x}_i - \mathbf{m}_r)\|\mathbf{x}_i - \mathbf{m}_r\|^{n-2} \sum_{j \neq r} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^p} \\ &= (\mathbf{x}_i - \mathbf{m}_r) a_{ir} \ , \end{aligned} \tag{5.23}$$

$$\frac{\partial J_2(\mathbf{x}_i)}{\partial \mathbf{m}_k} = p(\mathbf{x}_i - \mathbf{m}_k) \frac{\|\mathbf{x}_i - \mathbf{m}_r\|^n}{\|\mathbf{x}_i - \mathbf{m}_k\|^{p+2}} = (\mathbf{x}_i - \mathbf{m}_k) b_{ik} \ . \tag{5.24}$$

At convergence, $E(\frac{\partial J_2}{\partial \mathbf{m}_r}) = 0$ where the expectation is taken over the data set. If we denote by V_k the set of points, $\mathbf{x}$ for which $\mathbf{m}_k$ is the closest, we have

$$\frac{\partial J_2}{\partial \mathbf{m}_r} = 0 \Longleftrightarrow \int_{\mathbf{x}\in V_r} \{(n-p)(\mathbf{x}_i - \mathbf{m}_r)\|\mathbf{x}_i - \mathbf{m}_r\|^{n-p-2} + n(\mathbf{x}-\mathbf{m}_r)\|\mathbf{x}-\mathbf{m}_r\|^{n-2} \sum_{j\neq r} \frac{1}{\|\mathbf{x}-\mathbf{m}_j\|^p} P(\mathbf{x})\}\, \mathrm{d}\mathbf{x} + \sum_{k\neq r} \int_{\mathbf{x}\in V_k} p(\mathbf{x}-\mathbf{m}_k) \frac{\|\mathbf{x}-\mathbf{m}_r\|^n}{\|\mathbf{x}-\mathbf{m}_k\|^{p+2}} P(\mathbf{x})\, \mathrm{d}\mathbf{x} = 0\,, \quad (5.25)$$

where $P(\mathbf{x})$ is the probability measure associated with the data set. This is, in general, a very difficult set of equations to solve. However it is readily seen that, for example, in the special case that there are the same number of prototypes as there are data points, that one solution is to locate each prototype at each data point (at which time $\frac{\partial J_2}{\partial \mathbf{m}_r} = 0$). Again solving this over all the data set results in

$$\mathbf{m}_r = \frac{\sum_{i\in V_r} \mathbf{x}_i a_{ir} + \sum_{i\in V_j, j\neq r} \mathbf{x}_i b_{ir}}{\sum_{i\in V_r} a_{ir} + \sum_{i\in V_j, j\neq r} b_{ir}}\,. \quad (5.26)$$

From (5.25), we see that $n \geq p$ if the direction of the first term is to be correct and $n \leq p+2$ to ensure stability in all parts of that equation. In practice, we have found that a viable algorithm may be found by using (5.24) for all prototypes (and thus never using (5.23) for the closest prototype). We will call this the Inverse Weighted K-Means Algorithm.

5.3 Topology Preserving Mappings

5.3.1 Generative Topographic Map

The Generative Topographic Mapping (GTM) is a non-linear latent variable model, intended for modeling continuous, intrinsically low-dimensional probability distributions, embedded in high-dimensional spaces. It provides a principled alternative to the self-organizing map resolving many of its associated theoretical problems. An important, potential application of the GTM is visualization of high-dimensional data. Since the GTM is non-linear, the relationship between data and its visual representation may be far from trivial, but a better understanding of this relationship can be gained by computing the so-called magnification factors [5].

There are two principal limitations of the basic GTM model. The computational effort required will grow exponentially with the intrinsic dimensionality of the density model. However, if the intended application is visualization, this will typically not be a problem. The other limitation is the initialisation of the parameters, that can lead the algorithm to a local optimum.

The GTM defines a non-linear, parametric mapping $\mathbf{y}(\mathbf{x}; W)$ from a q-dimensional latent space to a d-dimensional data space $\mathbf{x} \in R^d$, where normally $q < d$. The mapping is defined to be continuous and differentiable. $\mathbf{y}(\mathbf{t}; W)$ maps every point in the latent space to a point in the data space. Since the latent space is q-dimensional, these points will be confined to a q-dimensional manifold non-linearly embedded into the d-dimensional data space. If we define a probability distribution over the latent space, $p(\mathbf{t})$, this will induce a corresponding probability distribution into the data space. Strictly confined to the q-dimensional manifold, this distribution would be singular, so it is convolved with an isotropic Gaussian noise distribution, given by

$$p\left(\mathbf{x}|\mathbf{t}, W, \beta\right) = \left(\frac{\beta}{2\pi}\right)^{d/2} \exp\left\{-\frac{\beta}{2}\sum_{d=1}^{d}(\mathbf{x}_d - y_d(\mathbf{t}, W))^2\right\}, \tag{5.27}$$

where $\mathbf{x}$ is a point in the data space and β denotes the noise variance. By integrating out the latent variable, we get the probability distribution in the data space expressed as a function of the parameters β and W,

$$p\left(\mathbf{x}|W, \beta\right) = \int p\left(\mathbf{x}|\mathbf{t}, W, \beta\right) p(\mathbf{t})\, \mathrm{d}\mathbf{t}\ . \tag{5.28}$$

Choosing $p(\mathbf{t})$ as a set of K equally weighted delta functions on a regular grid,

$$p(\mathbf{t}) = \frac{1}{K}\sum_{k=1}^{K}\delta(\mathbf{t} - \mathbf{t}_k)\ , \tag{5.29}$$

the integral in (5.28) becomes a sum,

$$p\left(\mathbf{x}|W, \beta\right) = \frac{1}{K}\sum_{k=1}^{K} p\left(\mathbf{x}|\mathbf{t}_k, W, \beta\right)\ . \tag{5.30}$$

Each delta function centre maps into the centre of a Gaussian which lies in the manifold embedded in the data space. This algorithm defines a constrained mixture of Gaussians[11, 12], since the centres of the mixture components can not move independently of each other, but all depend on the mapping $\mathbf{y}(\mathbf{t}; W)$. Moreover, all components of the mixture share the same variance, and the mixing coefficients are all fixed at $1/K$. Given a finite set of independent and identically distributed (i.i.d.) data points, $\{\mathbf{x}_{i=1}^N\}$, the log-likelihood function of this model is maximized by means of the Expectation Maximisation algorithm with respect to the parameters of the mixture, namely W and β. The form of the mapping $\mathbf{y}(\mathbf{t}; w)$ is defined as a generalized linear regression model $\mathbf{y}(\mathbf{t}; W) = \phi(\mathbf{t})W$ where the elements of $\phi(\mathbf{t})$ consist of M fixed basis functions $\phi_i(\mathbf{t})_{i=1}^M$, and W is a $d \times M$ matrix.

If we strip out the probabilistic underpinnings of the GTM method, the algorithm can be considered as a non-linear model structure, to which a clustering technique is applied in data space to update the prototypes, in this case the K-Means algorithm. In the next sections we present four algorithms that share this model structure.

5.3.2 Topographic Product of Experts ToPoE

Hinton [10] investigated a product of K experts with

$$p(\mathbf{x}_i|\Theta) \propto \prod_{k=1}^{K} p(\mathbf{x}_i|k) \ , \tag{5.31}$$

where Θ is the set of current parameters in the model. Hinton notes that using Gaussians alone does not allow us to model e.g. multi-modal distributions, however the Gaussian is ideal for our purposes. Thus the base model is

$$p(\mathbf{x}_i|\Theta) \propto \prod_{k=1}^{K} \left(\frac{\beta}{2\pi}\right)^{\frac{D}{2}} \exp\left(-\frac{\beta}{2}||\mathbf{m}_k - \mathbf{x}_i||^2 \ .\right) . \tag{5.32}$$

To fit this model to the data we can define a cost function as the negative logarithm of the probabilities of the data so that

$$J = \sum_{i=1}^{N}\sum_{k=1}^{K} \frac{\beta}{2}||\mathbf{m}_k - \mathbf{x}_i||^2 \ . \tag{5.33}$$

In [7] the Product of Gaussian model was extended by allowing latent points[2] to have different responsibilities depending on the data point presented:

$$p(\mathbf{x}_i|\Theta) \propto \prod_{k=1}^{K} \left(\frac{\beta}{2\pi}\right)^{\frac{D}{2}} \exp\left(-\frac{\beta}{2}||\mathbf{m}_k - \mathbf{x}_i||^2 r_{ik}\right) \ , \tag{5.34}$$

where r_{ik} is the responsibility of the k^{th} expert for the data point, $\mathbf{x}_i$. Thus all the experts are acting in concert to create the data points but some will take more responsibility than others. Note how crucial the responsibilities are in this model: if an expert has no responsibility for a particular data point, it is in essence saying that the data point could have a high probability as far as it is concerned. We do not allow a situation to develop where no expert accepts responsibility for a data point; if no expert accepts responsibility for a data point, they all are given equal responsibility for that data point (see below).

[2] The latent points, $\mathbf{t}_k$, generate the $\mathbf{m}_k$ prototypes, which are the latent points' projections in data space; thus there is a bijection between the latent points and the prototypes. $\mathbf{m}_k$ act as prototypes of the clusters.

We wish to maximise the likelihood of the data set $X = \{\mathbf{x}_i : i = 1, \cdots, N\}$ under this model. The ToPoE learning rule (5.36) is derived from the minimisation of $-\log(p(\mathbf{x}_i|\Theta))$ with respect to a set of parameters which generate the $\mathbf{m}_k$.

We may update W either in batch mode or with online learning. To change W in online learning, we randomly select a data point, say $\mathbf{x}_i$. We calculate the current responsibility of the k^{th} latent point for this data point,

$$r_{ik} = \frac{exp(-\gamma d_{ik}^2)}{\sum_{k=1}^{K} exp(-\gamma d_{ik}^2)} , \tag{5.35}$$

where $d_{pq} = ||\mathbf{x}_p - \mathbf{m}_q||$, the Euclidean distance between the p^{th} data point and the projection of the q^{th} latent point (through the basis functions and then multiplied by W). If no prototypes are close to the data point (the denominator of (5.35) is zero), we set $r_{ik} = \frac{1}{K}, \forall k$. γ is known as the width of the responsibilities and is usually set to 20.

We wish to maximise (5.34) so that the data is most likely under this model. We do this by minimising the -log() of that probability: define $m_d^{(k)} = \sum_{\omega=1}^{M} w_{\omega d}\phi_{k\omega}$, i.e. $m_d^{(k)}$ is the projection of the k^{th} latent point on the d^{th} dimension in data space. Similarly let $x_d^{(n)}$ be the d^{th} coordinate of $\mathbf{x}_i$. These are used in the update rule

$$\Delta_i w_{\omega d} = \sum_{k=1}^{K} \eta\phi_{k\omega}(x_d^{(i)} - m_d^{(k)})r_{ik} , \tag{5.36}$$

where we have used Δ_i to signify the change due to the presentation of the i^{th} data point, $\mathbf{x}_i$, so that we are summing the changes due to each latent point's response to the data points. Note that, for the basic model, we do not change the Φ matrix during training at all.

5.3.3 The Harmonic Topograpic Map

The HaToM has the same structure as the GTM, with K latent points that are mapped to a feature space by M Gaussian basis functions, and then into the data space by a matrix of weights W. In HaToM the initialisation problems of GTM are overcomed replacing the arithmetic means of K-Means algorithm with harmonic means, i.e. using K-Harmonic Means [22].

The basic batch algorithm often exhibited twists, such as are well-known in the Self-organizing Map (SOM) [13], so we developed a growing method that prevents the mapping from developing these twists. The latent points are arranged in a square grid in a similar manner to the SOM grid.

We developed two versions of the algorithm [17]. The main structure for the data-driven HaToM or D-HaToM is as follows:

1. Initialise K to 2. Initialise the W weights randomly and spread the centres of the M basis functions uniformly in latent space.
2. Initialise the K latent points uniformly in latent space.
3. Calculate the projection of the latent points to data space. This gives the K prototypes, $\mathbf{m}_k$.
   a) count=0
   b) For every data point, $\mathbf{x}_i$, calculate $d_{ik} = ||\mathbf{x}_i - \mathbf{m}_k||$.
   c) Recalculate prototypes, $\mathbf{m}_k$, using (5.6).
   d) If count<MAXCOUNT, count= count +1 and return to 3b
4. Recalculate W using $(\Phi^T\Phi + \delta I)^{-1}\Phi^T\Xi$ where Ξ is the matrix containing the K prototypes, I is identity matrix and δ is a small constant[a], necessary because initially $K < M$ and so the matrix $\Phi^T\Phi$ is singular.
5. If $K < K_{\max}$, $K = K + 1$ and return to 2.

[a] usually 0.001 but other values gave similar results

We do not randomise W each time we augment K, but we use the value from the previous iteration to update the prototypes $\mathbf{m}_k$ with the increased number of latent points.

If we wish to use the mapping for visualisation, we must map data points into latent space. We define the responsibility as in (5.35), and the i^{th} data point is represented by $\mathbf{y}_i$ where

$$\mathbf{y}_i = \sum_{k=1}^{K} r_{ik}\mathbf{t}_k \ , \tag{5.37}$$

where $\mathbf{t}_k$ is the position of the k^{th} latent point in latent space.

In the model-driven HaToM or M-HaToM, we give greater credence to the model by recalculating W and hence the prototypes, $\mathbf{m}_k$, within the central loop each time. Thus we are explicitly forcing the structure of the M-HaToM model on the data. The visualisation of the $\mathbf{y}_i$ values in latent space is the same as above.

In [17], we showed that this version had several advantages over the D-HaToM: in particular, the M-HaToM creates tighter clusters of data points and finds an underlying data manifold smoothly no matter how many latent points are used in creating the manifold. The D-HaToM, on the other hand, is too responsive to the data (too influenced by the noise), but this quality makes it more suitable for outlier detection.

Generalised Harmonic Topographic Map (G-HaToM)

The generalised version of K-Harmonic Means can be applied also to the HaToM algorithm. The advantage of this generalisation is the utilisation of a "p" value that, when bigger than 2, gives a boosting-like property to the

updating of the prototypes. The recalculation of the prototypes in this case is:

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} \frac{1}{d_{ik}^p (\sum_{l=1}^{K} \frac{1}{d_{il}^2})^{p-2}} \mathbf{x}_i}{\sum_{i=1}^{N} \frac{1}{d_{ik}^p (\sum_{l=1}^{K} \frac{1}{d_{il}^2})^{p-2}}} , \tag{5.38}$$

so that p determines the power of the L^2 distance used in the algorithm.

This generalised version of the algorithm includes the p^{th} power of the L^2 distance which creates a "dynamic weighting function" [20] that determines how data points participate in the next iteration to calculate the new prototypes $\mathbf{m}_k$. The weight is bigger for data points further away from the prototypes, so that their participation is boosted in the next iteration. This makes the algorithm insensitive to initialisation and also prevents one cluster from taking more than one prototype.

Some resuls for the generalised version of HaToM can be seen in [16].

5.3.4 Topographic Neural Gas

Topographic Neural Gas (ToNeGas) [18] unifies the underlying structure in GTM for topology preservation, with the technique of Neural Gas (NG). The prototypes in data space are then clustered using the NG algorithm. The algorithm has been implemented based on the Neural Gas algorithm code included in the SOM Toolbox for Matlab [15].

We have used the same growing method as with HaToM but have found that, with the NG learning, we can increment the number of latent points by e.g. 10 each time we augment the map whereas with HaToM, the increase can only be one at a time to get a valid mapping. One of the advantages of this algorithm is that the Neural Gas part is independent of the non-linear projection, thus the clustering efficiency is not limited by the topology preservation restriction.

5.3.5 Inverse-Weighted *K*-Means Topology-Preserving Map

As with KHM and NG, it is possible to extend the IWKM clustering algorithm to provide a new algorithm for visualization and topology-preserving mappings, by using IWKM with the GTM structure. We called the new algorithm Inverse-weighted K-Means Topology-Preserving Map (IKToM).

5.4 Experiments

We use a dataset containing results of a high-throughput experimental technology application in molecular biology (microarray data [8])[3]. The datasets

[3] `http://www.ihes.fr/~zinovyev/princmanif2006/` Dataset II - "Three types of bladder cancer".

contains only 40 observations of high-dimensional data (3036 dimensions) and the data is drawn from three types of bladder cancer: T1, T2+ and Ta. The data can be used in the gene space (40 rows of 3036 variables), or in the sample space (3036 rows of 40 variables), where each sample contains the profiles of the 40 genes. In these experiments we consider the first case. The dataset has been preprocessed to have zero mean; also, in the original dataset some data was missing and these values have been filtered out.

We use the same number of neurons for all the mappings, a 12*12 grid. We used a value of $p = 3$ for HaToM, and $p = 7$ for IKToM. For several of the results below we have utilised the SOMtoolbox [15] with default values.

5.4.1 Projections in Latent Space

The four algorithms are able to properly separate the three types of cancer in the projection (see Figs. 5.1 and 5.2), but ToPoE requires to run for 100,000 iterations while the others do it with only 20 passes or less.

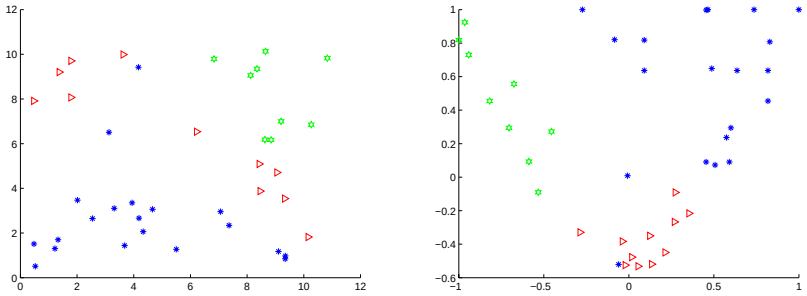

Fig. 5.1. The ToPoE (left) and HaToM (right) projection for the gene data with p=6. T1 in triangles (red), T2+ in circles (green) and Ta in stars (blue)

5.4.2 Responsibilities

The responsibilities of each latent point for each datapoint are shown in Fig. 5.3.

5.4.3 U-matrix, Hit Histograms and Distance Matrix

The U-Matrix assigns to each cell the average distance to all of its neighbors. This enables the identification of regions of similarity using different colors for different ranges of distances.

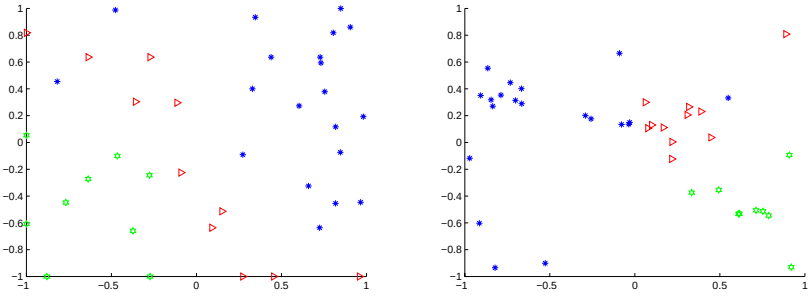

Fig. 5.2. The ToNeGas (left) and IKToM (right) projection for the gene data. T1 in triangles (red), T2+ in circles (green) and Ta in stars (blue)

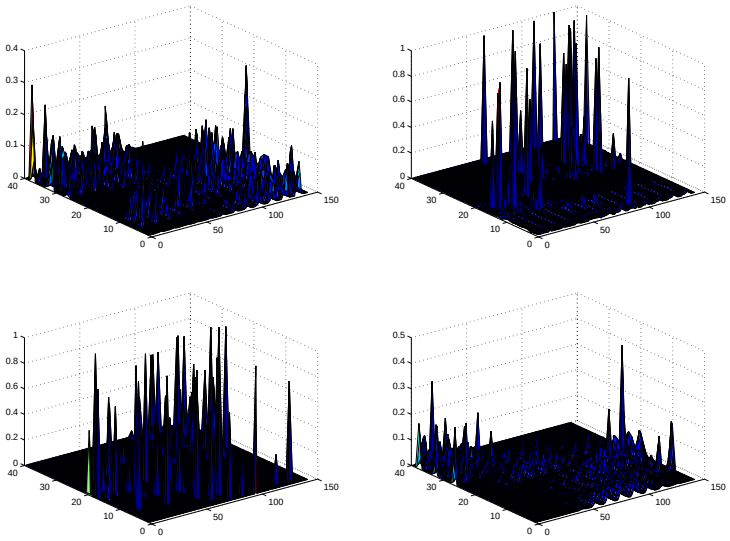

Fig. 5.3. Responsibilities for ToPoE, HaToM, ToNeGas and IKToM. In each diagram the latent points are on the right axis, the data points on the left and the vertical axis measures the responsibilities

The hit histogram are formed by finding the Best Matching Unit (BMU) of each data sample from the map, and increasing a counter in a map unit each time it is the BMU. The hit histogram shows the distribution of the data set on the map. Here, the hit histogram for the whole data set is calculated and visualized with the U-matrix (Figs. 5.4, 5.5, 5.6).

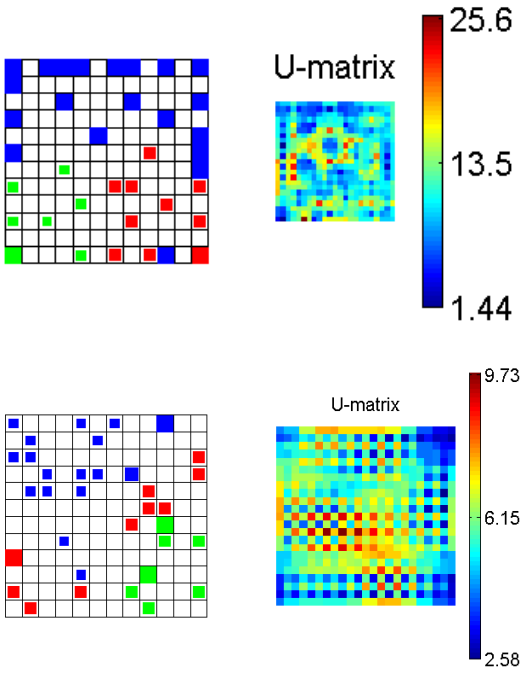

Fig. 5.4. Hit histogram and U-matrix for SOM (top) and ToPoE (bottom)

The hits histograms show that all SOM, ToPoE, HaToM, ToNeGas and IKToM have separate areas in the grids that are responsible for the different classes of genes (with higher distances in between clusters shown in the U-matrix), with only one blue point that appears as outlier for both of them.

Surface plot of distance matrix (Fig. 5.7): both color and vertical-coordinate indicate average distance to neighboring map units. This is closely related to the U-matrix.

The distance matrix is similar to the U-matrix and therefore gives similar results.

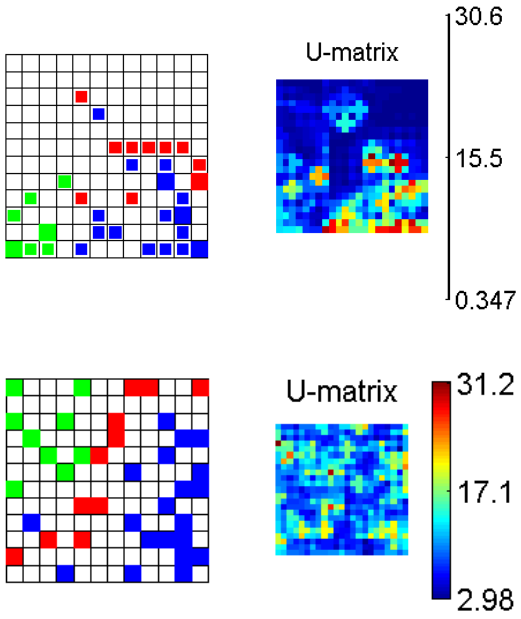

Fig. 5.5. Hit histogram and U-matrix for HaToM (top) and ToNeGas (bottom)

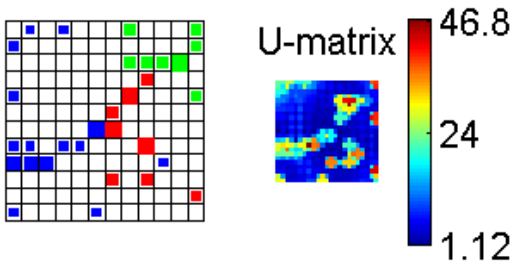

Fig. 5.6. Hit histogram and U-matrix for IKToM

5.4.4 The Quality of The Map

Any topology preserving map requires a few parameters (such as size and topology of the map or the learning parameters) to be chosen a priori, and this influences the goodness of the mapping. Typically two evaluation criteria are used: resolution and topology preservation. If the dimension of the data set is higher than the dimension of the map grid, these usually become contradictory goals.

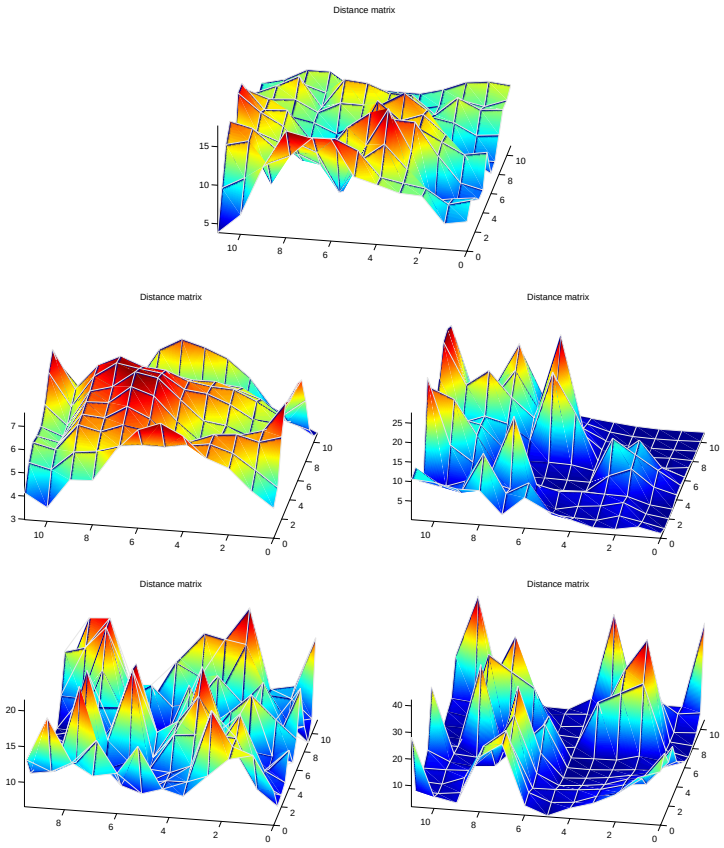

Fig. 5.7. Distance matrix for SOM, ToPoE, HaToM, ToNeGas, and IKToM

We first analyze the quantization error for each datapoint with the distance to its Best Matching Unit (BMU). The mean quantization error q_e is the average distance between each data vector and its BMU; it measures then the resolution of the mapping.

$$q_e = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - (BMU(i), k)\| \ . \tag{5.39}$$

ToPoE and HaToM are much worse than the other three mappings suggesting that their prototypes are further from the data.

The distortion measure which measures the deviation between the data and the quantizers is defined as:

$$E = \sum_{i=1}^{N} \sum_{k=1}^{K} h(BMU(i), k) \|\mathbf{m}_k - \mathbf{x}_i\|^2. \tag{5.40}$$

We first calculate the total distortion for each unit, and then average for the total number of neurons.

Another important measure of the quality of the mapping is the topology preservation. In this case we calculate the topographic error, t_e, i.e. the proportion of all data vectors for which first and second BMUs are not adjacent units.

$$t_e = \frac{1}{N}\sum_{i=1}^{N} u(\mathbf{x}_i) \tag{5.41}$$

where $u(\mathbf{x}_i)$ is equal to 1 if first and second BMU are adjacent and 0 otherwise.

This t_e does not consider diagonal neighbors, thus the hexagonal case in SOM always gives lower values of t_e due to its six neighbors for each unit in comparison to the four in the rectangular mapping used for the other three algorithms. We may use a different topographic error, such as the Alfa error [1] which considers also the diagonal neighbors in the rectangular case (though now the rectangular mappings have an advantage in that they have 8 neighbours). The formula for the alpha error is as follows:

$$Alfa = \frac{1}{N}\sum_{i=1}^{N} \alpha(\mathbf{x}_i) \tag{5.42}$$

where $\alpha(\mathbf{x}_i)$ is equal to 1 if first and second BMU are adjacent or diagonals and 0 otherwise.

Table 5.1. Quantization error and topology preservation error with topology-preserving Mappings for the gene data

Algorithm	SOM	ToPoE	HaToM	ToNeGas	IKToM
Map Size	(12*12)	(12*12)	(12*12)	(12*12)	(12*12)
Mean Quantization Error	11.8813	22.4106	22.3085	8.8433	13.7959
Average total distortion for each unit (e+003)	0.597	0.8924	1.3571	0.8514	1.1074
Topology preservation error	0.0250	0.2500	0.7500	0.2000	0.4500
Alfa error	0	0.0250	0.6750	0.1000	0.4000

The lower clustering errors are for ToNeGas, closely followed by SOM. The topology is completely preserved in SOM, but also quite well in ToPoE and ToNeGas. The other two have higher topology errors in this experiment.

5.5 Conclusions

We have developed four new Topology preserving mappings based on the Generative Topographic Mapping. Each algorithm applies a different clustering technique, providing the whole with particular advantages: HaToM can be

used in a data-driven or model-driven version, which are useful for different situations; both HaToM and ToNeGas overcome the problem of local minima with their clustering technique. ToPoE does not require a growing mode, while ToNeGas can apply the growing faster. Finally the Inverse Weighted K-Means has proven to improve situations where the initialisation of the prototypes are not randomly positioned. All four algorithms were applied to a high dimensional dataset, and all properly separated the clusters in the projection space.

References

1. Arsuaga-Uriarte, E. and Daz-Martn, F.: Topology preservation in SOM. Transactions On Engineering, Computing And Technology, **15**, 1305–5313 (2006)
2. Barbakh, W., Crowe, M., and Fyfe, C.: A family of novel clustering algorithms. In: 7th international conference on intelligent data engineering and automated learning, IDEAL2006 (2006)
3. Barbakh, W. and Fyfe, C.: Performance functions and clustering algorithms. Computing and Information Systems, **10** (2), 2–8 (2006) ISSN 1352-9404.
4. Bishop, C.M., Svensen, M., and Williams, C.K.I.: Gtm: The generative topographic mapping. Neural Computation, **10** (1), 215–234 (1997)
5. Bishop, C.M., Svensen, M., and Williams, C.K.I.: Magnification Factors for the GTM Algorithm. In: Proceedings of the IEE 5th International Conference on Artificial Neural Networks, Cambridge, U.K., 64–69P (1997)
6. Friedman, J., Hastie, T., and Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics, **28**, 337–374 (2000)
7. Fyfe, C.: Two topographic maps for data visualization. Mining and Knowledge Discovery (2007)
8. Dyrskjot, L., Thykjaer, T., Kruhoffer, M. et al.: Identifying distinct classes of bladder carcinoma using microarrays. Nat Genetics **33** (1), 90–96 (2003)
9. Hamerly, G. and Elkan, C.: Alternatives to the k-means algorithm that find better clusterings. In: CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management, pages 600–607, New York, NY, USA, ACM Press (2002)
10. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Computational Neuroscience Unit, University College, London (2000)
11. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation, **3**, 79–87 (1991)
12. Jordan, M.I. and Jacobs, R.A.: Hierarchical mixtures of experts and the em algorithm. Neural Computation, **6**, 181–214 (1994)
13. Kohonen, T.: Self-Organising Maps. Springer (1995)
14. Martinetz, T.M., Berkovich, S.G., and Schulten, K.J.: 'neural-gas' network for vector quantization and its application to time-series prediction. IEEE Transactions on Neural Networks., **4** (4), 558–569 (1993)
15. Neural Networks Research Centre. Helsinki University of Technology. Som toolbox. www.cis.hut.fi/projects/somtoolbox.
16. Peña, M. and Fyfe, C.: Developments of the generalised harmonic topographic mapping. *WSEAS Transactions On Computers*, **4** (11), 1548–1555 (2005)

17. Peña, M. and Fyfe, C.: Model- and data-driven harmonic topographic maps. *WSEAS Transactions On Computers*, **4** (9), 1033–1044, (2005)
18. Peña, M. and Fyfe, C.: The topographic neural gas. In: *7th International Conference on Intelligent Data Engineering and Automated Learning, IDEAL06.*, pages 241–249 (2006)
19. Tipping, M.E.: Topographic Mappings and Feed-Forward Neural Networks. PhD thesis, The University of Aston in Birmingham (1996)
20. Zhang, B.: Generalized k-harmonic means – boosting in unsupervised learning. Technical Report HPL-2000-137, HP Laboratories, Palo Alto, October (2000)
21. Zhang, B.: Generalized k-harmonic means– dynamic weighting of data in unsupervised learning. In: First SIAM international Conference on Data Mining (2001)
22. Zhang, B., Hsu, M., and Dayal, U.: K-harmonic means - a data clustering algorithm. Technical Report HPL-1999-124, HP Laboratories, Palo Alto, October (1999)