K

KERNEL-BASED TOPOGRAPHIC MAPS: THEORY AND APPLICATIONS

INTRODUCTION

One of the most compelling features of the mammalian brain is the *topographical* organization of its sensory cortex and the parcellation of the latter into cytoarchitectonic areas: neighboring nerve cells (neurons) that can be driven by stimuli originating from neighboring positions in the sensory input space, and neighboring neurons in a given area that project to neighboring neurons in the next area. In other words, the connections establish a so-called *neighborhood-preserving* or *topology-preserving* map, *topographic map* for short. In the case of the visual cortex, such a map is said to be *retinotopic*, and as a result of the neighborhood-preserving mapping, the two-dimensional retinal image is repeatedly mapped out at different levels in the visual cortical hierarchy (1–3). Similarly, the somatosensory cortex contains a *somatotopic map* of the body surface (4, 5), and the auditory cortex contains a *tonotopic map* of the spectrum of possible sounds, laid out according to pitch (6, 7).

Already in 1956, Sperry had begun his pioneering inquiries into the general question of how visual pathways might maintain topological order during embryonic development and had put forward a model for the construction of retinotopic maps. Von der Malsburg also realized that the connections cannot be entirely genetically preprogrammed and introduced a self-organizing process for the *local* ordering of neurons (8)—self-organization refers to the genesis of globally ordered structures out of local interactions. His computer simulations were perhaps the first to demonstrate self-organization.

The study of topographic map formation from a theoretical (i.e., formal modeling) perspective, started with basically two types of self-organizing processes, gradient-based learning and competitive learning, and two types of network architectures (Fig. 1) (for a review, see Ref. 9). In the first architecture, two sets of formal (abstract) neurons are arranged in two separate (one-or) two-dimensional layers or *lattices*[1] (Fig. 1a). The problem is then to learn a mapping for which neighboring neurons in the input lattice are connected to neighboring neurons in the output lattice. This determination is called *graph matching*, and it could play an important role in cortical organization. The network structure is commonly known as the Willshaw–von der Malsburg model (10).

The second architecture is less meant to be biological but is far more studied. We now have continuously valued inputs taken from an input region that need not be rectangular nor have the same dimensionality as the output lattice to which it projects (Fig. 1b). The problem here is to learn a mapping from the input space onto the lattice in such a way that neighboring lattice neurons code for neighboring *positions* in the input space. This network structure is often referred to as Kohonen's model because the popular self-organizing map (SOM) algorithm is applied to it (11, 12). We will introduce the basic version of the SOM algorithm and further introduce an alternative formulation that shows the connection to kernel-based topographic map formation, the central topic of this article.

SOM ALGORITHM

In its purest form, the SOM algorithm distinguishes two stages: the *competitive* stage and the *cooperative* stage. In the first case, the best matching neuron is selected, that is, the "winner," and in the second stage, the weights of the winner are adapted as well as those of its immediate lattice neighbors. We consider the minimum Euclidean distance version of the SOM algorithm only (also the dot product version exists; see Ref. 13).

Competitive Stage. Let A be a lattice of N neurons with weight vectors $\mathbf{w}_i = [w_{ij}] \in \mathbb{R}^d$, $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)$. All neurons receive the same input vector $\mathbf{v} = [v_1, \ldots, v_d] \in V \subseteq \mathbb{R}^d$. For each input $\mathbf{v}$, we select the neuron with the smallest Euclidean distance ("winner-takes-all", WTA):

$$i^* = \arg\min_i \|\mathbf{w}_i - \mathbf{v}\| \tag{1}$$

By virtue of the minimum Euclidean distance rule, we obtain a Voronoi tessellation of the input space: To each neuron corresponds a region in the input space, the boundaries of which are perpendicular bisector planes of lines that join pairs of weight vectors (the gray shaded area in Fig. 2 is the Voronoi region of neuron j). Note that the neuron weights are connected by straight lines: They represent the neurons that are nearest neighbors in the lattice. They are important for verifying whether the map is topology preserving.

Cooperative Stage. It is now crucial to the formation of topographically ordered maps that the neuron weights are not modified independently of each other but as topologically related subsets on which similar kinds of weight updates are performed. During learning, not only the weight vector of the winning neuron is updated but also its lattice neighbors, which, thus, end up responding to similar inputs. This updating is achieved with the neighborhood function, which is centered at the winning neuron and decreases with the lattice distance to the winning neuron.

[1] A lattice is an undirected graph in which every nonborder vertex has the same, fixed number of incident edges and which usually appears in the form of an array with a rectangular or simplex topology.

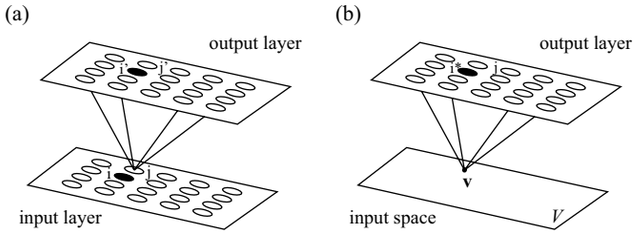

Figure 1. Two types of network architectures used for studying topographic map formation. (a) Willshaw–von der Malsburg model. Two isomorphic, rectangular lattices of formal neurons are shown: One represents the input layer, and the other represents the output layer. Neurons are represented by circles: Filled circles denote active neurons ("winning" neurons); open circles denote inactive neurons. Because of the weighted connections from the input to the output layer, the output neurons receive different inputs from the input layer. Two input neurons are labeled (i, j), and their corresponding output layer neurons are labeled (i', j'). Neurons i and i' are the only active neurons in their respective layers. (b) Kohonen model. The common input all neurons receive is directly represented in the input space, $\mathbf{v} \in V$. The "winning" neuron is labeled as i^*: Its weight (vector) is the one that best matches the current input (vector).

The weight update rule in incremental mode[2] is given by:

$$\Delta \mathbf{w}_i = \eta \Lambda(i, i^*, \sigma_\Lambda(t))(\mathbf{v} - \mathbf{w}_i), \quad \forall_i \in A \tag{2}$$

with Λ as the neighborhood function, that is, a scalar-valued function of the lattice coordinates of neurons i and i^*,$\mathbf{r}_i$ and $\mathbf{r}_i^*$, mostly a Gaussian:

$$\Lambda(i, i^*) = \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{i^*}\|^2}{2\sigma_\Lambda^2}\right) \tag{3}$$

with range σ_Λ (i.e., the standard deviation). (We also drop the parameter $\sigma_\Lambda(t)$ from the neighborhood function to simplify our notation.) The positions $\mathbf{r}_i$ are usually taken to be the nodes of a discrete lattice with a regular topology, usually a two-dimensional square or rectangular lattice. The parameter σ_Λ and usually also the learning rate η, are decreased gradually over time. When the neighborhood range vanishes, the previous learning rule reverts to standard unsupervised competitive learning (note that the latter cannot form topology-preserving maps, which points to the importance of the neighborhood function).

As an example, we train a 5×5 square lattice with the SOM algorithm on a uniform square distribution $[-1,1]^2$, using a Gaussian neighborhood function of which the range $\sigma_\Lambda(t)$ is decreased as follows:

$$\sigma_\Lambda(t) = \sigma_{\Lambda 0} \exp\left(-2\sigma_{\Lambda 0} \frac{t}{t_{\max}}\right) \tag{4}$$

with t the current time step, $t_{\max}$ the maximum number of time steps, and $\sigma_{\Lambda 0}$ the range spanned by the neighborhood function at $t = 0$. We take $t_{\max} = 100{,}000$, and $\sigma_{\Lambda 0} = 2.5$, and the learning rate $\eta = 0.01$. The initial weights (i.e., for $t = 0$) are chosen randomly from the same square distribution. Snapshots of the evolution of the lattice are shown in Fig. 3. We observe that the lattice initially is tangled, then contracts, unfolds, and expands to span the input distribution. We will reuse this example when illustrating the other topographic map formation algorithms discussed in this article. The astute reader has noticed that at the end of the learning phase, the lattice is smooth but then suddenly becomes more erratic (around $t = 50{,}000$). This example is an example of a phase transition, and it has been widely studied for the SOM algorithm (see, e.g., Ref. 14).

[2]By incremental mode it is meant that the weights are updated each time an input vector is presented. This mode is contrasted with batch mode in which the weights are only updated after the presentation of the full training set ("batch").

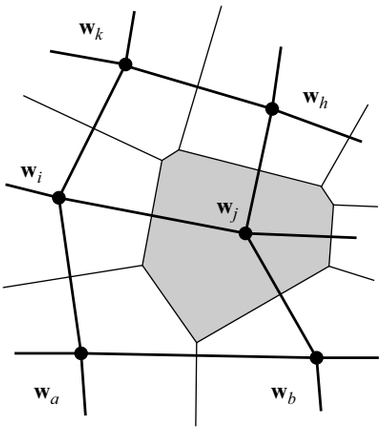

Figure 2. Definition of quantization region in the self-organizing map (SOM). Portion of a rectangular lattice (thick lines) plotted in terms of the weight vectors of neurons $a, \ldots, k$, in two-dimensional input space, for example, $\mathbf{w}_a, \ldots, \mathbf{w}_k$.

It is important to note that no well-defined objective function is found on which the SOM algorithm performs gradient descent. Kohonen (13) motivated the lack of an exact optimization by a link to an approximate one, based on the Robbins–Munro stochastic optimization. This lack of a well-defined objective function has motivated Luttrell (16) and Heskes and Kappen (17) to develop topographic map rules from energy functions. This being said, the use of the Euclidean distance in the algorithm because it also forms the argument of a Gaussian kernel has motivated several authors to connect the SOM algorithm to homoscedastic Gaussian mixture density modeling (18–22), among others.

The weight distribution of the SOM is a power law of the true input density (for a review, see Ref. 9), which means that when estimating the input density from the weight distribution directly [(e.g., using averaged nearest-neighbor neuron weight distances, such as in the heuristical U-matrix (23)], the low-density regions are underestimated (and vice versa), which causes a potential masking of cluster boundaries. Attempts have been introduced to modify the SOM algorithm so that the weight density is a linear function of the input density (such as in the BDH algorithm (24), among others). But a principled density estimation requires the estimation of the Voronoi volumes, which is very difficult for densities larger than 2 (several approximations for this have been suggested, e.g., Refs. 24 and 25). A more elegant solution involves using kernels locally adapted to the input

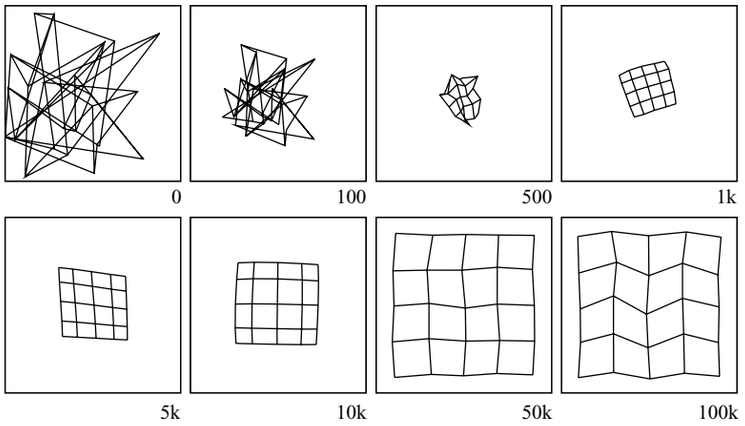

Figure 3. Evolution of a 5×5 lattice with a rectangular topology as a function of time. The outer squares outline the uniform input distribution. The values given below the squares represent time.

density, thereby avoiding the estimation of Voronoi volumes or other heuristics, and providing a natural interpolating facility. This solution is actually the motivation to use kernel-based topographic maps: combining the visualization properties of topographic maps with improved density estimation facilities. The latter is particularly important when clustering applications are envisaged.

Finally, the SOM algorithm has led to literally thousands of applications in areas that range from automatic speech recognition, condition monitoring of plants and processes, cloud classification, and microarray data analysis, to document and image organization and retrieval. They will not be reviewed here; instead we refer to Ref. 26 (http://www.cis.hut.fi/research/-som-bibl/).

KERNEL-BASED TOPOGRAPHIC MAPS

Rather than developing topographic maps with disjoint and uniform activation regions (Voronoi tessellation), such as in the case of the SOM algorithm (Fig. 2), and its adapted versions, algorithms have been introduced that can accommodate neurons with overlapping activation regions, usually in the form of kernel functions, such as Gaussians (Fig. 4). For these *kernel-based topographic maps,* or *kernel topographic maps,* as they are called (they are also sometimes called *probabilistic topographic maps* because they model the input density), several learning principles have been proposed. In the next sections, we will review the kernel-based topographic map formation algorithms.

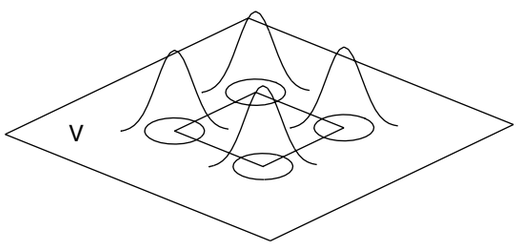

Figure 4. Kernel-based topographic maps. Example of a 2×2 map (cf. rectangle in V-space) for which each neuron has a Gaussian kernel as output function. We will use the more condensed representation in which a circle is drawn with its center the neuron weight vector and its radius the kernel range, for each neuron.

One motivation to use kernels, besides the biological relevance, is to improve the density estimation properties of topographic maps. In this way, we can combine the unique visualization properties of topographic maps with an improved modeling of clusters in the data using kernels. An early example is the elastic net of Durbin and Willshaw (27), which can be viewed as an equal-variance or homoscedastic Gaussian mixture density model, fitted to the data by a penalized maximum likelihood term. Other examples of the density modeling approach are the algorithms introduced by Bishop and coworkers (28, 29) (generative topographic map, based on constrained, homoscedastic Gaussian mixture density modeling with equal mixings), Utsugi (18) (also using equal mixings of homoscedastic Gaussians), and Van Hulle (30) (equiprobabilistic maps using heteroscedastic Gaussian mixtures). Furthermore, we should also mention the fuzzy membership in clusters approach of Graepel and coworkers (31) and the maximization of local correlations approach of Xu and coworkers (32), both of which rely on homoscedastic Gaussians. Heskes (20) showes the connection between minimum distortion topographic map formation and maximum likelihood homoscedastic Gaussian mixture density modeling (GMM). A unifying account of the heteroscedastic case was introduced in Ref. 33.

Kernel-based topographic map formation algorithms can be classified in several ways: by the type of objective function (likelihood, distortion, free energy, and information-theoretic criteria), the type of learning principle (incremental, batch, and fixed point learning[3]), and the

[3]With fixed point learning we mean that the solution is obtained by computing the fixed point of an iterated function. For example, an incremental learning rule can be converted into a fixed point rule by iterating directly on the equilibrium values of the parameter sought.

type of kernel, to mention a few. We will adopt a simple division into homoscedastic and heteroscedastic Gaussian kernels, because the majority of the algorithms are for homoscedastic Gaussians, and the heteroscedastic case is considered as an extension. Where possible, we will list the objective functions and the fixed point rules used for optimizing them. We will show the correspondences between the different algorithms and indicate when a proposed algorithm is unlikely to unfold lattices. We will also list several applications of the homoscedastic algorithms (but without going into details) and show the performance of the heteroscedastic algorithms. Finally, we will mention algorithms that use other kernel types and also thresholded kernels.

Homoscedastic Kernel Topographic Map Formation

In this section, we will review several topographic map formation algorithms that use homoscedastic Gaussian kernels as activation functions. The order in which they appear does not indicate any preference.

SOM Algorithm. The starting point is again Kohonen's SOM algorithm. We can show that the SOM algorithm is a limiting case of a kernel-based topographic map algorithm. We adopt the formalism of Kostiainen and Lampinen (21) (put into our notation); a version starting from the batch map (13) is shown later. Theoretically, the SOM algorithm does not perform gradient descent on the error function in Equation (5) but rather performs approximates of it in the sense of a Robbins-Munro stochastic approximation (13). For the sake of the current discussion. we consider the converged state of an SOM as a local minimum in the error function (average distortion; "energy") (34) and assume that it can be reached or approximated:

$$E(\mathbf{W}) = \sum_{\mu} \sum_{i} \lambda\,(i^*, i) \|\mathbf{v}^{\mu} - \mathbf{w}_i\|^2 \tag{5}$$

Given this distortion error, if we do not make any assumption about the distribution of the data points, then we can apply Jaynes' principle of maximum entropy, which yields the Gibbs distribution as the probability distribution. The Gibbs distribution is an exponential function of the distortion error (energy):

$$p(\mathbf{v}) \approx Z \exp\left(-\beta \sum_{i} \lambda\,(i^*, i) \|\mathbf{v} - \mathbf{w}_i\|^2 \right) \tag{6}$$

with $p(\mathbf{v})$ as the true density and Z as a normalization constant. A standard procedure to estimate the parameters $\mathbf{w}_i$ is by maximizing the likelihood for the sample $\mathcal{S} = \{\mathbf{v}^{\mu} | \mu = 1, \ldots, M\}$ (for details, see the original article):

$$\mathcal{L} = Z' \exp(-\beta \sum_{u} \sum_{i} \lambda(i^*, i) \|\mathbf{v}^{\mu} - \mathbf{w}_i\|^2) \tag{7}$$

with Z' as a normalization constant and β as a constant (noise variance; see Ref. 21). One can easily verify that for a vanishing neighborhood range, this model is in fact a homoscedastic Gaussian mixture density model (because β is a constant) with one Gaussian kernel centered on each neuron weight: Indeed, one could drop the definition of the winner, when the lattice is trained, and estimate $p(\mathbf{v})$ by the sum of all trained Gaussian kernels (with homogeneous prior probabilities, $\frac{1}{N}$). However, because the weight distribution of the SOM is a power law of the true input density (for a review, see Ref. 9) and because the kernel radii are not adapted to the local density, one can expect this density estimate to be inferior to what can be achieved with a regular fixed kernel method such as Parzen's.

It is important to note that the maximum likelihood procedure described here is actually different from the traditional one (e.g., see Ref. 35) because a winner needs to be selected (whereas in traditional maximum likelihood procedures, all kernels are updated). This difference means that, for example for a vanishing neighborhood range, a given Gaussian kernel's center is only updated when that neuron is the winner [the definition of the "winner" i^* in Equation (1) is equivalent to looking for the Gaussian kernel with the largest activity]. Hence, contrary to the classical case, the tails of the Gaussian kernels do not lead to center updates (they disappear "under" other kernels), which means that the kernel radii will be underestimated (as we will see later).

Elastic Net. Durbin and Willshaw's elastic net (27) can be considered one of the first accounts of kernel-based topographic map formation. The elastic net was used for solving the traveling salesman problem (TSP). In TSP, the idea is to find the shortest, closed tour that visits each city once and that returns to its starting point. When we represent the location of each city by a point $\mathbf{v}^{\mu}$ in the two-dimensional input space $V \subseteq \mathbb{R}^2$, and a tour by a sequence of N neurons—which comprise a ring or closed chain A—then a solution to the TSP can be envisaged as a mapping from V-space onto the neurons of the chain. Evidently, we expect the neuron weights to coincide with the input points ("cities") at convergence. An example of the convergence process for a 100 city case is shown in Fig. 5.

Without going into details, the error function that is minimized is given by:

$$E(\mathbf{W}) = -2\sigma_{\Lambda} \sum_{\mu} \log\left(\sum_{i} \exp[-\frac{1}{2\sigma_{\Lambda}^2} \|\mathbf{v}^{\mu} - \mathbf{w}_i\|^2] \right) + \kappa \sum_{i} \|\mathbf{w}_i - \mathbf{w}_{i+1}\|^2 \tag{8}$$

with κ as a parameter. The algorithm of the elastic net can be written as follows (in our format):

$$\Delta \mathbf{w}_i = 2\eta \left(\sum_{\mu} \Lambda^{\mu}(i)(\mathbf{v}^{\mu} - \mathbf{w}_i) + \kappa(\mathbf{w}_{i+1} - 2\mathbf{w}_i + \mathbf{w}_{i-1}) \right), \ \forall i \tag{9}$$

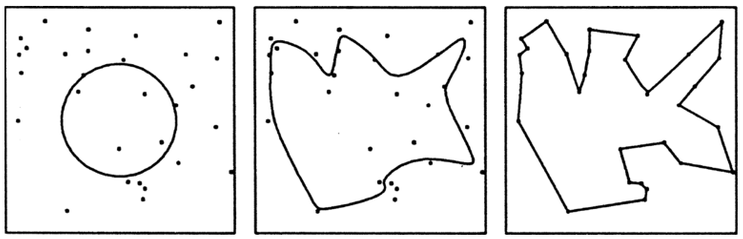

Figure 5. Elastic net used for solving the traveling salesman problem. The lattice has a ring topology (chain); the points represent cities and are chosen randomly from the input distribution demarcated by the square box. The evolution of the lattice is shown for three time instants, at $t = 0$ (initialization), 7000, and 10,000 (from left to right). The weights of the lattice at $t = 0$ form a circle positioned at the center of mass of the input distribution. (Reprinted from Ref. 34; ©1988 IEEE.)

where each weight $\mathbf{w}_i$ represents a point on the elastic net. The first term on the right-hand side is a force that drags each point $\mathbf{w}_i$ on the chain A toward each city $\mathbf{v}^\mu$, and the second term is an elastic force that tends to keep neighboring points on the chain close to each other (and thus tends to minimize the overall tour length). The function $\Lambda^\mu(i)$ is a *normalized* Gaussian:

$$\Lambda^\mu(i) = \frac{\exp(-\|\mathbf{v}^\mu - \mathbf{w}_i\|^2/2\sigma_\Lambda^2)}{\sum_j \exp(-\|\mathbf{v}^\mu - \mathbf{w}_j\|^2/2\sigma_\Lambda^2)} \tag{10}$$

with $\mathbf{w}_i$ the center of the Gaussian and σ_Λ its range, which is gradually decreased over time (as well as η and also κ). By virtue of this kernel, the elastic net can be viewed as a (equal-variance) homoscedastic Gaussian mixture density model, fitted to the data points by a penalized maximum likelihood term, that is, the elastic force that encourages the centers of neighboring Gaussians to be close in the input space (for a formal account, see Ref. 36).

The elastic net algorithm looks similar to Kohonen's SOM algorithm except that $\Lambda(i, j)$ has been replaced by $\Lambda^\mu(i)$ and that a second term is added. Interestingly, the SOM algorithm can be used for solving the TSP even without the second term (37), provided that we take more neurons in our chain than cities and that we (carefully) initialize the weights, for example, on a circle (a so-called N-gon) positioned at the center of mass of the input distribution $\{\mathbf{v}^\mu\}$ and whose diameter is half of the range spanned (as in the left panel of Fig. 5).

The elastic net has been used as a model for the development of both topography and ocular dominance in the mapping from the lateral genicu-late nucleus to the primary visual cortex (38). It is also applied in different areas of optimization problem solving. For example, it has been used for finding trajectories of charged particles with multiple scattering in high-energy physics experiments, such as the ring recognition problem of baryonic matter experiments (39) (Fig. 6), and has also been used for protein structure prediction [protein folding prediction, (40)] (Fig. 7). Furthermore, it has been used for clustering applications (41). Finally, because it also has a close relationship with "snakes" in computer vision (42) (for the connection, see Ref. 43), the elastic net has also been used for extracting the shape of a closed object from a digital image, such as finding the lung boundaries from magnetic resonance images (44) (Fig. 8).

Maximum Local Correlation. Sum and coworkers developed a new interpretation of topographic map formation, namely in terms of maximizing the local correlations between the activities of neighboring lattice neurons (32). The neurons have overlapping kernels that cause their activities to be correlated.

The kernels are defined as Gaussian *probabilities* that the corresponding neurons are active:

$$P_i(\mathbf{v}^\mu) \triangleq P(Act_i = 1|\mathbf{v}^\mu) = \frac{1}{\sqrt{2\pi\tau}} \exp\left(-\frac{\|\mathbf{v}^\mu - \mathbf{w}_i\|^2}{\tau}\right) \tag{11}$$

else $Act_i = 0$, $\forall i \in A$ (note that the definition of the range τ is slightly different). Furthermore, the range of the probability function, τ, is decreased during learning. When two neighboring neurons i and j are jointly active, a coupling

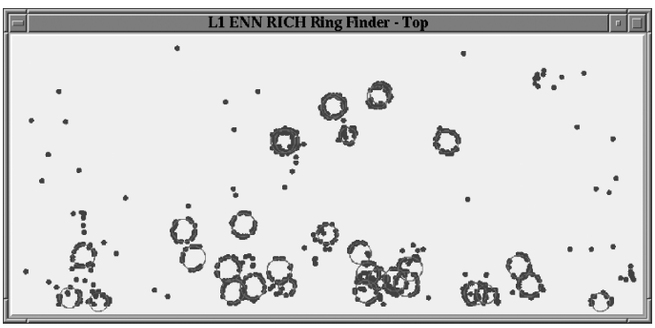

Figure 6. Elastic nets used for finding rings in scattering patterns. The found rings are indicated in red; the scattering patterns are indicated in blue (courtesy of Ivan Kisel).

Figure 7. Elastic net used for protein folding prediction. Shown are three types of proteins with the elastic net solutions shown in green and the true native states in red (40). (Reprinted with permission from John Wiley & Sons, Inc.)

energy ε is generated. The following average coupling energy[4] for this pair of neurons is defined for input $\mathbf{v}^\mu$:

$$\langle \epsilon_{ij}(\mathbf{v}^\mu)\rangle = \langle Act_i(\mathbf{v}^\mu) Act_j(\mathbf{v}^\mu)\epsilon\rangle = \epsilon P_i(\mathbf{v}^\mu)P_j(\mathbf{v}^\mu) \tag{12}$$

Topological information enters the learning process in the following manner:

$$\langle \epsilon_{ij}(\mathbf{v}^\mu)\rangle = \epsilon\, P_i(\mathbf{v}^\mu)P_j(\mathbf{v}^\mu)q_{ij} \tag{13}$$

with $q_{ij} = 1$ when neurons i and j are lattice neighbors, or it is zero. Taken over all neuron pairs, the following average coupling energy can be defined:

$$E(\mathbf{v}^\mu) = \epsilon \sum_i \sum_j P_i(\mathbf{v}^\mu)P_i(\mathbf{v}^\mu)q_{ij} \tag{14}$$

Finally, when taken over all training samples, the following likelihood function is defined:

$$C = \sum_\mu \log E(\mathbf{v}^\mu) \tag{15}$$

and on which gradient ascent is performed. This results in the following incremental update rules for the kernel centers (albeit that also a fixed point version could have been derived, see further):

$$\Delta \mathbf{w}_i = \eta \sum_j d_{ij}(\mathbf{v}^\mu - \mathbf{w}_i), \quad \forall i \tag{16}$$

with d_{ij} defined as:

$$\frac{d_{ij} = q_{ij}\exp(-\frac{\|\mathbf{v}^\mu - \mathbf{w}_i\|^2}{\tau})\exp(-\frac{\|\mathbf{v}^\mu - \mathbf{w}_j\|^2}{\tau})}{\sum_{l,m} q_{lm}\exp(-\frac{\|\mathbf{v}^\mu - \mathbf{w}_l\|^2}{\tau})\exp(-\frac{\|\mathbf{v}^\mu - \mathbf{w}_m\|^2}{\tau})} \tag{17}$$

[4]Note from the author: The transition to the rightmost term in Equation (12) is only valid when the activities are independent, which is not the case when the kernels overlap, especially when they are lattice neighbors, in which case, this is a heuristical procedure.

The algorithm is called maximum local correlation (MaxCorr). When comparing with the SOM algorithm, one can say that the term dij fulfills the role of the neighborhood function, albeit it is defined in the input space rather than in the lattice space. Because it depends on the product of two nearest-neighbor kernels, when they are far apart in the input space, the product could be small. The weight update rule then simply reduces to standard unsupervised competitive learning (which has no unfolding capacity). In the opposite case, when τ is too large, then, almost all neurons will have an equal probability to be active, and, by consequence, the weights will converge to the sample set mean (centroid). Therefore, τ should be chosen with care (but this is also the case when choosing the parameters of, for example, the SOM algorithm). Unfortunately, because of its statistical nature, the algorithm is relatively slow in unfolding and spanning the input space compared with the SOM algorithm. In addition, boundary effects are found: The border of the input space is not filled by neurons (foreshortening, as in the example shown next).

To speed up the algorithm and to free the user from defining the learning rate η (and also its cooling scheme), we can derive a fixed point version of Equation. (16) by taking the equilibrium kernel centers (obtained by solving for $\Delta \mathbf{w}_i = 0$, given our sample set):

$$\mathbf{w}_i = \frac{\sum_\mu \sum_j d_{ij}\mathbf{v}^\mu}{\sum_\mu \sum_j d_{ij}} \tag{18}$$

As an example, we reconsider the two-dimensional uniform distribution $[-1,1]^2$ but take from it a batch of 1000 samples, and a two-dimensional square lattice of $N = 5 \times 5$ neurons; the kernel centers are initialized by taking samples from the same distribution. The parameter τ is decreased as follows:

$$\tau(t) = 10\,\tau(0)\exp(-2\mathcal{T}(0)\frac{t}{t_{\max}}) \tag{19}$$

with $\tau(0)$ the initial value, t the epoch number (one epoch means that all data points in a sample set have been shown), and $t_{\max}$ the maximum number of epochs; we take $\tau(0) = 2.5$, that is, half the linear size of the lattice, and $t_{\max} = 100$. The result at $t_{\max}$ is shown in Fig. 9a. We observe the boundary effects. By choosing an amplitude

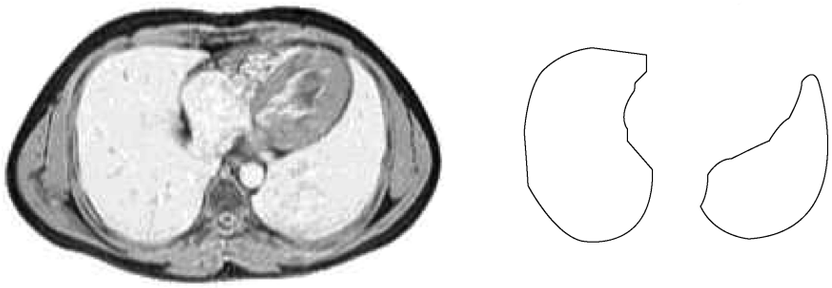

Figure 8. Elastic net applied to finding the lung boundaries from magnetic resonance images (44). The left panel shows the original image; the right panel shows the extracted lung boundaries. Note the mismatch at the high curvature region of the right lung. (Reprinted with permission.)

larger than 10 in the previous equation, the map becomes smoother with stronger foreshortening of the kernel centers at the border and, thus, an even larger unfilled boundary range. We have also noticed that a phase transition occurs in the map at about $\mathcal{T} = 1.3$: Above this value, the algorithm pursues topology-preservation; below it, distortion minimization is pursued (remember the link with unsupervised competitive learning). Phase transitions have been much less studied for kernel-based algorithms.

Generative Topographic Map. The generative topographic map (GTM) algorithm was introduced by Bishop and coworkers (28, 29) and develops a topographic map that attempts to find a representation for the input distribution $p(\mathbf{v})$, $\mathbf{v} = [v_1, \ldots, v_d]$, $\mathbf{v} \in V \subseteq \mathbb{R}^d$, in terms of a number L of latent variables $\mathbf{x} = [x_1, \ldots, x_L]$. This effort is achieved by considering a nonlinear transformation $\mathbf{y}(\mathbf{x}, \mathbf{W})$, governed by a set of parameters $\mathbf{W}$, which maps points in the latent variable space to the input space, much the same way as the lattice nodes in the SOM relate to positions in V-space. If we define a probability distribution $p(\mathrm{x})$ on the latent variable space, then this will induce a corresponding distribution $p(y \,|\, \mathbf{W})$ in the input space.

As a specific form of $p(\mathrm{x})$, Bishop and coworkers take a discrete distribution that consists of a sum of delta functions located at the N nodes of a regular lattice:

$$p(\mathrm{x}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\mathrm{x} - \mathrm{x}_i) \tag{20}$$

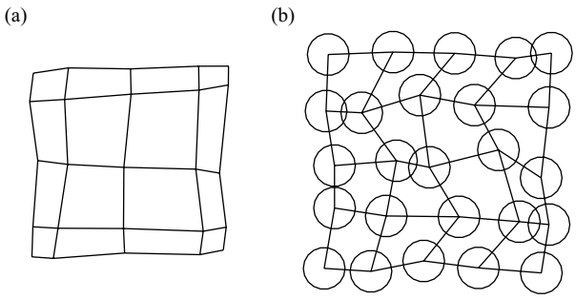

Figure 9. (a) Lattice-sized 5×5 neurons obtained with the fixed point version of the maximum correlation algorithm applied to a batch of 1000 samples taken from the two-dimensional uniform distribution $[-1, 1]^2$ (square box). (b) Lattice obtained with the regularized Gaussian mixture algorithm in Equations (26) and (27).

The dimensionality L of the latent variable space is typically less than the dimensionality d of the input space so that the transformation $\mathbf{y}$ specifies an L-dimensional manifold in V-space. Because $L < d$, the distribution in V-space is confined to this manifold and, thus, is singular. To avoid this, Bishop and coworkers introduced a noise model in V-space, namely, a set of radially symmetric Gaussian kernels centered at the positions of the lattice nodes in V-space. The probability distribution in V-space then can be written as follows:

$$p(\mathbf{v}|\mathbf{W}, \sigma) = \frac{1}{N} \sum_{i=1}^{N} p(\mathbf{v}|\mathbf{x}_i, \mathbf{W}, \sigma) \tag{21}$$

which is a homoscedastic Gaussian mixture model. In fact, this distribution is a *constrained* Gaussian mixture model because the centers of the Gaussians cannot move independently from each other but are related through the transformation y. Moreover, when the transformation is smooth and continuous, the centers of the Gaussians will be ordered topographically by construction. Therefore, the topographic nature of the map is an intrinsic feature of the latent variable model and is not dependent on the details of the learning process. Finally, the parameters $\mathbf{W}$ and σ are determined by maximizing the log-likelihood:

$$\ln \mathcal{L}(\mathbf{W}, \sigma) = \ln \prod_{\mu=1}^{M} p(\mathbf{v}^{\mu}|\mathbf{W}, \sigma) \tag{22}$$

and which can be achieved through the use of an expectation-maximization (EM) procedure (45): First, the probabilities are estimated (expectation step), and then these probabilities are used for estimating the parameters sought (maximization step). This procedure is repeated till convergence.

The GTM has been applied to visualizing oil flows along multiphase pipelines in which the phases are oil, water, and gas, and the flows can be one of three types; stratified, homogeneous, or annular (28) (Fig. 10, right panel); also it has been applied to visualizing electropalatographic (EPG) data for investigating activity of the tongue in normal and pathological speech (47) (Fig. 11). More recently, it has been applied to the classification of *in vivo* magnetic resonance spectra of controls and Parkinson patients (48) (Fig. 12), to word grouping in document data sets (using the newsgroup data set benchmark) (Fig. 13) and the exploratory analysis

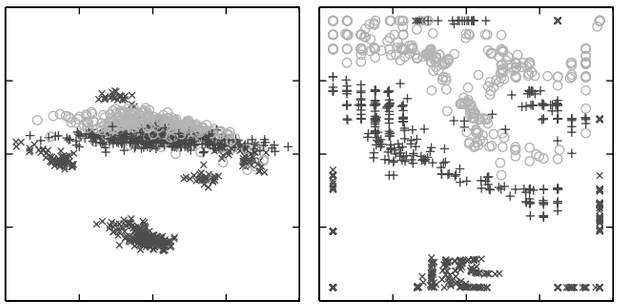

Figure 10. Oil flow data set visualized using principal components analysis (PCA) (left panel) and the GTM (right panel). Because the GTM performs a nonlinear mapping, it can better separate the three types of flow configurations: laminar (red crosses), homogeneous (blue plusses), and annular (green circles) (46). (Reprinted with permission.)

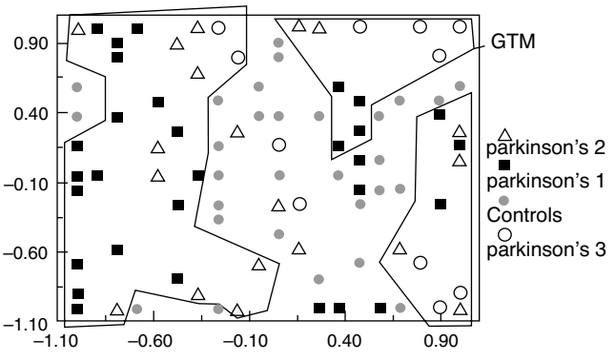

Figure 12. Magnetic resonance spectra of controls and three types of Parkinson patients visualized in a 15×15 GTM map (48). (Reprinted with permission from Wiley-Liss, Inc., a subsidiary of John Wiley & Sons, Inc.)

of web navigation sequences (49), and to spatio-temporal clustering of transition states of a typhoon from image sequences of cloud patterns (50). In another case, the GTM is used for micro-array data analysis (gene expression data) with the purpose of finding low-confidence value genes (Fig. 14), which, in the map, become more spread away from the region representing the bulk of the data (51) than an alternative method, NeuroScale[5]. Furthermore, a single two-dimensional visualization plot may not be sufficient to capture all of the interesting aspects of complex data sets. Therefore, a hierarchical version of the GTM has been developed (53). A final note on the GTM and its application to data mining: The mapping function used by GTM could not be meaningful as an explanation of the possible mappings, as it is based on a very artificial and arbitrarily constructed nonlinear latent space. For this reason, the prior will have to be learned from data rather than created by a human expert, as is possible for spring-based models.

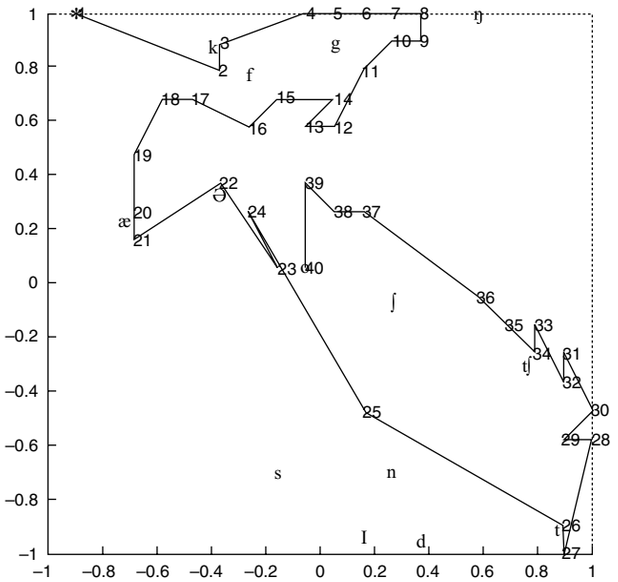

Figure 11. Visualization of the trajectory in a 20×20 GTM lattice of the activity of the tongue (electropalatographic data) of speaker RK for the utterance fragment "I prefer *Kant to* Hobbes for a good bedtime book" (47). (Reprinted with permission from Elsevier Limited.)

Regularized Gaussian Mixture Modeling. Heskes (20) showed the direct correspondence between minimum distortion topographic map formation and maximum likelihood Gaussian mixture density modeling (GMM) for the homoscedastic case. The starting point was the traditional distortion (vector quantization) formulation of the self-organizing map:

$$F_{\text{quantization}} = \sum_{\mu} \sum_{i} P(i|\mathbf{v}^{\mu}) \sum_{j} \lambda(i,j) \frac{1}{2} \|\mathbf{v}^{\mu} - \mathbf{w}_j\|^2 \quad (23)$$

with $P(i \mid \mathbf{v}_{\mu})$ the probability that input $\mathbf{v}^{\mu}$ is assigned to neuron i with weight $\mathbf{w}_i$ (i.e., the posterior probability, and with $\sum_i P(i|\mathbf{v}^{\mu}) = 1$ and $P(i|\mathbf{v}^{\mu}) \geq 0$). Even if we assign $\mathbf{v}^{\mu}$ to neuron i, there exists a confusion probability $\lambda(i, j)$ that $\mathbf{v}^{\mu}$ is assigned to neuron j. An annealed version of the self-organizing map is obtained if we add an entropy term:

$$F_{\text{entropy}} = \sum_{\mu} \sum_{i} P(i|\mathbf{v}^{\mu}) \log\left(\frac{P(i|\mathbf{v}^{\mu})}{Q_i}\right) \quad (24)$$

with Q_i the prior probability (the usual choice is $Q_i = \frac{1}{N}$, with N the number of neurons in the lattice). The final (free) energy is now:

$$F = \beta F_{\text{quantization}} + F_{\text{entropy}} \quad (25)$$

with β playing the role of an inverse temperature. This

[5]NeuroScale is an extension of the classical distance preserving visualization methods of Sammon mapping and multidimensional scaling. It uses radial basis function networks. For more information, see Ref. 52.

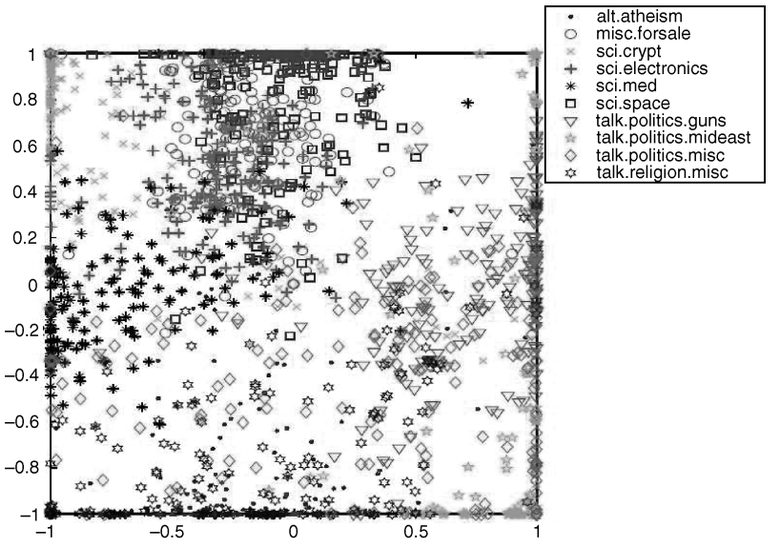

Figure 13. Visualization of word grouping in the newsgroup benchmark data set (49). (Reprinted with permission from the IEEE ©2005 IEEE)

formulation is very convenient for an EM algorithm. The expectation step leads to:

$$P(i|\mathbf{v}^\mu) = \frac{Q_i \exp(-\frac{\beta}{2}\sum_j \lambda(i,j)\|\mathbf{v}^\mu - \mathbf{w}_j\|^2)}{\sum_s Q_s \exp(-\frac{\beta}{2}\sum_j \lambda(s,j)\|\mathbf{v}^\mu - \mathbf{w}_j\|^2)} \tag{26}$$

and the maximization step to:

$$\mathbf{w}_i = \frac{\sum_\mu \sum_j P(j|\mathbf{v}^\mu)\lambda(j,i)\mathbf{v}^\mu}{\sum_\mu \sum_j P(j|\mathbf{v}^\mu)\lambda(j,i)}, \quad \forall i \tag{27}$$

which is also the result reached by Graepel and coworkers (54) for the soft topographic vector quantization (STVQ) algorithm (see the next section). Plugging Equation. (26) into Equation (25) leads to the standard error function:

$$E(\mathbf{W}) = -\sum_\mu \log \sum_i Q_i \exp\left(-\frac{\beta}{2}\sum_j \lambda(i,j)\|\mathbf{v}^\mu - \mathbf{w}_j\|^2\right) \tag{28}$$

and allows for the connection with a maximum likelihood procedure for a mixture of homoscedastic Gaussians when the neighborhood range vanishes ($\lambda(i,j) = \delta_{ij}$):

$$p(\mathbf{v}) \approx \frac{1}{N}\sum_i K_i(\mathbf{v}, \mathbf{w}_i) \tag{29}$$

of the true input density $p(\mathbf{v})$, with N the number of Gaussian kernels (we assume a homogeneous model; $Q_i = \frac{1}{N}$):

$$K_i(\mathbf{v}, \mathbf{w}_i) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \exp\left(\frac{-\|\mathbf{v} - \mathbf{w}_i\|^2}{2\sigma^2}\right) \tag{30}$$

with center $\mathbf{w}_i = [w_{i1}, \ldots, w_{id}] \in \mathbb{R}^d$ and radius σ. The likelihood can be written as:

$$\mathcal{L} = \sum_\mu \log \frac{1}{N}\sum_i K_i(\mathbf{v}, \mathbf{w}_i) \tag{31}$$

which is equal to $-E$ [Equation (28)]. When the neighborhood is present, Heskes showes that:

$$E = -\mathcal{L} + E_{\text{regularization}} \tag{32}$$

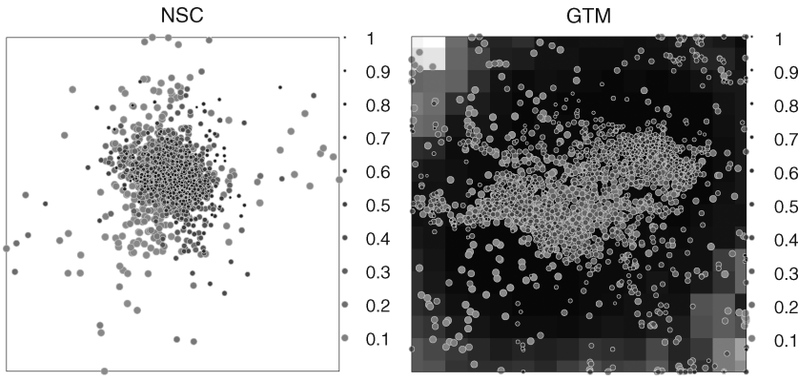

Figure 14. Distribution of genes with their confidence indicated (cf., the vertical scale), for standard NeuroScale (left panel) and the GTM. The latter seems to distinguish the low-confidence genes better (51). (Reprinted with permission.)

with the latter term capturing the neighborhood relations only. This result is an important result that shows the connection between minimum distortion topographic map formation and a regularized form of maximum likelihood homoscedastic Gaussian mixture density modeling.

Let us now reconsider our two-dimensional uniform distribution example and take a Gaussian neighborhood function of which the range is decreased exponentially, $\sigma_\lambda(t) = \sigma_\lambda(0)\exp(-2\sigma_\lambda(0)t/t_{\max})$; the initial range $\sigma_\lambda(0) = 2.5$. The kernel centers are initialized by sampling the same distribution. For the kernel range, we take $\sigma = 0.15$. The result is shown in Fig. 9b. We observe that the lattice disentangles and that it covers well the input space.

As an application, Heskes considers market basket analysis. Given are a list of transactions that correspond to the joint set of products purchased by a customer at a given time. The goal of the analysis is to map the products onto a two-dimensional (2-D) map (lattice) such that neighboring products are "similar." Similar products should have similar conditional probabilities of buying other products. In another application, he considers the case of transactions in a supermarket. The products are summarized in product groups and given are the co-occurrence frequencies. The result is a 2-D density map that showes clusters of products that belong together, for example, a large cluster of household products (Fig. 15).

Soft Topographic Vector Quantization. Another approach that considers topographic map formation as an optimization problem is the one introduced by Graepel and coworkers (31; 54). They start from the following cost function:

$$E(\mathbf{W}) = \frac{1}{2}\sum_{\mu}\sum_{i} c_{\mu,i} \sum_{j} \lambda(i,j)\|\mathbf{v}^{\mu} - \mathbf{w}_j\|^2 \qquad (33)$$

with $c_{\mu,i} \in \{0,1\}$ and for which $c_{\mu i} = 1$ if $\mathbf{v}^{\mu}$ is assigned to neuron i, else $c_{\mu,i} = 0 (\sum_i c_{\mu,i} = 1)$; the neighborhood function obeys $\sum_j \lambda(i,j) = 1$. The $\mathbf{w}_i$, $\forall i$, for which this function is minimal, are the optimal ones. However, the optimization is a difficult task, because it depends both on binary and continuous variables. Also, it will possess many local minima. To avoid this situation, a technique known as deterministic annealing is applied by doing the optimization on a smooth function parametrized by a parameter β, the so-called free energy: When β is small, the function is smooth and only one global minimum remains; when it is large, more of the structure of the original cost function is reflected in the free energy. Deterministic annealing begins by determining the minimum of the free energy at low values of β and attempts to keep track of the minimum through higher values of β.

The application of the principle of maximum entropy yields the free energy (31), which is given by:

$$F = -\frac{1}{\beta}\log\sum_{c_{\mu,i}} \exp(-\beta E) \qquad (34)$$

which leads to probabilistic assignments of inputs $\mathbf{v}^{\mu}$ to neurons, $P(i|\mathbf{v}^{\mu})$, $\forall i$, that is, the posterior probabilities, and which are given by:

$$P(i|\mathbf{v}^{\mu}) = \frac{\exp(-\frac{\beta}{2}\sum_j \lambda(i,j)\|\mathbf{v}^{\mu} - \mathbf{w}_j\|^2)}{\sum_s \exp(-\frac{\beta}{2}\sum_j \lambda(s,j)\|\mathbf{v}^{\mu} - \mathbf{w}_j\|^2)} \qquad (35)$$

which is termed by the authors a fuzzy assignment of inputs to clusters (neurons) (from which comes the alternative name for the algorithm: fuzzy membership in clusters). The fixed point rule for the kernel centers is then:

$$\mathbf{w}_i = \frac{\sum_{\mu}\sum_j P(j|\mathbf{v}^{\mu})\lambda(j,i)\mathbf{v}^{\mu}}{\sum_{\mu}\sum_j P(j|\mathbf{v}^{\mu})\lambda(j,i)}, \quad \forall i \qquad (36)$$

The updates are done through an EM scheme. We observe that the latter equation is identical to Heskes' rule for regularized Gaussian mixture modeling, Equation (27).

Figure 15. Visualization of market basket data in which 199 product groups are clustered based on their co-occurrence frequencies with other products (20). (Reprinted with permission from the IEEE © 2001 IEEE)

	1	2	3	4	5	6	7
1	economy computers	economy car industry	economy nature traffic	nature traffic	travel gambling		
2		gardening	balance economy	nature traffic	gambling racing	communication telephone	
3	financial politics	banking farming	international trade	cold war politics		military war	politics education
4	politics law	law court	ecology nuclear energy		military politics		lottery
5	research women's health	public health	public housing	military history		international politics unions	politics strike
6		public education	university education		border politics	high school sports	sports hockey
7	high school education	high school arts	arts	crime news	television		sports baseball

Figure 16. Document map generated with a modified version of the STMP (55). (Reprinted with permission from Elsevier Science & Technology).

Most topographic map algorithms are applied to objects described by Euclidean feature vectors. It is therefore important to signal that the STVQ was generalized to the soft topographic mapping for proximity data (STMP), which can be used for clustering categorical data given a matrix of pairwise proximities or dissimilarities, which is one of the first accounts of this nature in the topographic map literature. A candidate application is the DNA micro-array data set in which the data can be described by matrices with the columns representing tissue samples and the rows representing genes and with the entries in the matrix corresponding to the strength of the gene expression. In Ref. 55, a modified version of the STMP is used for clustering documents ("document map"). Every neuron points to several documents, characterized by the 10 index words that occur most often (Fig. 16). The map shows that the documents are grouped by content.

Conclusion. The diversity in algorithms for homoscedastic kernel-based topographic map formation reflects the differences in strategies behind them. Some algorithms have been introduced to exploit the use of chains to solve specific optimization problems (such as the traveling salesman problem in the elastic net), others to exploit deterministic annealing in an attempt to avoid local minima (such as the STVQ), others to develop a manifold that avoids topological defects because the map is topographically ordered by construction (such as the GTM), others to show that topographic maps can be developed based on local correlations in neural activity (such as MaxCorr), and still others to show and exploit the connection with mixture density modeling, to obtain density maps (such as Heskes' algorithm). Thus, by this diversity in strategies, these topographic map algorithms have their specific strengths (and weaknesses) and, thus, their own types of applications. Heteroscedastic kernel-based topographic maps could also be developed with different strategies in mind. We opt to review the algorithms that yield extended mixture density models, extended because it occurs in a topographic map, which means that the data density can be visualized.

Heteroscedastic Kernel Topographic Map Formation

In the literature, only few approaches exist that consider heteroscedastic kernels, perhaps because the kernel radius in the homoscedastic case is often used in an annealing schedule, as in the STVQ, the maximum local correlation, and the elastic net algorithms. When using heteroscedastic kernels, a better density estimate is expected. In the following, we restrict ourselves to spherically symmetric Gaussians, but one could also consider the general case where the Gaussians are specified by covariance matrices. As a reference, we take the homogeneous, heteroscedastic Gaussian mixture density model:

$$p(\mathbf{v}) \approx \frac{1}{N}\sum_i K_i(\mathbf{v}, \mathbf{w}_i, \sigma_i) \tag{37}$$

with N the number of Gaussian kernels:

$$K_i(\mathbf{v}, \mathbf{w}_i, \sigma_i) = \frac{1}{(2\pi\sigma_i^2)^{\frac{d}{2}}}\exp\left(\frac{-\|\mathbf{v}-\mathbf{w}_i\|^2}{2\sigma_i^2}\right) \tag{38}$$

with center $\mathbf{w}_i = [w_{i1}, \ldots, w_{id}] \in \mathbb{R}^d$, radius σ_i, and $\mathbf{v} = [v_1, \ldots, v_d]$, a random vector in $\mathbb{R}^d$ generated from the probability density $p(\mathbf{v})$. A standard procedure to estimate the parameters $\mathbf{w}_i$ and σ_i, $\forall i$, is by maximizing the (average) likelihood or by minimizing the (average) negative log-likelihood for the sample $S = \{\mathbf{v}^\mu | \mu = 1, \ldots, M\}$(35) through an EM approach (45).

This reference leads to the following fixed point rules:

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu P(i|\mathbf{v}^\mu)\mathbf{v}^\mu}{\sum_\mu P(i|\mathbf{v}^\mu)} \\ \sigma_i^2 &= \frac{\sum_\mu P(i|\mathbf{v}^\mu)\|\mathbf{v}-\mathbf{w}_i\|^2/d}{\sum_\mu P(i|\mathbf{v}^\mu)}, \ \forall i \end{aligned} \tag{39}$$

where we have substituted for the posterior probabilities $P(i|\mathbf{v}^\mu) = \frac{K_i^\mu}{\sum_j K_j^\mu}$. Note that in the expectation step, we update the posterior probabilities before the maximization step, which is the update of the kernel centers and radii. Note also that we can easily extend the current format to nonhomogeneous GMMs by considering the prior probabilities as additional parameters.

To compare the density estimation performance of the various heteroscedastic algorithms with that of heteroscedastic Gaussian mixture modeling, we again consider the two-dimensional uniform distribution $[-1,1]^2$ from which

we take 1000 samples. We consider $N = 25$ kernels of which we initialize the centers by taking samples from the same distribution; the kernel radii are initialized $\sigma_i = 0.2, \forall i$. We train the GMM during $t_{\max} = 100$ epochs. At $t_{\max}$, the average log-likelihood is -1.440, and the average kernel radius is 0.147 (see Figs. 18a and b, stippled lines). The resulting distribution of the kernels is shown in Fig. 17a.

Batch Map and its Heteroscedastic Extension. The original batch map (13), called here BMo, is defined as follows:

$$\mathbf{w}_i = \frac{\sum_\mu \Lambda(i^*, i)\mathbf{v}^\mu}{\sum_\mu \Lambda(i^*, i)}, \quad \forall i \tag{40}$$

Bearing in mind what we have said earlier about the SOM in connection to Gaussian mixture modeling, we can extend this rule to the heteroscedastic case:

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu \Lambda(i^*, i)\mathbf{v}^\mu}{\sum_\mu \Lambda(i^*, i)} \\ \sigma_i^2 &= \frac{\sum_\mu \Lambda(i^*, i)\|\mathbf{v} - \mathbf{w}_i\|^2/d}{\sum_\mu \Lambda(i^*, i)}, \quad \forall i \end{aligned} \tag{41}$$

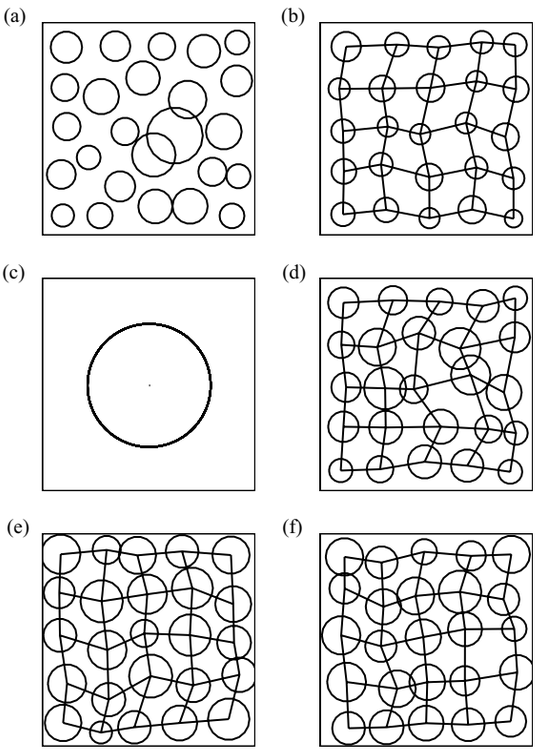

Figure 17. (a) Fixed point solution for a GMM with 25 kernels and for 1000 samples taken from the two-dimensional uniform distribution $[-1, 1]^2$ (boxes). The circles correspond to the standard deviations of the Gaussian kernels ("radii"). (b) Fixed point solutions for a 5×5 lattice of Gaussian kernels using the extended batch map BMe; (c) two update rules that correspond to the GMM case when the neighborhood has vanished, ERGMM1; (d) ERGMM2; (e) LDE; and (f) BM-kMER (F).

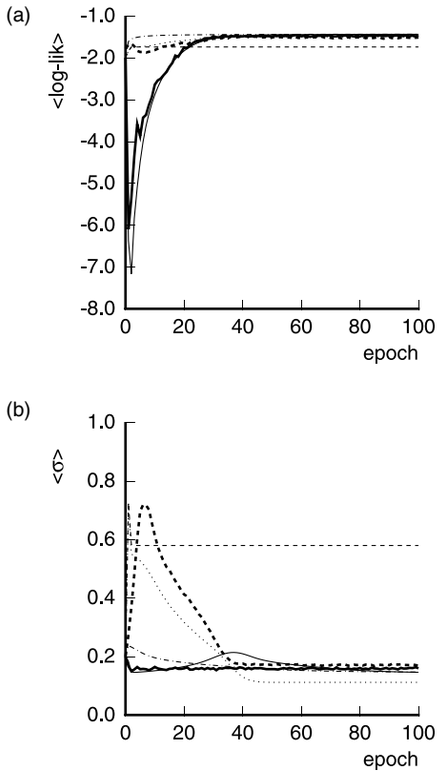

Figure 18. Average log-likelihood (a) and average kernel radius (b) as a function of training epochs for the update rules shown in Fig. , GMM (stippled line), BMe (dotted line), ERGMM1 (thin dashed line), ERGMM2 (thin full line), LDE (thick dashed line), and BM-kMER (thick full line).

with $i^* = \text{argmax}_i K_i$ (which is no longer equivalent to $i^* = \text{argmin}_i \mathbf{v} - \mathbf{w}_i$ but which is required because we now have heteroscedastic kernels), that is, an *activity-based* definition of "winner-takes-all" rather than a minimum *Euclidean distance-based* definition. We call this new EM algorithm the extended batch map (BMe). Notice again that, by the definition of the winner, the tails of the kernels are cut off, because the kernels overlap.

To continue with our example, we consider a 5×5 lattice of which the kernels are initialized as above and that we train with a Gaussian neighborhood function of which the range is decreased exponentially $\sigma_\lambda(t) = \sigma_\lambda(0)\exp(-2\sigma_\lambda(0)t/t_{\max})$; the initial range $\sigma_\lambda(0) = 2.5$. The average log-likelihood and average kernel radii are shown in Figs. 18a and b (dotted lines).

We observe that the radii quickly grow from their initial 0.2 value to over 0.7 and then converge to a value that is smaller than what is expected for the maximum likelihood approach (the average radius is now 0.111). This result is a direct result of the tails of the Gaussian kernels being cut off. The lattice at $t_{\max}$ is shown in Fig. 17b.

Extended Regularized Gaussian Mixture Modeling, Version 1. We will discuss in this section and the next two possible heuristic versions of heteroscedastic Gaussian kernel-based topographic map formation. The fixed point weight

update rule proposed for the homoscedastic case in regularized Gaussian mixture modeling (20), Equation (27), and which was also proposed in the soft topographic vector quantization (STVQ) algorithm (31, 54), Equation (36), can be easily extended to the heteroscedastic case by observing the similarities of the latter with the maximum likelihood approach of Equation (39)(56):

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)\mathbf{v}^\mu}{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)} \\ \sigma_i^2 &= \frac{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)\|\mathbf{v}^\mu - \mathbf{w}_i\|^2/d}{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)}, \quad \forall i \end{aligned} \tag{42}$$

We call this the extended regularized Gaussian mixture modeling version 1 (ERGMM1).

We again consider the uniform distribution example with the same lattice and neighborhood settings as in the previous case. We now observe that this approach leads to rapidly growing kernel radii (thin dashed line in Fig. 18b), which causes the kernels to map the whole input space instead of the kernel centers, which stay at the centroid of the distribution (Fig. 17c). This result is clearly a nonoptimal maximum likelihood result (the thin dashed line in Fig. 18a is lower than the GMM result). However, this result does not necessarily mean that, for example for multimodal input densities, the kernel radii would not cover the whole space, as we have shown (56). One way to solve this problem is to smooth the radii updates over time, using a leaky integrator such as Wegstein's, $\sigma_i(t) \leftarrow (1-\alpha)\sigma_i(t-1) + \alpha\sigma_i(t)$(13).

Extended Regularized Gaussian Mixture Modeling, Version 2. In a recent publication (33), we introduced a learning algorithm for kernel-based topographic map formation of heteroscedastic Gaussian mixtures that allows for a unified account of distortion error (vector quantization), log-likelihood, and Kullback–Leibler divergence. The distortion error is given by:

$$\begin{aligned} E(\mathbf{W}\sigma) = & \sum_\mu \sum_i \sum_j \Lambda_{ji} P(j|\mathbf{v}^\mu) \frac{\|\mathbf{v}^\mu - \mathbf{w}_i\|^2}{2\sigma_i^2} \\ & + \sum_\mu \sum_j P(j|\mathbf{v}^\mu)(-\frac{d}{2})\log \sum_i \frac{\Lambda_{ji}}{\sigma_i^2} \\ & + \sum_\mu \sum_i P(i|\mathbf{v}^\mu)\log \frac{P(i|\mathbf{v}^\mu)}{1/N} \end{aligned} \tag{43}$$

with $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_N)$ and $\sigma = (\sigma_1, \ldots, \sigma_N)$ and can be shown to be equal to a scaled version of the log-likelihood and a regularization term:

$$E = (-\log \mathcal{L} + \mathcal{R}) M (\sqrt{2\pi})^d \tag{44}$$

$$\mathcal{R} = -\frac{1}{M} \sum_\mu \log\Big(\sum_i Q_i \exp(-\sum_j \frac{\Lambda_{ij}}{2\sigma_j^2} \overline{\mathbf{w}}_i - \mathbf{w}_j\|^2) \Big) \tag{45}$$

that is, the part that collects the topological information, with Q_i kernel i's prior probability and where we applied a "bias-variance" decomposition of the weighted and normalized error term:

$$\sum_r \frac{\Lambda_{ri}}{2\sigma_r^2} \|\mathbf{v}^\mu - \mathbf{w}_r\|^2 = \frac{\|\mathbf{v}^\mu - \overline{\mathbf{w}}_i\|^2}{2\overline{\sigma}_i^2} + \sum_r \frac{\Lambda_{ri}}{2\sigma_r^2} \|\overline{\mathbf{w}}_i - \mathbf{w}_r\|^2 \tag{46}$$

with:

$$\overline{\mathbf{w}}_i = \frac{\sum_r \frac{\Lambda_{ri}\mathbf{w}_r}{\sigma_r^2}}{\sum_r \frac{\Lambda_{ri}}{\sigma_r^2}} \tag{47}$$

$$\frac{1}{\sigma_i^2} = \sum_r \frac{\Lambda_{ri}}{\sigma_r^2} \tag{48}$$

Thus, we take as the ith kernel radius $\overline{\sigma}_i$, which we estimate from the σ_js in Equation. (48). The σ_js are updated as in Equation (42). In this way, because we perform this weighted sum of inverse variances, we avoid the initially large radii generated by Equation (42). We call this algorithm the extended regularized Gaussian mixture modeling version 2 (ERGMM2).

The EM algorithm for our uniform density now behaves better (thin full lines in Figs. 18a and b): Both the average log-likelihood and the kernel radius are close to those of the GMM approach. The lattice is shown in Fig. 17d.

Other Versions. One could also consider the format suggested in Ref. 33:

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)\mathbf{v}^\mu}{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)} \\ \sum_\mu \sum_j P(j|\mathbf{v}^\mu)(\overline{\sigma}_j)^2 \Lambda_{ij} &= \sum_\mu \frac{1}{d} \sum_j P(j|\mathbf{v}^\mu)\Lambda_{ij} \|\mathbf{v}^\mu - \mathbf{w}_i\|^2, \quad \forall i \end{aligned} \tag{49}$$

with $\overline{\sigma}_i$ as in Equation (48), which was derived from an objective function (distortion minimization and log-likelihood maximization) but which does not lead to a closed form solution for updating the kernel radii. This format leads to a complex iterative update scheme because for large neighborhood ranges, the determinant becomes close to singular and because one should guarantee non-negative solutions for the σ_is. We do not consider this update scheme more.

Another heuristic version is obtained by adopting a mixed strategy, namely, to update the kernel radii as in maximum likelihood Gaussian mixture modeling but to update the kernel centers with the neighborhood function present:

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)\mathbf{v}^\mu}{\sum_\mu \sum_j \Lambda(j,i) P(j|\mathbf{v}^\mu)} \\ \sigma_i^2 &= \frac{\sum_\mu P(i|\mathbf{v}^\mu)\|\mathbf{v} - \mathbf{w}_i\|^2/d}{\sum_\mu P(i|\mathbf{v}^\mu)}, \quad \forall i \end{aligned} \tag{50}$$

However, it also leads to rapidly increasing kernel radii and thus behaves similarly to ERGMM1; in addition,

because it does not use neighborhood information for its kernel radii, it is more prone to getting trapped in solutions that do not show the correct the number of data clusters (as was observed in Ref. 56).

Also, an heuristic approach is suggested by Yin and Allinson (19) that minimizes the Kullback–Leibler divergence, based on an idea introduced by Benaim and Tomasini (57) for the homoscedastic case. Albeit that these authors only suggested an incremental, gradient-based learning procedure (thus, with a learning rate), we can easily cast their format into a fixed point learning scheme:

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu \Lambda(i^*,i) P(i|\mathbf{v}^\mu)\mathbf{v}^\mu}{\sum_\mu \Lambda(i^*,i) P(i|\mathbf{v}^\mu)} \\ \sigma_i^2 &= \frac{\sum_\mu \Lambda(i^*,i) P(i|\mathbf{v}^\mu)\|\mathbf{v}^\mu - \mathbf{w}_i\|^2/d}{\sum_\mu \Lambda(i^*,i) P(i|\mathbf{v}^\mu)}, \quad \forall i \end{aligned} \tag{51}$$

with the winner neuron defined as $i^* = \text{argmax}_i P(i|\mathbf{v}^\mu)$, thus, the neuron with the largest posterior probability. However, this learning scheme has very limited lattice unfolding capacity: For example the lattice does not unfold for our uniform distribution example (Fig. 19a). There seems to be a confusion between the posterior probability and the neighborhood function (because their products are taken for the same kernel i): Omitting the posteriors leads to the extended batch map algorithm, which can unfold the lattice. The neighborhood functions in the ERGMM1 and ERGMM2 EM rules act as smoothing kernels (summing over all posteriors) and do not impede lattice unfolding. To remedy the problem with Yin and Allinson's, one could replace $P(i|\mathbf{v}^\mu)$ by $P(i^*|\mathbf{v}^\mu)$ in the above equations. The lattice unfolds, but the solution is now similar to that of BMe, also with the underestimated kernel radius size (Fig. 19b). As applications, Yin and Allinson consider density estimation of X-ray diffraction data and capillary electrophoresis data; for the latter, the width of the peaks in the density estimate relate to the diffusion coefficients of the chemical analysis, which shows the benefit of heteroscedastic kernels.

In a still different approach, an input to lattice transformation Ψ is considered that admits a kernel function, a Gaussian (58) $\langle \Psi(\mathbf{v}), \Psi(\mathbf{w}_i)\rangle = K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$

$$K(\mathbf{v}, \mathbf{w}_i, \sigma_i) = \exp\left(-\frac{\|\mathbf{v} - \mathbf{w}_i\|^2}{2\sigma_i^2}\right) \tag{52}$$

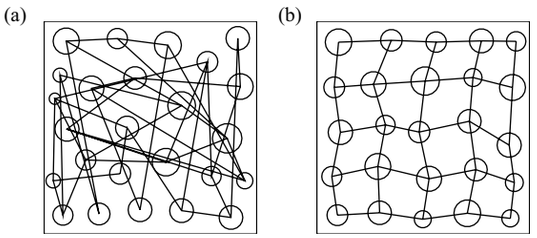

Figure 19. (a) Fixed point solution that corresponds to Yin and Allinson's approach in Equation (51); the lattice did not unfold. (b) Idem but now when replacing the posterior probability $P(i|\mathbf{v}^\mu)$ by $P(i^*|\mathbf{v}^\mu)$; the lattice now unfolds.

similar to kernel support vector machines (SVMs), when applying the kernel trick (for a tutorial on the SVM, see Ref. 59). When performing topographic map formation, we require that the weight vectors are updated to minimize the expected value of the squared Euclidean distance $\|\mathbf{v} - \mathbf{w}_i\|^2$ and, thus, following our transformation Ψ, we instead wish to minimize $\|\Psi(\mathbf{v}) - \Psi(\mathbf{w}_i)\|^2$, which we will achieve by performing gradient descent with respect to $\mathbf{w}_i$. This descent leads to the following fixed point rules to which we have added a neighborhood function:

$$\begin{aligned} \mathbf{w}_i &= \frac{\sum_\mu \Lambda(i,i^*) K(\mathbf{v}^\mu, \mathbf{w}_i, \sigma_i)\mathbf{v}^\mu}{\sum_\mu \Lambda(i,i^*) K(\mathbf{v}^\mu, \mathbf{w}_i, \sigma_i)} \\ \sigma_i^2 &= \frac{1}{\rho d}\frac{\sum_\mu \Lambda(i,i^*) K(\mathbf{v}^\mu, \mathbf{w}_i, \sigma_i)\|\mathbf{v}^\mu - \mathbf{w}_i\|^2}{\sum_\mu \Lambda(i,i^*) K(\mathbf{v}^\mu, \mathbf{w}_i, \sigma_i)}, \quad \forall i \end{aligned} \tag{53}$$

with ρ a scale factor (a constant) designed to relax the local Gaussian (and d large) assumption in practice and with $i^* = \arg\max_{\forall i \in A} K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$. We refer to this algorithm as local density modeling (LDE). As an example, we reconsider the uniform distribution example with the same lattice and neighborhood settings as in the previous case. We take $\rho = 0.4$ as recommended in Ref. 58. The kernel radii rapidly grow in size, span a considerable part of the input distribution, and then decrease to smaller values, all of which is evidenced by the evolution of the average radius in Fig. 18b (thick dashed line). The converged lattice is shown in Fig. 17e. For larger ρ the average radius becomes smaller, but the lattice fails to unfold.

Finally, rather than having a real-valued neural activation, one could also threshold the kernel into a binary variable: In the kernel-based maximum entropy rule (kMER), a neuron i is activated by input $\mathbf{v}$ when $\|\mathbf{w}_i - \mathbf{v}\| < \sigma_i$, where σ_i is the kernel radius of neuron i and which defines a hyperspherical activation region, $\mathcal{S}_i$(30). The membership function, $\mathbb{1}_i(\mathbf{v})$, equals unity when neuron i is activated by $\mathbf{v}$, else it is zero. When no neurons are active for a given input, the neuron that is positioned closest to that input is defined active. The incremental learning rules for the weights and radii of neuron i are as follows:

$$\begin{aligned} \Delta\mathbf{w}_i &= \eta \sum_j \Lambda(i,j)\Xi_i(\mathbf{v})\text{sign}(\mathbf{v} - \mathbf{w}_i) \\ \Delta\sigma_i &= \eta\left(\frac{\rho_r}{N}(1 - \mathbb{1}_i(\mathbf{v})\right) - \mathbb{1}_i(\mathbf{v})), \quad \forall i \end{aligned} \tag{54}$$

with sign(.) the sign function taken componentwise, η the learning rate, $\Xi_i(\mathbf{v}) = \dfrac{\mathbb{1}_i}{\sum_j \mathbb{1}_j}$ a fuzzy membership function, and $\rho_r = \dfrac{\rho N}{N - \rho}$. It can be shown that the kernel ranges converge to the case in which the average probabilities become equal, $\langle \mathbb{1}_i \rangle = \dfrac{\rho}{N}, \forall i$. By virtue of the latter, kMER is said to generate an equiprobabilistic topographic map. The algorithm has been considered for a wide range of applications, such as shape clustering (60), music signal clustering (9), and the linking of patent and scientific publications databases (61). More recently, also a fixed point version called batch map kMER (BM-kMER) was

introduced (62) and applied to handwritten numerals clustering. To illustrate BM-kMER, we reconsider the uniform distribution example with the same lattice and neighborhood settings. We take $\rho = 0.5$. (Note that ρ acts as a smoothness constraint for the corresponding density estimate; for a strategy to choose it, see Refs. 9 and 60.) The converged lattice is show in Fig. 17f. From Figs. 18a and b we observe that the average log-likelihood and kernel ranges quickly converge to their asymptotic vales (thick full lines).

Correspondences Between Algorithms. A final note on the algorithms based on Gaussian mixture models: We observe that the kernel center update rules usually either consist of normalized versions of the neighborhood function multiplied with the posterior probability or, when a winner is defined, consist of the same but without the posterior probability. For the heteroscedastic case, we have usually a normalized function that contains a term related to the local variance estimate, $\|\mathbf{v}^{\mu} - \mathbf{w}_i\|^2/d$, scaled by the neighborhood function and the posterior probability; when a winner is defined, we can do without the posteriors. Because these update rules were defined from different objective functions, they lead us to conclude that the number of variations are limited, at least when lattice unfolding is a concern (although we only have verified the latter through simulation not through formal proofs, which is not likely to be possible; see also the next section). For example, summing over the product of the neighborhood function and the posterior probability of the same kernel leads to a confounding of the contributions of the two, which could hamper a proper lattice unfolding.

Therefore, different algorithms exist for the Gaussian heteroscedastic case, but because for the algorithms in this section the vanishing neighborhood range corresponds to Gaussian mixture modeling, the choice depends on the computational complexity and the lattice unfolding capacity.

Local Minima in Kernel-Based Topographic Map Formation

Convergence of nonprobabilistic topographic map formation rules is usually explained in terms of convergence to (albeit in some cases only approximately, as in the SOM algorithm) local minima of an objective function. If a well-defined objective function is being optimized, then the existence of local minima supports the application of fixed point rules rather than incremental update rules, which leads to a substantial increase in speed of convergence and a freeing of the user from having to choose the learning rate (and its cooling scheme). Local minima are often studied from the perspective that they could correspond to topographic errors or topological defects (17, 63), such as kinks in the one-dimensional case and twists in the two-dimensional case. These defects are difficult to iron out, if at all, when the neighborhood range vanishes. This discussion goes back to the proof of the SOM ordering process by Kohonen (12) and Cottrell and Fort (64) and the role played by the neighborhood function, albeit that the proof applies to the one-dimensional case only [a proof is unlikely to exist for the higher-dimensional case (65)]. When using kernel-based (or probabilistic) topographic maps, these defects are of a similar concern (except for the GTM, which is topology-preserving by construction), but this is much less studied because the connection between local minima and topographic defects is less clear (see e.g., Fig. 1 in Ref. 21). Indeed, for kernel-based topographic maps, an additional issue develops because of their connection to mixture modeling, thus, in the absence of the neighborhood function: Global optimization (maximization of the likelihood function) in mixture modeling is a major concern on its own and has been thoroughly studied (for a standard reference, see Ref. 66). The presence of the neighborhood function is thought to perform a deterministic annealing of the optimization process, which is viewed as a means to avoid local minima of the objective function (but not necessarily to avoid topological defects). This performance has been studied formally by Obermayer and coworkers (31) for the homoscedastic case, in which the kernel radius is driven by the annealing procedure. The neighborhood range can even be kept constant during lattice unfolding. For the heteroscedastic case, for some class of rules, the estimation of the kernel radius depends on the gradual neighborhood range decrease (e.g., in Equation (42), kernel radii), but this annealing is less efficient in avoiding minima because it is subject to the same problems as mixture modeling: Kernel radii can shrink to zero when the data distribution is sparse; thus, for this reason, one often prefers to keep a minimum neighborhood radius present (the neighborhood acts as a statistical kernel smoother; for a discussion, see Ref. 9). For another class of rules, local minima exist for which the kernels span the whole input range, especially when a large neighborhood range at the onset also implies a large kernel range (as in the ERGMM1 rule). Monitoring the log-likelihood and comparing it with what can be achieved with the mixture modeling case (thus, training the kernels without using a neighborhood function at all) could be a solution to detect the latter.

Kernels Other than Gaussians

In principle, kernels other than Gaussians could be used in topographic map formation. For example, Heskes (20) pointed out that his regularized mixture modeling approach in principle, could accommodate any kernel of the exponential family, such as the Gamma, multinomial, and the Poisson distribution.

In another case, the kernel is considered for which the differential entropy of the kernel output will be maximal given a Gaussian input, that is, the incomplete gamma distribution kernel (67). The fixed point rules are the same as for the extended batch map in Equation (41), but with the winner i^* defined as $i^* = \arg\max_{\forall i \in A} K(\mathbf{v}, \mathbf{w}_i, \sigma_i)$. The update of the winner leads to a decrease of its radius and thus a decrease in overlap with its neighboring kernels, which is used as an heuristic to maximize the joint entropy of the kernel outputs.

Another type of kernels is the Edgeworth-expanded Gaussian kernel, which consists of a Gaussian kernel multiplied by a series of Hermite polynomials of increasing order and of which the coefficients are specified by (the second- but also higher-order) cumulants (68). However, similar to the case considered earlier, when attempting to derive the kernel update rules in a maximum likelihood

format, the presence of the neighborhood means that we do not have a closed form solution for updating the kernel moments (and, thus, the cumulants), except for the first moment. To remedy this lake, an heuristic procedure based on the extended batch map was proposed.

In still another case, a mixture of Bernouilli distributions is taken (69) for the specific purpose to better encode binary data (e.g., word occurrence in a document). This approach also leads to an EM algorithm for updating the posteriors and the expected joint log-likelihood with respect to the parameters of the Bernouilli distributions. However, as the posteriors become quite peaked for higher dimensions, for visualization purposes, a power function of them was chosen. Several applications were demonstrated, including word grouping in document data sets (newsgroup data set) and credit data analysis (from the UCI repository).

CONCLUSION AND OUTLOOK

In this article we have reviewed kernel-based topographic map formation. We showes that several types of objective functions, learning principles, and kernel types are found. We started with the homoscedastic Gaussian kernels, considered the heteroscedastic case mostly as an extension of the former, and briefly discussed non-Gaussian kernels. A foreseeable development is the integration of kernel-based activation functions into toroidal or spherical topographic maps [which were first introduced by Helge Ritter (70)] to avoid boundary effects, especially when the natural data model is spherical.

Another expected development is to go beyond the limitation of the current kernel-based topographic maps in which the inputs need to be vectors (we already saw the extension toward categorical data). But in the area of structural pattern recognition, more powerful data structures can be processed, such as strings, trees, and graphs. The SOM algorithm has already been extended towards strings (71) and graphs, which include strings and trees (55, 72, 73). However, also new types of *kernels* for strings, trees, and graphs have been suggested in the support vector machine literature (thus, outside the topographic map literature) (for reviews, see Refs. 74 and 75). The integration of these new types of kernels into kernel-based topographic maps (such as the LDE algorithm that adopted the format of kernel SVMs) is yet to be done, but it could turn out to be a promising evolution for biochemical applications, such as visualizing and clustering sets of structure-based molecule descriptions, protein structures, and long DNA sequences.

ACKNOWLEDGMENTS

The author is supported by the Excellence Financing program (EF 2005) and the CREA Financing program (CREA/07/027) of the K.U. Leuven, the Belgian Fund for Scientific Research—Flanders (G.0248.03, G.0234.04), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), the Belgian Science Policy (IUAP P5/04), and the European Commission (NEST-2003-012963, STREP-2002-016276, IST-2004-027017, and ICT-2007-217077).

BIBLIOGRAPHY

1. P. M. Daniel and D. Whitteridge, The representation of the visual field on the cerebral cortex in monkeys, *J. Physiol. (Lond.)*, **159**: 203–221, 1961.
2. D. C. Van Essen, J. H. R. Maunsell, and J. L. Bixby, The middle temporal visual area in the macaque: Myeloarchitecture, connections, functional properties and topographic organization, *J. Comp. Neurol.*, **199**: 293–326, 1981.
3. R. Gattass, A. P. B. Sousa, and C. G. Gross, Visuotopic organization and extent of V3 and V4 of the macaque, *J. Neurosci*, **8**: 1831–1845, 1988.
4. V. B. Mountcastle, Modality and topographic properties of single neurons of cat's somatic sensory cortex, *J. Neurophysiol.*, **20**: 408–434, 1957.
5. J. H. Kaas, R. J. Nelson, M. Sur, C. S. Lin, and M. M. Merzenich, Multiple representations of the body within the primary somatosensory cortex of primates, *Science*, **204**: 521–523, 1979.
6. M. M. Merzenich, P. L. Knight, and G. L. Roth, Representation of cochlea within primary auditory cortex in the cat, *J. Neurophysiol.*, **38**: 231–249, 1975.
7. N. Suga and W. E. O'Neill, Neural axis representing target range in the auditory cortex of the mustache bat, *Science*, **206**: 351–353, 1979.
8. C. von der Malsburg, Self-organization of orientation sensitive cells in the striate cortex, *Kybernetik*, **14**: 85–100, 1973.
9. M. M. Van Hulle, *Faithful representations and topographic maps: From distortion- to information-based self-organization*, New York: Wiley, 2000.
10. D. J. Willshaw and C. von der Malsburg, How patterned neural connections can be set up by self-organization, *Proc. Roy. Soc. Lond. B*, **194**: 431–445, 1976.
11. T. Kohonen, Self-organized formation of topologically correct feature maps, *Biol. Cybern.*, **43**: 59–69, 1982.
12. T. Kohonen, *Self-organization and associative memory*. Heidelberg Germany: Springer, 1984.
13. T. Kohonen, *Self-organizing maps*. Heidelberg Germany: Springer, 1995.
14. R. Der and M. Herrmann, Phase transitions in self-organizing feature maps, *Proc. ICANN'93,* Amsterdam, The Netherlands, 1993, pp. 597–600.
15. E. Erwin, K. Obermayer, and K. Schulten, Self-organizing maps: Ordering, convergence properties and energy functions, *Biol. Cybern.*, **67**: 47–55, 1992.
16. S. P. Luttrell, Code vector density in topographic mappings: Scalar case, *IEEE Trans. Neural Networks*, **2**: 427–436.
17. T. Heskes and B. Kappen, Self-organization and nonparametric regression, *Proc. ICANN'95*, Vol. I, 1995, pp. 81–86.
18. A. Utsugi, Hyperparameter selection for self-organizing maps, *Neural Computat.*, **9**: 623–635, 1997.
19. H. Yin and N. M. Allinson, Self-organizing mixture networks for probability density estimation, *IEEE Trans. Neural Networks*, **12**: 405–411, 2001.
20. T. Heskes, Self-organizing maps, vector quantization, and mixture modeling, *IEEE Trans. Neural Networks*, **12** (6): 1299–1305, 2001.
21. T. Kostiainen and J. Lampinen, Generative probability density model in the self-organizing map, in *Self-Organizing Neural Networks: Recent Advances and Applications*,

U. Seiffert and L. Jain (eds.), 75–94, Heidelberg, Germany: Physica Verlag, 2002.

22. J. J. Verbeek, N. Vlassis, and B. Kröse, *The generative self-organizing map: A probabilistic generalization of Kohonen's SOM*, Informatics Institute, University of Amsterdam, The Netherlands, IAS-UVA-02-03, 2002.
23. A. Ultsch and H. P. Siemon, Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis, *Proc. Intl. Neural Networks*, Paris, France, 305–308, 1990.
24. H. U. Bauer, R. Der, and M. Herrmann, Controlling the magnification factor of self-organizing feature maps. *Neural Computat.*, **8**: 757–771, 1996.
25. B. Fritzke, Growing cell structures – a self-organizing network in *k* dimensions, in *Artificial Neural Networks 2*, I. Alexander and J. Taylor (eds.), Amsterdam, The Netherlands: Elsevier-Science Publishers, 1992.
26. N. N. R. Centre, Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ), Helsinki University of Technology, 2003, available; http://liinwww.ira.uka.de/bibliography/Neural/SOM.LVQ.html.
27. R. Durbin and D. Willshaw, An analogue approach to the travelling salesman problem using an elastic net method, *Nature*, **326**: 689–691, 1987.
28. C. M. Bishop, M. SvensÕn, and C. K. I. Williams, GTM: A principled alternative to the self-organizing map, *Proc. 1996 International Conference in Artificial Neural Networks (ICANN'96)*, 165–170, 1996.
29. C. M. Bishop, M. SvensÕn, and C. K. I. Williams, GTM: The generative topographic mapping, *Neural Computat.*, **10**: 215–234, 1998.
30. M. M. Van Hulle, Kernel-based equiprobabilistic topographic map formation, *Neural Computat.*, **10** (7): 1847–1871, 1998.
31. T. Graepel, M. Burger, and K. Obermayer, Phase transitions in stochastic self-organizing maps, *Physical Rev. E*, **56** (4): 3876–3890, 1997.
32. J. Sum, C. S. Leung, L. W. Chan, and L. Xu, Yet another algorithm which can generate topography map, *IEEE Trans. Neural Networks*, **8** (5): 1204–1207, 1997.
33. M. M. Van Hulle, Maximum likelihood topographic map formation, *Neural Comp.*, **17** (3): 503–513, 2005.
34. H. Ritter and K. Schulten, Kohonen's self-organizing maps: Exploring their computational capabilities, *Proc. IEEE Int. Conf. on Neural Networks (ICNN)*, San Diego, CA, 1988, I. pp. 109–116.
35. R. A. Redner and H. F. Walker, Mixture densities, maximum likelihood and the EM algorithm, *SIAM Review*, **26** (2): 195–239, 1984.
36. R. Durbin, R. Szeliski, and A. L. Yuille, An analysis of the elastic net approach to the traveling salesman problem, *Neural Computat.*, **1**: 348–358, 1989.
37. H. Ritter, T. Martinetz, and K. Schulten, *Neural computation and self-organizing maps: An introduction*, Reading, MA: Addison-Wesley, 1992.
38. G. J. Goodhill and D. J. Willshaw, Application of the elastic net algorithm to the formation of ocular dominance stripes, *Network*, **1**: 41–59, 1990.
39. S. Gorbunov and I. Kisel, Elastic net for stand-alone RICH ring finding, *Nuclear Instruments Methods Phys. Res. A*, **559**: 139–142, 2006.
40. K. D. Ball, B. Erman, and K. A. Dill, The elastic net algorithm and protein structure prediction, *J. Computat. Chem.*, **23** (1): 77–83, 2002.
41. K. Rose, E. Gurewitz, and G. C. Fox, Constrained clustering as an optimization Method, *IEEE Trans. Pattern Analysis Mach. Intell.*, **15** (8): 785–794, 1993.
42. M. Kass, A. Witkin, and D. Terzopoulos, Active contour models, *Int. J. Comput. Vision*, **1** (4): 321–331, 1987.
43. A. J. Abrantes and J. S. Marques, Unified approach to snakes, elastic nets, and Kohonen maps, *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP'95*, 1995, pp. 3427–3430.
44. S. J. Gilson, I. Middleton, and R. I. Damper, A localised elastic net technique for lung boundary extraction from magnetic resonance images, *Proc. Fifth International Conference on Artificial Neural Networks* Cambridge, UK, 1997, pp. 199–204.
45. A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum likelihood for incomplete data via the EM algorithm, *J. Roy. Statist. Soc., B*, **39**: 1–38, 1977.
46. C. M. Bishop, *Pattern Recognition and Machine Learning*, New York: Springer, 2006.
47. M. A. Carreira-Perpiñán and S. Renals, Dimensionality reduction of electropalatographic data using latent variable models, *Speech Commun.*, **26** (4): 259–282, 1998.
48. D. Axelson, I. J. Bakken, I. S. Gribbestad, B. Ehrnholm, G. Nilsen, and J. Aasly, Applications of neural network analyses to in vivo ^{1}H magnetic resonance spectroscopy of Parkinson disease patients, *J. Mag. Res. Imag.*, **16** (1): 13–20, 2002.
49. A. Kabán and A Scalable, Generative topographic mapping for sparse data sequences, *Proc. of the International Conference on Information Technology: Coding and Computing (ITCC'05)*, 2005, pp. 51–56.
50. A. Kitamoto, Evolution map: Modeling state transition of typhoon image sequences by spatio-temporal clustering, *Lecture Notes in Computer Science*, **2534**/**2002**: 283–290, 2002.
51. D. D'Alimonte, D. Lowe, I. T. Nabney, and M. Sivaraksa, Visualising uncertain data, *Proc. European Conference on Emergent Aspects in Clinical Data Analysis (EACDA2005)* Pisa, Italy, 2005, Available: http://ciml.di.unipi.it/EACDA2005/papers.html
52. D. Lowe and M. E. Tipping, NeuroScale: Novel topographic feature extraction using RBF networks, in Advances in Neural Information Processing Systems, Mozer, M. C. , Jordan, M. I. , Petsche, T. (eds.) Cambridge, MA: MIT Press, 1997.
53. P. Tiňo and I. Nabney, Hierarchical GTM: Constructing localized non-58 linear projection manifolds in a principled way, *IEEE Trans. Pattern Anal. Mach. Intell.*, **24** (5): 639–656, 2002.
54. T. Graepel, M. Burger, and K. Obermayer, Self-organizing maps: Generalizations and new optimization techniques, *Neurocomputing*, **21**: 173–190, 1998.
55. S. Seo and K. Obermayer, Self-organizing maps and clustering methods for matrix data, *Neural Networks*, **17** (8,9): 1211–1229, 2004.
56. M. M. Van Hulle, Fixed point rules for heteroscedastic Gaussian kernel-based topographic map formation, *WSOM07*, Bielefeld, Germany, September, 2007, CD ROM, 2007, Available: http://biecoll.ub.uni-bielefeld.de.
57. M. Benaim and L. Tomasini, Competitive and self-organizing algorithms based on the minimization of an information criterion, *Proc. ICANN'91*, 1991, pp. 391–396.
58. M. M. Van Hulle, Kernel-based topographic map formation by local density modeling, *Neural Computat.*, **14** (7): 1561–1573, 2002.

59. C. J. C. Burges. A tutorial on support vector machines for pattern recognition, *Data Mining Knowledge Discovery*, **2**: 121–167, 1998.

60. M. M. Van Hulle and T. Gautama, Optimal smoothing of kernel-based topographic maps with application to density-based clustering of shapes, *J. VLSI Signal Proc. Sys. Signal, Image, Video Tech.*, **37**: 211–222, 2004.

61. F. F. Deleus and M. M. Van Hulle, Science and technology interactions discovered with a new topographic map-based visualization tool, *Proc. 7th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, San Fransisco, 2001, pp. 42–50.

62. T. Gautama and M. M. Van Hulle, Batch Map Extensions of the Kernel-based Maximum Entropy Learning Rule, *IEEE Trans. Neural Networks*, **16** (2): 529–532, 2006.

63. T. Geszti, *Physical models of neural networks*, Singapore: World Scientific Press, 1990.

64. M. Cottrell and J. C. Fort, Etude d'un processus d'auto-organization, *Ann. Inst. Henri Poincare?*, **23**, 1–20, 1987.

65. E. Erwin, K. Obermayer, and K. Schulten, Self-organizing maps: Ordering, convergence properties and energy functions. *Biol. Cybern.*, **67**: 47–55, 1992.

66. G. J. McLachlan and D. Peel, *Finite Mixture Models*, New York: Wiley, 2000.

67. M. M. Van Hulle, Joint entropy maximization in kernel-based topographic maps, *Neural Computat.*, **14** (8): 1887–1906, 2002.

68. M. M. Van Hulle, Edgeworth-expanded topographic map formation, *WSOM05* Paris, France, September, 2005, pp. 719–724.

69. J. J. Verbeek, N. Vlassis, and B. J. A. Kr—se, Self-organizing mixture models, *Neurocomputing*, **63**: 99–123, 2005.

70. H. Ritter, Self-organizing maps in non-Euclidian space, in *Kohonen maps*, Oja E., Kaski, S. (eds.). Amsterdam, The Netherlands: Elsevier, 1998.

71. T. Kohonen, and P. Somervuo, P., Self-organizing maps on symbol strings. *Neurocomputing*, **21**: 19–30, 1998.

72. S. Günter and H. Bunke, Self-organizing map for clustering in the graph domain, *Pattern Recognition Letters*, **23**: 4–5–417, 2002.

73. J. J. Steil and A. Sperduti, Indices to evaluate self-organizing maps for structures, *WSOM07*, Bielefeld, Germany, September, 2007, CD ROM, 2007, Available: http://biecoll.ub.uni-bielefeld.de.

74. J. Shawe-Taylor and N. Cristianini, *Kernel Methods in Computational Biology*, Cambridge, MA: MIT Press, 2004.

75. B. Jin Y. Q. Zhang, and B. Wang, Evolutionary granular kernel trees and applications in drug activity comparisons, *Proc. of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'05)*, pp. 1–6.

MARC M. VAN HULLE
K.U. Leuven, Laboratorium voor
Neuro- en Psychofysiologie
Campus Gasthuisberg, Belgium