7

Self-Organizing Maps and Unsupervised Classification

F. Badran, M. Yacoub, and S. Thiria

This chapter is dedicated to the second group of neural networks: Topological self-organizing maps. Those models are subject to unsupervised learning, in contrast with multilayer perceptrons, which were described in previous chapters. Primarily, the purpose of those models is purely descriptive: some structure is sought in given data. There is neither precise action to perform, nor desired response to obtain. Alternatively, information compression can be considered as the purpose of unsupervised learning: a compact description of the data, with minimal distortion, is sought.

The unsupervised learning methods that are used by topological self-organizing maps stemmed from techniques that were first designed for competitive learning. Among pioneering works in the field, one may quote [Didday 1976] and [von der Malsburg 1973]. The models were made of parallel filters that analyzed the same observation. For that observation, the filters' responses were different, and the filter that generated the highest response was said to win the competition. That winner is then favored by competitive learning, and the training algorithm enhanced the response of that filter to that observation. The same operation is performed for all observations of the training set until stabilization of the parameters of the filters. At that stage, each filter has been made sensitive to features that are specific to a subset of the data set: it operates as a feature detector.

Topological maps or self-organizing maps were first introduced by T. Kohonen in 1981. The first models were designed for processing high-dimensional data. Very large data sets with high dimensional data vectors were involved in the applications under consideration. In order to process such data, the topological map visualization methodology is designed to partition available data into clusters of data that exhibit some similarity. The training process is driven by the data set. The specificity of topological maps is to provide the clusters with a neighborhood structure, which is actually a graph structure on a discrete set. Low-dimension lattices (1D, 2D or 3D grid) are most frequently considered.

The most important feature of self-organizing maps is the possibility of comparing clusters, which summarize the data. Each observation is allocated to a cluster. Each cluster is projected onto a node of the map. The comparison of projections stemming from different observations allows estimating the proximity between their respective clusters: similar observations are projected onto the same node. Otherwise, the dissimilarity increases with the distance that separates the two projections; that distance is computed on the map. Thus, the cluster space is identified to the map, so that projection enables visualizing simultaneously the cluster space and the observation space.

Unsupervised classifiers and self-organizing maps are closely related; most such methods of clustering aim at aggregating similar data. In that context, similar means close with regard to the application field and the underlying metric. The topological ordering is the specific contribution of neural networks with unsupervised learning to clustering, a key theme in data analysis [Duda et al. 1973; Jain et al. 1988].

In current decision systems, any clustering may contribute to supervised classification as well. Most applications that use self-organizing maps are classifiers. Moreover, some of them are perform regression. Several explanations help to understand that fact:

- Straightforward modifications of the basic algorithm allow its use as a supervised training algorithm [Cerkassky et al. 1991].
- Results of unsupervised training algorithms may easily be integrated into data processing systems that touch the same areas of interest as multilayer Perceptrons. Therefore, self-organizing maps are used to pre-process data: information provided by self-organizing maps may be processed by other algorithms for regression or classification.

Actually, clustering or unsupervised classification turns out to be complementary to discrimination or supervised classification (as described in Chap. 6 of this book). It can be considered in a sense, that any application project uses supervised information to some extent. Any system needs to be validated before use: therefore, available expert knowledge must be used, since an expert has processed some available data so that the associated desired response is known and may be used to tune the automatic system. In particular, this knowledge may be used to improve unsupervised models. If expert knowledge is widely available, then it is possible to take advantage of it from the beginning of the analysis, using supervised forms of self-organizing maps. Conversely, if it is scarce, it can be only used to interpret results of the unsupervised analysis: expert knowledge will be used after achieving clustering tasks. Thus, the approach is sequential: first, a partition of the data set is sought; the recognition itself is subsequently performed.

Self-organizing maps and their theoretical foundations are presented in this chapter. Those algorithms are described under a unified formalism, in order to connect them with data analysis methods from which they actually stemmed: self-organizing map algorithms may be viewed as extensions of well-known

algorithms of pattern recognition and clustering. The formalism that we use here is slightly different from the original Kohonen formalism. We will discuss all the necessary links between the various versions of the basic algorithm. Then a section will show in detail how expert knowledge can be used after performing unsupervised training.

This chapter is also application-oriented to a large extent. Two detailed studies of real-world applications are presented. Numerous self-organizing map based concrete projects were carried out in various application fields. Some recent books describe some of those applications [Oja et al. 1999; Kohonen 2001]. A review paper provides a fairly complete bibliography of all papers published between 1981 and 1997 ([Kaski et al. 1998] www.soe.ucsc.edu/NCS). The Helsinki University Web site (http://www.cis.hut.fi/ research/som-research/) addresses a large variety of topics: computer vision, image analysis, image compression, medical imagery, handwriting recognition, speech recognition, signal analysis, music analysis, process control, robotics, Web searching and so on.

The first application that is described in the present chapter deals with remote sensing. By analyzing the details of the modeling that was performed, we will help understand how self-organizing maps are used to perform data analysis. Kohonen's research group performed the second application: the Websom system, which is aimed at document searching on the Web. This application is interesting because the relevant data exhibit very large dimensionality. It is a striking example demonstrating the expected computational power of self-organizing maps.

7.1 Notations and Definitions

This section defines the notations that will be used throughout the present chapter. The set D denotes the observation space. We assume that the observations are real-valued and multidimensional; therefore, D is a subset of the n-dimensional vector space $\mathbb{R}^n$. Each vector belonging to D is associated to a particular encoding of an individual observation, which is taken from the given population. N observation vectors are assumed to be available: they are associated to N individuals. They form the subset $A = \{z_i; i = 1, \ldots, N\}$. Actually, A is included in D. Naturally, it is assumed that A is a representative sample of the considered population. According to that assumption, A is the training set that allows parametric estimation.

All the methods that will be described aim, in a first step, at reducing the information that is present in D. They do so

- by building a finite subset $W = \{\boldsymbol{w}_c; c = 1, \ldots, p\}$ of D; those n-dimensional vectors will be called reference vectors or simply reference throughout this chapter;

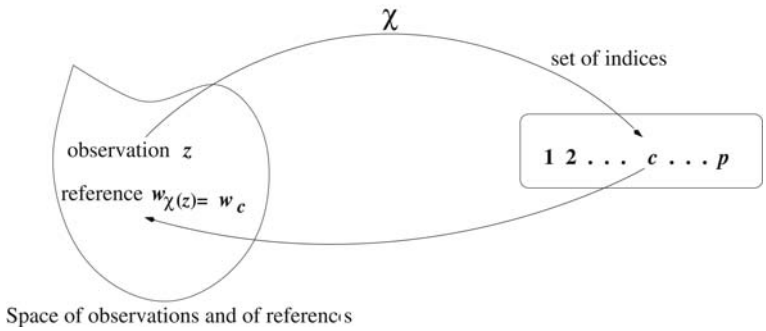

Fig. 7.1. General diagram of the modeling process: one observation $\boldsymbol{z}$ is associated to an index c that is selected among p indices using a function χ; that index allows defining the associated reference $\boldsymbol{w}_c$

- by defining an allocation function χ from D into the index set $\{1, \ldots, p\}$; that function performs a partition $P = \{P_1, \ldots, P_c, \ldots, P_p\}$ of D into p subsets, $P_c = \{\boldsymbol{z} \in D / \chi(\boldsymbol{z}) = c\}$.

Figure 7.1 describes graphically the modeling process: one observation $\boldsymbol{z}$ is associated to an index c that is selected among p indices using a function χ; that index allows defining the associated reference $\boldsymbol{w}_c$. Thus, the reference vector $\boldsymbol{w}_c$ is the representative example of the set P_c. It summarizes all the information contained in P_c. In the following, reference $\boldsymbol{w}_c$ or its associated index c will be used, depending on the context, for representing the observation subset P_c. We will estimate the model parameters from the observations of the training set A. Therefore, we denote by n_c the number of elements of P_c.

The knowledge of the reference vector set W and of the allocation function χ generates what is called a *vector quantization*. All known methods to determine W and χ can be derived from a variational principle and amount to a cost function minimization. Each method has a specific cost function. The latter incorporates the specific properties of the associated quantization. The vector quantization permits the allocation of a reference $w_{\chi(z)}$ to any observation $\boldsymbol{z} \in D$. That reference index is $\chi(\boldsymbol{z})$. Furthermore, the knowledge of the allocation function χ completely determines the partition of the set D into p subsets.

Although the cost functions are different for different methods, all methods that will be described share common features. In the following, the formalism of dynamic clustering will be used. That approach is iterative. Each iteration consists in two steps: a minimization step computes the reference vectors, and an allocation step changes the allocation function χ. Under some assumptions, the cost function decreases at each step and eventually converges towards a local minimum. That minimum depends strongly on the choice of the reference vector set that was selected to initialize the algorithm.

The k-means algorithm is a traditional unsupervised classification algorithm. It is the ancestor of self-organizing maps. In the next section, we describe both the most classical form of that algorithm, and its variants that give insight into the connections with self-organized maps.

For all methods, we first describe the standard version of each algorithm. Then we describe its most popular variants (stochastic or probabilistic versions).

7.2 The k-Means Algorithm

7.2.1 Outline of the k-Means Algorithm

The most known vector quantization method is the k-means algorithm. That method finds the set of reference vectors W and the allocation function χ by minimizing the cost function:

$$I(W,\chi) = \sum_{\boldsymbol{z}_i \in A} \left\| \boldsymbol{z}_i - \boldsymbol{w}_{\chi(\boldsymbol{z}_i)} \right\|^2 = \sum_c \sum_{\boldsymbol{z}_i \in P_c \bigcap A} \|\boldsymbol{z}_i - \boldsymbol{w}_c\|^2.$$

The quantity

$$I_c = \sum_{\boldsymbol{z}_i \in P_c \bigcap A} \|\boldsymbol{z}_i - \boldsymbol{w}_c\|^2$$

is the local inertia, with respect to the reference vector $\boldsymbol{w}_c$, of the observations of the learning set A that are allocated to that reference vector. Therefore, those observations belong to the subset P_c. That inertia is the squared quantization error performed when the observations of the subset P_c are replaced by the reference vector $\boldsymbol{w}_c$ that represents them. The total cost $I(W,\chi)$, which is to be minimized, is the sum of the local inertias I_c. In order to minimize $I(W,\chi)$, one must define the allocation function χ. The quantity to minimize becomes

$$I(W,\chi) = \sum_c I_c = \sum_c \sum_{\substack{\boldsymbol{z}_i \in A \\ \chi(\boldsymbol{z}_i)=c}} \|\boldsymbol{z}_i - \boldsymbol{w}_c\|^2.$$

The algorithm is implemented sequentially. An iteration is split into two phases. The first sep consists in minimizing $I(W,\chi)$: assuming that the reference vectors are kept fixed, it computes the activation function χ. In the second step, the value of the allocation function takes on the value that was just computed: the cost function is then minimized with respect to the parameters W of the reference set. In that two-phase iterative process, the value of the cost function $I(W,\chi)$ decreases at each step.

Thus, an iteration can be summarized as follows:

- *Allocation phase:* $I(W, \chi)$ is minimized with respect to the χ; during that phase, the reference vectors retain their previous values (or the initial values for the fist iteration). The minimization is performed when each observation $\boldsymbol{z}_i$ is allocated to the reference $\boldsymbol{w}_c$ by the allocation function χ:

$$\chi(\boldsymbol{z}) = \arg\min_r \|\boldsymbol{z} - \boldsymbol{w}_r\|^2 . \tag{1}$$

  In that relation, r varies from 1 to p (the number of reference vectors). By allocating the closest reference vector (in the sense of Euclidean distance) $\boldsymbol{w}_c$ to each observation $\boldsymbol{z}_i$, the cost function $I(W, \chi)$ is minimized. The new allocation function χ defines a new partition P of the set D (the closest reference vector has to be understood according to the Euclidean distance). In the following, n_c is the cardinal of the set $A \cap P_c$.
- *Minimization phase:* $I(W, \chi)$ is minimized with respect to the reference set W; the allocation function χ that was computed at the previous step is kept constant. The cost function $I(W, \chi)$ is then a convex quadratic function with respect to W. Its global minimum is reached for

$$\frac{\partial I}{\partial W} = \left[\frac{\partial I}{\partial \boldsymbol{w}_1}, \frac{\partial I}{\partial \boldsymbol{w}_2}, \ldots, \frac{\partial I}{\partial \boldsymbol{w}_p}\right]^T = 0.$$

The computation of the gradient that is associated to each reference vector $\boldsymbol{w}_c$ provides a new set of vector equations

$$2 \sum_{\substack{\boldsymbol{z}_i \in A \\ \chi(\boldsymbol{z}_i)=c}} (\boldsymbol{z}_i - \boldsymbol{w}_c) = 0,$$

which define the new reference vectors

$$\boldsymbol{w}_c = \frac{1}{n_c} \sum_{\boldsymbol{z}_i \in P_c \bigcap A} \boldsymbol{z}_i. \tag{2}$$

That algorithm can be proved to converge. If the allocation function that was computed in the first phase is applied, the class of an observation $\boldsymbol{z}$ changes only if its contribution to the global inertia that is computed with respect to the reference set W decreases. Therefore, that global inertia is smaller than the current value of $I(W, \chi)$. The second phase consists in updating the reference set W. Each reference vector $\boldsymbol{w}_c$ defines the center of inertia of the observation set $P_c \cap A$. That requires that $I(W, \chi)$ decrease, since it is the inertia with respect to the center of inertia of partition P. When the two phases are alternatively iterated, the cost function $I(W, \chi)$ decreases. $I(W, \chi)$ is expressed as a function of the trace of partition P on the data set A. That trace is a partition of A. Since the number of partitions of set A is finite, the iterative process converges to a local minimum of the cost function $I(W, \chi)$ with respect to the reference set and to the allocation function.

The implementation of the k-means algorithm can be summarized as follows:

k-Means Algorithm

1. Initialization phase: $t = 0$, choose the initial p reference vectors (randomly, in general), choose the maximal number of iterations N_{iter}.
2. Iteration: at iteration t, the reference set W^{t-1} is known from the previous step:
   Allocation phase: update the allocation function χ^t that is associated to the reference set W^{t-1}: a reference vector is allocated to each observation $\boldsymbol{z}$, as given in (1).
   Minimization phase: compute the new reference vectors W^t, as given in (2).
3. Iterate until the specified maximum number of iterations is reached, or until I stabilizes.

Note that the k-means algorithm may be considered as belonging to the family of dynamic clustering algorithms [Didday 1976]. It is a general method that provides a local minimum of a cost function. That method is based on using two entities: the set of partitions of the original data set into p subsets, and the space W of the representation (which may be different from the data set). Then, a subset P_k of the partition will be represented by an element $\boldsymbol{w}_k$, which will be its associated element of W. The discrepancy between an element $\boldsymbol{x}$ of the data set and its associated element $\boldsymbol{w}_k$ will be assessed by a positive dissimilarity function d such that the smaller $d(\boldsymbol{x}, \boldsymbol{w}_k)$, the better $\boldsymbol{x}$ agrees with $\boldsymbol{w}_k$. Thus, it is necessary to define a partition $P = \{P_k/k = 1 \dots p\}$ into p data subsets and jointly a set $W = \{\boldsymbol{w}_k/k = 1 \dots p\}$ of p representative elements such that they minimize a cost function. The latter will be defined from the training set by

$$H\,(P, W) = \sum_{k=1}^{p} \sum_{x_i \in P_k \bigcap A} d\,(\boldsymbol{x}, \boldsymbol{w}_k).$$

The dynamic clustering algorithm minimizes that function iteratively way. First, p representative elements are selected to initialize the process. Then, the general iteration consists of two phases: first an allocation phase that minimizes the cost function with respect to the partition, given the representative elements. Then, during the subsequent phase, the criterion is minimized with respect to the p representative elements, retaining the previous allocation function. In the particular case of k-means algorithm, the reference vectors are the representative elements and the dissimilarity function d is the Euclidean distance.

7.2.2 Stochastic Version of *k*-Means

The previous algorithm has all the shortcomings of deterministic optimization algorithms. Generally, those algorithms depend strongly on initial conditions, and converge to a local minimum. The optimization mechanism does not allow exploring all the local minima of the cost function. As shown in Chap. 2, optimization can be improved simply by running several optimization processes from various initial conditions, and selecting the best local minimum. In the case of unsupervised learning, the best reference vector set and the best partition will be selected, i.e., those which generate the smallest value of the cost $I(W, \chi)$.

At each iteration, during the minimization phase, the reference set that minimizes the cost function $I(W, \chi)$ for a given allocation function χ is determined. Yet, it is not necessary to complete the move towards the global minimum of the cost function to guarantee that it decreases. At time t, given the allocation function χ^t, finding a reference vector set W^t such that

$$I\left(W^t, \chi^t\right) \leq I\left(W^{t-1}, \chi^t\right)$$

is sufficient.

One may implement a simple gradient descent algorithm, which guarantees the decrease of $I(W, \chi)$ at each step. The computation of the gradient requires the computation of the partial derivatives of $I(W, \chi^t)$ with respect to all the components of each reference vector $\boldsymbol{w}_c$,

$$\frac{\partial I}{\partial \boldsymbol{w}_c} = \sum_{\substack{\boldsymbol{x}_i \in A \\ \chi^t(\boldsymbol{z}_i)=c}} 2\left(w_c - \boldsymbol{z}_i\right).$$

The computation of the reference vectors that was performed by relation $\boldsymbol{w}_c = 1/n_c \sum_{\boldsymbol{z}_i \in P_c \bigcap A} \boldsymbol{z}_i$ at each step is replaced by

$$\boldsymbol{w}_c^t = \boldsymbol{w}_c^{t-1} - \mu_t \frac{\partial I}{\partial \boldsymbol{w}_c} = \boldsymbol{w}_c^{t-1} - 2\mu_t \sum_{\substack{\boldsymbol{x}_i \in A \\ \chi^t(\boldsymbol{z}_i)=c}} \left(\boldsymbol{w}_c^{t-1} - \boldsymbol{z}_i\right).$$

That is the simple gradient descent optimization method that was described in Chap. 2. The allocation function χ^t that appears in the expression of the gradient is defined in the allocation phase of iteration t, the quantity μ^t is the training rate at iteration t, and the reference vector $\boldsymbol{w}_c^{t-1}$ was computed during previous iteration. That algorithm is not adaptive, since it minimizes the global cost function $I(W, \chi)$. To implement any change, the whole data set $\boldsymbol{A}$ has to be used.

The adaptive or stochastic version of the k-means algorithm is obtained by the following modification of the basic optimization procedure. The minimization is now performed stochastically: the terms of the sum in relation $I(W, \chi)$ are considered separately. At each iteration, a single observation $\boldsymbol{z}_i$ of

the data set is presented. It leads to the update of the closest reference vector $\boldsymbol{w}_{\chi(\boldsymbol{z}_i)}$. It amounts to decrease only the single term $\|\boldsymbol{z}_i - \boldsymbol{w}^t_{\chi(\boldsymbol{z}_i)}\|^2$ of $I(W, \chi)$ by gradient descent.

Then the partial gradient $2(\boldsymbol{w}^t_{\chi(\boldsymbol{z}_i)} - \boldsymbol{z}_i)$ is used to update the reference vector $\boldsymbol{w}_{\chi(\boldsymbol{z}_i)}$ as follows:

$$\boldsymbol{w}^t_{\chi^t(\boldsymbol{z}_i)} = \boldsymbol{w}^{t-1}_{\chi^t(\boldsymbol{z}_i)} - 2\mu_t \left(\boldsymbol{w}^{t-1}_{\chi^t(\boldsymbol{z}_i)} - \boldsymbol{z}_i \right).$$

A good minimum is obtained by presenting each observation of the data set $\boldsymbol{A}$ repeatedly (N_{iter} must be large enough). When updating the reference vectors, the gradient step μ decreases. When training starts, the value of μ is relatively large, and the decrease of the cost function is not strictly guaranteed. As training proceeds, μ^t becomes small enough, each reference vector update is small, and several updates must be performed to produce a significant change in the cost function. In that case, there is no major difference between the total gradient and the addition of several steps of the partial gradient. Then the stochastic gradient algorithm behaves as the classical version of the k-means algorithm does. The stochastic algorithm shows that k-means may be considered as a competitive algorithm, where each observation of the data set attracts the closest reference vector. Repeatedly presenting each observation while the gradient step μ decreases allows finding a satisfactory partition P such that each reference vector is the center of inertia of each subset of the partition.

The following summary of stochastic k-means may be useful for algorithm implementation:

Stochastic k-Means

1. Initialization: $t = 0$,
   Choose the initial p reference vectors (randomly, in general),
   choose the maximal number of iterations N_{iter} and the law of decrease of the gradient step μ^t.
2. Iteration t: keeping the reference set W^{t-1} constant, as computed at the previous iteration, choose randomly or sequentially an observation $\boldsymbol{z}_i$, and compute the gradient step (or learning rate) μ^t.
   Allocation phase: given W^{t-1}, $\boldsymbol{z}_i$ is assigned assign to the closest reference element of W^{t-1}, which defines a new allocation function χ^t.
   Minimization phase: the new reference vector $\boldsymbol{w}^t_{\chi(\boldsymbol{z}_i)}$ is computed (2).
3. Iterate until the specified maximum number of iterations is reached, or until I stabilizes.

The learning rate must decrease as the number of iterations t increases. It may be piecewise constant, equal to $1/\sqrt{t}$, or have any other ap propriate form.

The three experiments that are shown on Fig. 7.2 allow understanding the evolution of the k-means algorithm, classical and stochastic. They demonstrate the sensitivity of the solution to the number of reference vectors and to their initialization. For those experiments, the observations were generated randomly from spherical Gaussian laws with standard deviation $\sigma = 0.1$. Those laws are called the Gaussian modes. The first experiment is seeks a two-class partition; it shows the evolution of the set of reference vectors that capture observations from the four modes. During training, the two reference vectors are attracted by the two blocks made of the observations of the left-hand and right-hand sides. They stabilize at the centers of the two observation blocks. The second experiment makes use of the same observation data set, but seeks a four-class partition, with two different initializations of the reference vectors: at the center in the first experiment, and at the bottom right in the second experiment. In the first case, the position, which is symmetric with respect to the problem, allows finding the four classes produced by the four Gaussian modes. With the second initialization, three reference vectors are assigned to the data generated by the two right-hand Gaussian modes, and the last one is assigned to the data generated by the other two modes.

7.2.3 Probabilistic Interpretation of k-Means

The k-means is minimizes the cost function $I(W, \chi)$, which is the sum of the local inertias I_c. We defined that cost function by following geometric and kinetic intuition. It is possible to follow another approach. Actually, the cost function has a natural probabilistic interpretation. In order to get insight into that, a probabilistic model of data generation must be defined: we assume that the observations of the training set are an i.i.d. sample of a mixture of p Gaussian modes,

$$p(\boldsymbol{z}) = \sum_{c=1}^{p} \alpha_c f_c(\boldsymbol{z}), \quad \text{with } \sum_{c=1}^{p} \alpha_c = 1.$$

Each Gaussian mode has density f_c with expectation $\boldsymbol{w}_c$ and covariance matrix equal to Σ_c. Therefore, this density is given by

$$f_c(\boldsymbol{z}) = \frac{1}{(2\pi)^{n/2} \det(\Sigma_c)^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{w}_c)^T \Sigma_c^{-1} (\boldsymbol{z} - \boldsymbol{w}_c)\right].$$

It is well known that a Gaussian mixture model is a general formalism, which can be used for modeling complex probability distributions [Duda et al. 1973]. The mixture assumption states that each observation is a realization of one of the hidden random variables with normal density f_c. The mode is

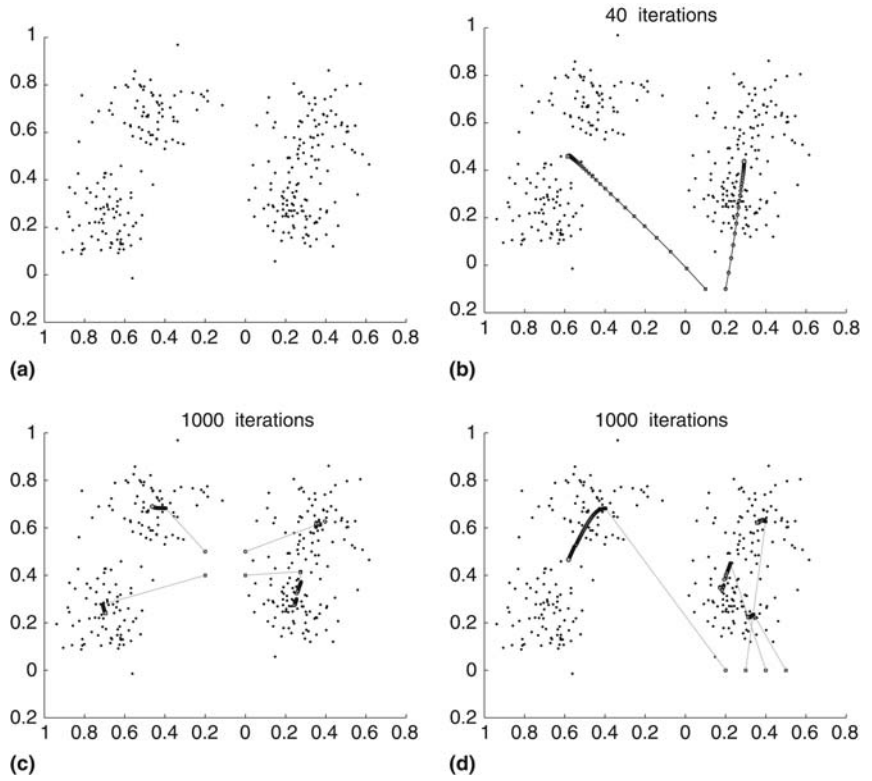

Fig. 7.2. Examples of operation of the k-means algorithm: sensitivity to initial conditions and to the number of reference vectors. Observations and reference vectors are shown on the same picture. (**a**) Representation of the learning set A: the data are generated from four Gaussian modes. (**b**) Evolution of the two reference vectors that were initialized at bottom right of the picture: each reference vector is assigned the observations that are generated from two Gaussian modes. Pictures (**c**) and (**d**) show the evolution of four reference vectors that were generated in two different ways. (**c**) The reference vectors are initialized at the center of the picture; each of them is assigned observations coming from one a Gaussian mode (**d**) The reference vectors are initialized at the bottom right of the picture: three reference vectors share the observations generated by two Gaussian modes; the last reference vector collects the observations generated by the other two modes

selected among p Gaussian modes with the prior distribution α_c. Equivalently, to generate the data, one must first choose randomly the mode according to the discrete probability α_c, and then to generate the observation from the probability law of the selected mode. Thus, that model generates a data set, which is partitioned by construction into p subsets. The subset that is labeled by index c contains about $N\alpha_c$ observations. Those observations are split around the reference vector $\boldsymbol{w}_c$. The subset has an ellipsoidal shape

that is defined by the eigenvectors and eigenvalues of the covariance matrix Σ_c. Note that the mixture model is general, since it can approximate any probability distribution with arbitrary accuracy when the number of modes p and the Gaussian mode parameters are selected appropriately. The geometric characteristics of the data set repartition may be described in an analytic way using the mixture model.

In that framework, the probabilistic k-means interpretation requires additional assumptions:

- The prior density on the mode set is uniform, i.e. all the α_c's are equal to $1/p$.
- The p normal densities f_c has the same covariance matrix equal to $\sigma^2 I$, where I is the identity matrix and σ is the common standard deviation. Therefore, those densities are given by

$$f_c(\boldsymbol{z}) = \frac{1}{(2\pi)^{n/2}\,\sigma^n} \exp\left[-\frac{\|\boldsymbol{z} - \boldsymbol{w}_c\|^2}{2\sigma^2}\right].$$

- The set A is an i.i.d. sample of a random variable that has the probability density $p(\boldsymbol{z})$.

Those assumptions restrict the validity domain of the interpretation. The observations must be assumed to be partitioned into p clusters. Those clusters are assumed to be isotropic, to have the same number of elements and to have the same probability distribution.

Thus, the probabilistic version of k-means amounts to estimating the reference vectors and the common standard deviation by maximizing the likelihood of the data set A. That estimation is performed by maximizing $p(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N)$ where $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N$ are the observations. Under the independence assumption one has

$$p(\boldsymbol{z}_{1,}, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N) = \prod_{i=1}^{N} p(\boldsymbol{z}_i).$$

As in the previous section, the allocation function χ is supposed to assign to each observation $\boldsymbol{z}_i$ its generating mode. The random generating modes are the mixture components. Therefore the allocation function χ defines a partition of the training set A into p subsets. If the classifying likelihood is defined by

$$p(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots \boldsymbol{z}_N \,|\, \chi) = \prod_{i=1}^{N} \alpha_{\chi(\boldsymbol{z}_i)} f_{\chi(\boldsymbol{z}_i)}(\boldsymbol{z}_i) = \left(\frac{1}{p}\right)^N \prod_{i=1}^{N} f_{\chi(\boldsymbol{z}_i)}(\boldsymbol{z}_i),$$

then maximizing the classifying likelihood amounts to minimizing

$$V(W, \sigma) = -\ln p(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N)$$

$$= \frac{1}{2\sigma^2} \sum_{i=1}^{N} \left\| \boldsymbol{z}_i - \boldsymbol{w}_{\chi(\boldsymbol{z}_i)} \right\|^2 + Nn \ln \sigma + \text{cte}$$
$$= \frac{1}{2\sigma^2} I\left(W, \chi\right) + Nn \ln \sigma + \text{cte}.$$

The minimization of $V(W, \sigma, \chi)$ may be performed in two steps:

- In the first step, the cost function $I(W, \chi)$ that appears in the expression of $V(W,\ \sigma)$ is minimized. One recognizes the global inertia term discussed in the previous section. The k-means algorithm is implemented (also in two steps as described above). That step leads to a local minimum of $I(W, \chi)$, denoted as $I_{\min}$.
- In the second step the quantity

$$\frac{1}{2\sigma^2} I_{\min} + Nn \ln \sigma$$

is minimized with respect to σ. That expression is minimum when its derivative is equal to zero. Therefore, one has

$$\sigma = \sqrt{\frac{I_{\min}}{Nn}}.$$

Thus, the k-means algorithm can be interpreted in a probabilistic framework. The minimization of the cost function $I(W, \chi)$ amounts to the parametric estimation of a probabilistic mixture model under very restrictive assumptions. The assumption of the isotropic identical distribution of the components of the mixture, with the single covariance matrix $\sigma^2 I$, should be emphasized. From a geometric point of view, that algorithm assumes that the data are split into p equally weighted spherical clusters with the same radius. Such is not always the case, so that the assumption is a severe limitation to the application range of the original k-means algorithm.

The following simulation gives insight into the behavior of k-means when the true data distribution does not comply with the assumptions of the probabilistic model. The observations that are shown on Fig. 7.3 are significantly non-isotropic and do not comply with the assumption of equal standard deviations. Therefore, the implementation of k-means in that case favors a solution, which is associated to a partition into two subsets that are as spherical as possible. Thus, the reconstructed partition is far from the original one (Fig. 7.3b).

In order to circumvent the problem, it may be efficient to display a larger number of reference vectors: Fig. 7.4 shows the reference vectors and the associated partition if five reference vectors are used. In that case, four reference vectors are allocated to the left mode while the last reference vector represents the other mode. Then, the problem of clustering the modes to reconstruct the two original classes must be solved. Alternative data analysis methods, such as hierarchical classification, can be taken advantage of. That methodology will be demonstrated in the section "Classification and topological maps," where the introduction of expert knowledge is addressed.

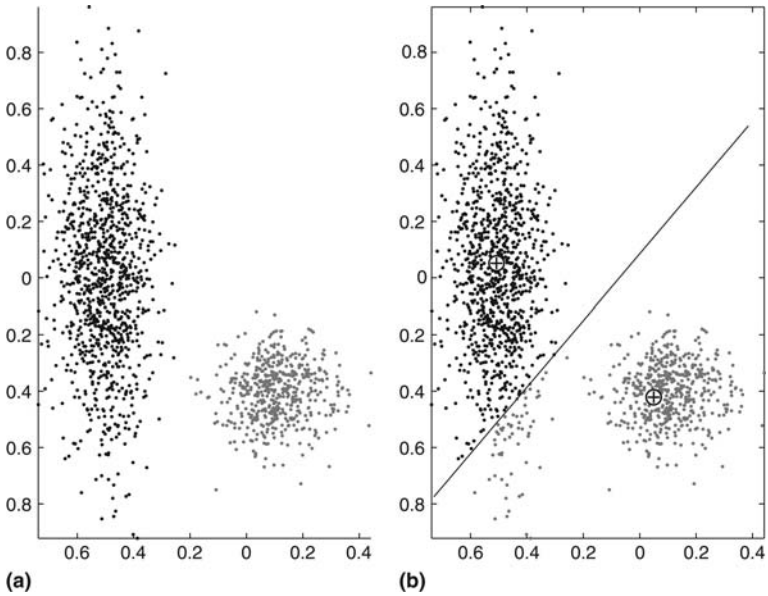

Fig. 7.3. Example of application of the k-means algorithm: a Gaussian mixture model of two modes with different anisotropic covariance matrixes generates the data. Crosses denote the positions of the reference vectors. (**a**) Training set A. (**b**) The reference vectors and the associated partition after stabilization of the algorithm; an oblique line separates the two sets. The algorithm has not reconstructed the true partition

In order to reconstruct the original distributions, the assumption of isotropic covariance must be relaxed. That is possible if the covariance matrices Σ_c of the Gaussians are not supposed to be identical (yet semi-positive definite). Then, the $n(n-1)/2$ elements of the matrices must be estimated, in addition to the reference vectors $\boldsymbol{w}_c$. The model is more complex since it has a larger number of parameters. A maximum likelihood methodology may perform the estimation, using the EM (Expectation-Maximization) algorithm [Dempster et al. 1977].

7.3 Self-Organizing Topological Maps

7.3.1 Self-Organizing Maps

In the early 1980's, Kohonen described a self-organization algorithm that defines a projection of the data space D onto a discrete, low-dimensionality space. That space has a non-oriented graph structure that is a generally a 1-, 2- or 3-dimensional mesh; that graph will be hereinafter termed the map. Actually, the set C is made of interconnected neurons: the connections between

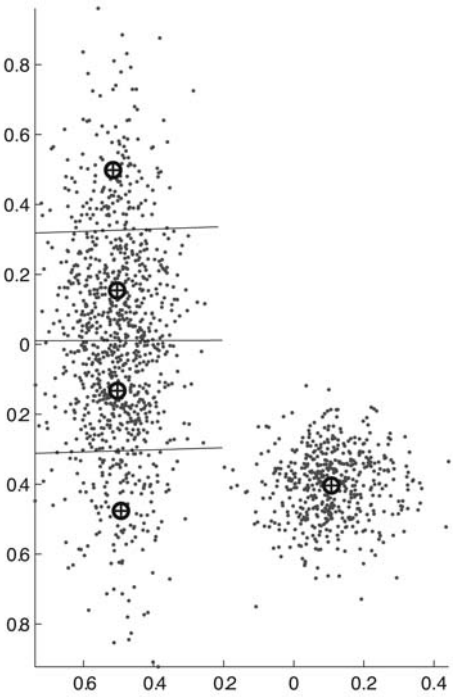

Fig. 7.4. Application of k-means with 5 reference vectors on the same data distribution as in Fig. 7.3; observations are distributed according two Gaussian modes with different anisotropic covariance matrices. Crosses denote the positions of the reference vectors. Four reference vectors are allocated to the first Gaussian mode. The last reference vector is allocated to the second Gaussian mode

neurons are the edges of the graph. For simplicity, we denote the whole graph and the set of its nodes with the same letter C. The graph structure allows the definition of an integer distance δ on C as follows: the length of a path on the graph is the number of edges of that path. For all the couple of neurons (c, r) of the map, $\delta(c, r)$ is the length of the shortest path on C between c and r. For any neuron c, that integer distance leads to defining the neighborhood of c of order d,

$$V_c\,(d) = \{r \in C, \delta\,(c, r) \leq d\}\ .$$

As mentioned above, the maps that are currently used are regular lattices. Therefore, the distance and the neighborhoods are quite easy to visualize, and they define the discrete topology on the map in a straightforward way. Examples of distance and neighborhoods are shown on Fig. 7.5 for a 2D grid.

For self-organizing maps, an association is sought between neurons of C and reference vectors in data space D, similarly to k-means. Training enables the set of reference vectors to sample the underlying probability distribution on the data set as faithfully as possible. In the case of topological maps,

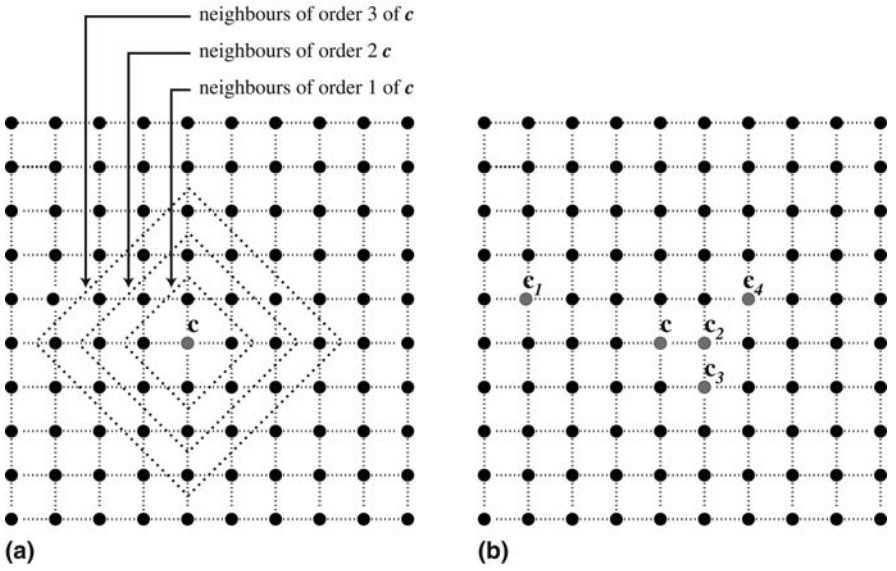

Fig. 7.5. The discrete topology of a 2D-topological map. The map features 10×10 neurons; each dot of the picture denotes a neuron c. The distance δ between two neurons is defined on the grid. (**a**) shows $V_c(1), V_c(2), V_c(3)$, which are neighborhoods of order 1, 2 and 3 of neuron c; (**b**) shows some distances between neurons: $\delta(c, c1) = 4, \delta(c, c2) = 1, \delta(c, c3) = 2, \delta(c, c4) = 3$

an additional constraint is imposed to retain the topology of the map: two neighboring neurons r and c are associated to reference vectors $\boldsymbol{w}_c$ and $\boldsymbol{w}_r$ that are close for the Euclidean distance in data space D.

That cursory description shows clearly that the self-organizing map algorithm is an extension of k-means. We will further show that it minimizes an appropriate cost function, which takes into account the inertia of the partition of the data set, and which guarantees that the topology of C is retained. In order to design such a cost function, the inertia function of the k-means algorithm will be generalized, by adding specific terms that take into account the topology of the map, through the distance δ and the associated neighborhoods.

The concept of neighborhood is taken into account through kernel functions K which are positive and such that $\lim_{|\chi|\to\infty} K(x) = 0$. Those kernels define influence regions around each neuron c. The distances $\delta(c, r)$ between neurons c and r of the map allow the definition of the relative influence of the neurons on elements of the data set. The quantity $K(\delta(c, r))$ quantifies that influence.

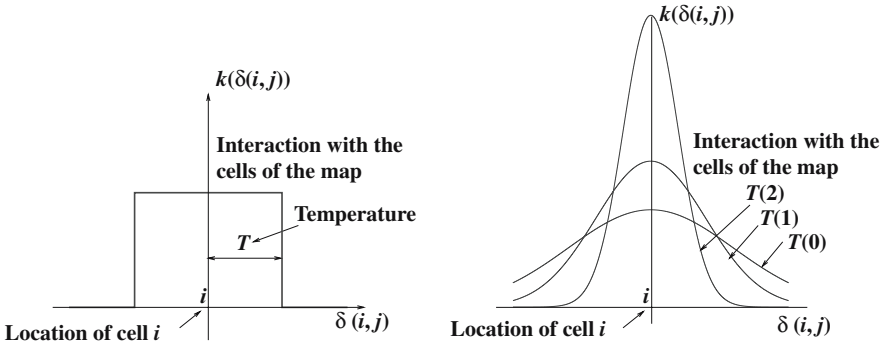

Fig. 7.6. Threshold neighborhood kernel (*left picture*) and Gaussian neighborhood kernel (*right picture*). In the case of the threshold kernel, neurons either belong to the neighborhood and share the same influence, or do not belong to the neighborhood, hence have no influence at all. In the case of Gaussian kernels, the influence between two neurons depends on their mutual distance

To take advantage of the size of the neighborhood, the family of kernels K^T that is parameterized by T (where T stands for temperature) will be used:

$$K^T(\delta) = K(\delta/T)$$

Figure 7.6 shows kernel functions that are commonly used in applications:

- $K(\delta) = \begin{cases} 1 & \text{if } \delta < 1 \\ 0 & \text{otherwise} \end{cases}$ hence $K^T(\delta) = \begin{cases} 1 & \text{if } \delta < T \\ 0 & \text{otherwise} \end{cases}$
- $K(\delta) = \exp(-|\delta|)$ hence $K^T(\delta) = \exp\left(-\frac{|\delta|}{T}\right)$
- $K(\delta) = \exp(-\delta^2)$ hence $K^T(\delta) = \exp\left(-\frac{\delta^2}{T^2}\right)$.

Figure 7.7 shows graphs of various kernels, for different values of parameter T. If we choose a level α such that the influence of a neuron that is below α is considered negligible ($K^T(\delta) < \alpha$), the radius of the effective neighborhood of a neuron can be computed for each value of T. For neuron c, that influence zone is exactly the ball $V_c^T = \{r \in C / K^T(\delta(c,r)) > \alpha\}$. Figure 7.7 shows that the size of the neighborhood decreases with T: the smaller T, the fewer the neurons that belong to the neighborhood V_c^T. The self-organizing map training algorithms minimize a cost function. When the minimum is reached, one gets a partition that is made of sets that are compact enough, and, in addition, it is possible to define an order that stems from the topology of the map. That cost function will be hereinafter noted as J_{som}^T. It plays the role of the cost function I of the k-means algorithm that was described in the previous section. We will now consider the most popular function J_{som}^T, which is

$$J_{\text{som}}^T(\chi, W) = \sum_{z_i \in A} \sum_{c \in C} K^T(\delta(c, \chi(\boldsymbol{z}_i))) \|\boldsymbol{z}_i - \boldsymbol{w}_c\|^2.$$

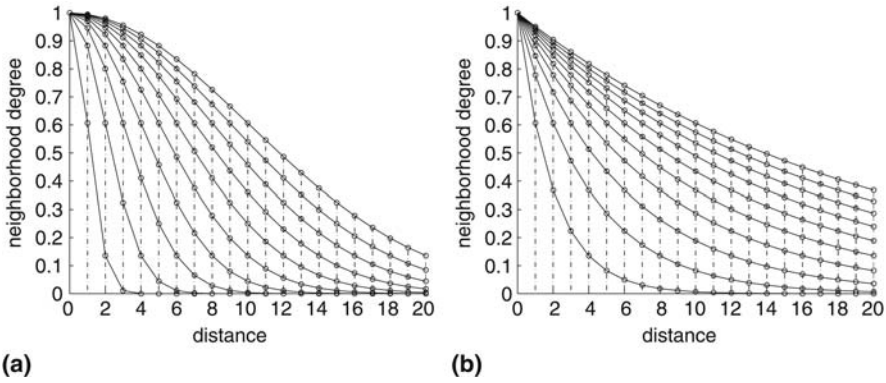

Fig. 7.7. Families of kernel functions that are used to control the neighborhood on the map; x-axis: distance on the map (length of the shortest path between two neurons). The curves show the kernels for different values of T; from top to bottom, T takes on values from 10 to 1 (**a**) $K^T(\delta(c_1, c_2)) = \exp(-0.5\ \delta(c_1, c_2)/T)$ (**b**) $K^T(\delta(c_1, c_2)) = \exp(-0.5\ \delta^2(c_1, c_2)/T^2)$

In that relation, χ is an allocation function, and W is the set of the p reference vectors of the map. $\chi(\boldsymbol{z}_i)$ stands for the neuron of the map C that is associated to the observation $\boldsymbol{z}_i$, and $\delta(c, \chi(\boldsymbol{z}_i))$ is the distance on the map C between a neuron c and the neuron that is allocated to observation $\boldsymbol{z}_i$. As for the k-means algorithm, it is possible to view the links between the map and the data space. Actually, the basic principles of those two algorithms are very similar, as shown on Fig. 7.8. The difference stems from the fact that the set of labels, shown on Fig. 7.1, is replaced by the label *graph* of the map. The cost function J^T_{som} is a mere extension of the k-means cost function $I(W, \chi) = \sum_{\boldsymbol{z}_i \in A} \|\boldsymbol{z}_i - \boldsymbol{w}_{\chi(\boldsymbol{z}_i)}\|^2$, where the Euclidean distance between an observation $\boldsymbol{z}_i$ and its associated reference vector is replaced by a generalized distance, denoted d^T, which takes into account all the neurons of the map

$$d^T\left(\boldsymbol{z}_i, \boldsymbol{w}_{\chi(\boldsymbol{z}_i)}\right) = \sum_{c \in C} K^T\left(\delta\left(c, \chi\left(\boldsymbol{z}_i\right)\right)\right) \|\boldsymbol{z}_i - \boldsymbol{w}_c\|^2.$$

Note that the distance between $\boldsymbol{z}$ and $\boldsymbol{w}_{\chi(\boldsymbol{z})}$, as expressed by the distance function d^T, is a weighted sum of the Euclidean distances between $\boldsymbol{z}$ and all the reference vectors of the neighborhood of the neuron $\chi(\boldsymbol{z})$. Function J^T_{som} is equal to the function $I(W, \chi)$ if parameter T is small enough. In that case, the distance d^T is identical to the Euclidean distance.

The minimization of the cost function $J^T_{\text{som}}(\chi, W)$ is performed in different ways, depending on whether an adaptive or a batch optimization is desired. In addition, a probabilistic formalism leads to a third version, which explicitly estimates probability densities. Those three versions of the topological map training algorithm are presented in the next sections.

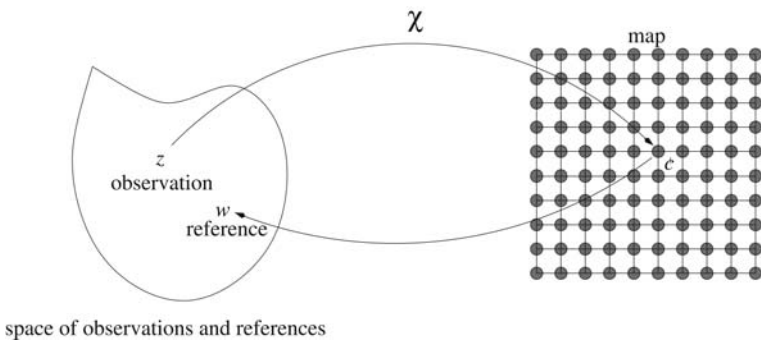

Fig. 7.8. Basic principle of self-organizing map modeling of data space. A label c, which is selected among P neurons of map C, is associated to any observation $\boldsymbol{z}$ of the data set D, using the allocation function $\chi(\chi(\boldsymbol{z}_i = c))$; that label allows the definition of the reference vector $\boldsymbol{w}_c$

7.3.2 The Batch Optimization Algorithm for Topological Maps

In the present section, we describe the minimization of the cost function $J^T_{\text{som}}(\chi, W)$. The only difference between the k-means and the self-organizing map algorithm is the difference between the two cost functions. When T is kept constant, the minimization of J^T_{som} may be written in the dynamic clustering formalism (see the section that is devoted to k-means). Here, just as in the previous section, that formalism provides a proof of convergence of the algorithm to a local minimum of the cost function.

When T is kept fixed, the minimization of J^T_{som} is performed iteratively. Each iteration has two phases. The first phase is an allocation phase and the second phase is a minimization phase where the cost function that is associated to the current partition is minimized:

- *Allocation phase.* $J^T_{\text{som}}(\chi, W)$ is minimized with respect to the allocation function χ. The set W of reference vectors is kept fixed during that phase. The expression of $J^T_{\text{som}}(\chi, W)$ and of $d^T\left(\boldsymbol{z}_i, \boldsymbol{w}_{\chi(\boldsymbol{z}_i)}\right)$ show that the best allocation function is defined for each observation $\boldsymbol{z}$ by

$$\chi^T(\boldsymbol{z}) = \arg\max_{r \in C} \sum_{c \in C} K_T(\delta(c, r)) \left\|\boldsymbol{z} - \boldsymbol{w}_c\right\|^2 = \arg\max_{r \in C} d^T(\boldsymbol{z}, \boldsymbol{w}_r).$$

  That phase allows defining an allocation function χ and the associated partition of data space D. Then the closest reference vector with respect to the weighted distance d^T is allocated to each observation.
- *Minimization phase.* $J^T_{\text{som}}(\chi, W)$ is minimized with respect to the reference vector set W. That minimization is performed while freezing the allocation function χ that was previously computed. Since J^T_{som} is convex with respect to the parameters from W, the minimization can be performed by

computing the value for which the gradient of the cost function is zero. That defines the new reference vector set

$$\boldsymbol{w}_c^T = \frac{\sum_{r\in C} K\left(\delta\left(c,r\right)\right)\boldsymbol{Z}_r}{\sum_{r\in C} K\left(\delta\left(c,r\right)\right)n_r},$$

where $\boldsymbol{Z}_r = \sum_{\boldsymbol{z}_i\in A,\,\chi(\boldsymbol{z}_i)=r} \boldsymbol{z}_i$ is the sum of all observations of the training set A that are allocated to neuron r. Note that each new reference vector is the center of mass of the mean vector of the subsets $P_r \cap A$, each center of mass being weighted by $K(\delta(c,r))n_r$.

To summarize, we get the following algorithm:

Batch Algorithm of Topological Maps: T Fixed

1. *Initialization*: $t = 0$. Select the p reference vectors (randomly, in general), the structure of the map and its size, the maximum number of iterations N_{iter}.
2. *Iteration t*. The reference vector set W^{t-1} is known from the previous step,
   Allocation phase: update the allocation function χ^t that is associated to W^{t-1}. Then each observation $\boldsymbol{z}_i$ is allocated to a reference vector according to

$$\chi^T(\boldsymbol{z}) = \arg\max_{r\in C} \sum_{c\in C} K_T(\delta(c,r))\|\boldsymbol{z}-\boldsymbol{w}_c\|^2 = \arg\max_{r\in C} d^T(\boldsymbol{z},\boldsymbol{w}_r);$$

   Minimization phase: apply relation

$$\boldsymbol{w}_c^T = \sum_{r\in C} K(\delta(c,r))\boldsymbol{Z}_r / \sum_{r\in C} K(\delta(c,r))n_r$$

   to compute the new set W^t of reference vectors.
3. Iterate until the maximum iteration is reached, or until J_{som}^T stabilizes in a local minimum according to a stopping criterion.

As for k-means, a close look at the behavior of self-organizing maps for simple examples gives insight into the implementation problems that may arise. The following numerical experiment illustrates the role of the temperature parameter T in the minimization. The data are the same as on Fig. 7.2 in the section on k-means. As mentioned before, the data are a sample of a uniform mixture of four Gaussian modes with partial pairwise overlap. On Fig. 7.9, the results (topological graph and quantization) are displayed in data space. Kohonen's representations are used. The observations and the reference vectors are shown on the same diagram. The map-induced topology of neighboring neurons is shown as well. Reference vectors that are relative to neighboring neurons on the graph are connected by edges on the picture. At initialization, reference vectors were selected randomly around the center of the observation

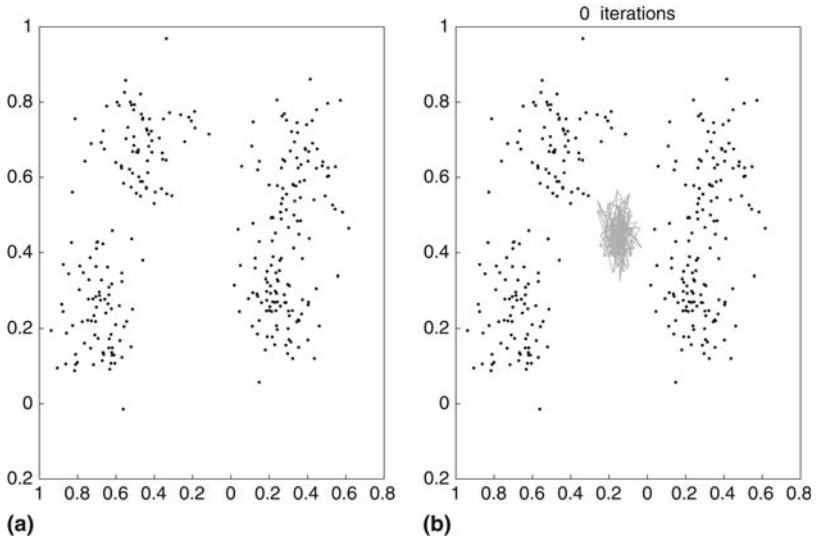

Fig. 7.9. Observation set and initial order generated on the map by random selection of the reference vectors

cloud according to a peaked Gaussian law (its standard deviation is equal to 0.01). Initially, no ordering between the positions of the reference vectors can be observed.

Figure 7.10 shows the maps that are obtained for four distinct values of $T : T = 10$, $T = 5$, $T = 3$ and $T = 1$.

For large values of T, the reference vectors are gathered around the center of mass of the observation cloud. For small values of T, the neighborhood interaction is weaker and the map is unfolded from the same initialization.

The above procedure, for a fixed value of the temperature parameter T, finds a local minimum of the cost function J_{som}^{T} with respect to χ and W. Actually, Kohonen originally suggested to repeat that minimization a number of times, with a monotonous decrease of T. In that approach, the process performs successive steps of Fig. 7.10. The reference vectors are randomly initialized and order appears when T value is still large: the map then unfolds until it covers the whole space of the observation distribution. The performance of the model on completion of training, and the associated partition, depend on the parameters of the minimization algorithm. The most important parameters are:

- the temperature variation interval $[T^{\min}, T^{\max}]$ of the temperature parameter T, i.e. the initial value of $T(T^{\max})$ and its terminal value $(T^{\min})$;
- the number of times N_{iter} the iterative step is repeated;
- the cooling schedule, i.e. how T decreases in time when it spans the temperature interval $[T^{\min}, T^{\max}]$.

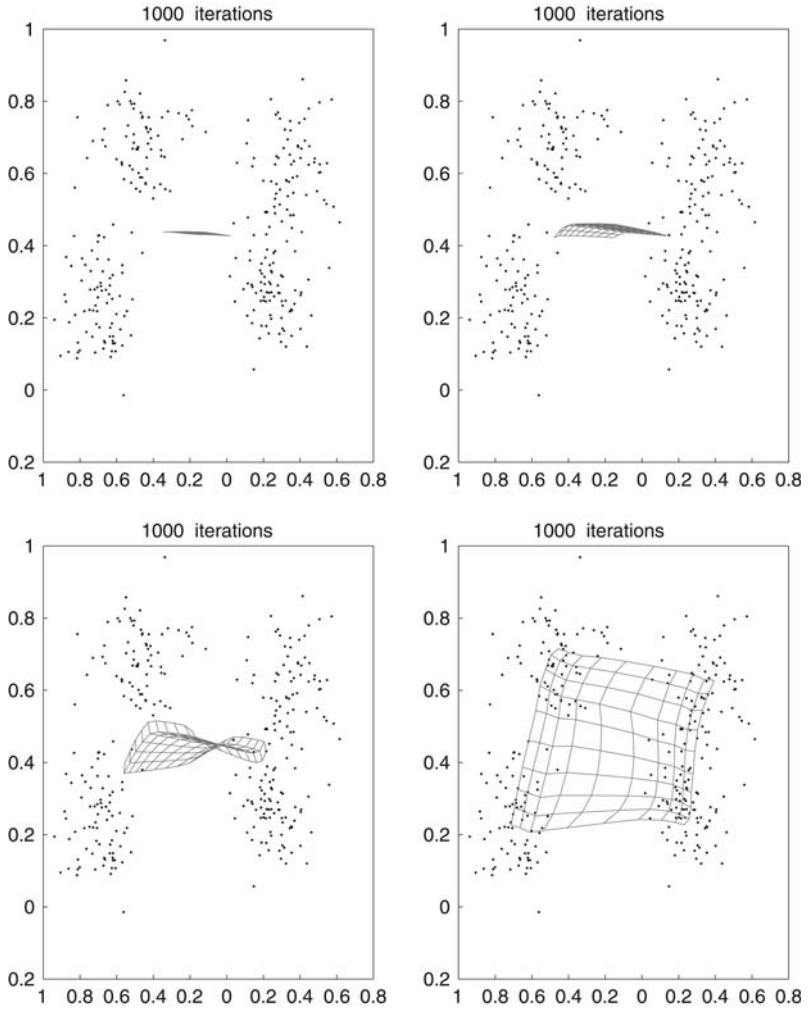

Fig. 7.10. Performing batch algorithm with T fixed (*from top to bottom and from left to right*: $T = 10$, $T = 5$, $T = 3$ and $T = 1$)

Figure 7.11 shows that the cooling schedule may be crucial. On that figure, several final results are displayed for the same training data set, the same temperature variation interval and different cooling schedules. If temperature decreases too fast (quenching), the self-organization is not efficient and the induced neighborhood relation among reference vectors from the data space is not a faithful representation of the graph-induced topology. Using neuronal analogy, one may say that the receptive fields of neurons that are close in cortical maps are not close in the perceptual space. Note that topological self-organization is highly sensitive to the whole parameter set of the algorithm. No

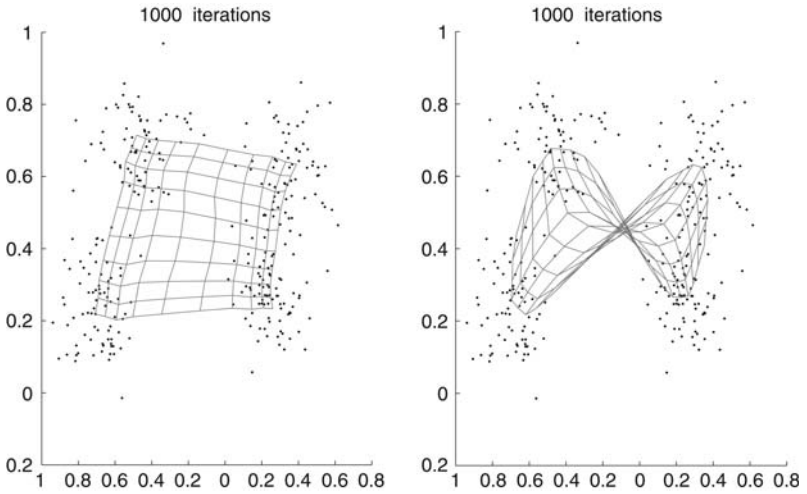

Fig. 7.11. Representation of the order that is induced by the topological map. The maps are obtained for different cooling schedules of T, the same random initialization at the center of the observation cloud and the same variation interval for T

general rule guarantees an appropriate self-organization of the reference vector set. Therefore, it is advisable to test that the algorithm has performed well and that the final set of reference vectors exhibits appropriate self-organization. A relevant real-life example is described in detail below.

For implementation purpose, the batch algorithm of topological maps for a usual cooling schedule can be summarized as follows:

Batch Optimization of Topological Maps (T Decreasing)

1. *Initialization:* perform the dynamic clustering SOM (Self-organizing map) algorithm for $T = T_{\max}$, set $t = 0$
2. *Iteration t.* The reference vector set W^{t-1} is known from previous step. Compute the new temperature value according to the cooling schedule:

$$T = T_{\max} \left(\frac{T_{\min}}{T_{\max}} \right)^{\frac{t}{N_{\text{iter}}-1}}$$

For that temperature T, perform sequentially the two following phases:

- *Allocation:* update the allocation function that is associated to W^{t-1}. A reference vector is allocated to each observation belonging to the training data from relation:

$$\chi^T(\boldsymbol{z}) = \arg\max_{r \in C} \sum_{c \in C} K_T\left(\delta\left(c, r\right)\right) \left\|\boldsymbol{z} - \boldsymbol{w}_c\right\|^2 = \arg\max_{r \in C} d^T\left(\boldsymbol{z}, \boldsymbol{w}_r\right).$$

- *Minimization:* apply relation

$$\boldsymbol{w}_c^T = \sum_{r \in C} K(\delta(c,r))\boldsymbol{Z}_r / \sum_{r \in C} K(\delta(c,r))n_r$$

in order to compute the new reference vector set W^t.

3. Repeat the iterative step until $T = T_{\min}$.

The expression of $\boldsymbol{w}_c^T$ shows that the SOM algorithm takes advantage of the neighborhood function $K^T(\delta)$, which is parameterized by T in order to introduce topological self-organization. At high temperatures, a single observation $\boldsymbol{z}_i$ generates a significant change of many reference vectors. Conversely, at low temperatures, $K^T(\delta(c,r))$ can be neglected if $c \neq r$: an observation influences only the updating of its closest reference vectors. While the algorithm proceeds, the various values of T lead to the gradual localization of the reference vectors. More specifically, relation

$$\boldsymbol{w}_c^T = \sum_{r \in C} K(\delta(c,r))\boldsymbol{Z}_r / \sum_{r \in C} K(\delta(c,r))n_r$$

shows that, for a given temperature, the update of a reference vector $\boldsymbol{w}_c$ depends on the observations of the training set that belong to the subset P_c of the partition, and also of the observations of P_r if r is located in a significant neighborhood of c, $r \in V_G^T = \{r | K^T(\delta(c,r)) \leq \alpha\}$.

The smaller T, the smaller the number of neurons belonging to a neighborhood V_c^T, and the smaller number of observations that have an influence on the updating of a reference vector. For T small enough, V_c^T is restricted to the single neuron c, and J_{som}^T strictly amounts to relation the k-means cost function; in that case, SOM is strictly identical to k-means.

Since Kohonen's original SOM training algorithm includes a cooling schedule within bounds that are defined by the temperature variation interval $[T^{\min}, T^{\max}]$, the convergence to a solution occur in two steps. The first step takes place for large values of T: repeated iterations of the dynamic clustering SOM algorithm (with T fixed) tends to guarantee a topological similarity between the reference vector set and the map. The second step takes place at low temperature T: the algorithm tends to get more similar to k-means until complete similarity when T is very small and when $K(\delta(c,r)) \equiv 0$ for any couple (c, r) of distinct neurons. Thus, the first step may be considered as the initialization step of the k-means, using initial reference vectors that retain the topological structure of the map.

The following experiments show how the maps unfold and finally cover the manifold of the training observations. Figure 7.12 describes the progress of training for two different topologies (1D chain and 2D grid) for the four-gaussian-mixture example. The 1-D chain features 50 neurons. The 2-D grid is a 10×10 square mesh. When the reference vectors are initialized randomly around the center of the training data set, the following behavior is observed:

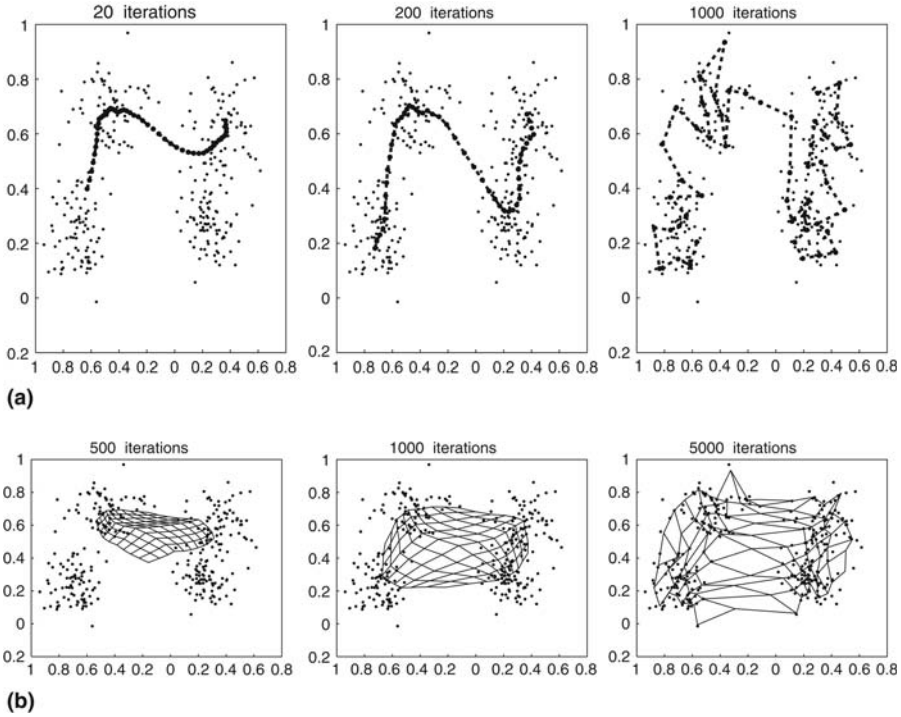

Fig. 7.12. Evolution of the batch training algorithm for the four Gaussian mixture example (pictures a and b) for two different topologies: 1-D with 50 neurons et 2-D with 10×10 neurons. Top pictures display the 1-D map after 20, 200, 2,000 iterations. The same experiment was performed for the 2-D map model; the bottom pictures show the evolution after 50, 500, 5,000 iterations. In the two cases, when convergence is reached, the map covers the whole support of the observation density

- During the first phase, when T is large, the map collapses onto the center of mass, and topological self-organization appears. Then, as T decreases, the map is organized in order to minimize the total inertia of the partition that is associated to the reference vector set. At the end of the algorithm, some reference vectors are positioned at the heart of the observation cloud. Others are trapped in void or low-density regions.
- A close look at the resulting partition provides an interpretation of the hidden structure of observations. Figure 7.13 displays the map. The neurons that have not captured any observation are shown as black dots. Thus, it is possible to separate the data set into two distinct clusters: the algorithm detects natural boundaries.

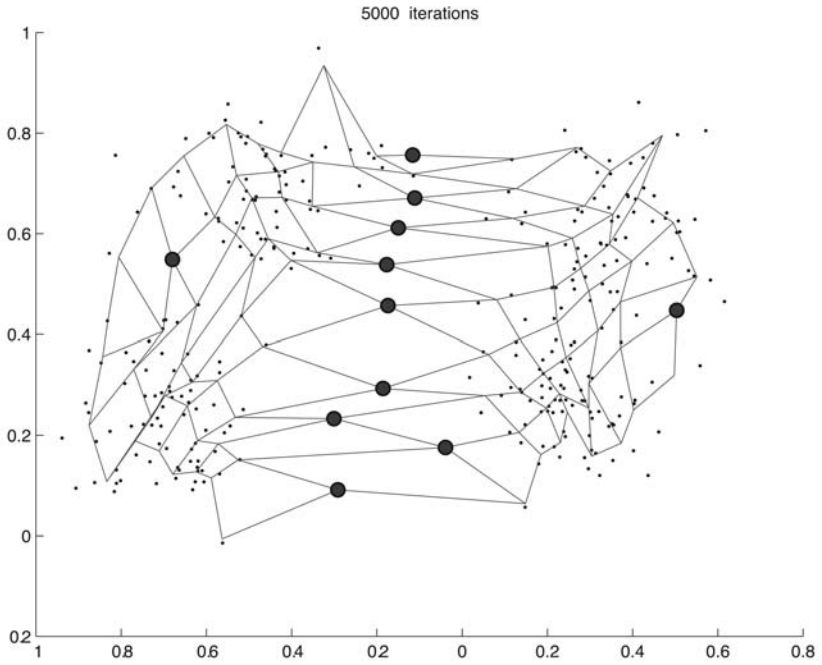

Fig. 7.13. Visualization of natural boundaries of the partition of the data set into two subsets. The neurons without any observation allocation are shown as black dots

7.3.3 Kohonen's Algorithm

The original SOM algorithm, as suggested by T. Kohonen, stems from the dynamic clustering version that was described above. We will now describe its specific features. As for k-means, a stochastic version of SOM is available. It suffices to observe that, during the minimization phase, it is not necessary to terminate the minimization process and compute the global minimum of $J^T_{\text{som}}(\chi, W)$ for a given allocation function χ: one just has to make it decrease. Therefore, relation $\boldsymbol{w}_c^T = \sum_{r\in C} K(\delta(c,r))\boldsymbol{Z}_r / \sum_{r\in C} K(\delta(c,r))n_r$ may be replaced by a simple gradient descent step. Thus, at iteration t and for neuron c, one has

$$\boldsymbol{w}_c^t = \boldsymbol{w}_c^{t-1} - \mu^t \frac{\partial J^T_{\text{som}}}{\partial \boldsymbol{w}_c^{t-1}},$$

where μ^t is the gradient step at iteration t,

$$\frac{\partial J^T_{\text{som}}}{\partial \boldsymbol{w}_c} = 2 \sum_{z_i \in A} K^T \left(\delta\left(c, \chi\left(\boldsymbol{z}_i\right)\right)\right)\left(\boldsymbol{z}_i - \boldsymbol{w}_c\right).$$

That batch algorithm requires that the whole data training set A be available. The contribution of the single observation $\boldsymbol{z}_i$ to parameter $\boldsymbol{w}_c$ to the update is $2K^T(\delta(c, \chi(\boldsymbol{z}_i)))(\boldsymbol{z}_i - \boldsymbol{w}_c^{t-1})$. Alternatively, one may use the stochastic gradient algorithm that computes the reference set once again, at each presentation of an observation $\boldsymbol{z}_i$. That adaptive version is closer to training processes in natural systems. It was the initial version that was suggested by Kohonen. It differs from the batch version that was presented above in two respects: first, the data flow is used instead of the stored data; second the allocation function χ is not the same; Kohonen's algorithm uses the same as in k-means: $\chi(\boldsymbol{z}_i) = \arg\min_c \|\boldsymbol{z}_i - \boldsymbol{w}_c\|^2$.

Therefore, at each presentation of an observation, the new reference vectors are computed for all the neurons of the map C, depending on the selected neuron,

$$\boldsymbol{w}_c^t = \boldsymbol{w}_c^{t-1} - \mu^t K^T \left(\delta\left(c, \chi_t\left(\boldsymbol{z}_i\right)\right)\right)\left(\boldsymbol{w}_c^{t-1} - \boldsymbol{z}_i\right).$$

Thus, Kohonen's algorithm may be summarized as follows:

Kohonen's Algorithm

1. Initialization
   - select the structure and size of the map;
   - choose the initial position of the p reference vectors (usually, this choice is random);
   - choose $T_{\max}, T_{\min}$ and the maximum number of iterations N_{iter};
   - initialize $t = 0$.
2. Iteration t: with the reference vector set W^{t-1}, as computed at the previous iteration:
   - take the current observation $\boldsymbol{z}_i$ (or select randomly an observation from the training set);
   - compute the new value of T according to the cooling schedule:

$$T = T_{\max}\left(\frac{T_{\min}}{T_{\max}}\right)^{\frac{t}{N_{\text{iter}}-1}}$$

   - For that value of T, the following two phases must be performed:
     - *Allocation phase*: W^{t-1} being known, neuron $\chi^t(\boldsymbol{z}_i)$ is assigned to the current observation $\boldsymbol{z}_i$ by the allocation function $\chi(\boldsymbol{z}) = \arg\min_r \|\boldsymbol{z} - \boldsymbol{w}_r\|^2$;
     - *Minimization phase*: the new set of reference set W^t is computed; the reference vectors are updated according to

$$\boldsymbol{w}_c^t = \boldsymbol{w}_c^{t-1} - \mu^t K^T(\delta(c, \chi_t(\boldsymbol{z}_i)))(\boldsymbol{w}_c^{t-1} - \boldsymbol{z}_i),$$

       depending on their distance to the neuron that was selected during the allocation phase.
3. Iterate with decreasing temperature T, until the maximum number of iterations N_{iter} is reached.

7.3.4 Discussion

An in-depth analysis of Kohonen's algorithm unravels its salient features.

- In the update rule for reference vectors, the gradient step μ^t decreases as the number of iterations increases. When the algorithm starts, μ^t is large, and J^T_{som} is not guaranteed to decrease. Later, when the gradient step becomes small enough, the reference vector updates are small for each iteration. In that situation, Kohonen's SOM algorithm behaves in a way similar to the dynamic clustering SOM version.
- If we assume that $K^T(\delta)$ becomes negligible when distance δ exceeds a given threshold d^T, then $K^T(\delta(c,r))$ is significant only for neurons that belong to a given neighborhood of neuron c, whose size is tuned by d^T. That neighborhood will be denoted as $V_c(d^T)$. Thus, when an observation $\boldsymbol{z}_i$ is taken into account, the reference vector $\chi(\boldsymbol{z}_i)$ will be updated, together with the reference vectors of all neurons of the neighborhood $V_{\chi(\boldsymbol{z}_i)}(d^T)$.
- From the point of view of the neuronal representation, the operation of Kohonen's maps can be understood by taking into account the lateral connections between neurons of the map: each neuron c is connected to neighboring neurons r, and any modification of the reference vector $\boldsymbol{w_c}$ generates updates for all reference vectors that are associated to neurons belonging to $V_c(d^T)$ with intensity $K^T(\delta(c,r))$, which decreases with increasing distance $\delta(c,r)$.
- If $K^T(\delta)$ is chosen as a threshold function (see Fig. 7.6), it is constant on the interval $[-d^T, d^T]$ and equal to zero elsewhere, the difference between Kohonen's SOM and k-means is clear. The weight update is the same for the two algorithms; however, in Kohonen SOM, not only is the closest reference vector r updated: the reference vectors associated to neurons of the neighborhood $V_c(d^T)$ are updated as well. Thus, topological self-organization arises: neurons that are close on the map represent observations that are close in data space.
- When temperature T is small, updates according to relation $\boldsymbol{w}_c^t = \boldsymbol{w}_c^{t-1} - \mu^t K^T(\delta(c, \chi_t(\boldsymbol{z}_i)))(\boldsymbol{w}_c^{t-1} - \boldsymbol{z}_i)$ are performed for a subset of all neurons, and, when $d^T < 1$, Kohonen's SOM algorithm is identical to stochastic k-means. Actually, in that case, the only neuron to be updated is the winner of the competition selected by the allocation function χ.

The fact that self-organizing maps are considered as belonging to the family of neural methods stems from the fact that the neural interpretation allows a crisp understanding of the training process. In the following section, we elaborate on that point.

7.3.5 Neural Architecture and Topological Maps

The training algorithms that were described in the previous section allow the determination of the reference vector set $W = \{\boldsymbol{w}_c; c \in C\}$ of a self-organizing

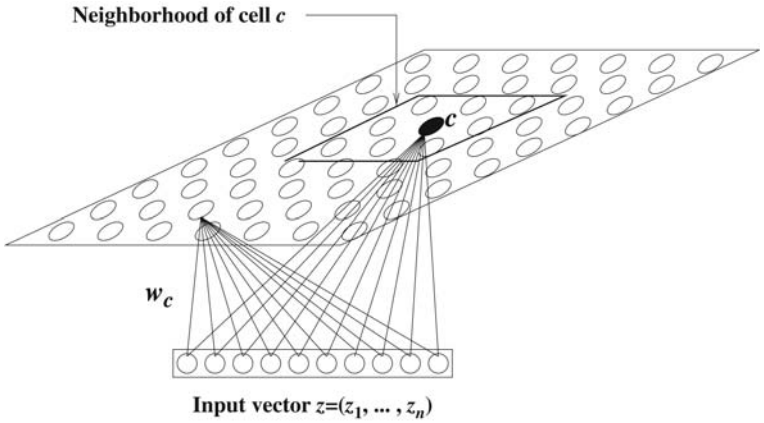

Fig. 7.14. 2-D topological map. The network has two layers: an input layer contains the observations, and a representation layer, for which a topology must be defined (distance δ between neurons and neighborhood function). Each neuron c stands for a reference vector $\boldsymbol{w}_c$; it is fully connected to the input layer. The connection weight vector of each neuron c is the reference vector $\boldsymbol{w}_c$ associated to neuron c

map. A 2-layer neural network provides a joint representation of the map and of the reference vectors (see Fig. 7.14):

- Observations are present in the input layer of the network. The state of each unit is a component of one observation. Therefore, the number of neurons of that layer is equal to the dimension of input space.
- The second layer is the neuronal map. The structure of the map may be decided a priori. In more flexible versions, the structure can evolve during training. The neurons simply compute a distance. Each neuron c is connected to all input units. The reference vector that is associated to the current neuron c of the second layer is actually the vector of connection weights afferent to neuron c. Each neuron has n afferent connections since it is connected to all units of the input layer. When an observation $\boldsymbol{z}$ is presented to the input layer, the output of neuron c of the map is $\|\boldsymbol{z} - \boldsymbol{w}_c\|^2$.

During training, the network connection weights change using various updating rules. Thus, the neurons of the map compute their distances to the current observation in parallel. The main feature of the self-organization process is to focus the adaptation process on the most active area of the map. Kohonen's original algorithm, which is the simplest one, considers that the active zone is the neighborhood of the neuron c that is closest to the observation under consideration, i.e. whose output $\|\boldsymbol{z} - \boldsymbol{w}_c\|^2$ is smallest. That neighborhood generates topological constraints that lead to self-organization.

As indicated in the previous section, it models in a simple way, the lateral coupling between an active neuron and its neighbors on the connection graph

of the cortical map. Thus, at the end of training, neuronal connection weights have converged in order to guarantee that a neuron has discriminative abilities, i.e. that it is active only for a subset of the observations of the training set. A neuron c, which is represented by the reference vector $\boldsymbol{w}_c$, may be considered as an average observation that is a compressed representation of the data set P_c of the observations that it has been assigned. Thus, the whole neuronal map performs a vector quantization of the whole data set D, which is obtained by the analysis of the training set A. The quality of the quantization (faithful or not) strongly depends on whether the training set is representative or not.

7.3.6 Architecture and Adaptive Topological Maps

Self-Organizing Maps produce simple representations of data that are embedded in spaces of very large dimension. That representation is performed in a low-dimension discrete set C with a graph structure. The problem of the choice of architecture consists in selecting a suitable graph structure for the map, i.e. a structure that is appropriate for the specific problem of interest. Therefore, one must define a measure of the adequacy of a map to the problem of interest. The data set D and the map C are related in two ways: the embedding of C into D that maps each neuron c of C onto a reference vector $\boldsymbol{w}_c$ of C, and the allocation function χ of D into C, which associates to each observation vector in D a neuron c of the map. Those two mappings have to be topologically consistent in the following sense:

- Two neurons that are neighbors in the map C must be represented by two reference vectors that are close in D.
- Reciprocally, data that are approximately similar must be allocated by χ to the same neuron or to neighboring neurons.

If the dimension of the map does not fit with the underlying dimension of the data cloud (dimension of the manifold that is generated by the observations), two observations that are close in data space D may be allocated to distant neurons in the map. Yet, the topological consistency is an interesting property because it allows reducing the dimension of the data while retaining similarities. In previous sections, it was assumed that the graph structure of the map was given a priori. That choice was not data-driven, which has shortcomings: it does not guarantee the adequacy between the structure of the map and the internal structure of the data distribution.

Usually, in applications, the dimension of the data space may be very large if the number of features describing the data is large, but the observations are not distributed uniformly in the data set. They are located in specific regions with various concentrations. Reference vectors must be located in high-density regions, and one must avoid wasting reference vectors by locating them in void regions. The choice of the graph structure of the map is very important because, when it is appropriate, it guarantees the topological consistency of the map and a good representation of the underlying data probability distribution.

In order to solve that problem, one may consider an oversize map (with respect to the underlying dimension of the problem) and apply Kohonen SOM algorithm to it. After training, neurons that capture no observation of the training set are discarded. Kohonen's algorithm is performed on the resulting new structure. That *pruning* process is iterated as long as necessary.

Another technique consists in defining the map (size and dimension) during training while updating the reference vectors. Thus, the map is built incrementally, allowing the addition of some neurons and the deletion of others. Several methods have been proposed in the literature. They can be classified into two categories:

- In the first category, the dimension k of the map is decided a priori, and the map is built adaptively by addition and deletion of neurons. In order to perform those operations in a systematic way, simple graphical structures such as segments for $k = 1$, triangles for $k = 2$, tetrahedrons for $k = 3$ and so on, are processed. [Oja et al. 1999].
- A second category allows the data to drive the selection of the dimension of the map, which may vary from one region to another. The neural gas algorithm [Oja et al. 1999; Fritzke 1995] is building the graph by introducing the connection links in the data space itself. In that method, whenever an observation is presented, the two closest reference vectors are selected; if they are connected, the connection is activated, otherwise the corresponding connection link is created. The connection links that are inactive during a fixed number of iterations are deleted.

7.3.7 Interpretation of Topological Self-Organization

The structure of the cost function J_{som}^T gives insight into the topological self-organization during training. The subsets $P_r \cap A$ generate a partition of the training set A, so that J_{som}^T can be written as follows:

$$\begin{aligned} J_{\text{som}}^T &= \sum_r \sum_{z_i \in P_r \bigcap A} \sum_c K^T\left(\delta\left(c,r\right)\right) \left\|\boldsymbol{z}_i - \boldsymbol{w}_c\right\|^2 \\ &= \sum_c \sum_r \sum_{z_i \in P_r} K^T\left(\delta\left(c,r\right)\right) \left\|\boldsymbol{z}_i - \boldsymbol{w}_c\right\|^2. \end{aligned}$$

Decomposing that relation shows that the cost function J_{som}^T generates a vector quantization and guarantees topological consistency

$$\begin{aligned} J_{\text{som}}^T = &\left[\sum_c \sum_{r \neq c} \sum_{z_i \in P_r} K^T\left(\delta\left(c,r\right)\right) \left\|\boldsymbol{z}_i - \boldsymbol{w}_c\right\|^2\right] \\ &+ K^T\left(\delta\left(c,c\right)\right) \sum_c \sum_{z_i \in P_r} \left\|\boldsymbol{z}_i - \boldsymbol{w}_c\right\|^2 \end{aligned}$$

$$= \frac{1}{2} \sum_{c} \sum_{r \neq c} K^T \left(\delta \left(c, r \right) \right) \left[\sum_{z_i \in P_r} \| \boldsymbol{z}_i - \boldsymbol{w}_c \|^2 + \sum_{z_i \in P_c} \| \boldsymbol{z}_i - \boldsymbol{w}_r \|^2 \right]$$
$$+ K^T \left(\delta \left(c, c \right) \right) \left[\sum_{c} \sum_{z_i \in P_c} \| \boldsymbol{z}_i - \boldsymbol{w}_c \|^2 \right].$$

This decomposition gives two terms, the sum of which must be minimized:

- The second term is I of k-means, weighted by $K^T(\delta(c,c)) = K(0)$. Its influence is controlled by the temperature parameter T: the smaller the temperature, the more influential that term during minimization. It tends to build a partition into compact subsets, and the reference vectors tend to be the centers of mass of the partition subsets.
- The first term enforces the topological consistency constraint: if two neurons r and c are close on the map, $K^T(\delta(c,r))$ is large, because $\delta(c,r)$ is small. Minimizing that term decreases the distance between the subsets P_c and P_r that are allocated to c and r. Thus, proximity on the map enforces proximity in the data set.

The above form of J^T_{som} also gives insight into the presentation of the algorithm as consisting in two different steps that depend on the temperature T (see above the section on batch optimization algorithm of topological maps). The first step occurs when T is large: the first term is dominant, and the task of the algorithm is mainly to guarantee the topological consistency of the map. The second step occurs at lower temperature. In that case, the second term becomes dominant and the algorithm essentially minimizes the inertia of the partition. The temperature allows performing the appropriate tradeoff between the two terms of J^T_{som}. Since the topological self-organization occurs during the first part of training, then the minimization is useful to obtain subsets that are as compact as possible. It is the k-means phase of the algorithm that consists in approximating locally the data distribution. Thus, the algorithm may be cursorily described as a version of the k-means algorithm subject to the constraint of topological consistency of the reference vectors with the map.

The following experiment gives insight into the difference between SOM and k-means. We consider again the example that was displayed on Fig. 7.2[d] to illustrate k-means. In that case, a topological map with a 1D chain structure is used with four neurons, and the parameters of the map are estimated from the training set, generated from a mixture of four Gaussians.

The four reference vectors were initialized at the bottom right of the figure just as for the previous k-means experiment. The two solutions that are obtained by k-means and SOM are shown on Fig. 7.15. The map topology constraint allows locating the four neurons at the centers of the four Gaussian modes. Thus, the SOM algorithm was able to determine the solution of the k-means problem under the topological consistency constraint (Fig. 7.15 [b]);

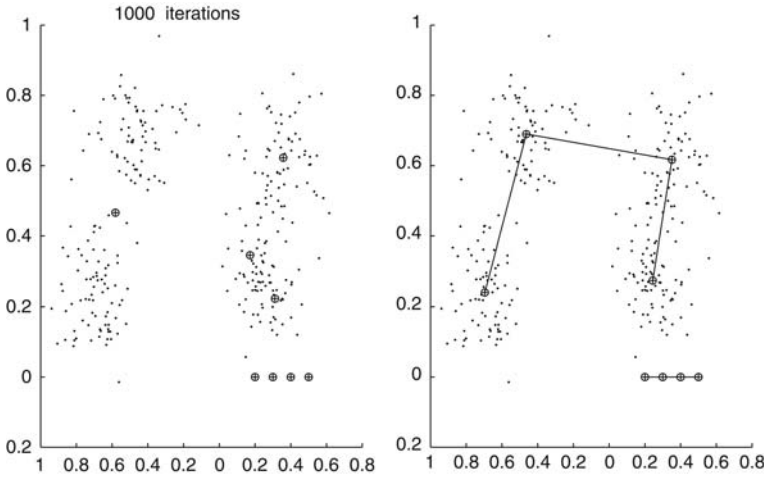

Fig. 7.15. Comparison k-means (**a**) and SOM (**b**) for the same initialization. The reference vectors are initialized in the right bottom of the picture

that solution is different from the solution that was found by the straightforward implementation of k-means (Fig. 7.15 [a]). To summarize, the map provided a better representation of the training set.

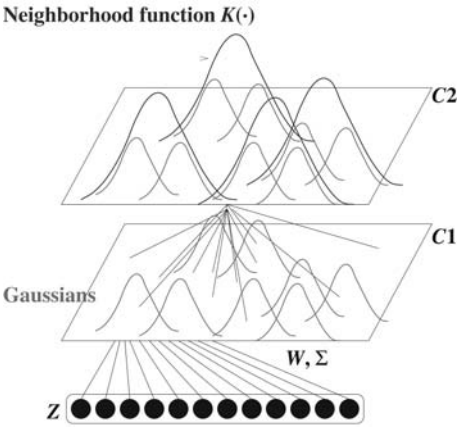

Fig. 7.16. Modeling of the SOM according to a probability density mixture. The map is shown in the neural formalism: 3 layer architecture: an input layer and two layers that are maps with similar size and similar topology. A neuron of C_1 represents a gaussian with expectation vector $\boldsymbol{w}_c$ and scalar covariance matrix $\sigma_c I$; a neuron of C_2 represents a Gaussian mixture, whose density is given by $p(\boldsymbol{z}) = \sum_{c_2} p(c_2) p_{c_2}(\boldsymbol{z})$ where $p_{c_2}(\boldsymbol{z}) = \sum_{c_1} p(c_1 \mid c_2) p(\boldsymbol{z} \mid c_1)$

7.3.8 Probabilistic Topological Map

Similarly to k-means algorithm, a probabilistic version of SOM, called PRSOM, can be defined [Anouar et al. 1997; Gaul et al. 2000]. The difference between SOM and PRSOM is essentially that, for PRSOM, a Gaussian density f_c is associated to each neuron c of the map. Each Gaussian density function f_c is completely defined by the mean vector (the equivalent of reference vector of SOM) $\boldsymbol{w}_c = (w_c^1, w_c^2, \ldots, w_c^n)$, and by its covariance matrix that is a square symmetric positive-definite matrix Σ_c, restricted to isotropic densities: $\Sigma_c = \sigma_c^2 I$, where I is the (n, n) unit matrix. Then the density functions can be written as

$$f_c(\boldsymbol{z}) = \frac{1}{(2\pi)^{n/2}\,\sigma_c^n} \exp\left(-\frac{\|\boldsymbol{z} - \boldsymbol{w}_c\|^2}{2\sigma_c^2}\right).$$

Thus, in the PRSOM, each neuron c of the map is allocated to the mean vector $\boldsymbol{w}_c$ and to the positive scalar σ_c. As for SOM, the data space D is partitioned into subsets of the family $\{P_c/c \in C\}$. The subset P_c is described by the density function f_c : $\boldsymbol{w}_c$ represents its associated reference vector, and σ_c estimates the standard deviation of the observation of $P_c \cap A$ around $\boldsymbol{w}_c$. The two parameter sets $W = \{\boldsymbol{w}_c; c \in C\}$ and $\sigma = \{\sigma_c; c \in C\}$ define completely the PRSOM. Their values must be estimated during training from the training set A.

If we assume that the data underlying distribution is a Gaussian mixture, the PRSOM allows an estimating of the parameters of the mixture. A neural interpretation of PRSOM can be given: the architecture that is associated to the PRSOM has three layers architecture (Fig. 7.16):

- Data is presented to the input layer.
- The map C is duplicated into two similar maps C_1 and C_2 that have the same topology as the map C in the SOM model. The generic neuron of maps C_1 (resp. C_2) will be denoted c_1 (resp. c_2).

That approach was first described by Luttrel [Luttrel 1994]; it assumes that a random propagation occurs forward and backward through the 3 layers of the network. In the backward direction, from the map to the data space, that propagation is described by the conditional probabilities $p(c_1|c_2)$ and $p(\boldsymbol{z}|c_1, c_2)$. Moreover, the Markov assumption is postulated, namely that $p(\boldsymbol{z}|c_1, c_2) = p(\boldsymbol{z}|c_1)$. Then the probability of each observation $\boldsymbol{z}$ can be computed explicitly as

$$p(\boldsymbol{z}) = \sum_{c_2} p(c_2) p_{c_2}, (\boldsymbol{z})$$

with

$$p_{c_2}(\boldsymbol{z}) = \sum_{c_1} p(c_1 \mid c_2) p(\boldsymbol{z} \mid c_1).$$

The probability density is fully determined by the network architecture, which provides an expression of the conditional probability $p(c_1|c_2)$ using the neighborhood relation on the map and the conditional density of the observation $p(\boldsymbol{z}|c_1) = f_{c_1}(\boldsymbol{z}, W_{c_1}, \sigma_{c_1})$. If we assume that the neighborhood relationships permit the definition

$$p(c_1|c_2) = \frac{1}{T_{c_2}} K^T(\delta(c_1, c_2)), \quad \text{with } T_{c_2} = \sum_r K^T(\delta(c_2, r));$$

then the posterior probability densities of the observations may be expressed as a function of the Gaussian distributions of the neurons:

$$p_{c_2}(\boldsymbol{z}) = \frac{1}{T_{c_2}} \sum_{r \in C_1} K^T(\delta(c_2, r)) f_r(\boldsymbol{z}, \boldsymbol{w}_r, \sigma_r).$$

Thus, $p_{c_2}(\boldsymbol{z})$ can be interpreted as a local mixture of Gaussian densities that are associated to each neuron of the map. The set of average vectors $W = \{\boldsymbol{w}_c; c \in C\}$ and the set of scalar standard deviations $\sigma = \{\sigma_c; c \in C\}$ are the parameters to be estimated by training. The probabilistic formalism makes it possible now to maximize the likelihood of the observation set just as for the probabilistic version of k-means. If the observations of the training set A are assumed to be the independent, and that each observation $\boldsymbol{z}_i$ is generated by the Gaussian mode $p_{\chi(\boldsymbol{z}_i)}$ that is associated to neuron $\chi(\boldsymbol{z}_i)$, and if it is further assumed that neurons c_2 of C_2 have similar prior probabilities, the classifying likelihood can be written as

$$p(\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N \mid W, \sigma, \chi) = \prod_{i=1}^{N} p_{\chi(\boldsymbol{z}_i)}(\boldsymbol{z}_i),$$

which must be maximized with respect to the parameters of the model W, σ and the allocation function χ. According to the usual strategy, it is performed by a minimization process

$$E(W, \sigma, \chi) = -\sum_{i=1}^{N} \ln \sum_{r \in C} K^T(\delta(\chi(\boldsymbol{z}_i), r)) f_r(\boldsymbol{z}_i, \boldsymbol{w}_r, \sigma_r)$$

by using the dynamic clustering formalism. The phases of allocation and minimization are sequentially and alternatively iterated until convergence:

- *Allocation phase.* Assume that the parameters $\{W, \sigma\}$ have the values computed at the previous iteration or at initialization. Then E must be minimized with respect to the allocation function χ. A new allocation function must be found that assigns each observation $\boldsymbol{z}$ to a neuron. That step generates a new partition of the training data space D. It can easily be seen that the optimal allocation function associates to a given observation $\boldsymbol{z}_i$ the most probable neuron c according to the density p_{c_2}:

$$\chi(\boldsymbol{z}) = \arg\max_{c_2} p_{c_2}(\boldsymbol{z})$$

- Minimization phase. During that phase, the allocation function is kept constant, and $E(W, \sigma, \chi)$ is minimized with respect to W and σ.

The parameters W and σ are updated as in the batch version of the SOM algorithm by canceling the partial derivatives of the cost function $E(W^t, \sigma^t, \chi^t)$. To solve the equation, an iterative procedure is used as in [Duda et al. 1973], assuming that for ith iteration the initial values of the parameters are close to the optimal values. The update relations are the following:

$$\boldsymbol{w}_r^t = \frac{\sum_{i=1}^{N} \boldsymbol{z}_i K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}{\sum_{i=1}^{N} K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}$$

$$\left(\sigma_r^t\right)^2 = \frac{\sum_{i=1}^{N} \left\|\boldsymbol{w}_r^{t-1} - \boldsymbol{z}_i\right\|^2 K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}{n \sum_{i=1}^{N} K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}.$$

In both above relations, the parameters at iteration t are expressed as functions of the parameters at iteration $t-1$.

Since the model is complex, an appropriate initialization is desirable. Since PRSOM can be considered as extensions of SOM, one can first perform a SOM estimation of the reference vector set W in order to initialize the mean vector set of PRSOM.

Thus, the PRSOM training algorithm can be summarized as follows:

PRSOM Algorithm with Constant Temperature T

1. *Initialization*: $t = 0$. The initial values W^0 of the references are computed using a SOM training algorithm, the σ_r^0 is computed by the mean of the local inertia I_r (Sect. 7.2.1). The initial allocation function χ^0 is derived from the update relation

$$\boldsymbol{w}_r^t = \frac{\sum_{i=1}^{N} \boldsymbol{z}_i K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}{\sum_{i=1}^{N} K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}},$$

$$\left(\sigma_r^t\right)^2 = \frac{\sum_{i=1}^{N} \left\|\boldsymbol{w}_r^{t-1} - \boldsymbol{z}_i\right\|^2 K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}{n \sum_{i=1}^{N} K\left(\delta\left(r, \chi^{t-1}\left(\boldsymbol{z}_i\right)\right)\right) \frac{f_r\left(\boldsymbol{z}_i, \boldsymbol{w}_r^{t-1}, \sigma_r^{t-1}\right)}{P_{\chi^{t-1}(\boldsymbol{z}_i)}\left(\boldsymbol{z}_i\right)}}.$$

The maximal number of iterations N_{iter} is chosen.

2. *Iteration* t : W^{t-1} and σ^{t-1} were computed at the previous iteration.
   - *Minimization phase*: computation of the new parameters W^t and σ^t;
   - *Allocation phase*: update of the allocation function χ^t that is associated to W^t and σ^t from relation $\chi(\boldsymbol{z}) = \arg\max_{c2} p_{c2}(\boldsymbol{z})$.
2. Iterate until $t > N_{\text{iter}}$ or until stabilization of the cost function $E(W, \sigma, \chi)$.

As for SOM training, PRSOM uses a neighborhood whose size is controlled by the temperature parameter T. During training, the size of the neighborhood decreases according to the cooling schedule. At the end of training, the map provides an organized structure of the average vector set, and the partition associated to the map is defined by the final allocation function $\chi^{N_{\text{iter}}}$. As for other versions of SOM, the data space D is divided into M subsets: each neuron c of the map represents a data subset $P_c = \{\boldsymbol{z} / \chi^{N_{\text{iter}}}_{(\boldsymbol{z})} = c\}$. That map and that partition were determined from probability distributions, whereas SOM just uses Euclidean distances. The probability density estimation gives access to additional information that may be useful for application purposes. Actually, that information is crucial as far as classification problems are concerned. No stochastic version of PRSOM is available: a large sample of the data is necessary to estimate the initial variance before updating the parameters.

PRSOM provides a lot of additional information about the training data (tracking outliers, computing probabilities, etc.). However, that model can be used only if the training observation set is large enough to allow an accurate estimation of the standard deviations of the Gaussian modes of the mixture in the initialization phase. Remote sensing, where a tremendous amount of data is available, is ideally suited to applications of SOM. The detection of ocean color is described in the next section.

7.4 Classification and Topological Maps

Among the various applications of SOM, many of them are classification tasks. As stated above, classification is not a straightforward application of self-organization: unsupervised learning provides an allocation function that assigns any observation to a cluster of a partition of the training set, irrespective of the semantics of the data. In such problems, it is assumed that a lot of noise-corrupted observations are available with not knowledge of their class. The partition that is obtained depends on the probability density underlying the training set. Regions that contain a high density of data will be covered by a fine partition; low-density regions will be covered by a coarse partition. The large amount of data available in high-density regions provides accurate information on those regions. On the other hand, the geometry of the partition depends on the nature of the encoding of the observations. Thus, for a given problem and a given data set, several different encodings may generate several partitions of the data space. With the SOM algorithm, the selection

of the code provides information about the problem of interest. The basic principle of the algorithm is to favor the emergence of clusters (the partition subsets) that are appropriate for the application under consideration. If the application involves a classification task into S classes, each partition subset must be included in one class as completely as possible. Then, one can assign one of the S classes to a whole cluster. Since each subset is assigned to one neuron of the map, the classification problem amounts to labeling each neuron of the map. The label set is the set of the S classes of the problem. Labeling can be performed in two different ways. Since each reference vector represents a subset of the partition P, and since the reference vector may be interpreted as an average experiment, it is possible to use expert knowledge to recognize the class of the reference vector on the basis of its characteristics:

1. by asking an expert of the domain to classify some data extracted from the training set,
2. by first aggregating the neurons on a statistical basis and then use the expert knowledge to label the clusters.

7.4.1 Labeling the Map Using Expert Data

Assume that a S-class classification task must be performed, and that the labels of those classes must belong to a label set $L = \{l_i, i = 1, \ldots, S\}$. At the end of SOM training, when all parameters of the map are estimated, each observation $\boldsymbol{z}$ is assigned to a neuron $c = \chi(\boldsymbol{z})$, so that the label l_c of that neuron can be assigned to the observation. Therefore, the problem is: how to label the neurons of the map with the labels of L?

Labeling the neurons of the map is the first step in the design of a classifier from a SOM. If the amount of data classified by the expert is very large, labeling may be performed by majority voting (see hereafter Fig. 7.17):

- Assign the expert-classified data to the various neurons of the map using the allocation function provided by the SOM training.
- For every neuron c, select the label l_i that is the most commonly used label for the expert classified-data assigned to neuron c.
- All the data belonging to the subset that is represented by neuron c are now labeled by label l_i.

At the end of the labeling phase, the set of neurons c that have the same label l can be used to approximate the probability distribution of the data of class l. The larger the amount of expert-classified data, the better the classifier. Of course, neurons that represent data lying on the boundaries of the classes may get the wrong label. Another source of error is the lack of expert-classified data in some subset represented by a given neuron: the corresponding region of the data space is thus poorly identified.

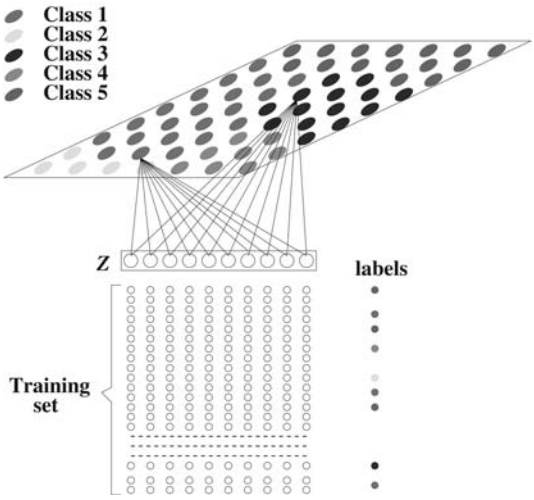

Fig. 7.17. Map labeling using expert-classified data. Classified data are assigned to the relevant neurons of the map. Then each neuron is labeled using majority voting among classified data that are allocated to that neuron

7.4.2 Searching a Partition that Is Appropriate to the Classes

If the amount of expert-labeled data is too small, the above labeling method is inappropriate. Majority voting result has a large variance and may generate classification error with significant probability. The presence of a single wrongly labeled observation may lead to assigning a wrong label to the associated neuron. Thus, a whole region of the data space may be wrongly classified. Furthermore, due to the small amount of labeled data, a significant number of subsets of the partition may include no labeled data, so that the algorithm is not able to provide them with any label.

In that case, it is possible to take into account an additional phase, in which the various observation subsets are clustered as appropriately as possible. A coarser partition is sought, and labeling will be performed after that additional clustering phase. When one fuses several partition subsets, more expert-classified data are available to label a larger subset. Of course, as before, the whole process is valid only if the original clustering is consistent with the classification, so that majority voting can select the right label.

If the map and the partition that are provided by SOM are assumed to be relevant, then the two following additional assumptions are taken into consideration:

- The data quantization is correct, so that each reference vector is a good representative of its allocated data.
- Topology is relevant: two subsets that are represented by neighboring neurons on the map contain observations that are close in data space.

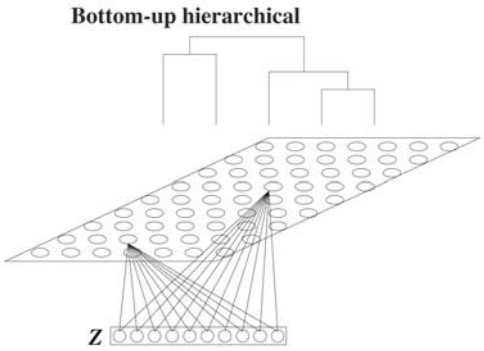

Fig. 7.18. Looking for a partition that is appropriate to the desired classes. The method consists in clustering the neurons of the map by bottom-up hierarchical classification, and testing the obtained partition with respect to the expert-classified data

Underlying the second assumption is the hypothesis that there exists a structure on the dataset that fits the classification problem, and that it is possible to exhibit that structure with the topological self-organization of the map. Thus, two subsets that are represented by neighboring neurons have a strong probability of representing observations that belong to the same class.

Of course, those assumptions are very strong. It is implicitly supposed that a right data encoding is already known to perform the classification. Therefore, that point must be solved in a preliminary analysis, providing an appropriate data representation, stemming from an appropriate variable selection and the design of a relevant coding. The effect of the coding process on the classification result will be shown in the section devoted to applications.

Bottom-up hierarchical classification [Jain et al. 1988] may perform the second stage of the process by appropriately clustering the neurons (see Figs. 7.18 and 7.19).

This method computes a partition hierarchy. The various partitions of the hierarchy are found iteratively. The initial partition is the finest one. It is made of all the singletons of the map. From that initial partition, two subsets of the current partition are clustered at each iteration. To select the two subsets that are going to be clustered, a measure of the similarity between two subsets is defined. Among all the possible subset pair, the pair that is made of the most similar subsets, with respect to the chosen similarity criterion, is selected.

Summary of the hierarchical classification algorithm:

Hierarchical Classification Algorithm

3. *Initialization.* Consider the finest partition that is made by all the singletons; each neuron is allocated to a distinct subset. Choose the desired number of subsets K.

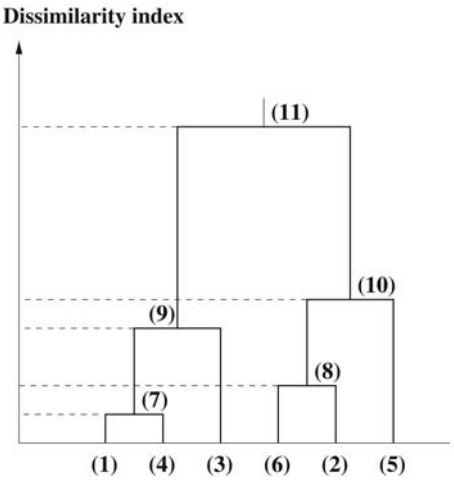

Fig. 7.19. Clustering the neurons of the map using bottom-up hierarchical classification: the leaves of the tree represent the neurons (here there are six neurons); for each cluster, the vertical axis provides the clustering index for the selected similarity

4. For a given partition, find the closest subsets in the sense of the selected similarity criterion, and cluster them together in order to get a single subset.
5. If the number of clusters of the current partition is larger than K, go to step (2), otherwise terminate the algorithm.

Several similarity measures are proposed in the literature [Jain et al. 1988]. The most popular one is due to Ward. It consists in aggregating clusters in such a way that the sum of the cluster inertia is as small as possible. That is a way of favoring clusters that are as compact as possible in the (Euclidean) data space. If that criterion is selected to cluster the neurons of the map, the working space is the data space and the associate reference vectors are the neurons. Conversely, since the neurons are distributed on the map with its awn discrete graph topology, one may choose to favor aggregation in a way that takes into consideration that discrete structure. Those clusters will be made of neurons that tend to form connected sets on the map [Murtagh 1985; Yacoub et al. 2001]. The choice one of those two strategies, or the use of a hybrid strategy that combines both of them, may have a crucial influence on the results.

Hierarchical classification allows generating an arbitrary number of subsets, since clustering may be stopped at any iteration. For a given similarity measure, the number of partition elements depends on the number S of classes that are sought. That number also depends on the agreement between the unsupervised statistical partitioning and the partition into S classes that is determined by the application. That number may be higher than S if a class

is not homogeneous from a statistical point of view. Then it is inferred that the expert has assigned to the same class instances of the observation space that are quite different. The analysis of the most consistent partition, obtained by hierarchical classification, allows analyzing the homogeneity of the classification performed by the expert. Therefore, a refinement of the expert classification into S' classes, with $S' > S$, may be designed.

7.4.3 Labeling and Classification

After labeling of the map, the probabilistic version of self-organizing maps (PRSOM) can perform a probabilistic classification. As mentioned above, a normal law is associated to each neuron. An observation $\boldsymbol{z}$ is assigned to a neuron according to the probability $p(c|\boldsymbol{z})$, which is defined by Bayes relation as shown below. A probabilistic assignment is thus obtained. Since the map is labeled according to one of the above procedures, the *posterior* probability that the considered observation $\boldsymbol{z}$ belongs to class l can be estimated. The PRSOM stem from a probabilistic modeling where it is assumed that the observations are generated according to the mixture distribution:

$$p(\boldsymbol{z}) = \sum_c p\,(c)\, p_c\,(\boldsymbol{z})$$

where $p_c(\boldsymbol{z})$ is also a normal law local mixture

$$p_c(\boldsymbol{z}) = \frac{1}{T_c} \sum_c K^T\,(\delta\,(c, r))\, f_r\,(\boldsymbol{z}, \boldsymbol{w}, \sigma_r)$$

where $T_c = \sum_c K^T(\delta(c, r))$ and f is a normal law with mean $\boldsymbol{w}_r$ and scalar covariance matrix $\sigma_r^2 I$. The quantities $p_c(\boldsymbol{z})$ are computed from the neurons of the map and the quantities $p(c)$ are computed from the partition that has been proposed by PRSOM. If N stands for the observation number of the training set A and n_c is the number of observations that are assigned to neuron c by the allocation rule $\chi(\boldsymbol{z}) = \arg\max_c p(\boldsymbol{z} \mid c)$, the prior probability $p(c)$ of neuron c can be estimated as n_c/N. Then Bayes rule allows computing the posterior probability of neuron c given observation $\boldsymbol{z}$:

$$p\,(c\,|\boldsymbol{z}\,) = \frac{p\,(c)\, p_c(\boldsymbol{z})}{p(\boldsymbol{z})} = \frac{n_c p_c\,(\boldsymbol{z})}{\sum_{r \in C} n_r p_r(\boldsymbol{z})}.$$

After training, the topological map that is proposed by PRSOM determines the parameters of the normal laws that characterize the various neurons. For any observation $\boldsymbol{z}$, it is then possible, applying the above relation , to compute the posterior probability of an obervation being assigned to a given neuron. Since a class is represented by a subset of neurons, the posterior probability that the observation $\boldsymbol{z}$ belongs to the class l_i is derived from the neurons that are labeled by l_i. If the subset of those neurons is denoted C_i, one gets

$$p\left(l_i \mid \boldsymbol{z}\right) = \sum_{c \in C_i} p\left(c \mid \boldsymbol{z}\right) = \frac{\sum_{c \in C_i} n_c p(\boldsymbol{z})}{\sum_{r \in C} n_r p_r\left(\boldsymbol{z}\right)}$$

where

$$p_c\left(\boldsymbol{z}\right) = \frac{1}{T_c} \sum_{c} K^T\left(\delta\left(c, r\right)\right) f_r\left(\boldsymbol{z}, \boldsymbol{w}, \sigma_r\right).$$

Note that this probability relies on the labeling of the map. That step is crucial for the computation of the posterior probabilities. Their consistency depends on the quality of the map. Thus, the classifier performances depend jointly on the amount of expert data, on the accuracy of the approximation of the observation density, and on the topological order that is built by the self-organization process.

The knowledge of posterior probabilities leads to a classification rule that is based on probabilistic estimation. Using those relations, the vector of class membership probabilities can be computed for each observation $\boldsymbol{z}$. Finally, the assignment of the observation to a class is performed by the application of Bayes rule: choose the class for which the membership probability is highest.

7.5 Applications

Self-organizing maps gave rise to a large number of applications. Specific developments were required for some of them, but they are in actual operation. At the moment, the most important research center for those topics is located at University of Technology of Helsinki (UTH). The major part of the research that is developed in its computer science laboratory (Laboratory of Computer and Information Science) is performed by the Neural Network Research Center, created by T. Kohonen in 1994, and now headed by E. Oja. The description of a large number of applications is now available on the Web site of NNR (http://www.cis.hut.fi/research/). The main research axis and the current applications are generally focused on self-organizing maps. Companies now exploit many applications. They arose from original, multidisciplinary research, and several research groups specialized such fields as bioinformatics, speech and writing analysis and recognition, and image analysis.

Actually, the implementation of self-organizing maps into larger systems widely uses the specific features of the application domain. The coding of the information, the organization of data bases, the analysis and visualization of the data, require specific, multidisciplinary research whose results are crucial for the performance of the self-organizing maps.

In the following, two applications will be described in detail. They were selected as representative of the domains to which self-organizing maps are relevant. The target of this presentation is twofold:

- The first section describes a satellite remote sensing application. It is a field of growing importance, and a lot of statistical problems must be solved by physicists and research engineers who are in charge of designing models. Considering that a very large amount of data is now available, this field is particularly suited to neuronal modeling. That application will fully illustrate the methodology that was described in previous sections. It uses the probabilistic model of self-organizing maps (PRSOM).
- The second section gives a brief account of one of the most popular applications that was developed at UTH: the WEBSOM system. That is devoted to information research on the Web. The earlier version was implemented in 1995. The salient feature of that application is the high dimensionality of the data. Dimensioning the topological map with a very large set of neurons and tuning the algorithm (regarding computing time and convergence accuracy) were the basic issues that were successfully faced at UTH. The development of WEBSOM spurred research oriented towards shortening the training phase for the design, and towards shortening the document research time during the exploitation phase.

7.5.1 A Satellite Remote Sensing Application

A lot of data is generated by the observation of earth with on-board sensors, and handed to geophysicists. All the neural methods that are presented in this book are helpful to process those data because they solve multidimensional statistical problems. Among those methods, unsupervised training is especially useful, because it allows extracting information even when expert information is scarce. Gathering expert information often requires costly analyses (ground mission, sophisticated biological and chemical analyses). That explains why expert-appraised data is scarce as compared to the amount of available satellite data.

Self-organizing bring valuable contributions to satellite data analysis, because estimating the observation probability density, and designing representative data partitioning, can be performed in a relatively straightforward way. Such information provides new insights into the physical phenomena of interest:

1. PRSOM estimate the variance and local uncertainty of the observations.
2. The partitions that are obtained are useful to the expert of the various application fields (physicists, chemists ...) because they may serve as an accurate summary of the observation set. Investigating such a summary may be crucial for understanding the phenomena of interest.
3. In all the fields that are concerned by experiments, heavy and expensive experimental campaigns are carried out regularly. With respect to the amount of satellite data, the expert-assessed observations are scarce, but they contain extremely valuable information. A few expert-labeled observations allow the identification of subsets of the partition from the

topological map. The classification methodology that is presented below in the section ≪Classification and PRSOM≫ allow that identification.

4. In order to demonstrate the various capabilities of self-organizing maps, the presentation of the next application is organized as follows:
   - Description of the application field, of the problems of interest, and of the available data;
   - Description of experiments that allow the understanding of the influence of data encoding on the partition and on the topological order that is obtained;
   - Description of experiments that allow the assessment of the impact of expert knowledge

7.5.1.1 The Color of the Ocean

The biological activity of the ocean is crucial for the natural economy of the earth, for it is strongly connected to the fishing resource, and is part of the biochemical cycles with strong climate consequences. During the past few years, several multi-spectral sensors dedicated to measure the color of the ocean were launched on different satellites ((MOS, POLDER-1, OCTS, SEAWIFS, MODIS) or will be in the future (MERIS, POLDER-2, GLI). They are expected to estimate the chlorophyll contents of the upper layer of the ocean, and to assess its space and time variability.

Two steps are needed to recover the pigment fields from the ocean color satellite data. The first step deals with atmospheric correction (indeed, the atmosphere contributes more than 80% of the measured signal); the second step is dedicated to ocean interaction (see Fig. 7.20). The atmospheric correction algorithm currently computes sea-level reflectance by canceling atmospheric effects (aerosols and air molecules). The second step aims at inverting that reflectance to provide the chlorophyll pigment concentration. This is a tricky issue, because one has to take into account both uncertainties that arise from atmospheric corrections, and uncertainties that are intrinsically related to the variation of biological population. Thus, the knowledge of the atmospheric aerosol composition and of the water constituent concentration is crucial. The following experiments aim at assessing various classes of aerosols and waters from the top of atmosphere spectra, which are provided by the sensors.

7.5.1.2 The Data

The data that are used hereafter were provided by US radiometer SeaWifs, which was located on the SeaStar satellite. That sensor has eight spectral bands in the visible and the near infrared spectrum (see Table 7.1).

For each measurement location at the ocean surface, the observation vector is 8-dimensional. Its components are the eight radiances that are measured on the top of atmosphere. The following results are a representative example

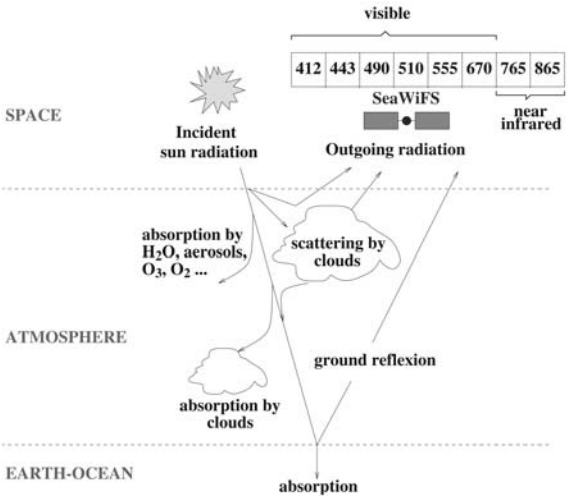

Fig. 7.20. Outline of the physical phenomena that are relevant to explain the observations. The observations consist in the set of reflectance spectra that are measured by the SeaWifs radar. That radar uses eight wavelengths for analyzing the surface

of ocean color data processing. They come from a SeaWifs image. This image was over West Africa and the Canarie islands on January 5th, 1999. Two pictures of that zone, which were obtained on the same day with different resolutions, are available: a LAC (*Local Area Coverage*) image, with $2141 \times 793 = 1,697,813$ pixels (Fig. 7.21) and lower resolution, GAC (*Global Area Coverage*) image with $536 \times 199 = 106,664$ pixels. The topological map design was performed from a sampled LAC image. Sampling was performed by line decimation of the original LAC image. Thus the training set had $238 \times 793 = 188,734$ pixels. The quality of the resulting vector quantization, was assessed from the full LAC picture. Since 90% of the pixels had no contribution to

Table 7.1. Spectral bands of SeaWifs

Bands K	Wavelengths (nanometers) λ_k
1	412
2	443
3	490
4	510
5	555
6	670
7	765
8	865

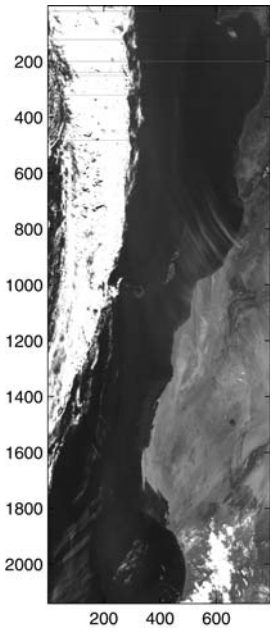

Fig. 7.21. ThiS SeaWifs image has been taken over West Africa and the Canarie islands on January 5th, 1999. LAC picture 2141 × 793

the training phase, and considering the large amount of available data, the performances thus estimated may be considered as fully significant.

The expertise was provided through the GAC image. That image had additional information, from two sources:

- Information that was provided by SeaWifs was available, such as land masking, presence of clouds.
- A pixel classification of the GAC image, using different optical models provided from atmosphere experts, was also available. Figure 7.22 shows the expert-processed GAC picture. On that image, five classes were identified by the expert: aerosols that arise from the desert, so-called case 2 waters, which are waters with high organic matter contents, sea aerosols, clouds and land. Actually the pixels that are assigned the land label are residual pixels for which no label was provided by the expert
- Note that both types of information may contain errors, just as any expert-based classification of highly complex phenomena. For instance, the expert sought five classes. Therefore, different aerosols may have been clustered in the same class if he had no appropriate physical model to discriminate between them.

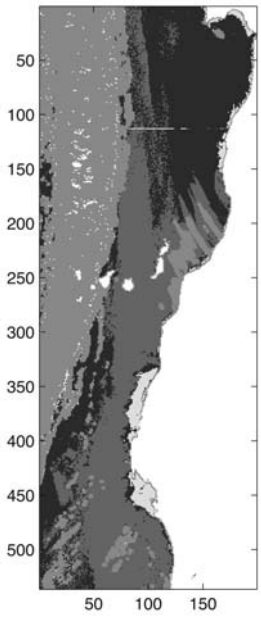

Fig. 7.22. Expert-processed GAC image. The picture represented five zones whose boundaries had been defined by the expert: the aerosols coming from the desert (*black*), the case-2 waters (*light grey*), the marine aerosols (*dark grey*), the clouds (*medium grey*) and the earth (*white*)

7.5.1.3 The Role of Encoding

Numerical experiments were performed using different encodings of SeaWifs spectra.

The first encoding uses directly the reflectance at the top of the atmosphere. In order to normalize the influence of the wavelengths, the reflectance spectrum values were normalized to the $[-1, +1]$ interval. The reflectance for wavelength λ_k, is denoted $\rho(\lambda_k)$, the normalization has been computed from the learning set for each wavelength, $(k = 1, \ldots, 8)$. Thus, each observation was encoded by an 8-dimensional vector, each component of that vector being the normalized reflectance for a given wavelength. Since the reflectance values $\rho(\lambda_k)$ are numbers between 0 and 1, they were normalized between -1 and 1, using to $2\rho(\lambda_k) - 1$.

The sampled LAC image that was used for training (actually one picture line out of ten), after coding as described above, will be denoted by $\mathrm{App}_{\mathrm{cod1}}$. A second coding was performed, in order to highlight the shape of the spectra of interest. To that effect, the slopes of spectra were taken into account. They were computed for each wavelength. Thus, the kth component of the new coding vector was computed from the reflectance as

$$\Delta\rho\left(\lambda_k\right) = \frac{\rho\left(\lambda_{k+1}\right) - \rho\left(\lambda_k\right)}{\lambda_{k+1} - \lambda_k}.$$

Seven slopes were thus computed for k from 1 to 7. In order to keep information about the spectrum intensity, a component of the observation vector was the norm $\|\rho\|$ of the spectrum. Thus, each observation was encoded in an 8-dimensional vector

$$[\Delta\rho(\lambda_1), \ldots, \Delta\rho(\lambda_7), \|\rho\|]^{\mathrm{T}}.$$

The sampled LAC image that was used for training was encoded according to that second scheme; the result will be denoted $\mathrm{App}_{\mathrm{cod2}}$. Normalization between –1 and 1 was performed as previously. Since the slopes and the norm do not have the same order of magnitude, the normalization was implemented separately for each component as

$$2\frac{x - \min}{\max - \min} - 1,$$

where x is a derivative (namely $(\Delta\rho(\lambda_k)k = 1, \ldots, 7)$, min (resp. max) are the minimum (resp. the maximum) over the set of all the derivatives in $\mathrm{App}_{\mathrm{cod2}}$.

For all test data, the same encodings were used. The following numerical experiments are an illustration of the methodology, which has been described in the section "Classification and PRSOM." They use quantizations, which are followed by classifications. The quantizations are obtained from the probabilistic maps, and clustering is performed by hierarchical classification. All self-organizing maps have the same architecture:

- The input layer has 8 units
- The map is 2-dimensional, with 10×10 neurons. The neighborhoods are defined from the exponential kernel family $K(\delta) = \exp(-\delta^2)$.

7.5.1.4 Quantization Using PRSOM

In the first part of the study, PRSOM was used for determining the patterns that are representative summaries of the set of all observed spectra. In that case, a fine quantization of the training set is sought. Actually, the result is a summary of the training set; if it is statistically representative, it is also a summary of all observations. Otherwise, the generalization may be poor since a subset of the set of observations was overlooked. The two encoding schemes that were described above (normalized radiance vales for the first one, slopes + norm for the second one), resulted in different maps. Those maps will illustrate the importance of the encoding process for quantization and topological order. Each map quantizes the observation set intro 100 subsets. Figure 7.23 shows the map that was obtained with $\mathrm{App}_{\mathrm{cod1}}$; on that figure, the number that is located above the neuron indicates how many pixels from the training set are allocated to that neuron. Figure 7.24 shows the

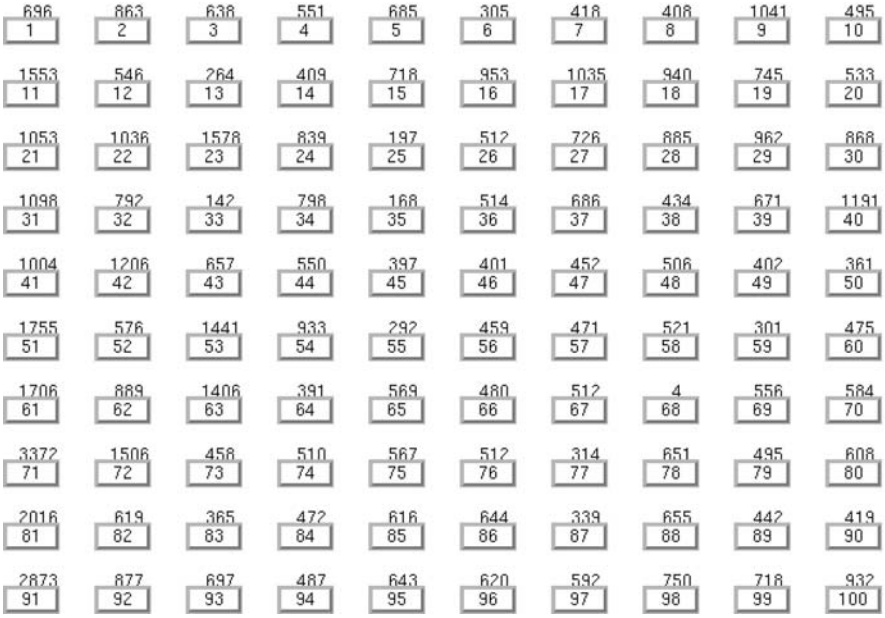

Fig. 7.23. PRSOM (10×10) map obtained using App_{cod1} as the training set. The map representation displays the 2D topological ordering. Each square contains a number that identifies the neuron, and the figure above each square is the number of pixels of the training set that are assigned to that neuron

same map, and, for each neuron, the associated variance. It is clear that scattered values of the reflectance are represented at the upper right corner of the map, whereas the lower left corner contains representative of similar spectra. Physical considerations provide an interpretation of the various zones of the map:

- The spectra are more stable if the sky is clear and if the signal permits an analysis of the ocean
- High variability zones may stem from atmospheric influence that is due to aerosols or to cloud reflection

The hundred patterns associated to the neurons are characterized by spectra that are of the same kind as the observations (vectors of $\mathbb{R}^8$). Figure 7.25 is presenting for the first coding, the set of patterns and their topological connections through the map. It is the same map than in Figs. 7.23 and 7.24 but each neuron is now attached to its encoded pattern. Each encoded pattern is an average spectrum over the allocated reflectance observations. The patterns are organized according an order, which is visualized through the map. The patterns that are associated to weak variance local densities have similar shapes: the observation space is sampled there in a very fine way. Patterns that belong to the high variance regions are sampling the observation space

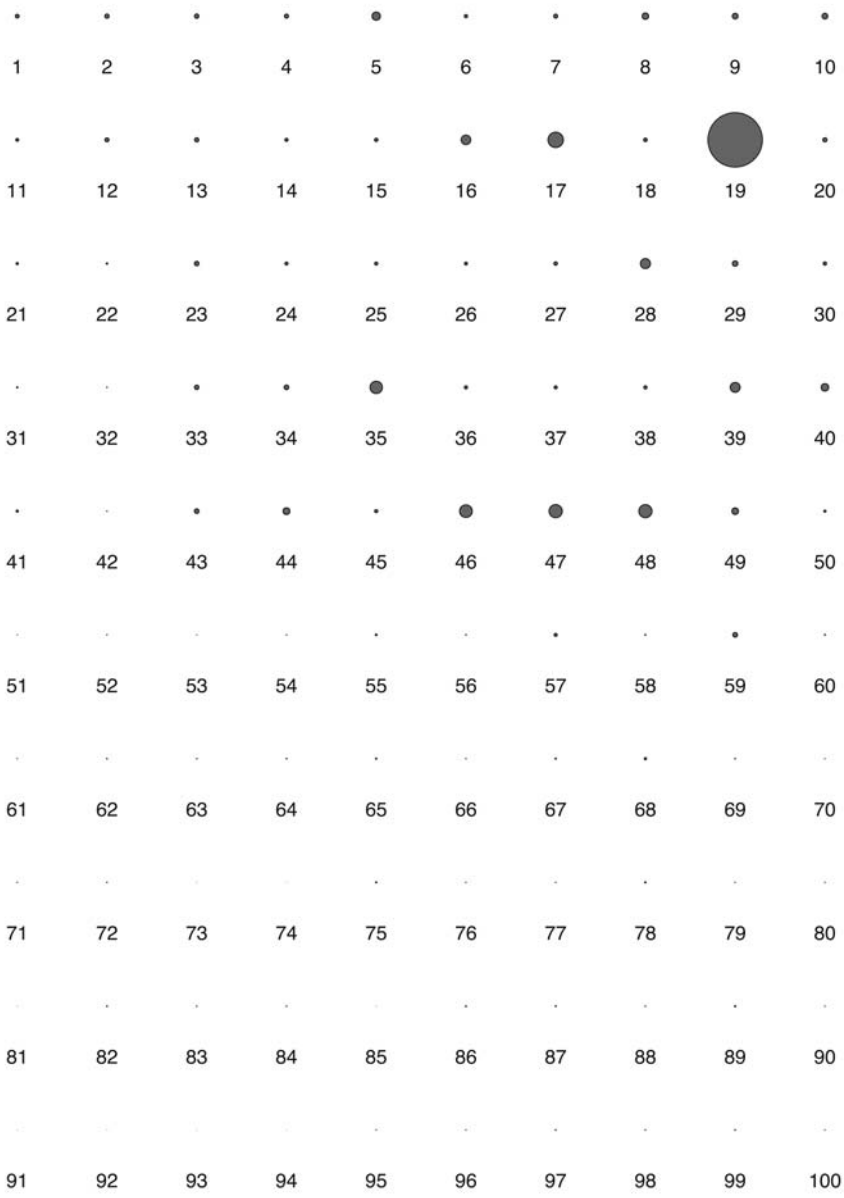

Fig. 7.24. Estimated variance for PRSOM. The map is the (10×10) map of the previous figure. The area of the disk above the neuron is proportional to the estimated variance of the associated Gaussian distribution

in a coarser way. The visualization technique enables to select some patterns and then to specifically study the set of observations that are allocated to one of these patterns or to locate them on the SeaWifs image.

A first investigation is useful to check the quality of the process (Fig. 7.25): it is possible to identify wrong spectra with respect to the measurement process. Actually, on this map, patterns that are associated to neurons 17,28,35 and 39 have a vanishing wavelength reflectance. If one gathers the concerned information, it appears that they all present the same fault. It is then possible to infer that in these cases a channel was defective and that some neurons have specialized in detecting this fault. Figure 7.26 displays the spectral patterns that are associated to neurons 17 and 35. and their variance.

One may perform a similar analysis for any of the 100 neurons of the map. Figure 7.27 shows the spectrum that is associated to neuron 51, which is located in a high-density zone. Then the set of allocated radiance spectra is displayed as well as the associated geographical zone on SeaWifs image. When it is compared with SeaWifs image of Fig. 7.21 one may notice that neuron 51 controls a light colored zone located on the sea and for which there is apparently neither desert aerosols nor clouds. When the ordering of spectra that is proposed on Fig. 7.25 is inspected, one notices that the proposed coding is governed by the spectral intensities. Thus the ordering favors the emergence of underlying physical properties. The same experiments have been performed using another encoding process that takes into account both the intensity and the shape of the spectra ($\mathrm{App}_{\mathrm{cod2}}$). Figure 7.28 shows the new ordered patterns that were obtained. (On that figure, the patterns have been decoded in order to show their original spectrum profile). The organization of the neurons is now performed with respect to their intensities and their shapes.

7.5.2 Classification and PRSOM

The first experiment group allowed us to assess the quality of the vector quantizations, which were obtained using PRSOM. We shall use now these quantizations to achieve classification tasks.

A first possibility was displayed in previous section. Recall it amounts to study separately the physical property of each neuron pattern of the map. This study has to be performed by an expert, who is able to recognize the aerosol class and thus to label the patterns from their spectral properties. If all the neurons are identified, then the partition that is obtained through the self-organizing map enables to use it straight to classify the whole image SeaWifs. Moreover, if the learning set is representative of the physical problem, it may be used to label other SeaWifs images that share the same physical properties.

If this identification process is not possible, i.e. if the expert is not able to label accurately every neuron of the map, it is possible to cluster the neurons according to an unsupervised way. One can proceed as it was demonstrated in the previous section "Classification and topological map," by aggregating the

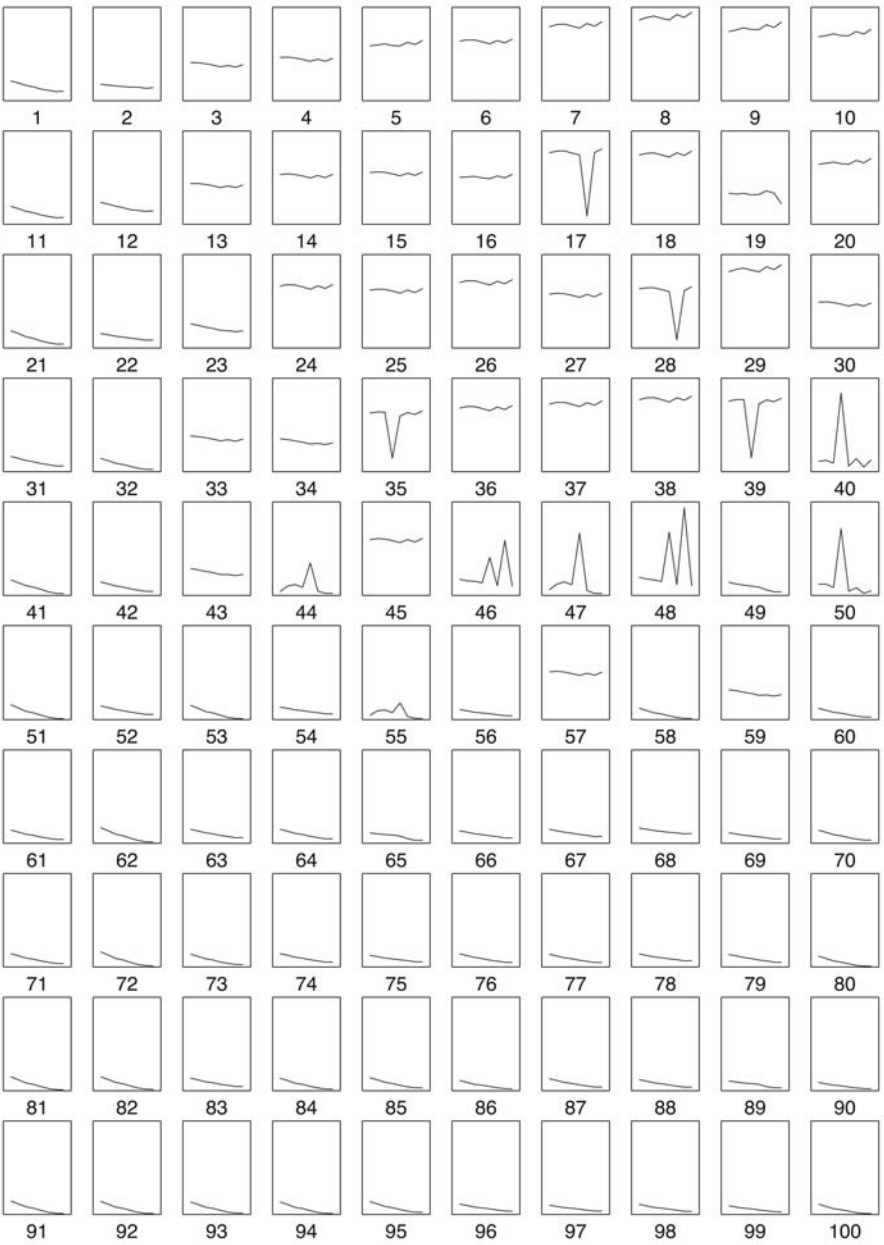

Fig. 7.25. Vector quantization from the previous PRSOM (10×10) map from the training set App_{cod1}. The number that identifies the neuron is above the square, the associated spectral pattern is drawn inside each frame

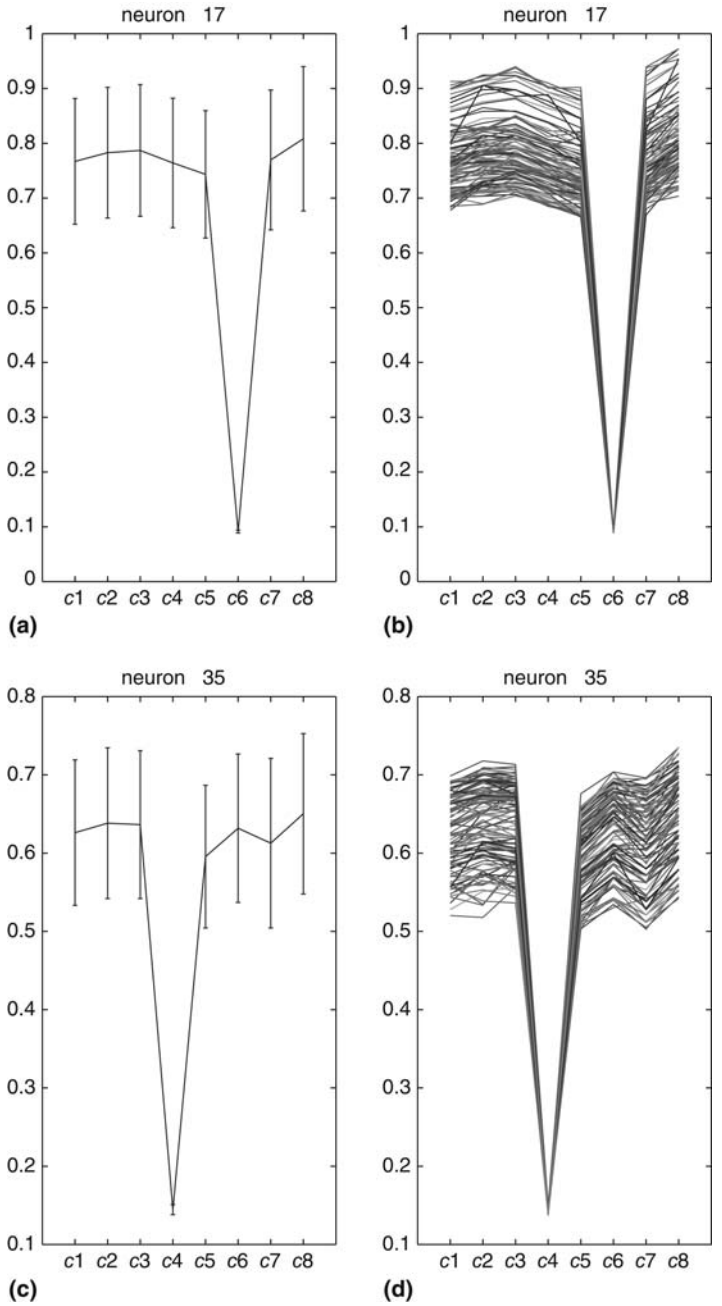

Fig. 7.26. Figures (**a**) and (**c**) represent the spectral patterns that are associated to neurons 17 and 35. Vertical bars represent the variance for each wavelength. Figures (**b**) and (**d**) represent the radiance subset that are respectively represented by neurons 17 and 35. (PRSOM 10×10 map obtained from $\mathrm{App}_{\mathrm{cod1}}$)

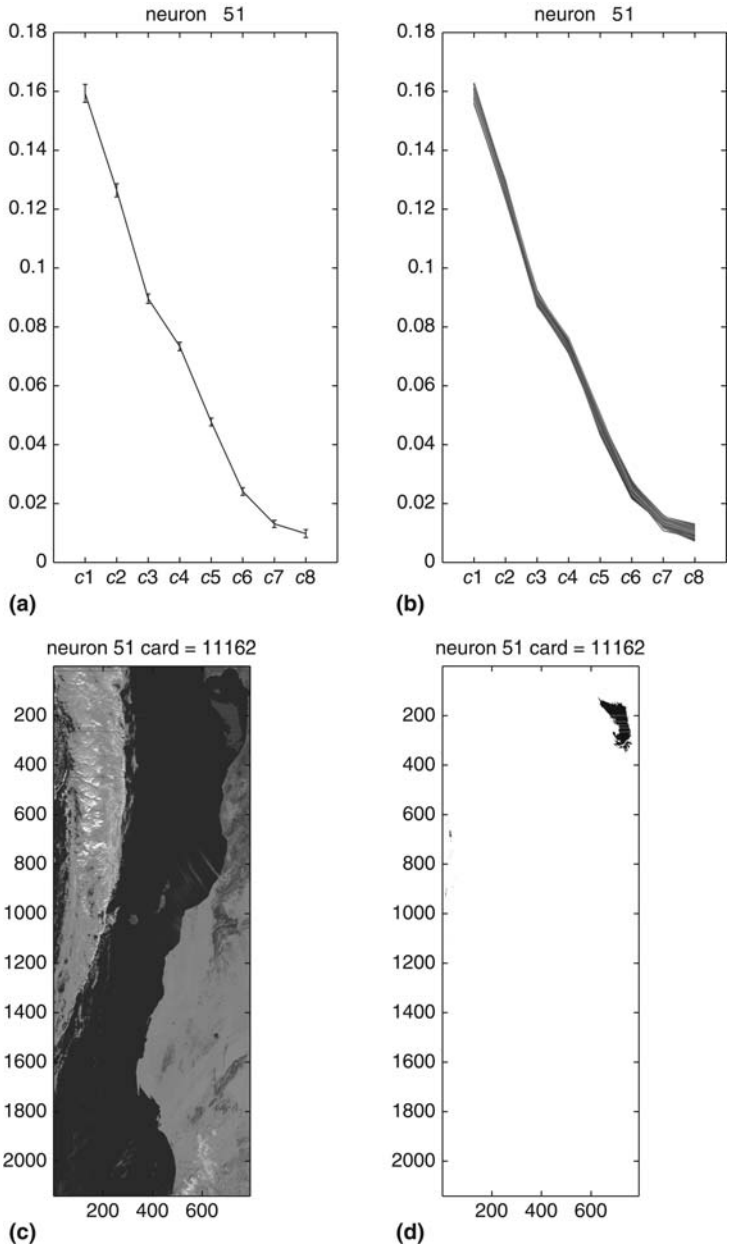

Fig. 7.27. Representation of neuron 51 spectral characteristics: (**a**) representation of the spectral observation vectors of the learning set $\mathrm{App}_{\mathrm{cod1}}$ that are allocated to neuron 51 (**b**) representation of the spectral observation vectors of the geographical zone, which is allocated to neuron 51, (**c**) representation of the geographical zone in the total image, (**d**) representation in black of this geographical zone. (PRSOM 10×10 map obtained from $\mathrm{App}_{\mathrm{cod1}}$)

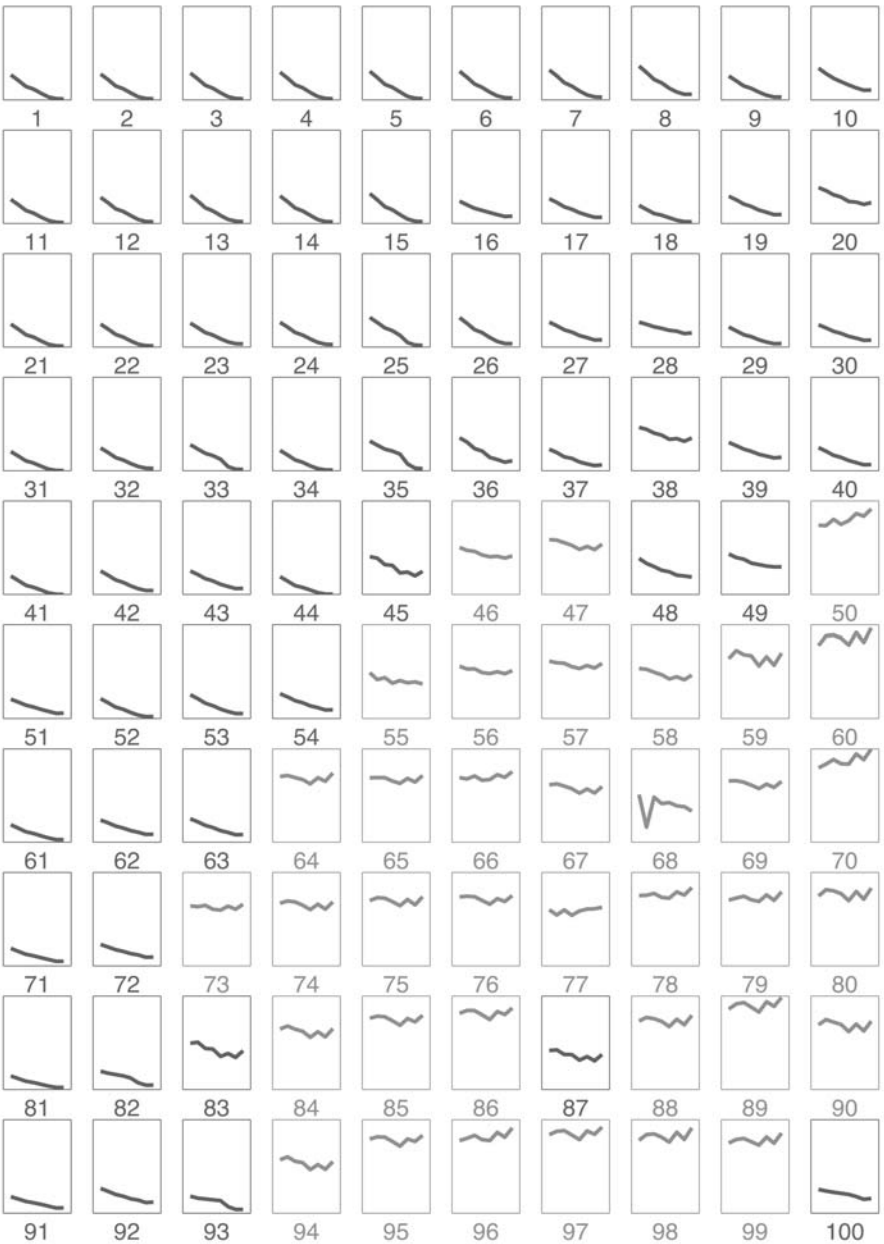

Fig. 7.28. Representation of vector quantization, which is associated to PRSOM map that is obtained from App_{cod2}. The identification number of the neuron is above the frame. The associate pattern spectrum is depicted inside the frame

neurons into classes and trying to label the classes that were obtained through hierarchical classification. In order to illustrate as best as possible the quality of results when PRSOM and BUHC (Bottom-Up Hierarchical Classification) are sequentially processed, two experiments of different complexity are presented:

- The first experiment is relative to the determination of a mask that is able to detect the thick clouds and to discriminate them from other spectra. It is known that clouds are strongly reflecting the signal: the top of atmosphere signal that are registered by the satellite sensors are presenting stronger and more variable intensities than when sea or aerosols are concerned. The discrimination between thick clouds and other constituents amounts to build a binary classifier. Since clearly distinct properties in the observation set physically characterize this problem, the two classes that are searched have to be fully separated.
- In the second experiment one tries to recover the five classes, which have been identified by the expert; these classes have been determined by comparing the data with aerosol physical models. Actually, the number of classes is higher and the expert has possibly introduced a lot of mistakes, so this problem is far more complex.

The two vector quantizations that have been obtained using PRSOM will be used to recognize the expert-identified classes. Class determination will be performed through bottom-up hierarchical classification using the Ward index that has been previously defined in the paragraph ≪Looking for a partition that suits the classes of interest≫.

In the first experiment, bottom-up hierarchical classification is performed on the PRSOM 10×10 map obtained from App_{cod1}. Since the searched classification is a binary classification to select thick clouds, clustering has been pursued up to the obtention of two classes. Figures 7.29 and 7.30 show the classifications that were obtained on the topological map and on the image. The visualization of the map enables to observe the neurons of each class. Clearly, the associated zones are well connected on the map. This classification has been compared with the expert-based classification by computing the confusion matrix. Here, SeaWifs provided the expert-based classification since the cloud mask is available. The confusion matrix is represented on Table 7.2; it

Table 7.2. Confusion matrix that compares the SeaWifs labeled classification and the classification that was obtained from PRSOM + BUHC. PRSOM was obtained from Appcod1 and BUHC uses Ward index

	PRSOM + BUHC	
	Clouds	Apparent sea
SeaWifs-labeled clouds	0.91	0.09

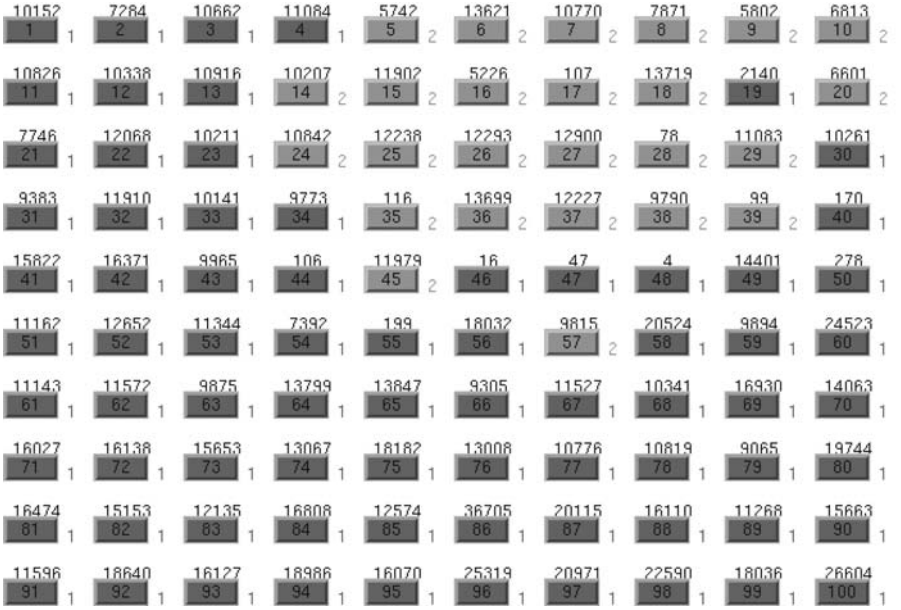

Fig. 7.29. Presentation on the map of the two classes that are proposed by PRSOM + BUHC: PRSOM was determined from $\mathrm{App}_{\mathrm{cod1}}$, and BUHC uses the Ward index. The number at the right of the neuron frame represents the class that is obtained through hierarchical classification. The set of dark grey neurons constitutes class 1 and the set of light grey neurons constitutes class 2

allows to compare the two classifications. The division between two geographical zones that is displayed on Fig. 7.30 shows a good adequacy to the division that comes straight from the satellite data. It shows that the two classes have been correctly demarcated by using PRSOM algorithm and then bottom-up hierarchical classification onto the raw physical measurements without any additional expert information. The nice consistency of hierarchical classification results at the top level of the classification tree allows expecting that clustering is consistent at any level of the hierarchy.

The second experiment is searching a zone that is already provided by expert knowledge. Actually this class is case 2 water. This experiment confirms the quality of vector quantization. In Fig. 7.22, this zone, which is endowed with particular optical properties is colored in light grey. Investigation of the various geographical zone that are associated to the 100 neurons of the map enables to select three neurons. The subset of their associated spectra allows to find a geographical zone that can be superimposed to the expert-based region. Figure 7.31 shows the three patterns (w_{33}, w_{82}, w_{93}) and the associated geographical zones.

If the full hierarchy that arises from BUHC is investigated, one notices that the three neurons of interest (namely 33, 82 and 93 constitute a subset that is

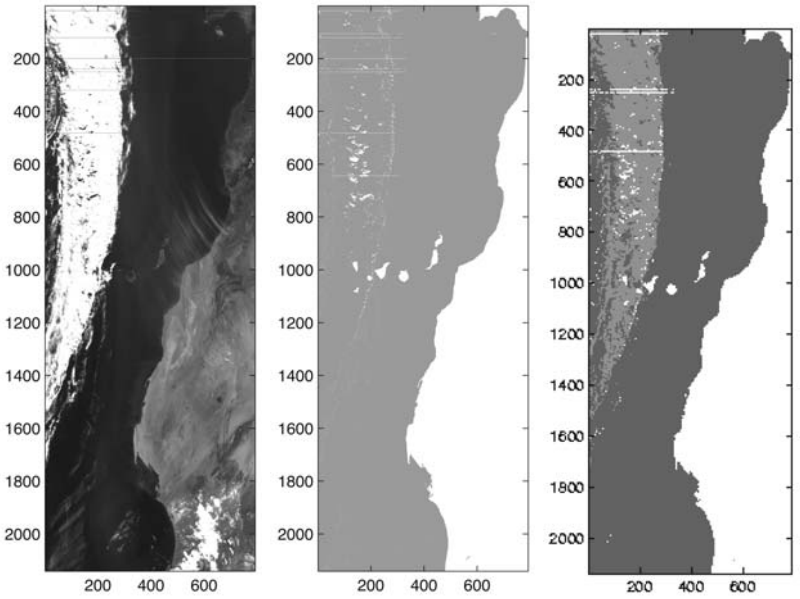

Fig. 7.30. Presentation of the two classes that are proposed by PRSOM + BUHC: the *left* image is representing the original SeaWifs image the *center* image represents the land mask that is provided by SeaWifs (in *white* on the image) and the *right* image presents the binary classification that is provided by PRSOM + BUHC (the light zone is similar to the class 2 of Fig. 7.29 and represent thick clouds, the dark zone includes various aerosols); PRSOM was achieved using App_{cod1} as learning set and BUHC uses the Ward index

clustered at the level 35 of the hierarchy. Thus, it is possible to propose from this result a classifier that enables to automatically identify the case 2 water labeled pixels. The neurons 33, 82 and 93 will be labeled case 2, all the other neurons will be labeled by a negative label that points out they do not belong to case 2. Then, it appears that in that case, using the hierarchical clustering without additional expert knowledge may allow retrieving information which comes from the physics of measure.

The last experiment is using straight expert knowledge. Knowledge is used to label neurons according to the methodology that was presented previously in the section about classification and topology. The test set is the expert-labeled GAC image. Each neuron is then mastering pixels of GAC image that are allocated to that neuron. These pixels constitute a subset of labeled elements. The label of the associate neuron is determined by a majority voting. The two classifications are compared on Table 7.3 where the confusion matrix is represented. It is clear that the neurons correctly provide the expert knowledge.

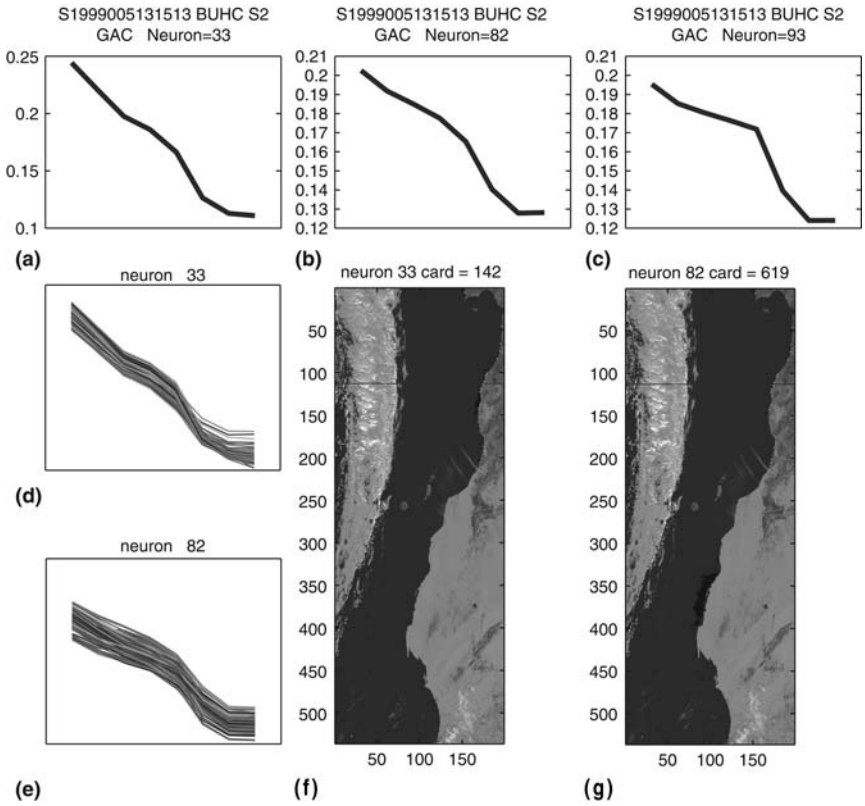

Fig. 7.31. Representation of the patterns, of the observation subsets and of the geographical zones that are allocated to neurons 33, 82 and 93 (PRSOM map 10×10 using App_{cod2} as learning set). Figures (**a**), (**b**), (**c**) represent the neuron spectra, Figures (**d**) and (**e**) represent the associated learning set spectra for neurons 33 and 82. The zones that are associated to neurons 33 and 93 have been colored in black in figure (**f**) and (**g**)

Table 7.3. Confusion matrix allowing the comparison between the expert-based classification (GAC image) and the PRSOM-based classification. On this map, the hundred neurons have been labeled using majority vote from expert data

		PRSOM + majority vote			
		Class 1	Class 2	Class 3	Class 4
Expert	Marine aerosols	0.8	0.04	0	0.16
	Clouds	0.03	0.91	0.01	0.05
	Case 2	0.03	0.22	0.71	0.03
	Desert aerosols	0.1	0.04	0	0.86

The bunch of results that were obtained in that application demonstrates that topological maps are correctly operating when they are used to process numerical data.

The following application is due to T.Kohonen. It shows that the algorithm is performing well when it is used for textual processing.

7.5.3 Topological Map and Documentary Research

This last paragraph presents another real-world application in a field that is completely different: documentary research. The general objective of the Websom system that was created by Kohonen and his collaborators is to create a content-based labeling of a set of texts. The current working version allows organizing 7,000,000 texts in a single documentary data basis. Thus documents with close-by semantics are endowed with neighbor label. A visual inspection of the basis representation provides a global hint about the content of the documents that are stored in a particular zone of the basis. Looking for the keywords that are associated to the zone and considering the topics of the different documents allow document searching in an original way. Considering his short description of Websom system's main characteristics, one feels how self-organizing maps are used: semantically close observations (texts) are allocated to neighbor neurons on the map. In order for the application to be operational, several additional properties have to be checked:

- As for the remote-sensing satellite data application, the quality of the system depends closely on the semantics of the texts of interest.
- Documentary research is useful only if the number of stored texts is large enough and if the visualization is fine enough. Thus the dimension of the map has to be very high.
- The system is supposed to be operated on-line, thus it has to work fast.
- The basis algorithms has to be changed to allow
  1. introducing a linguistic knowledge that enables textual manipulation,
  2. training high-dimensional maps to be able to process as many documents as possible,
  3. using a friendly interface which really helps the user to perform document research,
  4. reducing the duration of an average research session.

7.5.3.1 Information Coding

When a text is preprocessed significant information is extracted that depends on the specificities of the general field of the research. Of course, the encoding has to be made according the specifications of the topological maps: Kohonen's algorithm is processing numerical multidimensional data. Thus any text has to be represented by an n-dimensional numerical vector. The current version of Websom system is processing a corpus that contains 6,840,568 English

summaries. The average length of such abstract is 132 words. To process also number and symbols, it has been decided to cancel the words that are too scarce (less that 50 occurrences) and also to cancel 1355 words that are semantically poor. Eventually a set of 43,222 words has been considered for the whole corpus.

Several versions of the system exist. The earlier was coding straight the text histogram by a vector, the length of which was the word number of the corpus. According to that coding, each component of the text representative vector represents the weighted occurrence frequency of the associated word in the text. The weights were fixed according to the influence of the word on the document global meaning. This dimension was too large to allow further processing. Several data compression methods were proposed to cope with this high dimension problem: projection reduction (principal component analysis) or random projection. Eventually a random projection method was implemented. A 500-dimensional vector represents each text. Such a vector is a text summary coming from a statistical analysis of the text vocabulary. The coding complexity is O(NL)+(n), where N is the document number, L is the average number of distinct words in a document and n is the initial histogram dimension. To appreciate the reduction, it is interesting to point out that the simpler projection compression method has a complexity equivalent to *Nld*. Thus, the reduction is quite important and enables to extend Websom over the whole corpus.

7.5.3.2 Specific Features of Learning Process

A visual representation of the corpus organization is possible through the two-dimensional map. That is a great help for documentary research. At the end of the learning phase, the allocation phase that associates a neuron to a document enables to locate a given document with respect to the global corpus: texts with similar meanings are supposed to be located in close zones on the map. In Websom's last version, the corpus is divided into 21 sections (agriculture, transportation, chemistry, electricity, etc.). To extract this information, each neuron is endowed with one of the section labels and a set of key words. These keywords are extracted from the subset of texts that are allocated to this neuron. More precisely, the type is determined through a majority voting over the text subset and the keywords are selected by building the intersection of the key-words set of every text of the text subset.

When Websom is used, texts with close meaning are projected in closed regions of the two-dimensional map. So, projecting the text onto the map enables to locate its meaning with respect to the whole set of texts of the learning basis, actually the whole corpus. Using the map labeling enables to interpret a new text through an automatic process. The neighbor neurons provide subsidiary information that allows a finer understanding.

Considering the very large number of documents that lie in the basis, a large amount of neurons are required in order to perform a fine enough

document analysis. The more important change that was introduced enables to train fast high-dimensional maps. The topological map that is used for Websom is composed by 1,002,240 neurons. It is impossible to train this map because the connection number is too large: 1,002,240 × 500. The new idea relies on the simple fact that a good initialization considerably increases the convergence speed. This good initialization is found through a hierarchical procedure that enables to guide the training from one step to the next one. In the Websom implementation, the parameters are tuned using a first rectangular map of 435 neurons. This first map is extracted form the learning basis. Then a second map that uses a finer sampling is initialized using the results of the first one: the initial values of a parameter of the second map are obtained through an interpolation of the values of the three closest neighbors extracted from the 435 neurons of the first map. In that way, the number of neurons increases from step to step up to 1,002,240 neurons. For each step, there is a new learning phase of the whole corpus. The initial learning phase (for the 435 neurons of the first map) requires 300,000 iterations; every further learning phase only requires five iterations of the "dynamical clouds" versions of the algorithms. In such a way, it is possible to train very large maps. Moreover, the hierarchical order that was found in the previous steps is used to find the closest neighbors during the successive learning steps.

7.5.3.3 Discussing Websom Performances

The various improvements, which were implemented in Websom are quite efficient with respect to the time complexity of the computation. The methodology that was previously detailed allows reducing the number of operations from $O(dN^2)$ for the original Kohonen algorithm to $O(dN^2)$ $O(dM^2)$+ $O(dN)$ +$O(M^2)$ for Websom. In that expression, N is the number of neurons in the actual map, M is the number of neurons in the initial map and d is the dimension of the input layer ($d = 500$ for Websom). The comparisons that were achieved with the original Kohonen methodology show that the last version of the implementation has the same performances than the original algorithm with respect to the quantization error and the classification error. The final version of the map was obtained through a six-week learning phase that was performed on a six-processor computer (SGI O2000). The performances over the seven millions text basis go up to 64% of correct classification. As it is generally the case for data mining applications, the interface was very carefully designed : the map is presented as a sequence of HTML pages. It is easy to explore it by using the mouse. A simple click enables to reach the documents and then to visualize and to read them.

References

1. Anouar F., Badran F., Thiria S. [1997], Self Organized Map, A Probabilistic Approach, *Proceedings of the Workshop on Self-Organized Maps*, Helsinki University of Technology, Espoo, Finlande, 4–6 juin 1997

2. Bock H.H. [1996], Probabilistic Models in Data Analysis, Computational Statistics and Data Analysis, *23*, pp 5–28
3. Bock H.H. [1998], Clustering and neural networks, *in* Rizzi et al. (éd.), *Advances in data science and classification*, Springer verlag, pp 265–278
4. Cerkassky Y., Larmnajafih [1991], Constrained topological mapping for non parametric regression analysis, *Neural Network*, vol. 4, pp 27–40
5. Dempster A.P., Laird N. M., Rubin D. [1977], Maximum Likelihood from incomplete data via the E.M. algorithm (with discussion), *Journal of the Royal Statistical Society*, series B 39, pp 1–38
6. Didday E., Simon J.C. [1976], Clustering Analysis, *in Digital Pattern Recognition*, K.S. Fu, Springer verlag
7. Duda R.O., Hart P.E. [1973], *Pattern Classification and Scene Analysis*, John Wiley
8. Fritzke B. [1995], *A growing Neural Gaz Network learns topology*, D.S. Touretzky and T.K. Leen (editors), Advanced in Neural Information Processing Systems 7, MIT Press, Cambridge MA
9. Gaul W., Opitz O., Schader M. (éd.) [2000], *Data Analysis Scientific Modeling and Practical Application*, Springer
10. Jain A.K., Dubes R.C. [1988], *Algorithms for Clustering Data*, Prentice Hall
11. Kaski S., Honkela T., Lagus K., Kohonen T. [1998], WEBSOM-self-organizing maps of document collections, *Neurocomputing*, vol. 21, pp 101–117
12. Kaski S., Kangas J., Kohonen T. [1998], Bibliography of self organizing map (SOM) papers 1981–1997, Neural Computing Survey, vol. 1, pp 102–350. On peut trouver cet article à l'adresse: http://www.icsi.berkeley.edu/∼JAGOTA/ncs/
13. Kohonen T. [1984], Self organization and associative memory, Springer Series *in Information Sciences*, 8, Springer Verlag, Berlin (2^{nd} éd. 1988)
14. Kohonen T., Kaski S., Lagus K., Salojrvi J., Honkela J., Paatero V., Saarela A. [2000], Self organization of a massive document collection, *IEEE transaction on neural networks*, vol. 11, no 3
15. Kohonen T. [2001], *Self Organizing Maps*, Springer, 3^{e} édition
16. Luttrel S.P. [1994], A bayesian analysis of self-organizing maps, *Neural Comput*, 6
17. Murtagh F. [1985], A survey of algorithms for contiguity-constrained clustering and related problems, *The Computer Journal*, vol. 28, pp 82–88
18. Oja E., Kaski S. [1999], *Kohonen Maps*, Elsevier
19. Vichi M., Bock H.H. [1998], Advances in Data Science and Classification, Springer, Heidelberg, pp 397–402
20. Von der Malsburg C. [1973], Kybernetik 14, 85
21. Yacoub M., Badran F., Thiria S. [2001], *Topological Hierarchical Clustering : Application to Ocean Color Classification*, ICANN'2001, Springer 2001, Proceedings, pp 492–499