LECTURE 8: UNSUPERVISED METHODS

Data Segmentation

A *data segmentation* is the partition of a data space into a set of non-overlapping regions whose union is the data. Often we will perform such a segmentation as we believe that it will:

- decompose the data set into a series of *meaningful components*. However....
- there is a problem in telling a computer what a 'meaningful segmentation' is
- A set of rather heuristic 'rules' is often quoted
    1. Segmentations should be homogeneous in some quantity.
    2. Partitions should be 'simple'.
    3. Boundaries should be 'simple'.

There are, primarily, two domains in which we may achieve a practical segmentation, the *observation domain* and the *feature domain*.

Observation domain segmentation looks to describe edges between regions of an observations set, such as a time series or image, based on the temporal or spatial properties of the data.
Feature space segmentation performs a mapping of the observations to some feature space on the basis of measurements from each observation and its environment. Once a feature space is created we look to *partition* this space. These partitions are mapped back onto the observation space and form the basis of segmentation.

The latter methodology has the advantage that, in principle, both features of the obervations and information regarding their context (local relationships) may be taken into account in the partitioning.

Unsupervised methods

To recap, there are two approaches to the segmentation of a pattern space, namely *unsupervised* and *supervised* methods.

Given a space in which to perform a partitioning, unsupervised partitioning relies on the fact that *patterns (feature vectors) belonging to the same object or state tend to have similar feature vectors* and *vice versa*. This means that segmentation of the data set relies on finding the 'blobs' or 'clumps' which dominate feature space (we hope). In the case of unsupervised partitioning :

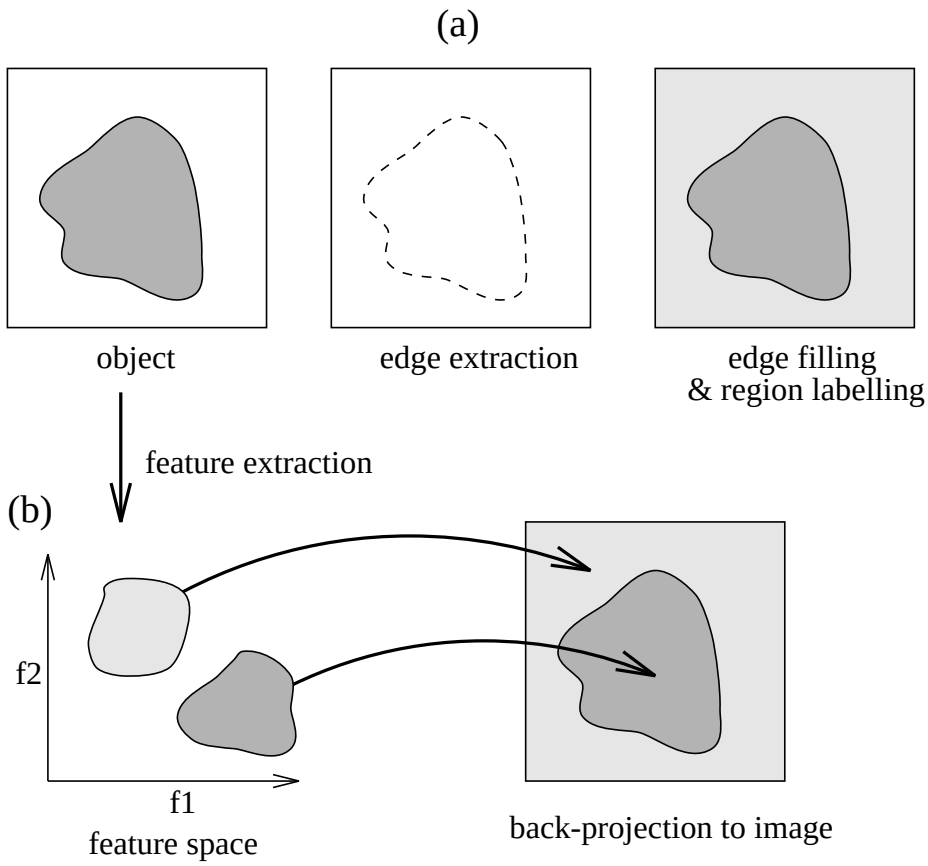

Figure 1: An example of image segmentation: Observation space (a) and feature space (b) segmentation.

> *Each node/cluster/blob is taken to be a class. Hence if we search for the positions and shapes of K clusters, the data will be segmented into K regions. Generally, we do not know K ahead of time, and methods for assessing the probable number of clusters in a data set are a subject of active research.*

Clustering

On a broad, descriptive level cluster analysis algorithms can be broken into two distinct phases.[1] Firstly, a model fitting phase, whereby some *partition hypothesis* of complexity K, $\mathcal{H}_K$ say, is 'optimally' fitted to the data set. Secondly a model validation phase, whereby the set of $\{\mathcal{H}_K\}$ are assessed according to some cluster validity criterion and the 'optimal' partition hypothesis selected.

We make the distinction between *hierarchical* and *partitional* fitting methods. For the most part, the former act by partitioning the data set into successively fewer structures, based upon merging structures which have sufficient similarity. Such a

[1] One could argue that, in the case of Bayesian inference methods, these two phases are so interelated as to form a single methodology, however.

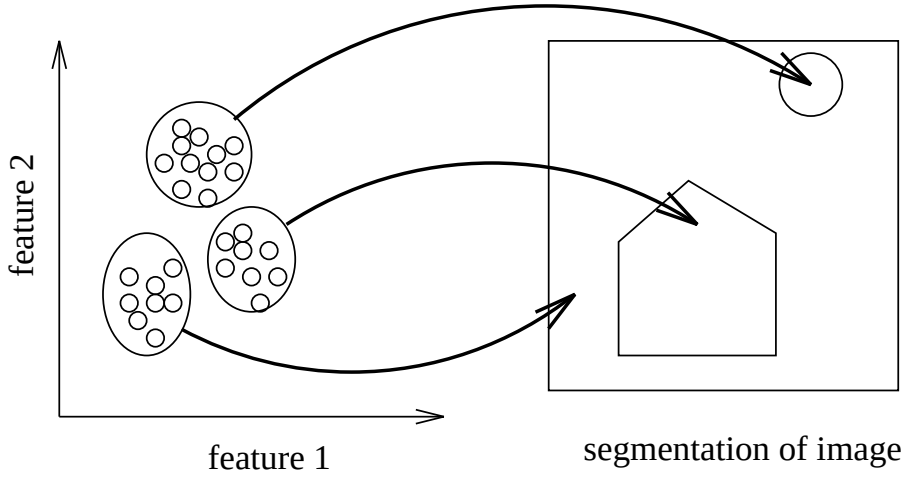

Figure 2: Blobs or clumps in a feature space – unsupervised partitioning.

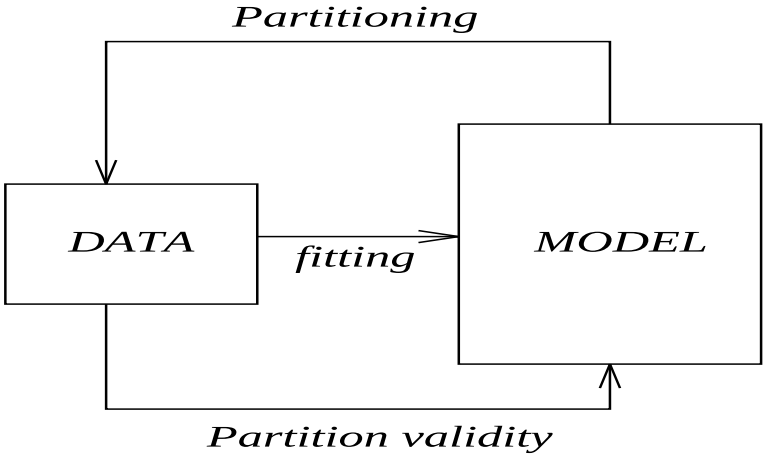

Figure 3:

method gives rise to a *dendogram* which, amongst other properties, details the number of structures obtained as a function of some merging threshold. With a threshold of zero the number of structures equals the number of data and a high threshold partitions the entire data set as a single cluster. Observation of the dendogram linkage may then be used to select an appropriate number of stuctures within the data (this is often based upon a partition's 'lifetime' – the range over which a partitioning is stable with respect to changes in the merging threshold). The major drawback of the hierarchical approach is that the entire dendogram is sensitive to (possibly erroneous) previous cluster merging i.e. data are not permitted to change cluster membership once assignment has taken place.

Partitional methods typically start with a data partitioning into a small number of clusters and increase the number of partitions into which the data is divided. The precise partitioning is performed so as to minimize or maximize some objective function. A datum is, furthermore, free to change its partition membership in such a scheme. The precise choice of objective function clearly has a very strong

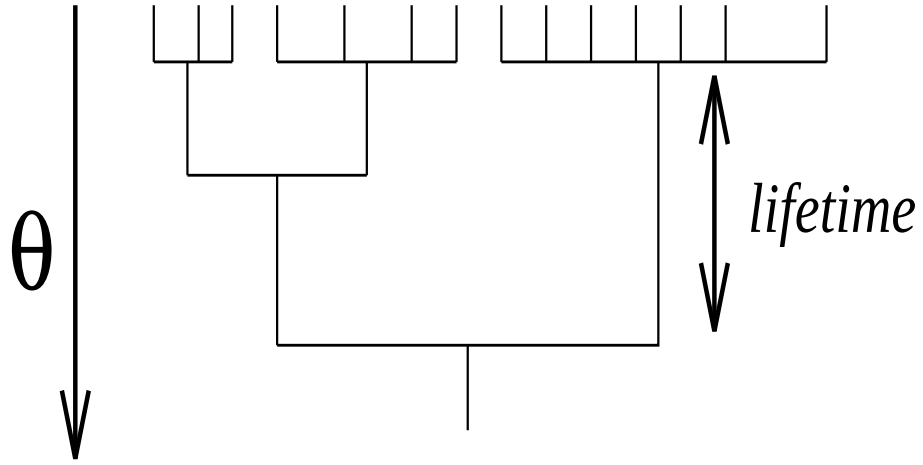

Figure 4:

bearing on the partitioning of the data, indeed the most popular, the total square error between $\mathcal{H}_K$ and the data, implies that the data will be modelled by the fitting of hyper-ellipsoidal clusters, and whether we may assume that clusters (if they exist) are multivariate Gaussian distributed is an open question for many data sets. Many methods for ranking the validity of data models of differing complexity (number of partitions) have been proposed. Most, however, rely (implicitly or explicitly) upon estimates of within- and between-cluster scatter matrices. The major problem with this approach, of course, is that if data do not conform to the assumptions made by the technique then the latter may impose structure on the data and not disclose the 'true' structure. This is also the case when clusters have widely differing numbers of members. The objective function may be improved by artificially splitting a cluster with a large number of members *more* rather than recognising one with a small number. The use of 'fuzzy' cluster methods (whereby a datum's membership may be distributed over many clusters) may be incorporated into the genre of partitional methods with relative ease and has proved popular.

Such an extension to the modelling process is certainly more representative of 'real' data. There are still, however, difficulties encountered. The 'optimal' number of clusters must be estimated, the location and shape of the clusters is invariably unknown and there may be a large variability in the number of data members in each cluster.

Maximum Likelihood mixture models: EM & K-means Algorithms

We consider the case of a data set, $X = \{\mathbf{x}_n\}, n = 1...N$ where $X \subset \Re^d$. Let the distribution of data in X form a probability density function denoted by $P(X)$. We may estimate this density function, in the limit, by considering it as a weighted combination of all possible data models, $\mathcal{M}$, each one of which is specified by a

parameter vector $\mathbf{w}^{\mathcal{M}}$

$$P(X) = \int_{\mathcal{M}} \int_{\mathbf{w}^{\mathcal{M}}} P(X \mid \mathcal{M}, \mathbf{w}^{\mathcal{M}}) P(\mathcal{M}, \mathbf{w}^{\mathcal{M}}) d\mathbf{w}^{\mathcal{M}} d\mathcal{M} \tag{1}$$

Such a complex specification is normally constrained such that we look only at a particular class of models, such as those with the property of universal approximation, the Gaussian mixture model (GMM) being a popular choice. We may then specify each GMM by a single parameter, K, which describes the complexity of the model (the number of Gaussian kernels). If, furthermore, we assume, as is common practise, that the probability distribution of the within-model free parameters, specified by $\mathbf{w}^K$, is dominated by a single, most probable, solution, $\mathbf{w}^K_{MP}$, then Equation (1) reduces to

$$P(X) = \sum_K P(X \mid K, \mathbf{w}^K_{MP}) P(K, \mathbf{w}^K_{MP}) \tag{2}$$

For ease of notation we assume that, for every model specified by K, dependence upon $\mathbf{w}^K_{MP}$ is implied. Bayes' theorem then states that (droppping the $\mathbf{w}$ terms)

$$P(K \mid X) = \frac{P(X \mid K) P(K)}{P(X)} \tag{3}$$

where the evidence term, $P(X)$, is given by Equation (2). As the number of Gaussian kernels, K, specifies the number of data partitions, we may use $P(K \mid X)$ as a partition validation measure.

Parameter Estimation

In previous chapters we have already looked at the Expectation-Maximisation (EM) algorithm for sucessive re-estimation of parameters e.g. in a mixture model and the formulation of the parameter updates follows this approach here.

The likelihood term in Equation (3), if we assume N independent data samples, may be written as

$$P(X \mid K) = \prod_{n=1}^{N} P(\mathbf{x}_n \mid K) \tag{4}$$

From Bayes' theorem for mixtures we may write the above as

$$P(X \mid K) = \prod_{n=1}^{N} \left[\sum_{k=1}^{K} P(\mathbf{x}_n \mid k) P(k) \right] \tag{5}$$

where $P(k)$ is the *a priori* probability of the k-th kernel in the GMM. The within-model parameter vector, $\mathbf{w}^K_{MP}$, however, still remains to be estimated for each model.

The EM approach ultimately seeks to maximise the logarithm of Equation (4), namely

$$\sum_{n=1}^{N} \log P(\mathbf{x}_n \mid K). \tag{6}$$

This maximisation is subject to the constraint that

$$\sum_{k=1}^{K} P(k) = 1. \tag{7}$$

and

$$\begin{aligned} P(\mathbf{x}_n \mid k, \mathbf{w}_k) &= P(\mathbf{x}_n \mid k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathsf{T}}\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k)\right] \end{aligned} \tag{8}$$

The auxilliary Equation in the EM update takes the form

$$Q' = -\sum_{n}\sum_{k} P_{old}(k \mid \mathbf{x}_n) \ln\{P_{new}(k) P_{new}(\mathbf{x}_n \mid k)\} \tag{9}$$

If we let the free parameters of each component, k, of the GMM be specified by a parameter vector $\mathbf{w}_k$ (i.e. the mean, $\boldsymbol{\mu}_k$ and covariance matrix, $\boldsymbol{\Sigma}_k$ for each k and the mixture priors $P(k)$) then combining Equations (5,6) and differentiating Equation 9 with respect to the parameters for the k-th component of the mixture gives:

$$\begin{aligned} \boldsymbol{\mu}_k^{new} &= \frac{\sum_{n=1}^{N} P_{old}(k \mid \mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} P_{old}(k \mid \mathbf{x}_n)} \\ \boldsymbol{\Sigma}_k^{new} &= \frac{\sum_{n=1}^{N} P_{old}(k \mid \mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})(\mathbf{x}_n - \boldsymbol{\mu}_k^{new})^{\mathsf{T}}}{\sum_{n=1}^{N} P_{old}(k \mid \mathbf{x}_n)} \\ P_{new}(k) &= \frac{1}{N}\sum_{n=1}^{N} P_{old}(k \mid \mathbf{x}_n) \end{aligned} \tag{10}$$

K-means

If we allow the set of posterior probabilities, $P(k \mid \mathbf{x}_n)$, to collapse to the set $P'(k \mid \mathbf{x}_n)$ such that

$$P'(k \mid \mathbf{x}_n) = \begin{cases} 1 & \text{iff } k = argmax_k\{P(k \mid \mathbf{x}_n)\} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

then the above maximum-likelihood (EM) update solution becomes the K-means approach with the caveat that normally the covariances and priors are not estimated as part of the optimisation - they are assigned *after* the means have converged. Like the ML approach, K-means may be implemented using batch or incremental algorithms. The latter invokes a simple adaption rate, η, such that at each iteration step, t and presentation of pattern $\mathbf{x}[t]$, the mean *closest* to $\mathbf{x}[t]$ is adapted via:

$$\boldsymbol{\mu}[t+1] = \boldsymbol{\mu}[t] - \eta(\boldsymbol{\mu}[t] - \mathbf{x}[t])$$

i.e. a stochastic gradient descent. An extension of K-means, the so-called fuzzy K-means, looks to update all the kernel means, not just the one closest to the pattern. Each mean is hence adapted, in a batch algorithm, using the same update procedure as EM, or for a stochastic algorithm, the adaption rate, η, of K-means is replaced by:

$$\eta'[t] = \eta P(k \mid \mathbf{x}[t])$$

i.e. adaption is governed by the membership to each kernel. A problem is that to estimate the posteriors (memberships) we need to have estimates for the covariances and priors. The value of the K-means approaches is mainly in their computational simplicity and hence speed.

Let us look at a simple example. Figures 5,6 shows the results of applying a simple segmentation of the (in)famous Iris data set. There are three types of iris and the features are measures of petal length and width.

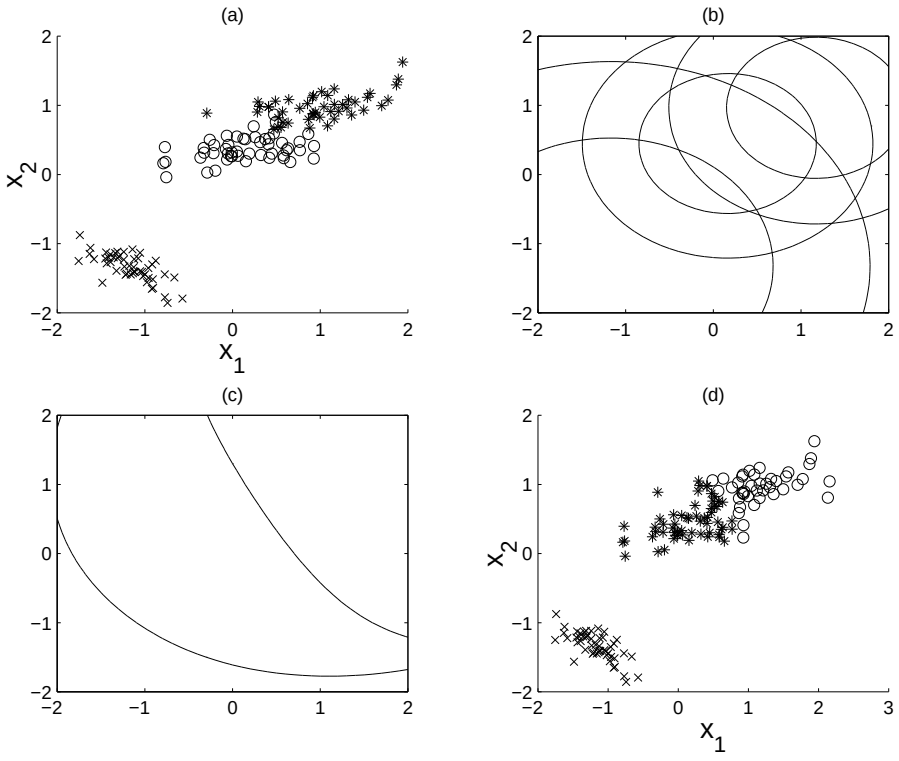

Figure 5: Partitioning of the Iris data using K-means.

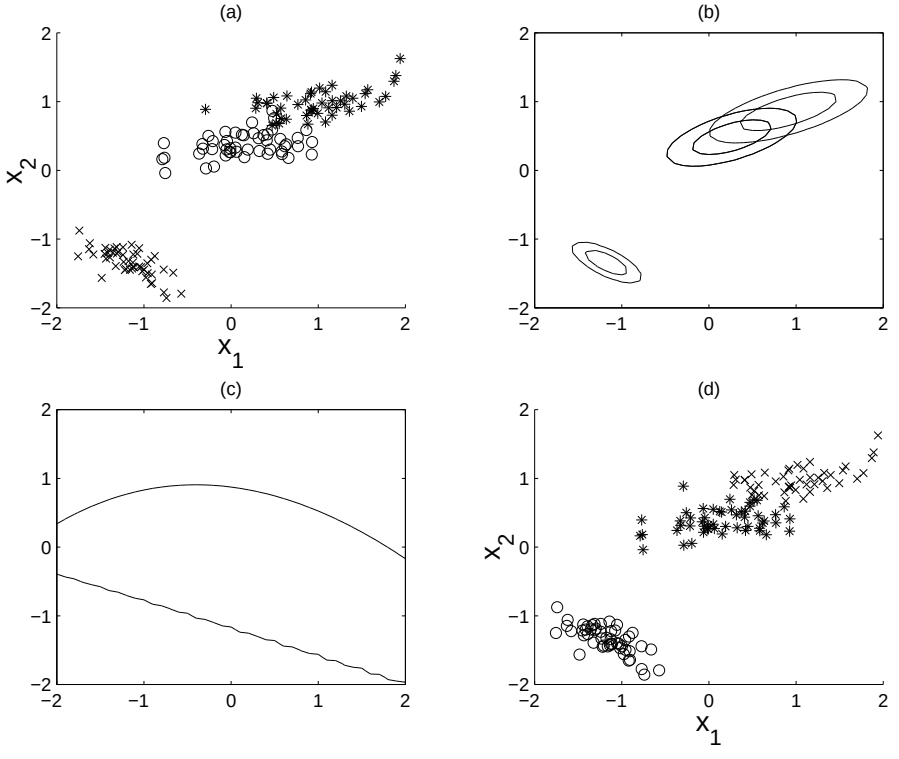

Figure 6: Partitioning of the Iris data using EM.

Cluster Validity

We must start by noting that the use of either the K-means or the ML approach imposes an implicit assumption of hyper-ellipsoidal clusters on the data model. Most cluster validity measures are based upon estimates of the kernel covariance matrices for a given model complexity (number of kernels in the GMM). This paper assesses four partition validity measures for the GMM, each of which is detailed in the following subsections. We note that all measures may be applied to both the ML and the K-means partitioning methods.

A comparison

The results from several clustering approaches are compared - all using the same model parameters (from the EM algorithm).

1. A Bayesian approach - defined by Roberts (1997) Within a Bayesian setting, the *evidence* of the data set may be obtained from the *marginal* integral:

$$P(X \mid M_K) = \int_{\mathbf{w}_K} P(X \mid M_K; \mathbf{w}_K) P(\mathbf{w}_K \mid M_K) d\mathbf{w}_K \tag{12}$$

where $P(\mathbf{w}_K \mid M_K)$ is the prior parameter probability density for the K-component model. For notational convenience, we will derive all results assuming a K-component Gaussian Mixture Model (GMM) and drop the explicit conditions on M_K in the above equation, thus Equation (12) is rewritten as

$$P(X) = \int_{\mathbf{w}} P(X \mid \mathbf{w}) P(\mathbf{w}) d\mathbf{w} \tag{13}$$

We denote $L(X \mid \mathbf{w})$ as the *log-likelihood* of the data given the parameters and the model i.e.

$$L(X \mid \mathbf{w}) = \ln P(X \mid \mathbf{w}) \tag{14}$$

and rewrite Equation (13) in the form

$$P(X) = \int_{\mathbf{w}} \exp\{-E(\mathbf{w})\} d\mathbf{w} \tag{15}$$

where the energy function, $E(\mathbf{w})$, is defined as

$$E(\mathbf{w}) = -L(X \mid \mathbf{w}) - \ln P(\mathbf{w}) \tag{16}$$

If we are to avoid the computational expense of numerically integrating a high-dimensional parameter space we may make the assumption that $E(\mathbf{w})$ has a local quadratic form (the posterior parameter distribution is Gaussian and sharply

peaked) around a *most-probable* state, represented via the most-probable parameter set, $\mathbf{w}_{MP}$. Expanding $E(\mathbf{w})$ as a Taylor series to second-order terms gives

$$\begin{aligned} E(\mathbf{w}) &= E(\mathbf{w}_{MP}) + (\mathbf{w}-\mathbf{w}_{MP})\left.\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_{MP}}\right|_{\mathbf{w}_{MP}} \\ &+ \frac{1}{2!}(\mathbf{w}-\mathbf{w}_{MP})^{\mathsf{T}}\left.\frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w}_{MP}^2}\right|_{\mathbf{w}_{MP}}(\mathbf{w}-\mathbf{w}_{MP}) \end{aligned} \tag{17}$$

Setting

$$\mathbf{H} = \left.\frac{\partial^2 E(\mathbf{w})}{\partial \mathbf{w}_{MP}^2}\right|_{\mathbf{w}_{MP}}$$

(so that $\mathbf{H}$ is the Hessian matrix of $E(\mathbf{w})$ evaluated at $\mathbf{w}_{MP}$) and noting that the first-order term of Equation (17) is equal to zero, we obtain the following from Equations (15,17)

$$P(X) = \exp\{-E(\mathbf{w}_{MP})\}\int_{\mathbf{w}} \exp\left\{-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^{\mathsf{T}}\mathbf{H}(\mathbf{w}-\mathbf{w}_{MP})\right\} d\mathbf{w} \tag{18}$$

which is of standard form and equates to

$$P(X) = (2\pi)^{N_p/2}|\mathbf{H}|^{-1/2}\exp\{-E(\mathbf{w}_{MP})\} \tag{19}$$

where N_p is the number of parameters in the model. Taking logarithms of Equation (19) and combining it with Equation (16) gives

$$\ln P(X) = L(X \mid \mathbf{w}_{MP}) + \ln P(\mathbf{w}_{MP}) + \frac{N_p}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathbf{H}| \tag{20}$$

This expression consists of two components. The first term, $L(X \mid \mathbf{w}_{MP})$, gets larger as N_p increases. The rest of the expression tends to *decrease* due to the Hessian component as the determinant of the $N_p \times N_p$ matrix can rapidly increase with N_p. The whole expression can hence be seen as finding a balance between fitting the data well and finding a simple model.

2. Fuzzy hyper-volume (FHV) : this looks at models with lowest total volume, defined via

$$V(K) = \sum_{k=1}^{K}\sqrt{|\mathbf{\Sigma}_k|} \tag{21}$$

3. Evidence density : this measure uses the FHV measure to penalise the data likelihood measure. It is defined as

$$\varrho(K) = \frac{L(K))}{V(K)} \tag{22}$$

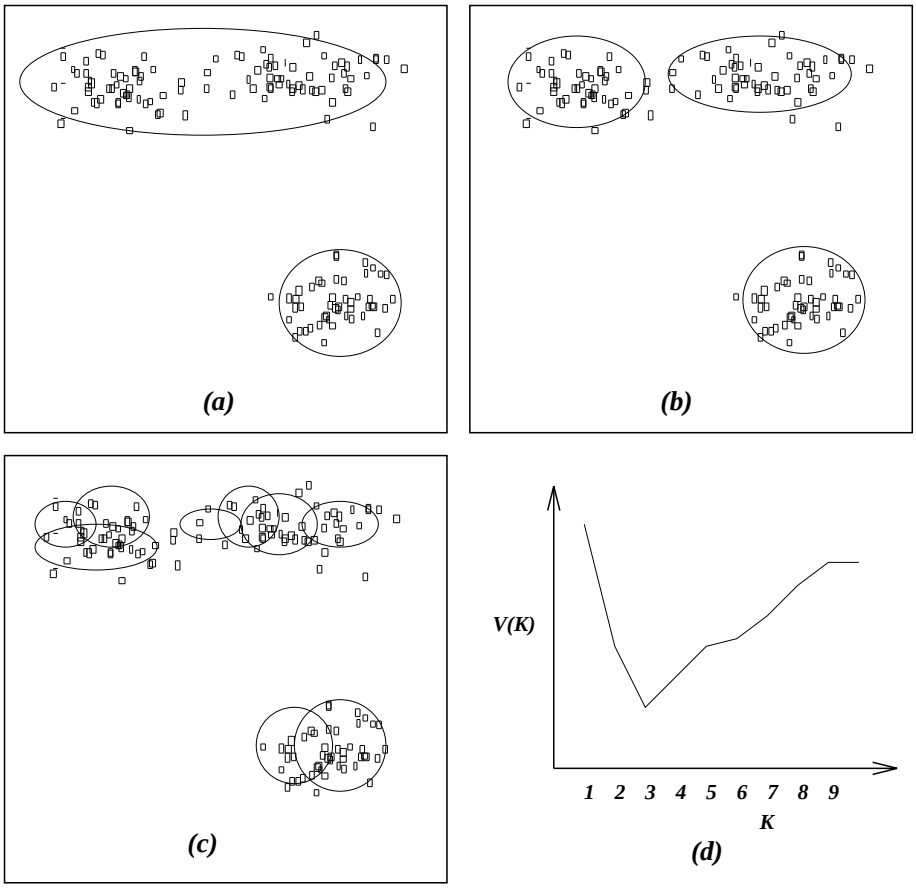

Figure 7: Fuzzy hyper-volume (FHV).

4. Minimum description length (MDL) : derived by Rissanen (1978) from an information-theoretic perspective, MDL is defined via

$$MDL(K) = -L(K) + \frac{1}{2} N_p(K) \log N \tag{23}$$

where, as before, $N_p(K)$ is the number of parameters in the K Gaussian model.

5. Partition coefficient (PC) : this is defined by Bezdek (1981)

$$PC(K) = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} P(k \mid \mathbf{x}^n)^2 \tag{24}$$

6. Minimum message length : this is defined in the work of Oliver *et al.* (Monash University in Australia) and is similar in essence to the Bayesian approach but derived in a different way.

Synthetic data 1

I first investigate the properties of these model selection methods on a simple two-dimensional toy problem. Data was generated from four Gaussians, each with a

common (isotropic) width, σ_{gen}, and means given by

$$\begin{aligned} \boldsymbol{\mu}_1 &= (0,0)^{\mathsf{T}} \\ \boldsymbol{\mu}_2 &= (2,\sqrt{12})^{\mathsf{T}} \\ \boldsymbol{\mu}_3 &= (4,0)^{\mathsf{T}} \\ \boldsymbol{\mu}_4 &= (-2,-\sqrt{12})^{\mathsf{T}} \end{aligned} \tag{25}$$

a total of 30 samples per Gaussian were taken, making $N = 120$. The width was set to $\sigma_{gen} = \{0.66, 1.0, 1.2\}$. The resultant data sets are presented in Figure (8)[2].

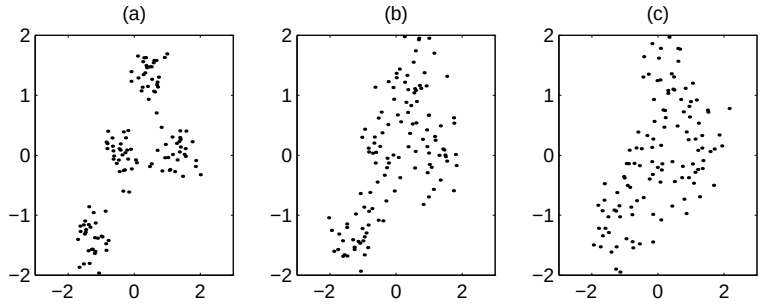

Figure 8: Synthetic data 1: (a) $\sigma_{gen} = 0.66$, (b) $\sigma_{gen} = 1.0$, (c) $\sigma_{gen} = 1.2$.

Results are presented over ten runs of the EM algorithm, each with a different random number seed. Figures (9,10 & 11) show the mean and ± 1 SD for $K = 2...7$ for:

- Bayesian method : we plot the negative log likelihood hence model support is higher for lower values of this quantity.
- Fuzzy hyper-volume (FHV).
- Evidence density.
- Minimum description length.
- Partition coefficient : we plot $1 - PC(K)$, hence model support is higher for lower values of this quantity.
- Minimum message length.

We see clearly that, in the simple case of $\sigma_{gen} = 0.66$, all methods give greatest support to the 'true' model. As σ_{gen} increases, however, all methods save for the Bayesian one, MML and MDL become erratic (higher variance) and make poor assessments.

[2]Note that the data sets are normalised such that they have a zero mean and unity intra-component covariance - this is why the data appear re-scaled in the plots.

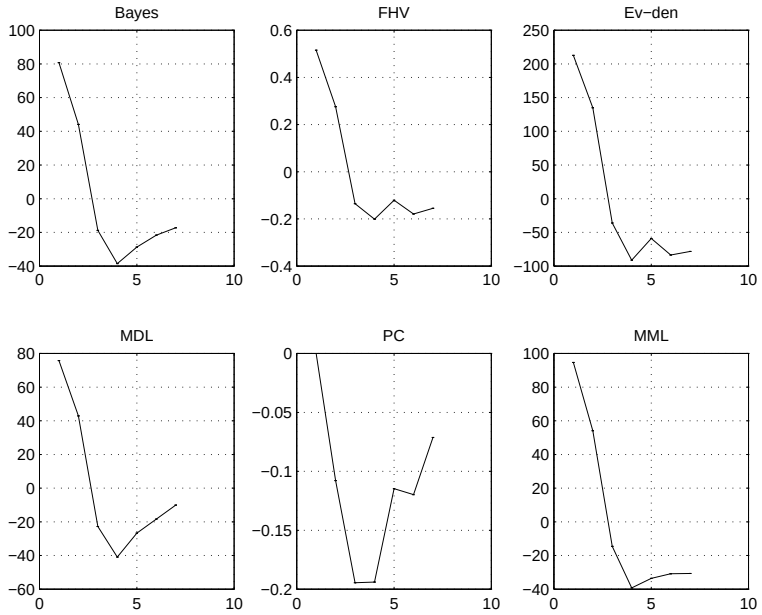

Figure 9: Model-selection functions for synthetic data set, $\sigma_{gen} = 0.66$.

Synthetic data 2

The second toy problem I show takes the form, once more, of data generated from four Gaussians. In this case, however, they are paired such that each pair has a common mean, i.e. $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_3 = \boldsymbol{\mu}_4$. I set $\sigma_1 = \sigma_3 = 1$ and $\sigma_2 = \sigma_4 = 5$. I sample 1000 points from this distribution, taking 250 from each Gaussian. Once more, the entire data set is normalised to zero mean and unit variance and this is shown in Figure (12). I apply the same methodology as the previous section and the results are shown in Figure (13). We note that both MDL and the Bayesian method clearly support two-cluster as well as the (true) four-cluster models. Other methods give the most support for two clusters, though there is a slight dip in the measures (more support) for $N_{clusters} = 4$. Note also that, due to large N, there is low variance in the plots - the ± 1 SD bars are barely visible.

Gaussian-mixture segmentation, some conclusions

- The Gaussian mixture model (GMM) is by far the most popular method. It is analytically simple and gives good results.
- The EM algorithm may be used to estimate the parameters.
- However, it looks for *Gaussian* clusters in data sets.
- The probable number of partitions may be estimated using simple methods which look to minimise *within-cluster* variance (so giving high likelihoods) and maximise *between-cluster* variance (so giving models with small volume) - methods such as the likelihood density measure do just this.

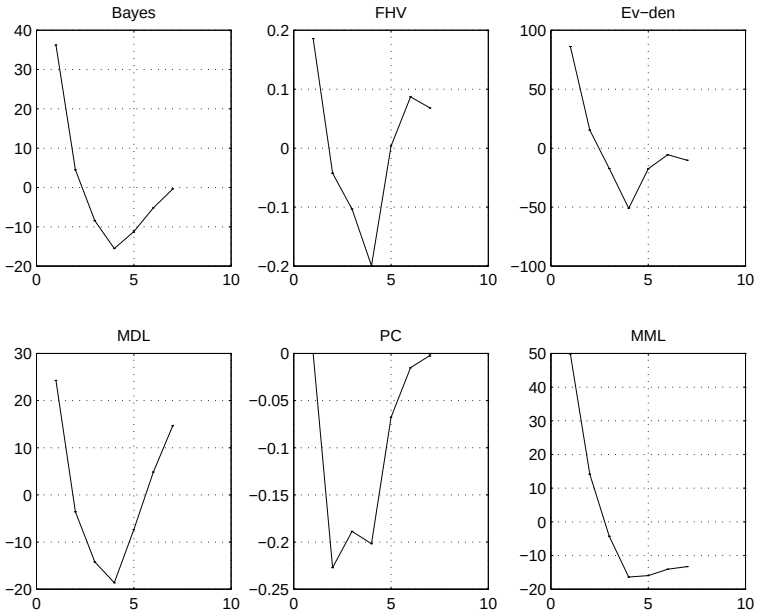

Figure 10: Model-selection functions for synthetic data set, $\sigma_{gen} = 1.0$.

- Methods of *model-order selection* based on *information theory* (incl. Bayesian statistics) are probably the best though.

A general problem with kernel-based methods is that each class is assumed to be *unimodal* and *Gaussian* (e.g.). This means that only *convex* data partitions may be found.

Scale-space approaches

Consider now the effect of blurring a data set with sucessively larger blurring filters. Peaks in the local data density gradually begin to be merged. This process is rather similar to finding a dendrogram for the data as the size of the blurring function is similar to a merging threshold. At some scales of blurring, though, the structure of the peaks in the data density remains fairly constant and this allows us to define a stable partitiong over some scale-space range. The larger this range, the more likely that we have discovered 'true' structure in the data. Figures 15, 16 show this process in operation on a simple data set. Note that Gaussian mixtures get it right also.

Figure 17 shows a difficult data set with three clusters. Gaussian mixture methods fail here, but the scale-spae approach succeeds.

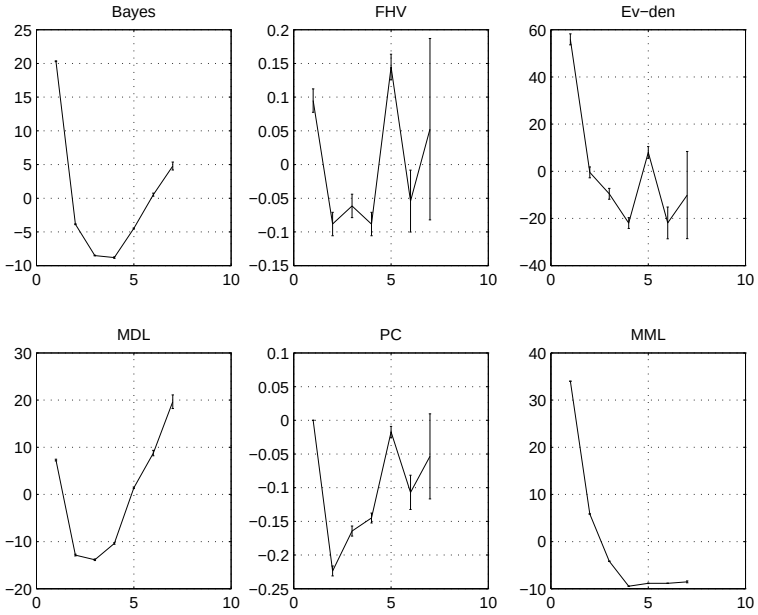

Figure 11: Model-selection functions for synthetic data set, $\sigma_{gen} = 1.2$. Note the graceful breakdown of both MDL and the Bayesian method.

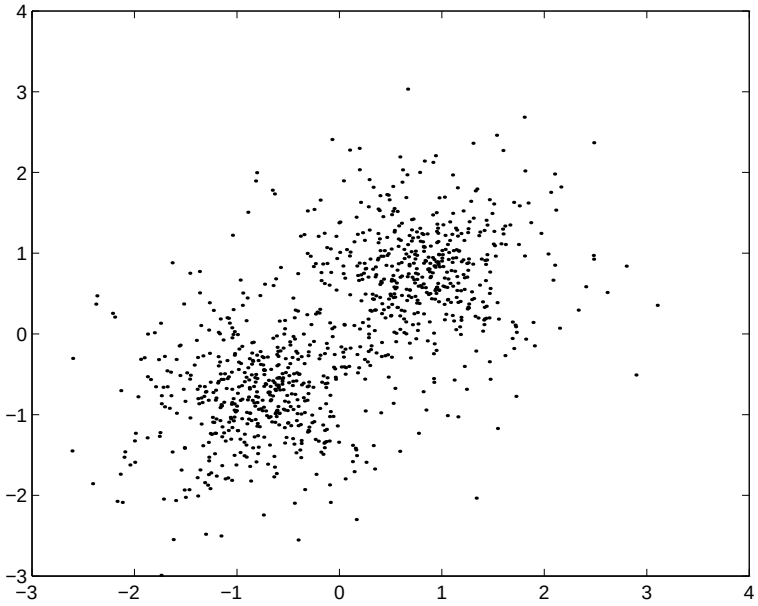

Figure 12: Synthetic data set 2: two pairs of joint-mean components (1000 data point).

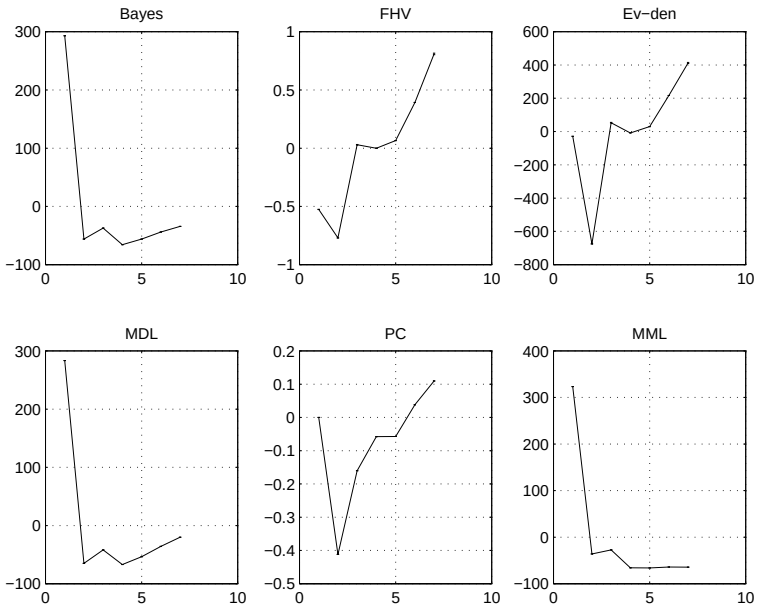

Figure 13: Synthetic data set 2: Model support results. Note that both MDL and the Bayesian method clearly support two-cluster as well as the (true) four-cluster model.

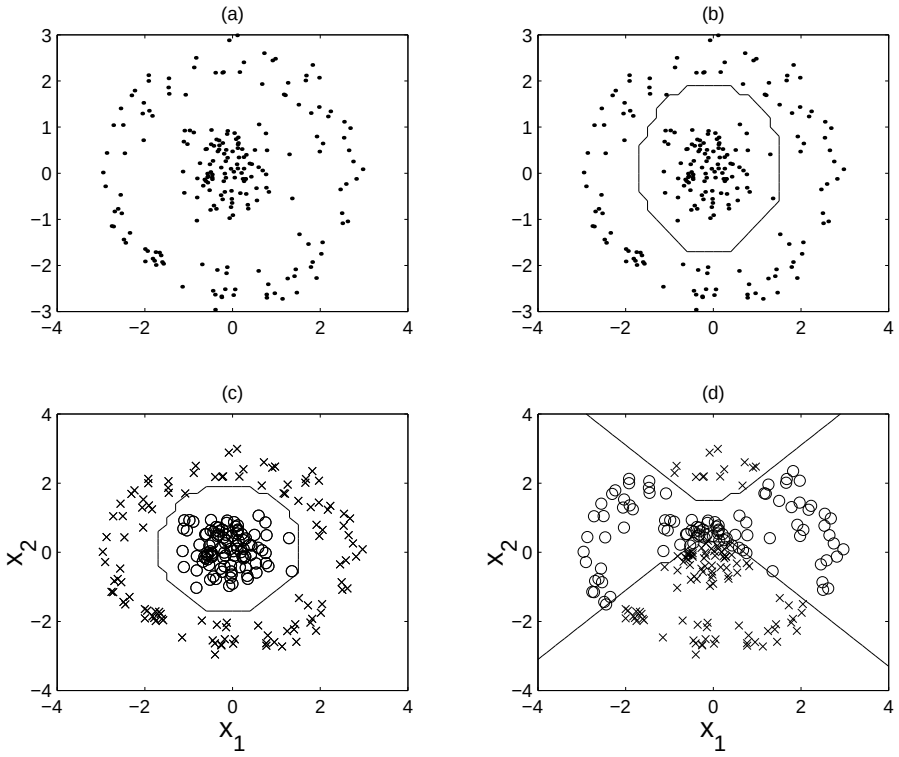

Figure 14: Partitioning of the Ring data set (a). The EM approach fails (d) and more spohisticated methods are needed to get the correct results (b,c).

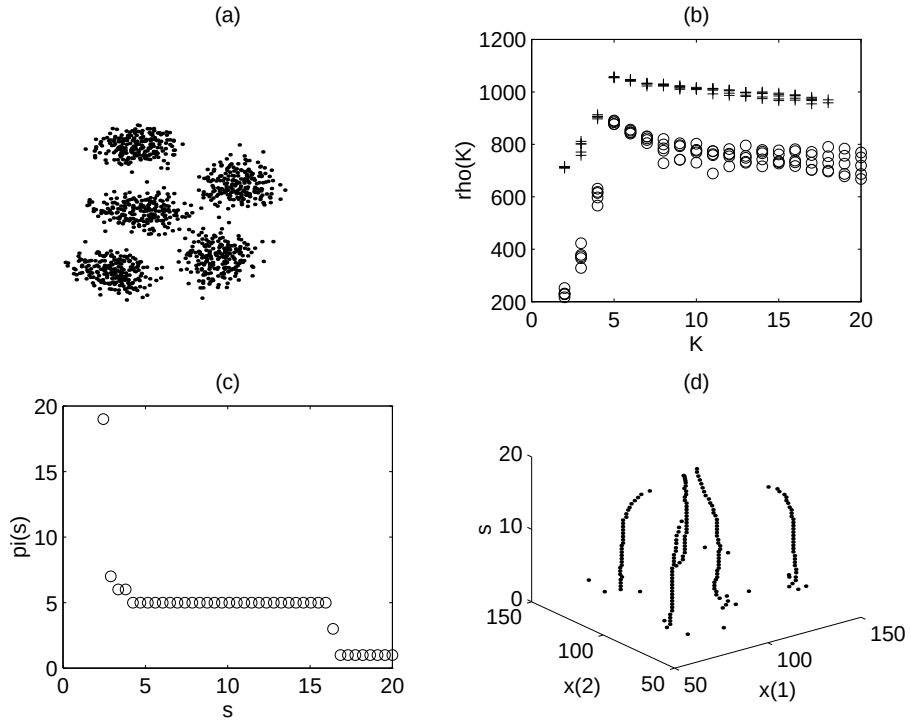

Figure 15: Sucessive blurring of a simple 5-cluster data set (a). K-means and EM with FHV penalties (b) and scale-space partitioning (c,d).

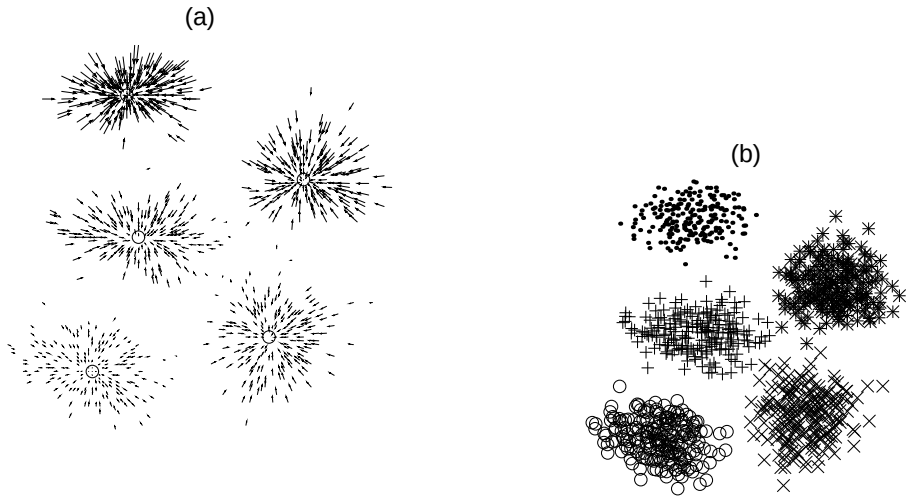

Figure 16: The gradients on the smoothed density surface let each datum point to its cluster membership (a) so giving a partitioning (b).

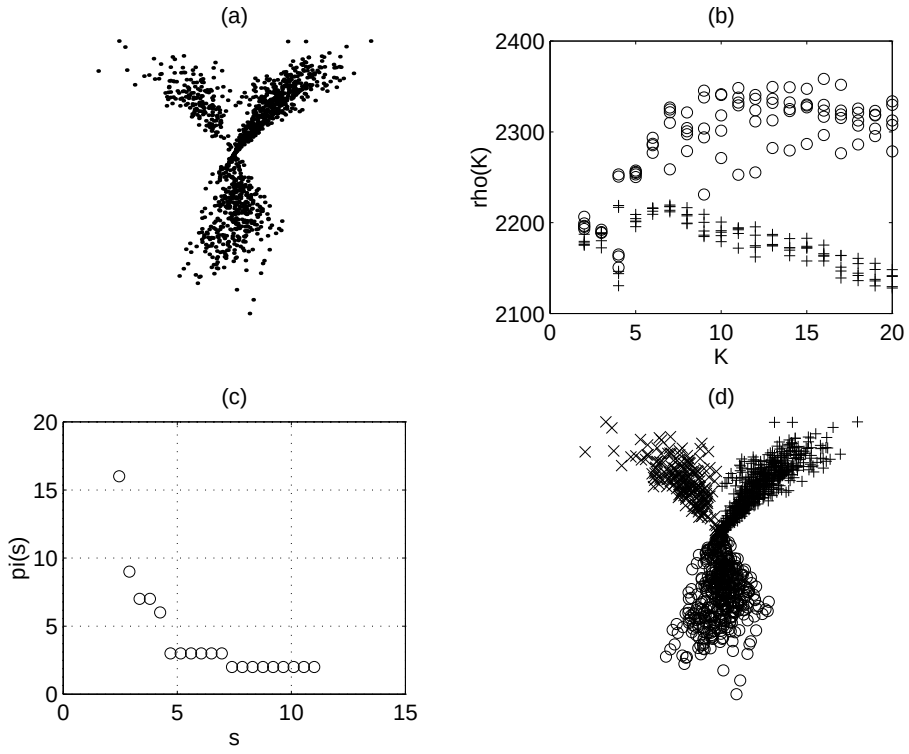

Figure 17: A difficult three cluster data set (a). GMM approaches fail (b), but scale-space methods work (c,d).

Topographic approaches

Topographic approaches are typified by the *Self-Organising Map* (SOM) originally proposed by Kohonen in the early 1980s. His inspiration came from the existence of maps in the brain, such as the motor-cortex homonculous in which physically nearby regions of neurons relate to physically nearby regions of the body.

The SOM attempted to produce a map (set of neurons or prototype vectors) which

- offered a 'natural' representation of data
- preserved topology from data to the map
- was un-supervised in its training (learning)

Consider a mapping $\phi : \mathbf{A} \rightarrow \mathbf{B}$ where $\mathbf{B}$ represents an array of output units labelled by the indices $[j, k]$, [3] and $\mathbf{A}$ an *input-space* with inputs represented by the vector set $\{\mathbf{x}\}$. Let each output unit have a weight vector associated with it, say $\mathbf{m}_{jk}(t)$, whose dimension is the same as that of the input vectors.

In order to obtain the mapping ϕ, the weight vectors, $\mathbf{m}_{jk}$, are initially given random values and the input data set, $\{\mathbf{x}\}$, is then presented, repeatedly and in random order, to the network. At each presentation of an input vector, $\mathbf{x}(t)$, we compute for each output unit the Euclidean distance, in input-space, between $\mathbf{m}_{jk}(t)$ and $\mathbf{x}_t$ and select the unit, $[j, k]^*$, with the minimum distance (figure). A neighbourhood $\mathcal{N}$ around $[j, k]^*$ is then defined [4] such that, within $\mathcal{N}$, weight vectors are adapted according to the following rule :

$$\mathbf{m}_{jk}(t+1) = \mathbf{m}_{jk}(t) + \alpha(t)[\mathbf{x}(t) - \mathbf{m}_{jk}(t)] \tag{26}$$

where $\alpha(t)$ is an adaption gain parameter, $0 \leq \alpha \leq 1$

For convergence to a smooth mapping, the size of $\mathcal{N}$ and the value of $\alpha(t)$ must be gradually decreased with time. These decreases are monotonic and governed by the following equations :

$$\alpha(t) = \alpha_o \exp\left[-\frac{t}{\tau_\alpha}\right] \tag{27}$$

where α_o is an initial adaption gain, normally $0.1 \leq \alpha_o \leq 0.5$, and τ_α is the time constant for the process. The value of $\mathcal{N}$ may decrease linearly, with time, from an initial value or may decrease in an exponential manner.

In most implementations of Kohonen's algorithm, including the one presented here, not all the units lying within $\mathcal{N}$ are updated equally and equation 26 is modified to :

$$\mathbf{m}_{jk}(t+1) = \mathbf{m}_{jk}(t) + \alpha(t)\beta(d, t)[\mathbf{x}(t) - \mathbf{m}_{jk}(t)] \tag{28}$$

[3] The array structure is two-dimensional in most of Kohonen's work for ease of visualisation

[4] One may represent the neighbourhood-boundary as a circle on the feature map, centred at $[j, k]^*$, with radius equal to $\sqrt{\mathcal{N}}$.

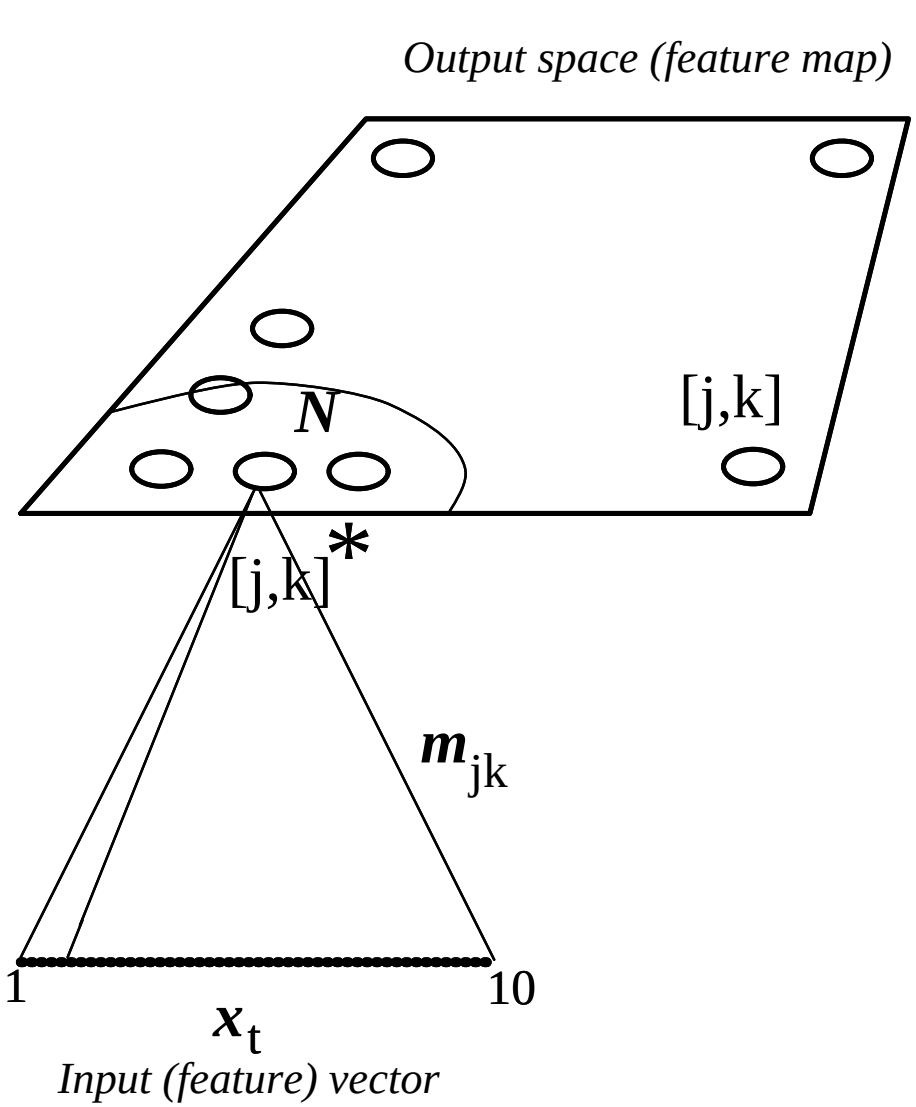

Figure 18: Schematic diagram of the self-organising network.

where $\beta(d,t)$ is a decreasing function of distance, d, such that $\beta(0,t) = 1, \forall t$, d being the Euclidean distance in *output-space* between $[j,k]^*$ and $[j,k]$. A popular choice is the truncated Gaussian.

Once the map is trained it may be used in lieu of the original space for

- visualisation of data
- determination of the probable number of sub-classes (clusters) in the data
- visualisation of data dynamics, as movement in the *high-dimensional* input space is mapped to a movement on (e.g.) a 2-D map.

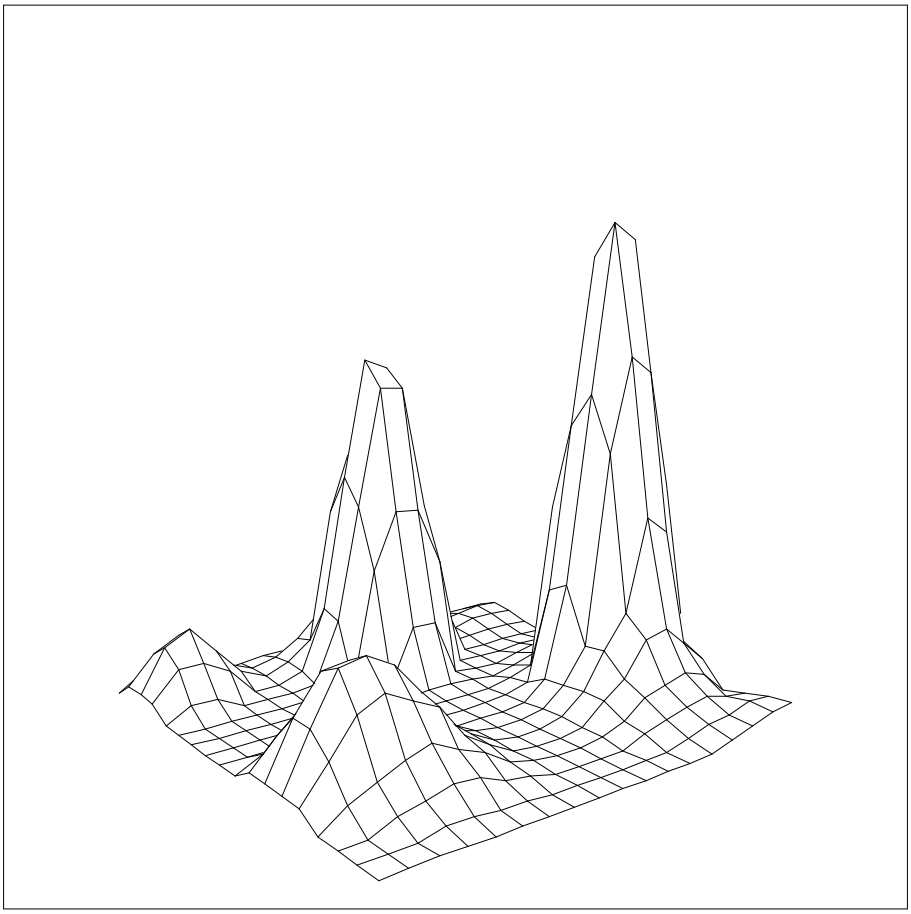

Figure 19: Response of trained SOM to a simple cluster set - the original data was 10-dimensional. Note the ease with which clusters are seen.