

## Chapter 3

# Cluster Analysis

### 3.1 Introduction

In this work, we are attempting to model consumer preferences and product attributes. To build models of individual consumers is of little value if one wants to generalize to new customers; conversely, to treat an entire population as a homogeneous collection is to assume that everyone has the same tastes, which is patently untrue. Cluster analysis can be seen as a compromise between these extremes, and an attempt to discover groups of people with similar tastes across a range of products. The principal purpose of clustering data is to find groups of consumers who can be successfully targeted with the same product.

More generally, clustering is an attempt to find groups of similar points within a set of data, such that members of the same group are similar to each other, while members of different groups are dissimilar. This requires definitions of similarity and dissimilarity, as well as a choice of the number of groups for which to search.

Cluster analysis is an example of unsupervised pattern recognition: before the study begins, the analyst does not know to which cluster each object belongs, doesn't know how many clusters there are, or even if there are any clusters at all. In supervised learning, the training data consists of pairs of records, with inputs matched to desired outputs. Such learning typically takes place by iteratively minimising the error, i.e. the deviation between the predictions that the model produces, and the true, target values. In unsupervised learning, there are no pre-specified outputs, and so no such error-minimisation can be defined. We can regard the data as being incomplete, with the target values (i.e. the class labels) missing. This is the assumption behind the EM algorithm, described later.

In the next section, we discuss some particular issues when clustering preference data, and list several stages of cluster analysis. In order to distinguish between similar and dissimilar points, we need some definition of distance, the subject of Section

3.3. We follow this with a discussion of several techniques for actually segmenting the data, focussing on  $k$ -means clustering and Gaussian mixture modelling, and compare linear and non-linear preference mapping techniques. In Section 3.6, we discuss several methods to discover the number of clusters present. The results of exploratory experiments highlight several issues, leading to novel approaches based on analysing the consistency of the results, and how two models that appear equally good can be usefully distinguished (Sections 3.8–3.9).

## 3.2 Clustering preference data

The beverage preference data set,  $P_b$  consists of the responses of 450 subjects to each of 20 samples of beverages. The subjects were asked to express a preference on a discrete scale of 1–9 for each sample, one being a low preference.

For this analysis, the data set defines a 20-dimensional space, with each dimension being the preference score for a different product; each point in this space represents a unique combination of scores. Any cluster found within this space would represent a group of subjects with a similar set of preferences over the samples. Each person is therefore represented as a vector in this feature space. Equivalent spaces can be defined for the meat preference data ( $P_m$  has 16 features, defining a 16-dimensional space) and the vegetable preference data ( $P_v$  has 17 features, defining a 17-dimensional space).

In order to carry out any cluster analysis, a number of choices must be made. Milligan [82] describes seven stages to cluster analysis, summarised here:

**Clustering elements** Choose representative data to analyse.

**Clustering variables** Choose attributes which will define the clusters.

**Variable standardisation** Normalise or standardise data, if required.

**Measure of association** Choose a similarity measure.

**Clustering method** Choose a clustering algorithm.

**Number of clusters** Determine the number of clusters for which to search.

**Interpretation, testing and replication** Interpret the results in context, test their significance and repeat the analysis with new data.

Not all of these choices are relevant to the current study. For example, the data sets have already been collected by Unilever, so there is no need to select which data to analyse. Similarly, all the data are on the same scale, so no standardisation is required. The most interesting areas are therefore feature selection; the measure of similarity; the

clustering method; determining the number of clusters; and interpreting and validating the results.

As mentioned, the data is discrete, which potentially leads to a considerable overlap between observations. For example, the beverage preference data shows that of the 450 responses to one particular sample, 315 people rated it just one out of nine. Similarly, 313 people rated another sample as a maximum 9 out of 9. Although every possible score was given at least once for every product, in several cases, only one person rated certain samples with certain scores. All in all, this means that the data is very sparse, with very uneven distributions across each dimension. This makes estimating covariance matrices (for example) very hard, as discussed in Section 3.4.5.

### 3.3 Similarity and distance measures

To find clusters of similar objects, one must first define the term “similar.” Within the data space of food preferences, similarity can be defined with a distance metric: similar objects are closer than dissimilar objects. We therefore require a definition of distance, of which there are many [88]. By far the most common way to measure distances is the Euclidean definition, where the distance between points  $a$  and  $b$  in  $m$ -dimensional space is given by:

$$d(a, b) = \sqrt{\sum_{i=1}^m |a_i - b_i|^2}$$

This is a special case of the Minkowski distances:

$$d(a, b) = \sqrt[p]{\sum_{i=1}^m |a_i - b_i|^p}, p \geq 1$$

(from Murtagh, [88, p.4].)

Although  $p$  can take on any value of one or more, only three values are used in practice: 1, 2 and  $\infty$ . These are known respectively as the “city-block” (or “Manhattan”) distance<sup>1</sup>, the Euclidean distance, and the Chebyshev (or “maximum coordinate”) distance. A further alternative is the Mahalanobis distance:

$$d(a, b) = (a - b)' \Sigma^{-1} (a - b),$$

where  $\Sigma$  is the covariance matrix of  $a$  and  $b$ . This matrix is difficult to estimate accurately given very few records, so we will not consider this metric further.

Given that the Euclidean distance is so widely used, and that many standard tech-

---

<sup>1</sup>This is known as the Hamming distance, if the variables are binary.

niques rely on it, one must carefully justify the use of any alternative measure. While it may be sensible to measure the every day objects using Euclidean distances, there is less reason to suppose that a 20-dimensional space of beverage preferences should be treated so.

In the case of food preference data, the vectors are all integers, bounded between one and nine, which is clearly not typical of continuous measurements in other fields. However, it is reasonable to assume that a preference score of 7.5 would be half way between scores of 7 and 8, even if no consumer actually gives such a fractional score. Stone and Sidel describe several studies using the standard nine-point hedonic scale, and conclude that the data “should not be assumed to violate the normality assumption” [121, p.85]. Bishop [15, p.208] derives the sum-of-squares error function (equal to the square of the Euclidean distance) by assuming a Gaussian distribution. Thus with the lack of any clear motivation to break with tradition, this work uses Euclidean distances.

## 3.4 Clustering methods

We now consider several approaches to clustering, starting with preference mapping, widely used in product design analysis, and introduce a non-linear variation. We then turn to  $k$ -means clustering and Gaussian mixture models, both widely used in data analysis.

### 3.4.1 Preference mapping

As we mentioned in Chapter 1 (Section 1.2.5), there are two forms of the widely used preference mapping technique, namely internal and external mapping [78]. We now describe these, and give several examples on both real and synthetic data sets.

Internal preference mapping projects the consumers’ preference scores into a low dimensional space, using principal components analysis (PCA) [52]. This dimension reduction can help group consumers, to show who likes which products. This is because with a fixed amount of data, fewer parameters are needed to specify a cluster model as the dimension is reduced. Dimension reduction also allows the data to be visualised more easily. The same PCA dimensions can then be correlated with the sensory attributes to determine *why* people like the products. In contrast, external preference mapping combines the two data sets from the outset. First, the sensory data is projected onto a lower dimensional space, typically using the first two or three principal components. Then for each consumer, a least-squares model is built and used to predict preferences from the projected sensory scores. The least-squares model is usually a linear or quadratic multivariate model. In the studies described below, a simple linear model is used. The regression analysis identifies each consumer’s drivers,

i.e. the features that make the consumer like or dislike each product. Thus we find  $f_i : S_{PC} \rightarrow p_i$ , given sensory data  $S$  transformed to the first few principal components to give  $S_{PC}$ , and preference  $p_i$  for each consumer  $i$ .

If two principal components are used, then a simple plot shows where each consumer lies in terms of which combination of sensory features affect their preferences. A single vector is drawn for each consumer, with the direction determined by the parameters of the least-squares model. (The co-ordinates of the vector are the two parameters of the linear regression model, with zero-offset.) The lengths can be normalised (so that every vector is of length one) or scaled according to accuracy (so that long vectors correspond to well-modelled consumers). The scale used in this case is the correlation between the individual consumer's preferences and the predictions made by the corresponding linear model [27].

Figure 3.1 shows a normalised external preference map for some artificial data<sup>2</sup>. Each vector has been normalised to unit length. This shows three groups of consumers, with vectors pointing roughly north-west, east, and south-west. The vectors were then clustered by applying  $k$ -medians clustering to their angles; the centre of each resultant cluster is shown as a dotted line ending in a circle. The  $k$ -medians algorithm uses data points as cluster centres, and iteratively assigns each data point to the nearest centre, and then re-selects the centres as being the points nearest the median of each cluster [26]. This is closely related to  $k$ -means clustering, described in Section 3.4.4, but does not require a definition of the mean of a set of points, which is difficult to define in the discontinuous space formed by measuring angles around a circle<sup>3</sup>.

Figure 3.2 shows a preference map for the same data, with each vector scaled using the correlation described above. Note that the only difference between these two maps is the length of each vector. The longer vectors in the south-west group show that these consumers are better modelled than the other two groups.

Figures 3.3–3.4 show the corresponding normalised and scaled plots for the meat data. The normalised map shows the vectors are spread almost uniformly around the unit circle, suggesting no clear clusters exist in this two-dimensional transformed space. The scaled map (Figure 3.4) also shows that most consumers preferences are not accurately predicted by the regression models used, indicated by the fact that most of the lines are very short compared to the longest.

---

<sup>2</sup>The artificial sensory data is uniformly distributed, of size  $p \times f$  with  $p = 15$  products and  $f = 60$  features. The preference data consisted of three groups of 50 points each, giving 150 consumers. Each group was drawn from a Gaussian distribution in a 15-dimensional space, each with a standard deviation of 10. The Gaussian component centres were 5 units apart in each dimension. This sample is multiplied by the sensory data to simulate the effect of sensory values on preferences, and it is this product which is used to create the preference map.

<sup>3</sup>The difficulty is caused by the periodicity of moving around a circle. For example, the distance between  $\alpha$  and  $2\pi - \alpha$  is  $|2\alpha|$ , and not  $|2\pi - 2\alpha|$ .

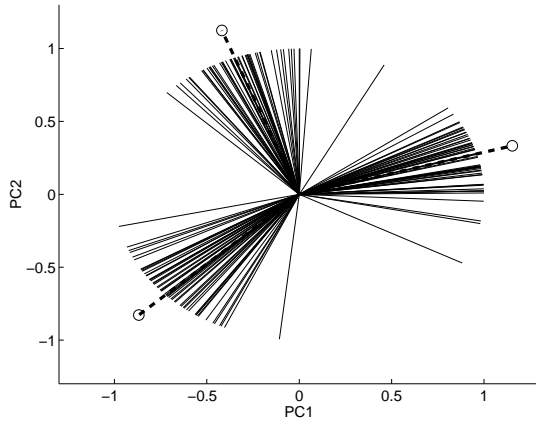


Figure 3.1: Normalised preference map, artificial data consisting of three clusters

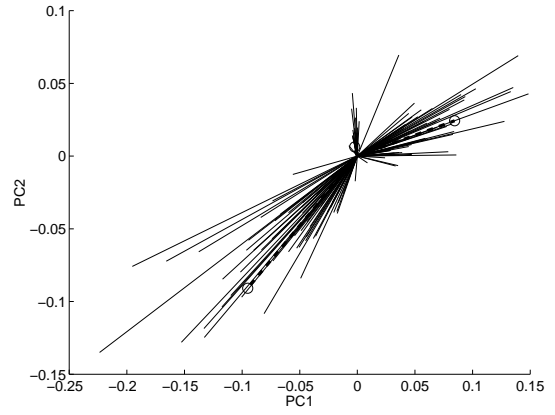


Figure 3.2: Scaled preference map, same artificial data as in Figure 3.1

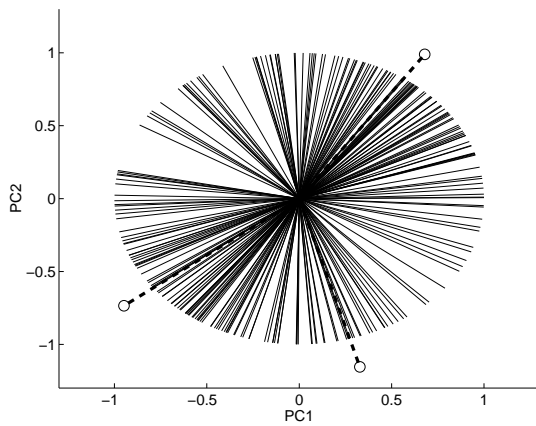


Figure 3.3: Normalised preference map, meat preference data

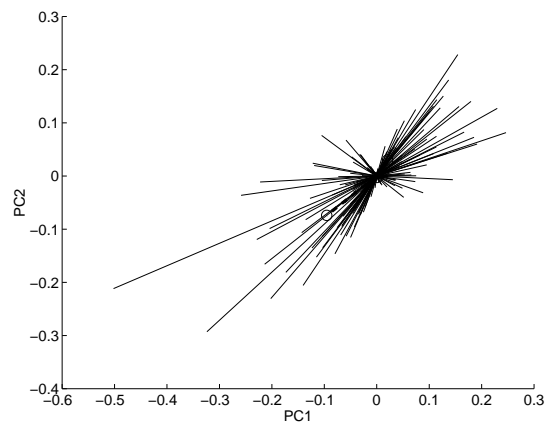


Figure 3.4: Scaled preference map, meat preference data

Given high dimension data  $\mathbf{X}$ , construct a low dimensional approximation  $\mathbf{Y}$ :

1. Find the neighbours  $\mathbf{X}_j$  of each point  $\mathbf{X}_i$ .
2. Compute weights  $W_{ij}$  that best reconstruct each point  $\mathbf{X}_i$  from its neighbours, minimising a quadratic error term.
3. Compute vectors  $\mathbf{Y}_i$  that are best reconstructed by weights  $W_{ij}$ .

Figure 3.5: Locally linear embedding algorithm, from Saul and Roweis [112]

### 3.4.2 Locally linear preference mapping

Traditional preference maps combine PCA to reduce dimensionality, regression to identify drivers, and clustering to group people with similar drivers together. PCA is linear in the sense that the derived features are linear combinations of the original features, which raises the question as to whether a non-linear combination would be more useful. Roweis and Saul [109] describe “locally linear embedding” (LLE), which combines local linearity with global non-linearity, and performs feature extraction and dimension reduction. The algorithm is in three stages, as shown in Figure 3.5. LLE preserves the distances between points within each local neighbourhood, rather than preserving the distances between all points, as attempted by (for example) multidimensional scaling [111].

The assumption behind LLE is that the data lies on or close to a smooth, non-linear and low dimensional manifold embedded within a high dimensional space. In food design, this high dimensional space is the sensory feature space. A low dimensional manifold here assumes that not all combinations of sensory feature values are plausible. (Perhaps no apple could be both very sweet and very crunchy.) Modelling this manifold using a locally linear (but globally non-linear) model suggests that in any small region of the sensory feature space, the relationship between these features is linear. Thus even if there is no global correlation between sweetness and crunchiness of apples, in one narrow range of sweetness values, the relationship to crunchiness could be linear.

We can use LLE to produce a low dimensional approximation to the sensory data, and then perform regression for each preference panellist, in the same way as the external preference mapping described above. This will allow us to produce a non-linear preference map, which is constrained to be locally linear. This limits the complexity of the model, and therefore reduces the risk of overfitting present when more general non-linear models are used in the high dimensional sensory space.

Two factors can be varied in LLE: the size of the neighbourhood used in calculating the weights, and the number of dimensions into which the data are projected. Given that the algorithm is deterministic, we need not repeat the experiments for each setting.

Data source	Locally linear embedding			Principal components analysis	
	Dimensions	Neighbours	Error	Dimensions	Error
Meat	1	7	2.47	1	2.61
Vegetable	2	7	2.56	1	2.21
Beverage	5	4	1.73	2	1.79

Table 3.1: Feature extraction results: LLE vs. PCA

For each type of food, we use LLE to project the sensory data onto  $l$  dimensions, with  $1 \leq l \leq 10$ . In each case, we vary the neighbourhood size, with  $1 \leq k \leq 15$ . For each setting, the leave one out cross validation error (Section 2.5.1) is calculated for every preference panellist, and the mean is calculated. Table 3.1 shows the results, comparing LLE preference mapping with conventional PCA preference mapping, and picking the model with the lowest test error in each case. Although LLE tends to use slightly more derived features (dimensions), the error scores are very similar, with LLE achieving marginally lower errors on two out of the three data sets. This suggests that any non-linearity in the data that LLE is successfully modelling is insufficient to significantly improve the final error scores. This could mean that the underlying distribution is linear; or that there is insufficient data to accurately account for the non-linearity; or that LLE is not powerful enough to exploit the non-linearity present. In both cases, the number of dimensions selected increases as the number of preference records available increases, suggesting that the smaller data sets may not provide enough evidence to justify building complex models.

### 3.4.3 Preference mapping summary

McEwan describes preference maps and offers the caveat that they tend to be “used for understanding and direction, not prediction” [78, p.79]. In this work, we are using predictive accuracy as our prime measure of model quality, limiting the usefulness of preference maps generally.

Scaled (i.e. non-normalised) preference maps can also be used to detect outliers: we could choose to remove consumers who are not well modelled by the preference map, i.e. those with short vectors in Figure 3.4. The question of how short is short is left to Chapter 4, which considers outlier detection.

A preference map is one type of *biplot* [55] — an attempt to display multivariate data to show relationships between samples and between measurements of those samples. This usually defines some form of dimension reduction, along with a graphical display. Besides a standard scatterplot, the most common examples of biplots are PCA and multidimensional scaling (MDS), both of which have linear and non-linear variants. Many variations of MDS are in common use [32] such as Kruskal’s “stress” minimisation

Given  $n$  data points and a integer  $k$ , do:

1. Randomly assign all  $n$  points into  $k$  groups;
2. Assign the centre of each group to the mean position of its members;
3. Reassign each point to its nearest group centre;
4. Repeat steps 2-3 until convergence.

Figure 3.6:  $k$ -means clustering algorithm

function and Procrustes analysis.

Non-linear biplot methods are inevitably harder to interpret than their linear counterparts, and as interpretability is one of our desired criteria, non-linear methods should be treated with caution. Kaski [71, p. 19] describes how principal curves (a common form of non-linear PCA) are, in practice, equivalent to self-organizing maps (SOMs), themselves very similar to the  $k$ -means clustering algorithm that we will discuss in the next section. The results presented in the previous section showed no clear benefit for either linear or non-linear preference mapping. Whether alternative biplot or MDS methods would in fact show consistently better results is a possible subject for future work.

As the name suggests, preference mapping (whether based on PCA or LLE) and biplots generally, emphasise visualisation of the data, by producing two- (or possibly three-) dimensional maps. However, if the intrinsic dimensionality of the data is — for example — four, then vital information will be lost from the map. In Chapter 2, we described many ways to perform feature selection, and we argue that it is preferable to other forms of dimension reduction due to ease of interpretation and the identification of drivers.

#### 3.4.4 $k$ -means clustering

The  $k$ -means clustering method assigns the data into  $k$  clusters, using the iterative algorithm shown in Figure 3.6. Duda and Hart describe this under the name “basic Isodata” [38, p.201], and it is also known as the Lloyd-Max algorithm [72].

Each iteration reduces the sum of the squared within-groups distances until convergence to a local — but not necessarily global — optimum. If the original random group assignment is very poor, then the final clustering solution will also tend to be poor. Because of this,  $k$ -means clustering is sometimes used in conjunction with another technique. For example, Fayyad et al. [44] describe repeatedly drawing samples from a data set, perform clustering on these samples, and then initialise the main clustering routine

to the average of the cluster centres of the samples. However, Meila and Heckerman [80] compare several such initialisation techniques empirically, and show that none of them consistently showed improvements compared to random initialisation. In this work, we initialise the centres by randomly selecting  $k$  data points as our centres, and assign each point to the nearest centre.

The clusters found by  $k$ -means analysis are spherical, and can only be found in continuous spaces. This is because we are minimising the sum-squared error, which is the Euclidean distance and only defined for continuous spaces<sup>4</sup>. Having estimated the cluster centres, we can classify a new data point as belonging to the cluster with the nearest centre.

Variations of the technique include:  $k$ -medians, which as the name suggests uses the median instead of the mean when estimating group centres, and was used in Section 3.4.1; fuzzy  $c$ -means, which uses fuzzy logic to allow each point to belong to more than one group; and adaptive resonance theory (ART), in which a neural network dynamically creates “prototype vectors” (similar to  $k$ -means centres) to represent the data [24].

A related but more flexible method is mixture modelling, described in the next section.  $k$ -means clustering is sometimes described as a “simple approximate” version of Gaussian mixture models (Duda and Hart [38, p.201]). As Kearns et al. show, this is not necessarily true [72]. Bishop shows that as the component widths (defined by the covariance matrices) tend towards zero, then the mixture model solution tends towards the  $k$ -means solution [15, p.190]. We shall return to this issue later in Section 3.4.6.

### 3.4.5 Gaussian mixture models and the EM algorithm

A Gaussian mixture model (GMM) is, as the name implies, a weighted sum of functions with normal distributions. Each component is defined by three sets of parameters: the means, the covariance matrices and the mixing proportions. They can be used to describe data, with the assumption that the data have arisen from populations mixed in certain proportions. McLachlan and Basford [79] describe their application to clustering.

Usually, the parameters of a mixture model are estimated using the Expectation-Maximisation (EM) algorithm, defined by Dempster, Laird and Rubin [35] and outlined in Figure 3.7. We are assuming that each data set is generated from a mixture of Gaussians, but that the information specifying which data point came from which component is missing. It is this cluster membership that the EM algorithm aims to discover.

---

<sup>4</sup>Other distance measures can be defined for  $k$ -means, which would lead to different shaped clusters.

- |   |
|---|
| <ol style="list-style-type: none"> <li>1. Initialise model parameters, producing the first model;</li> <li>2. Compute posterior probabilities of the data assuming the current model;</li> <li>3. Re-estimate model parameters using the posterior probabilities;</li> <li>4. If model is unchanged then terminate, else go to step 2.</li> </ol> |
|---|

Figure 3.7: General EM algorithm

Fraley and Raftery [48] describe a number of different ways of defining Gaussian clusters. They define the cluster density of a single component thus:

$$f_k(x|\mu_k, \Sigma_k) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right\}$$

This equation gives the density of the  $k^{th}$  component at position  $x$  with centre  $\mu_k$  and covariance matrix  $\Sigma_k$  in  $p$ -dimensions. Fraley and Raftery go on to describe different forms that the covariance matrix can take, and a geometric interpretation of these parameterisations. Each Gaussian component can be spherical or elliptical, of fixed or variable volume, and if elliptical, may be of fixed or variable shape and fixed or variable orientation. According to Fraley and Raftery [48], this gives six forms of GMM, determined by the nature of the covariance matrix. Having estimated the parameters associated with the mixture model, we can classify a new data point as belonging to the component that has the maximum likelihood score for that data point. I.e. the class  $c$  of point  $x$  given  $k$  mixture components defined by density functions  $f_i$  is  $c = \max_i f_i(x)$ .

During the EM algorithm, there is a danger that one or more covariance matrices may “collapse”, leading to the rank deficient case. The elements on the leading diagonal tend towards zero as fewer points are modelled closely by the component, so the likelihood scores of the points remaining within the cluster tend towards infinity. The implementation used here checks for this, and adds a small value to the leading diagonal if required. This forces a small Gaussian component to be slightly rounder and larger, and is a form of ridge regression [105, p.108].

Identifying very small elements on the leading diagonal could guide further analysis. Such points mean that the consumers assigned to this Gaussian component have a very low variance in their preferences for one product, and this consistency of opinion could be used to weight regression errors. If everyone agrees on one preference score for one product, then it is important to predict that score accurately, where as if everyone disagrees then it is less important to predict their mean preference. This approach is the subject of future work (Section 5.3.4).

Let us consider what effect the shape of such Gaussian components has on modelling

preferences for some hypothetical apples. The variance in one dimension is a measure of how much in agreement all the subjects in one cluster are regarding that apple. For example, members of one cluster may all agree that a Granny Smith apple is delicious, but may have a wider range of opinions regarding Cox's Orange Pippins. An alternative approach is to view each cluster as representing a prototypical apple eater, so the variance in each dimension represents how strongly the subject feels about the corresponding apple. A narrow variance represents a very strong opinion (regardless of the cluster mean, which is the actual preference), whereas a wide variance represents a less firm opinion.

A spherical cluster implies that the variance in preference scores across all apples is identical, for members of that cluster. This is a very strong assumption, suggesting that all preferences are of equal conviction.

A fixed-orientation ellipse allows the width of the cluster to vary across the range of apples, because a separate parameter is used to define the variance in each dimension. However, the variance is constrained to be parallel to each axis (the fixed orientation), with no correlation between dimensions. The alternative variable-orientation ellipse is free to form any angle with any axis. In order to choose between these options, we must ask if preference scores for each apple are independent of preferences for other apples.

Suppose Mr Doe comes to like Granny Smith's more than he used to. Then his point in the apple-space will move slightly along the "Granny Smith's" dimension, towards the high preference end. Can this cluster be parallel to the Granny Smith's axis, or must he move in other dimensions as well? In other words, if he increases his preference for Granny Smith's, is he likely to also increase (or decrease) his preference for, say, Cox's Orange Pippins? If so, then the cluster definitions must measure the covariance between the apple preferences. Thus a full covariance matrix must be defined, allowing each cluster to be orientated at any angle relative to each axis. If on the other hand we assume that the preferences are independent, then the points (and clusters) can move parallel with each axis without influencing other apple preferences. In this case, the covariance matrix for the cluster will be zero for all elements not on the leading diagonal, with the leading diagonal defining the variance parallel to each axis, i.e. the variance of each set of product preference scores for that cluster.

It is assumed that product preferences are related to sensory properties. If Mr Doe's preference for Granny Smith's increases because he likes their bitterness, then presumably his preference for any other bitter (or sweet) apples will increase (or decrease) accordingly. This effect can be captured using variable-orientation components.

The other factor when choosing the shape of Gaussian components is the number of parameters required to define them, and the difficulty in estimating these parameters

from a finite data set. A spherical Gaussian of fixed radius needs only its mean and mixing proportions specified. In  $d$ -dimensions, this is  $d + 1$  parameters per component. For example, in 20 dimensions, to define four such constrained clusters requires  $4 \times (20 + 1) - 1 = 83$  parameters<sup>5</sup>. At the other extreme, a fully variable ellipse requires  $\frac{1}{2}(d + 1)(d + 2)$  per cluster. Thus in 20 dimensions, to fully specify four clusters would require  $4 \times 0.5 \times (20 + 1) \times (20 + 2) - 1 = 923$  parameters. To find this many parameters from only a few hundred records is troublesome. Further consideration is given to this in Section 3.6.3. For comparison,  $k$ -means requires  $d$  parameters per cluster to define the mean, giving  $4 \times 20 = 80$  parameters in this example. As we noted earlier,  $k$ -means is implicitly constrained to find spherical clusters.

### 3.4.6 Biases in clustering algorithms

Kearns et al. [72] use information theory to demonstrate that in some simple but pathological cases, the  $k$ -means algorithm actually moves *away* from an initially correct solution. This problem occurs when two clusters overlap:  $k$ -means has no way of representing overlapping clusters, unlike GMMs, because each point is assigned to exactly one cluster. Thus when presented with overlapping samples from two Gaussian distributions, the GMM finds the “correct” model (i.e. it tends to find the means and standard deviations of the distribution from which the samples are drawn) while  $k$ -means estimates centres that are shifted away from the overlap. Thus compared to Gaussian mixture models,  $k$ -means finds clusters with less overlap, which raises the question: do clusters in preference data have overlaps? Does it make sense to talk about groups of consumers overlapping, or should we only consider non-overlapping, contiguous groups? This depends on how we regard the data and the nature of clustering. One interpretation regards the consumers’ preferences as being generated by some underlying process. For example, one consumer might have a sweet tooth and so prefer sugary foods. This consumer might belong to a distinct cluster of sweet-toothed people, as opposed to a cluster of “sour-toothed” people, perhaps. However, each individual will still have individual tastes, and so may like some products that most sweet toothed people do *not* like. In this model, clusters of consumers may overlap when mapped according to the preferences of a range of products. Thus consumers who actually belong to some intrinsic sweet-tooth cluster may appear closer to the “sour-toothed” cluster centre in some dimensions.

An alternative interpretation is to consider the data as being the only truth, and to consider any underlying generating process as either non-existent or unknowable. In this view, our aim is simply to split up the data into groups of consumers who

---

<sup>5</sup>The mixing proportions are constrained to sum to one. Hence having estimated  $k - 1$  of them, the last one needs no further calculation, and so we have one less free parameter to estimate.

expressed similar preferences over the range of test products. In this case, there can be no overlap of clusters. The clusters that we are attempting to discover will still (we hope) be useful for describing consumers, but they are not “real” divisions, in the sense that, for example, gender and species are. We are not attempting to discover pre-existing, underlying groups of consumers (which could in principle be generated from overlapping generating components), but merely to divide consumers into groups to enable us to design foodstuffs that more closely match their preferences.

One way to compare these two interpretations is to compare the results of clustering data using  $k$ -means clustering and GMMs. In general,  $k$ -means clustering tends to find equal-sized (hyper-)spheres, and can be misled by overlapping clusters. GMM algorithms can find overlapping, unequal-sized, non-spherical shapes, but require more free parameters to do so. Section 3.9 shows that these biases have a modest, though statistically significant effect when segmenting the preference data. The results we present there suggest that consumers should be divided into overlapping clusters of differing sizes, and that in at least some cases, these should be non-spherical.

A further point to note when comparing  $k$ -means with GMM is that  $k$ -means is typically much faster to converge. Typical timings on the spinach preference data, using a standard desktop PC, are 0.057s for  $k$ -means convergence, and 1.07s for EM convergence with a GMM, almost 20 times slower.

### 3.5 Comparing and validating segmentations

Stochastic clustering algorithms, such as  $k$ -means and Gaussian mixture models, tend to produce different solutions each time they are applied to a data set, due to the initially random values of their parameters. It is therefore important to compare solutions to see how much they differ, and to see whether these differences are significant. It is also interesting to see whether different algorithms consistently find different solutions to the same segmentation problems. Finally, if we know the true segmentation values (if we use synthetic data, for example), then we can compare these labels with the estimates provided by the clustering algorithms, and measure the accuracy of the algorithms.

Clustering algorithms explicitly or implicitly attempt to optimise some criterion function. For  $k$ -means, this is minimising the sum-of-squares-error (SSE) of the distance between data points and their cluster centre; for GMMs, this is maximising the likelihood,  $p(Data|Model)$ . When comparing two clustering techniques, it is inappropriate to use such a score. It would not be surprising if  $k$ -means minimised SSE better than GMMs or hierarchical clustering, because the  $k$ -means algorithm is *designed* to minimise SSE. We therefore require a measure of the accuracy of a solution that is not explicitly optimised by any of the algorithms that we are using. We discuss a number

		Partition V			
		1	2	3	4
Partition U		(65)	(60)	(71)	(44)
	1 (50)	39	2	0	9
	2 (85)	3	57	13	12
	3 (54)	23	1	7	23
	4 (51)	0	0	51	0

Table 3.2: Contingency table comparing two GMM partitions of the beverage preference data

of possibilities in Sections 3.5.1–3.5.6.

### 3.5.1 The Rand index

The Rand index [101, 68] is a way of measuring the similarity of two partitions of the same set of data. To calculate it, we require two partitions of the data, where a partition consists of a list that associates every data point with exactly one cluster. One way to achieve these two partitions is to perform clustering twice, using different algorithms or different random initialisations. Consider each pair of data points that are being clustered, and how they are grouped in two clustering solutions. If every data point is assigned to exactly one cluster, then each pair of points must be assigned either to the same cluster or to two different clusters, in each solution. By counting the number of these agreements and disagreements between the two solutions, we can derive an index of consistency. The index we use in this work is the adjusted Rand index proposed by Hubert and Arabie [68]. This gives a score of one for a perfect match, and zero for a match that is no better than random. In Section 5.3.3, we discuss a possible extension to the Rand index, that allows points to be assigned to more than one cluster, such as fuzzy clustering models, and GMMs. In this work, we assume that each point is assigned to exactly one cluster, and accordingly, use the maximum-likelihood estimate for GMM models.

Consider two partitions,  $U$  and  $V$ . With  $U$ , for each cluster label  $1 \leq i \leq k$ , we count how many members of the  $i^{\text{th}}$  cluster,  $U_i$ , belong to each of the clusters in  $V$ . These counts can then be used to estimate the similarity of two solutions. Table 3.2 is the result of two GMM partitions of the beverage preference data. The bracketed numbers show the total number of members assigned to each cluster. Although Brennan [21], notes that the number of clusters need not be the same in each partition, they are in this example.

Table 3.2 shows, for example, that  $U_2$  corresponds quite closely with  $V_2$ , with 57 out of the 85 members of  $U_2$  assigned to  $V_2$ . The remaining members of  $U_2$  were

mostly assigned to  $V_3$  and  $V_4$ . Similarly,  $U_4$  and  $V_3$  are a good match. In contrast, the 54 members of  $U_3$  were mostly divided equally between  $V_1$  and  $V_4$ , suggesting a poor match between  $U_3$  and any one of the  $V$  clusters. The adjusted Rand index for Table 3.2 is 0.441.

### Calculating the Rand index

Let us consider each pair of data points being clustered,  $x_i$  and  $x_j$ , and how they are grouped in two clustering solutions,  $U$  and  $V$ . Assuming that every entity is assigned to exactly one cluster, then each pair must be assigned to either the same cluster or to two different clusters. This gives four possible outcomes:

- a) Both  $U$  and  $V$  assign  $x_i$  and  $x_j$  to the same cluster
- b)  $U$  assigns  $x_i$  and  $x_j$  to the same cluster, but  $V$  assigns them to different clusters
- c)  $U$  assigns  $x_i$  and  $x_j$  to different clusters, but  $V$  assigns them to the same cluster
- d) Both  $U$  and  $V$  assign  $x_i$  and  $x_j$  to different clusters.

Exactly one of these four options holds for every possible pair of points. If we count how many times each option holds, we obtain four values,  $a$ ,  $b$ ,  $c$  and  $d$ , depending on which of the above four conditions holds. Then  $a$  and  $d$  represent consistent solutions and  $b$  and  $c$  represent conflicting solutions. Therefore a simple measure of consistency is:

$$R = \frac{a + d}{a + b + c + d}$$

This score is the (unadjusted) Rand index, proposed by William Rand [101], among others. Note that  $(a + b + c + d) = \binom{n}{2} = \frac{1}{2}n(n - 1)$ , i.e. the number of pairs that can be drawn from the  $n$  entities being clustered. We can define the total number of agreements as  $A = a + d$ , and the number of disagreements as  $D = b + c$ .

Several alternatives to the Rand index have been proposed. These are summarised by Hubert and Arabie [68, p.195–196]. While Rand proposed using the “probability of agreement”, Mirkin (and others) have suggested using the “probability of disagreement” [84]. Hubert and Arabie suggest using the difference between the probabilities of agreement and of disagreement.

### Adjusted Rand index

One drawback with the Rand index as described above, is that it is not “corrected for chance.” This means that it does not produce a suitable constant value (zero) in a

null case, when the two partitions are random. Hubert and Arabie [68] suggest such a correction, which theoretically gives a score of zero in the null case. This calculates the expected number of matching pairs of entities, assuming a random distribution in the contingency table. Milligan [83] empirically demonstrates that this corrected Rand index does indeed score (close to) zero on random data. This correction means that it becomes feasible to compare the actual magnitude of the index across different studies, irrespective of the number of clusters, the algorithm used, etc.

The Hubert and Arabie “corrected Rand index” is defined thus:

$$R_c = \frac{a + d - n_c}{a + b + c + d - n_c}$$

where the correction for chance,  $n_c$ , is a function of the values in the contingency table. Note that if only one cluster is defined, then the adjusted Rand simplifies to 0/0 because  $b$ ,  $c$  and  $d$  will all equal zero, and  $a$  will equal  $n_c$ . It seems sensible to define this as one, i.e. to assume that even two random partitions would cluster the data the same way.

### 3.5.2 External criterion

In discussing the Rand index, we considered one method of comparing two cluster partitions, assuming that we do not know the true classification. However, if we do know the true class labels of the data points, then we can use this “external” information to measure the quality of a partition. Essentially, we replace partition  $V$  with the true class labels, and measure how similar  $U$  is to it. This can then be interpreted as the probability that our partition  $U$  is correct. Although this can only work with artificial data (or any other data set where we know the true cluster membership), it is a useful way to measure the effectiveness of clustering algorithms.

One drawback of this approach is that the accuracy with which each data vector is modelled is taken as a crisp true or false. In other words, a partition that “only just” misclassifies a vector is penalised just as much as one that substantially misclassifies it. Future work therefore includes the development of a non-crisp variant of the Rand index, interpreting the assignment of data points to clusters probabilistically, which is more appropriate for clustering algorithms such as fuzzy  $c$ -means and GMMs. This is discussed further in Section 5.3.3.

### 3.5.3 Train and test

If the same data is used both in the training and the testing of any model, there is a strong risk that the resultant measure of accuracy will be an overestimate. If we split the data into two distinct sets, using one set to build the model and the remainder to

test it, any random noise in the data is unlikely to have the same pattern in both sets (see Section 2.5.1 and particularly Figure 2.7; also see Mitchell [85, p.69]). Therefore if the model overfits the training data, the accuracy over the test set will tend to be lower. In general, the error over the test set provides an unbiased estimate of model accuracy (assuming the two data sets are independently drawn from the same distribution).

This approach can be applied to cluster analysis by using some of the data to build the model (i.e. to find the clusters) and then using this model to label the remaining (test) data. The model’s estimated labels can then be compared with the known labels. The accuracy can then be calculated (e.g. the percentage of correct predictions). Before any predictions can be made however, the cluster labels may need to be permuted. (We don’t require the clustering algorithm to assign all fish to a cluster labelled “fish”, merely that it should assign all fish to the same, arbitrarily labelled cluster. We can provide labels by hand later.) Although this is an appealing approach, it once again requires labelled data and so can only be applied to artificial data sets (as we will describe in Section 3.6.1).

### 3.5.4 Replication analysis

Instead of relying on pre-existing labels, we can use one clustering model to estimate the labels, and then use a second to “re-discover” these same labels. Replication analysis can be used to validate the results of clustering and is reviewed by Milligan [82]. This is a form of cross validation (Section 2.5.1), where the data is divided into training and testing sets. The training data is used to find clusters, which are then used to classify the test data. The test data is then independently clustered to form a separate model. This gives two partitions of the test data, which can be compared using the Rand index (or some similar technique). If the two partitions of the test set are very similar, then the same structure has been found in both, suggesting that this structure is actually in the data, and not some artefact of the algorithm or noise. If this “real” structure has been found, then the original model (built from the training set) will generalize well. Milligan only covers hierarchical clustering, which is deterministic. For non-deterministic algorithms (e.g.  $k$ -means and GMM), multiple runs would be required to estimate the consistency.

This approach has the goal of rejecting models that overfit (or underfit) the data. It has the advantage that it does not require a labelled data set (i.e. one where the true class labels are known in advance), unlike the external criterion and “train and test” methods we described above. We use replication analysis in Section 3.6.1 to consider the possible existence of clusters.

### 3.5.5 Hypothesis testing

Milligan [82] distinguishes between external criterion analysis and internal criterion analysis. External analysis uses extra variables that are not used to build the cluster model. For example, we could remove one feature from a data set, and then clustering it using the remaining variables. Given this cluster model, we could measure the within-group and between-group distances for the feature that we held out. We can then perform a parametric statistical test to determine whether the clusters that we found also cluster the extra feature. An alternative external method uses different sample of data drawn from same distribution as the training set, which is equivalent to what we described in Section 3.5.3, above.

However, withholding a feature thought to be useful in discovering and describing the clusters will tend to produce less accurate models. If the feature is not related to the clusters, the statistical test will be meaningless. At the other extreme, we must consider correlated features. If two features are highly correlated, and we use one feature to build a model and the other to test it, then we are really only measuring the correlation between the features, and not the accuracy of the model's predictions.

Internal analysis re-uses the same data (with all the features) to measure the goodness of fit between the model and the data. One approach is to analyse consistent and inconsistent pairings. This measures whether entities assigned to the same cluster are more or less similar than entities assigned to different clusters. The null hypothesis is typically that the cluster labels are randomly assigned; the test measures the likelihood of the alternative hypothesis, namely that structure exists within the data. This gives a clear statistical result, interpretable as confidence in the model. The main difficulty with this approach, as noted by Milligan [82], is the choice of a null sample distribution for the test statistic. In this work, there is no obvious choice, so this approach is inappropriate.

### 3.5.6 Robustness

Rather than attempting to measure how accurate a clustering model is, it may be more profitable to measure how *consistently* the algorithm discovers the clusters. This is particularly important if the data points are unlabelled (such as the Unilever preference data sets) and, therefore, the underlying clusters<sup>6</sup> are unknown, in terms of position, size, number etc. If multiple runs of a clustering algorithm are performed, the models can then be compared. This could be done using the Rand index to compare the segmentations, or else by directly comparing the partitions in a model-specific fashion (e.g. the total distance between pairs of centres in two  $k$ -means clustering models).

---

<sup>6</sup>Assuming that underlying clusters exist — see Section 3.4.6

Similar analysis could be carried out as a method to estimate the number of clusters. We could partition the data into  $k$  clusters a number of times, where  $k$  varies (e.g. 2–20). If the algorithm discovers the underlying model, it should re-discover it repeatedly, giving high Rand index scores between solutions with the true  $k$  value, but lower Rand scores between solutions with the wrong  $k$  value.

Unlike some of the methods outlined above, this technique does not require labelled data. We discuss robustness further in Section 3.8, where we experimentally measure the robustness of both artificial data and preference data sets, in an attempt to ascertain the true number of clusters.

### 3.5.7 Summary of cluster validation methods

In this section, we have discussed a number of methods for comparing and validating the results of cluster analysis. We now consider which of these methods is appropriate for the current work.

The (adjusted) Rand index has the advantage that it does not require labelled data, although it can be used with labelled data to measure accuracy. The external criterion (Section 3.5.2) and “train and test” (Section 3.5.3) approaches require labelled data, which is not available for the preference data sets, and these techniques will not be considered further. Formal hypothesis testing with internal criteria (Section 3.5.5) requires a null sample distribution which is unavailable; this will also not be considered further.

Replication analysis is essentially a method to avoid overfitting [82, p. 368], and is a form of cross validation (c.f. Section 2.5.1, p. 51). An alternative to cross validation is the minimum description length method (Section 2.5.2 p. 54), which is used in Sections 3.6.2–3.6.4, without replication analysis.

Replication analysis splits the available data set in two, using only half to build the cluster model. In Section 3.6.3 (particularly Figure 3.14), we will show that using a small data set can lead to an underestimation of the number of clusters. Therefore, we do not use replication analysis in Section 3.9. We do use a form of replication analysis in Section 3.6.1 below when considering the very existence of clusters, rather than trying to estimate the number of clusters present.

We use robustness analysis (Section 3.5.6) in Section 3.8 as an alternative method to estimate the number of clusters present in a sample.

## 3.6 How many clusters?

Like most clustering algorithms, both  $k$ -means and Gaussian mixture models require the analyst to specify in advance how many clusters for which to search. In Section

3.6.2, we consider how to estimate the intrinsic number of clusters by using information criteria, but first we need to consider whether any clusters exist at all.

### 3.6.1 Existence of clusters

Before attempting to estimate how many clusters exist, we shall consider whether there are any clusters at all. In particular, if we compare the consistency of repeated clustering of a preference data set with the consistency of clustering an artificial, unstructured data set, we can discover if there may be some structure in the preference data. The assumption is that if clusters do exist in the preference data, then the clustering algorithms should find them with a “greater than random” frequency, as compared with an unstructured random data set.

To do this, we perform a form of replication analysis (Section 3.5.4). First, we randomly split the data set into two halves,  $d_1$  and  $d_2$ . Then we cluster each data set independently, producing cluster models  $c_1$  and  $c_2$ . We use each cluster model to classify the other half of the original data set, producing a second clustering of the data,  $c'_1$  and  $c'_2$ , where  $c'_1 = c_2(d_1)$ . I.e. the classification of the first half of the data, using the classifier trained on the second half of the data. Similarly,  $c'_2 = c_1(d_2)$ . If our cluster model is correct, then  $c'_1 \approx c_1$  and  $c'_2 \approx c_2$ . We can then use the Rand index (Section 3.5.1) to measure the similarity. Over a number of repeated experiments, we can produce a distribution of similarity measures. If we repeat the whole process with a second data set, we can then test to see if the two distributions are the same, and hence whether one data set has more structure than the other one does.

We use four data sets here: the three food preference sets, and an artificial set. The latter consists of 300 points in a 15 dimensional space, normally distributed with a standard deviation of 10. This can be thought of as a null preference data set, with 300 consumers measuring 15 products, but with no structure (i.e. clusters) present in the responses.

We take each data set in turn, split it into two, and cluster each half as described above, using the  $k$ -means clustering algorithm, with  $k = 4$  in all cases<sup>7</sup>. After clustering, we calculate the Rand index to measure the similarity between the two cluster models. For the three preference data sets, this value is in the range 0.30–0.35<sup>8</sup>. This shows for example, that there is an approximately 30% chance that two points assigned to the same cluster in one model will also be assigned to the same cluster in the other model. For comparison, the unstructured random data gives a consistency score of

---

<sup>7</sup>The arbitrary choice of four clusters is not necessarily an optimal value of  $k$ ; the aim here is not to identify the exact number of clusters, merely to see if they exist.

<sup>8</sup>The mean adjusted Rand index scores were: 0.307 for the beverage data; 0.342 for the meat data; and 0.337 for the vegetable data. Each figure is the average result from 100 random splits of each data set.

0.061 (averaged over 100 random splits). This shows that each pair of points drawn from an unstructured distribution are unlikely to be consistently assigned to clusters in two different models, consistent with the definition of the adjusted Rand index.

We now test whether or not the preference data contains *significantly* more structure than random data, with the assumption that any structure present will tend to lead to greater consistency in each pair of clustering results. By repeating the above clustering and comparison 100 times for each data set, we generate a distribution of the Rand index scores. We can then use a standard statistical hypothesis test to see if the scores are significantly different. Comparing the unstructured data set to each of the preference sets in turn, a non-parametric Wilcoxon rank sum test gives  $p \approx 0$  in all three cases, strongly suggesting that there is structure in these sets. Having established that clusters are likely to exist, we need to determine how many (while accepting that the answer may still be zero).

### 3.6.2 Information criteria

In Section 2.5.2 (p.54), we gave some background to complexity penalisation and information criteria. As we described there, the Minimum Description Length (MDL) requires a “large” data set to produce reliable results. The feature selection work was using data sets with dozens of sensory records, whereas the segmentation work now being discussed has hundreds of preference records. Therefore, we can sensibly consider using MDL to estimate the number of clusters present in the data.

The description length is the sum of the model likelihood,  $L$ , and the model complexity,  $C$  (e.g. Bishop [15, p.429]). For a Gaussian mixture model, the likelihood is taken as the negative log likelihood that each data point was generated by the model. Thus:

$$L = - \sum_{i=1}^n \ln p(x_i | M)$$

where  $M$  is the mixture model, and  $x_i$  is the  $i^{th}$  record from  $n$  records.

The complexity  $C$  depends on the nature of the model; it is a function of the number of records and the number of free parameters. The general case gives complexity  $C$  as:

$$C = \frac{1}{2} \ln(n) \times p$$

given  $n$  records and  $p$  free parameters.

The simplest model uses spherical clusters, with the covariance matrix being a scalar multiple of an identity matrix. This requires us to estimate the mean and radius of

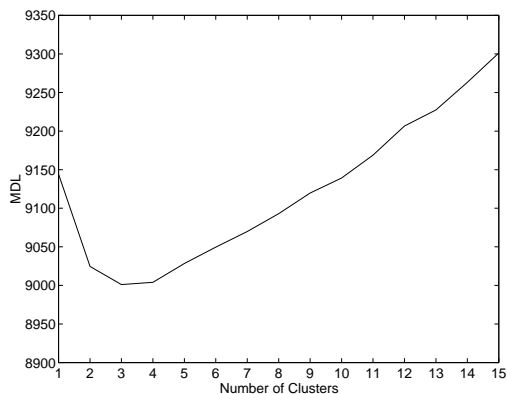


Figure 3.8: Description length for meat preference data, spherical GMM

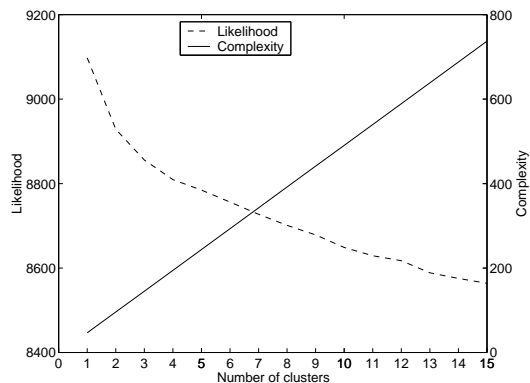


Figure 3.9: Complexity and likelihood for meat preference data

each component, and the mixing proportions. Thus:

$$C(k) = \frac{1}{2}(\ln n) \times (k(d + 2) - 1)$$

with  $n$  records,  $d$  dimensions (variables) and  $k$  clusters (c.f. the end of Section 3.4.5).

Figure 3.8 shows the description length for a spherical Gaussian mixture model, with between one and 15 clusters, applied to the meat data set,  $P_m$ . The minimum corresponds to assuming three clusters, although the description length score is barely greater when four clusters are assumed, or even two. Figure 3.9 shows the two components of the description length for the same data set. The dashed line (left axis) shows the likelihood, while the solid line (right axis) shows the complexity, a linear function of the number of clusters. The sum of these two lines gives the description length curve of Figure 3.8. Clearly, the likelihood error score drops monotonically as we increase the number of clusters. In the limit, if every data point is assigned to its own Gaussian component, then the components would tend to have a covariance of zero, and each data point would have a negative log likelihood approaching zero.

Figure 3.10 shows the description length for a spherical Gaussian mixture model, applied to the vegetable data set,  $P_v$ . The minimum description length corresponds with four clusters. Figure 3.11 shows the two components of the description length for the same data.

Figure 3.12 shows the description length for a spherical Gaussian mixture model, applied to the beverage data set,  $P_b$ . The minimum description length corresponds with seven clusters, although eight or nine are only slightly less likely. Figure 3.13 shows the two components of the description length for the same data.

The minimum description lengths have given us estimations of the number of clusters present in our three preference data sets. But how reliable are these estimates?

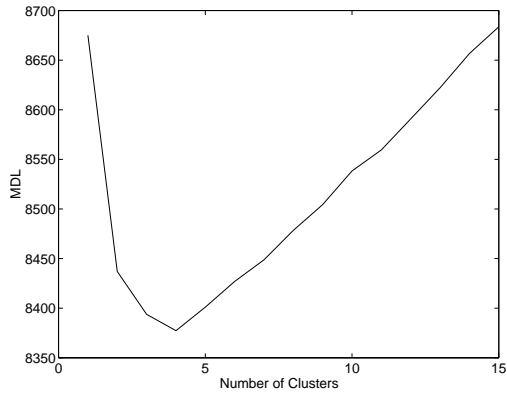


Figure 3.10: Description length for vegetable preference data, spherical GMM

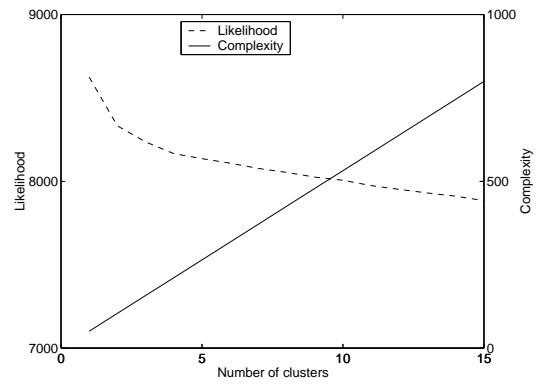


Figure 3.11: Complexity and likelihood for vegetable preference data

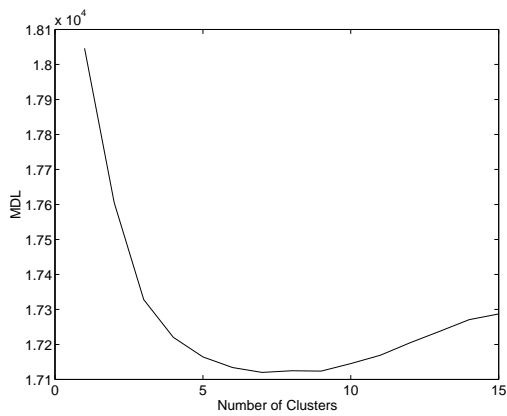


Figure 3.12: Description length for beverage preference data, spherical GMM

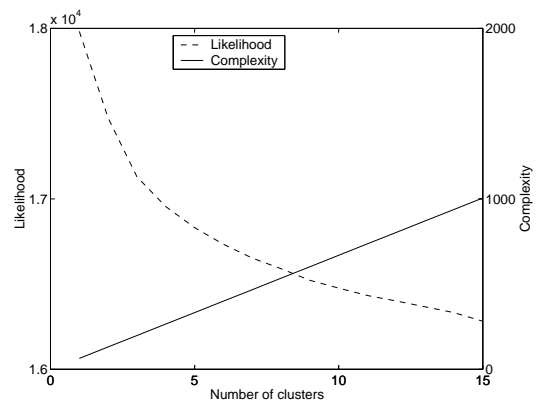


Figure 3.13: Complexity and likelihood for beverage preference data

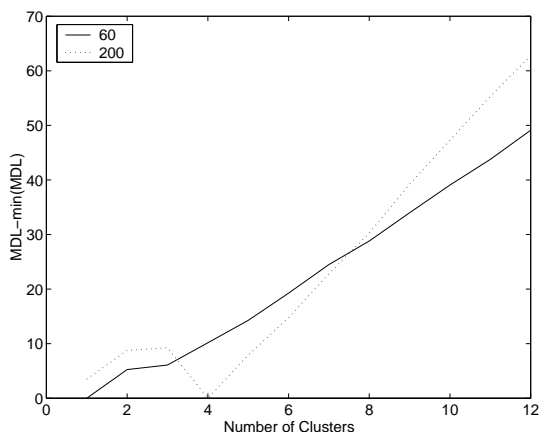


Figure 3.14: Description length for artificial data; various data set sizes. Each line is rescaled to have a zero minimum.

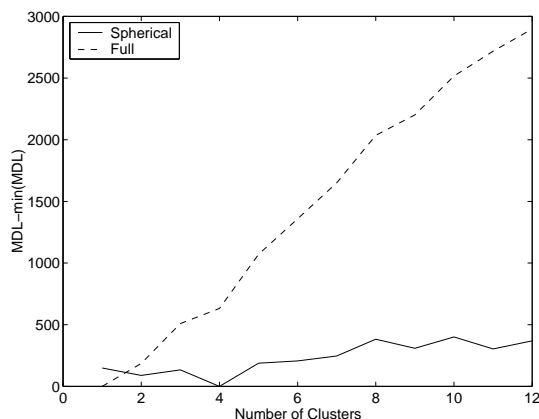


Figure 3.15: Description length for artificial data; non-spherical components. Each line is rescaled to have a zero minimum.

Hjorth [66, p. 46] suggests that even 500 records may not be enough to allow information criteria such as MDL to be used safely. In the next section, we examine the effects of small data sets on MDL predictions, and similarly, the choice of model complexity. In Section 3.8, we present an alternative method for estimating the number of clusters present that does *not* explicitly require a large data set.

### 3.6.3 The effect of small data sets

We now consider the effect of the size of the data sets on the estimation of the number of clusters present, when using the minimum description length. To do this, we will define an artificial data distribution, so that we know the true number of clusters, and we can control the number of samples drawn. The data set consists of  $n$  records, equally drawn from four Gaussian components in two dimensions. The standard deviation of each component is two. We select  $n = 15$  or 50 points per component, giving a total of 60 or 200 points.

Figure 3.14 shows the effect of the number of records on how accurately MDL selects the correct number of clusters<sup>9</sup>. Ideally, the lowest point of each line should be at four clusters, corresponding to the source of the data. The dotted line corresponding to the larger data set does indeed show a dip at four clusters; however, when fewer records are present, the description length line (solid) has a minimum corresponding to a single cluster. Thus with a small data set, the intrinsic number of clusters may be underestimated, due to a lack of information.

A similar effect can be seen if rather than reducing the number of records, we in-

<sup>9</sup>For clarity, each line has been rescaled to have a minimum at zero. The results are the average of 50 experiments with independent data sets.

crease the number of parameters to be estimated. If we reduce the spherical constraint on the Gaussian mixture model, allowing it to find elliptical components, many more parameters must be estimated (Section 3.4.5). The data are 100 points sampled from each of four spherical Gaussian components in 16 dimensions. Two types of Gaussian mixture models are used, one with spherical components, and one with fully elliptical components. Figure 3.15 shows the effect of searching for these mixture models with a limited data set<sup>10</sup>. The spherical GMM description length (solid line) has a minimum at four, correctly estimating the number of clusters present. The elliptical GMM description length (dashed line) has a minimum at one, underestimating the number of clusters. If we had used more data, no doubt both methods would have produced the same, accurate estimate, and given even less data, both would have underestimated the value. Nonetheless, these results demonstrate a potential bias whereby using complex models (such as elliptical GMMs) can lead the MDL principle to underestimate the intrinsic complexity of a finite sample of data.

Given this effect, and that we are constrained to using very small data sets, can we reduce the number of parameters further? Several variations of GMMs were introduced in Section 3.4.5; we now consider these in more detail, and examine the effect of choosing the “wrong” model.

### 3.6.4 Cluster shapes

We can not only constrain the Gaussian components to be spherical, we can also constrain them to be equally likely, by forcing the mixing proportion to be constant and equal to  $1/k$ <sup>11</sup>. This reduces the number of parameters by  $k - 1$ , leaving  $k(d + 1)$  parameters to estimate, if we have  $k$  clusters in  $d$  dimensions. Of course, if the clusters are of significantly different sizes, then the likelihood score will worsen, because the model can no longer represent them.

An alternative constraint we could impose on the Gaussian mixture model would be to force all the components to be the same size, by forcing all the covariance matrices to be equal. Assuming we are using spherical clusters, this means that all the covariance matrices will equal  $\lambda\mathbf{I}$ , with a single value of  $\lambda$  being used across all components, and where  $\mathbf{I}$  is a suitable identity matrix. This model also has  $k(d + 1)$  free parameters.

If we fix the mixing proportion and force the GMM components to have equal covariance matrices, we have  $kd + 1$  free parameters.

We can begin to investigate these differences using artificial data, and the Rand index. We described the Rand index in Section 3.5.1; in this context, we can use it as

<sup>10</sup>As before, the lines are rescaled to have a minimum at zero. The results are the average of 50 experiments with independent data sets.

<sup>11</sup>In principal, other priors could be used, but a uniform prior is appropriate as no other information is available.

GMM Version	Data source		
	Vary priors	Vary size	Vary neither
No constraint	0.662	0.560	0.624
Equal covariance	0.698	0.362	0.660
Equal prior	0.431	0.608	0.690

Table 3.3: Biases in Gaussian mixture models: Rand index showing correlation with truth

a measure of accuracy scaled from zero (no better than random) to one (perfect).

By changing the number of free parameters, we are changing the complexity penalty function used in the MDL calculations, and hence the gradient of the solid straight line in Figures 3.9, 3.11 and 3.13. Assuming the negative log likelihood drops monotonically as we increase the number of clusters (as it does in the experiments described in Section 3.6.2), then as we reduce the gradient or intercept of this line, the estimated number of clusters will increase. Of course, by making the Gaussian components less complex, we would expect the likelihood score to increase, as the simpler model will tend fit the data less closely.

We now consider the effect of using different Gaussian components in a GMM, given different sets of data. We use three different forms of GMM to estimate the true cluster memberships of three different sets of artificial data. The three data sets all consist of points drawn from four Gaussian components in two dimensions. The first set had a different number of points drawn from each component ( $n_i \in \{35, 75, 150, 300\}$ ), and all components had the same standard deviation ( $\sigma = 2$ ). The second set used the same number of points in each cluster ( $n_i = 50$ ), but with various standard deviations ( $\sigma \in \{1, 2, 3, 4\}$ ). The final set used the same number of points in each cluster ( $n_i = 50$ ) and a constant standard deviation ( $\sigma = 2$ ). These three sets are shown in Table 3.3 as “vary priors”, “vary size” and “vary neither” respectively.

The three Gaussian mixture models all consisted of four spherical components (corresponding to the four clusters). The first model allowed different priors and different covariance matrices for each component. The second was constrained to use a single covariance matrix for all components, but allowed differing priors. The third model could use different covariance matrices, but all components had the same prior. These three models are shown in Table 3.3 as “no constraint”, “equal covariance” and “equal prior” respectively.

The results in Table 3.3 show that when the constraints contradict the data, the accuracy measured with the Rand index is low. For example, when the model is constrained to use equal covariance matrices, but the data is drawn from different sized clusters, the Rand index was only 0.362 (the centre cell in Table 3.3). The last

column shows that when the data set is at its simplest (equal sizes, equal priors), then the more constrained models outperform the more general model. By introducing more constraints, we have fewer free parameters to estimate, allowing the remaining parameters (such as the means of the components) to be estimated more accurately. But this is data dependent: if we choose a model which is too heavily constrained, then we will underfit the data, and produce a poor model.

As with any data modelling problem, this shows a problem-specific trade-off between accuracy and complexity. There is some optimal level of model complexity required to maximise the accuracy of the model's predictions. However, with unlabelled data, we have no way to measure the accuracy. In Section 3.8, we consider one possible solution.

### 3.7 Which algorithm?

Having discussed clustering algorithms and how to validate them, it is now time to consider which algorithm is appropriate to use for these data sets. We use the Rand index here to compare  $k$ -means models with spherical GMMs on an artificial data set. We use artificial data here so that we can measure the accuracy of each method, and know in advance the true number of clusters. The data consisted of 300 points, sampled from four Gaussian components in 16 dimensions. Each Gaussian had a standard deviation of  $\sigma = 10$ , with a mean inter-centre distance of three. Both algorithms were used twice, to create a total of four partitions of the data. The Rand index between each pair of partitions was then calculated, as was the Rand index between each partition and the true classification, according to which of the four clusters actually generated each data point. Table 3.4 shows the resulting Rand index scores. The third row of the table shows the case when four clusters were sought, corresponding to the underlying distribution.

Several trends are apparent. Firstly, as we increase the number of clusters being sought (moving down the table), both  $k$ -means and GMMs become less consistent, as shown by the decreasing Rand indices in the first two columns. With more clusters, there are more possible models, so the chance of correspondence is reduced. Secondly, GMMs tend to be much more consistent than  $k$ -means: the values in the second column are always larger than those in the first. The artificial data used is created from a mixture of Gaussians, with some overlap, and so the GMM is more appropriate than  $k$ -means, as discussed in Section 3.4.6.

Thirdly, both methods are far less accurate than they are consistent, shown by the fact that columns four and five (the "truth" columns) have lower scores than columns one and two, respectively. If an algorithm fails to find a very good solution when run once, it is reasonable to suppose that a second execution is more likely to find the

Clusters	<i>k</i> -means vs. <i>k</i> -means	GMM vs. GMM	<i>k</i> -means vs. GMM	<i>k</i> -means truth	GMM truth
2	0.485	0.943	0.476	0.152	0.134
3	0.406	0.878	0.428	0.180	0.183
4	0.296	0.662	0.333	0.162	0.177
5	0.239	0.491	0.269	0.138	0.160
6	0.203	0.453	0.224	0.121	0.143
7	0.205	0.404	0.206	0.114	0.134
8	0.194	0.395	0.201	0.097	0.131
9	0.191	0.380	0.200	0.094	0.117
10	0.185	0.363	0.186	0.088	0.119

Table 3.4: Rand index scores within and between algorithms, for an artificial data set, generated from a mixture of four Gaussian components

same poor solution as before (and so be consistent) than it is to find a different, better solution (and so be accurate).

The middle column shows the similarity of the two classes of clustering models. Again, as we increase the number of clusters being sought, the solutions become less similar, because there are more possible solutions to choose from. These figures are very close to the first column, suggesting that the inconsistencies between the two algorithms are caused mostly by the larger variation in the solutions found by *k*-means clustering.

Finally, if we compare the last two columns, we can see that when seeking just two clusters, *k*-means is more accurate than a GMM (a sign test gives  $p \approx 0$ ). With more than 4 clusters, GMMs are significantly more accurate than *k*-means ( $p \ll 0.01$  in each case). With three or four clusters, the differences are not significant.

Overall, we can say that GMMs tend to be more consistent in their solutions than *k*-means models, but that they are not necessarily more accurate. In Section 3.8, we discuss whether these consistency measures can be used in estimating the intrinsic number of clusters, when this is not known a priori. In Section 3.9, we present an alternative view of accuracy, namely regression accuracy, which is more applicable to food data, because the class labels are unknown.

### 3.8 Consistency versus accuracy

In the previous sections, we used artificial data, because we needed labelled data to estimate the model accuracies. The food preference data is unlabelled, so we need to consider some alternative methods. Even when we can't measure the accuracy of clustering, we can measure consistency. If we repeat the same algorithm on the same data set, we can measure whether or not we get the same answer.

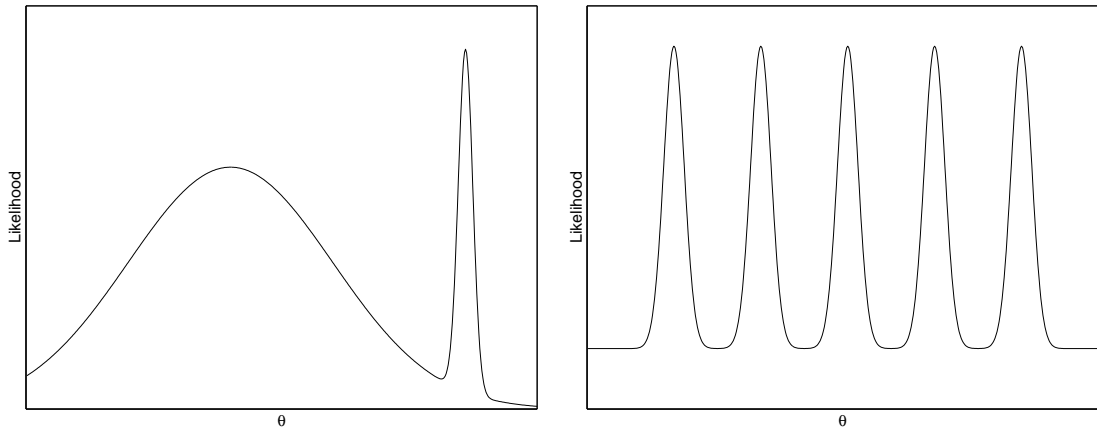


Figure 3.16: Pathological fitness landscape    Figure 3.17: Multi-modal fitness landscape

As Table 3.4 shows, consistency does not necessarily imply accuracy; accuracy does not necessarily imply consistency. It is not hard to imagine a pathological data set which tends to mislead the model building process, and consistently produces the same inaccurate result: Figure 3.16 shows such a landscape. Many iterative model-building algorithms — such as the EM algorithm for a GMM — will tend to converge to the large dome on the left of the graph, rather than finding the optimum solution towards the right. Conversely, a data set may have multiple optimal solutions, which although distinct are equally accurate. Figure 3.17 shows a multi-modal landscape, with five equally good solutions. In this case, a model with high accuracy could be on any of the peaks, which suggests that a set of solutions could have very low consistency, while still being accurate.

In the case of clustering, as we increase the model complexity (and in particular, the number of clusters sought), we expect:

1. The cluster error<sup>12</sup> will decrease. This is because with more cluster centres, the data points will be closer to the nearest centre (and therefore have greater likelihood in a Gaussian model).
2. The consistency with which solutions are found will also tend to decrease. This is because with more possible models to choose from, any particular model will be found less often.

An exception to the decrease in consistency occurs if we search for exactly as many clusters as are intrinsically present in the data. In this case, we would expect the same (correct) solution to be found almost every time, leading to greater consistency.

We propose an alternative method to estimate the number of clusters present, based on this increased consistency. We predict that when we search correctly for the intrinsic

<sup>12</sup>In  $k$ -means, this is the sum-squared-error; in GMMs, it is the negative log likelihood.

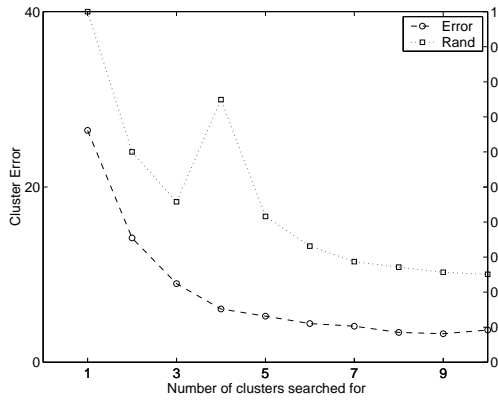


Figure 3.18: Accuracy and consistency: artificial data generated from four Gaussian components.

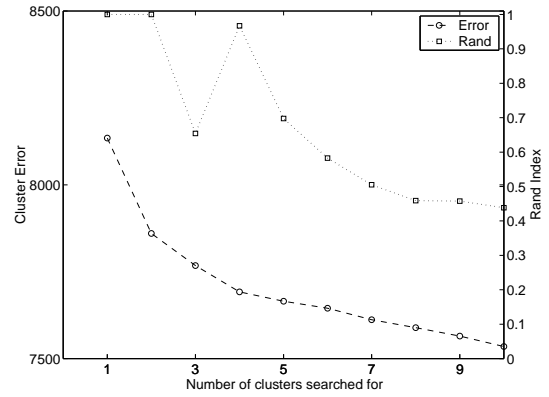


Figure 3.19: Accuracy and consistency: vegetable preference data

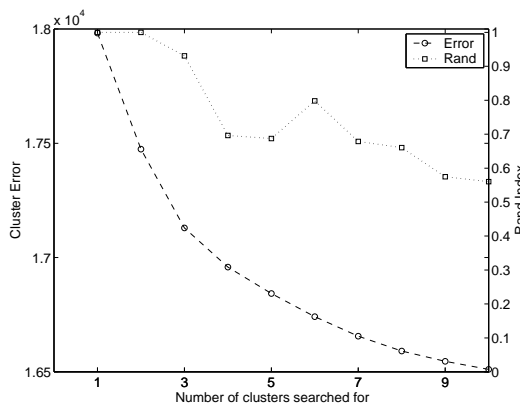


Figure 3.20: Accuracy and consistency: beverage preference data

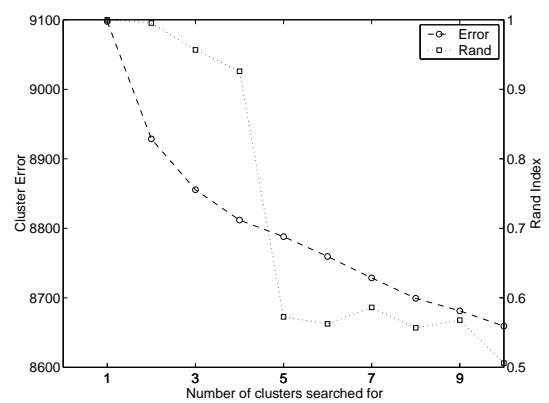


Figure 3.21: Accuracy and consistency: meat preference data

number of clusters, then we will find the same answer consistently, whereas when we search for the wrong number, we will tend to get different answers every time. To examine this, we measure both the error and the consistency of  $k$ -means algorithm solutions as we vary  $k$ . We define consistency here as the average Rand index score between pairs of segmentations found by independent applications of the  $k$ -means algorithm.

We start with a set of artificial data, where we know the intrinsic number of clusters, and we can choose a distribution that should be relatively easy for the  $k$ -means algorithm to model. Figure 3.18 shows both the sum-squared-error and the Rand index calculated across 50 runs of the  $k$ -means algorithm, with different random initialisations. The data consists of 75 points drawn from each of four Gaussian components in two dimensions; each Gaussian has a standard deviation of 2, with the centres being on the corners of a square of length 3. This graph shows monotonically decreasing error (dashed line), as expected. However, the consistency (dotted line) measured by the Rand index shows a clear peak at the four-cluster mark, corresponding to the true number of clusters.

Thus although more complex models (with five or more clusters) appear to be more accurate, they are also less consistent than the four-cluster models. The simpler models, (with two or three clusters) are less consistent, contrary to what would otherwise be expected. Trivially, when searching for just one cluster, the same model will always be found by  $k$ -means clustering, corresponding to the mean of the entire sample, so the Rand index is always one.

Figure 3.19 shows the same graph but using the vegetable preference data instead of artificial data. The Rand score shows a clear peak at four clusters, with an approximately identical segmentation being found more consistently than is the case with either three or five clusters. This corresponds with the MDL estimates given earlier (Section 3.6.2 and Figure 3.10).

Figure 3.20 shows the results using the beverage preference data. The small peak at six clusters is close to the MDL estimate in Figure 3.12, which suggested around seven clusters.

With both the vegetable and beverage data sets, seeking very few clusters (e.g. two or three) gives very consistent results. As noted earlier, there are fewer such models, and so there will be greater consistency. However, both the graphs show consistency scores that do *not* decrease monotonically, which is what we would expect from unstructured data. The deviations from a monotonic decrease correspond (approximately) to the solutions found using MDL earlier, but with several advantages. Firstly, we do not need to calculate a likelihood score, so we can use any clustering algorithm. (Here, we used  $k$ -means, for example.) This gives us greater flexibility, and can save time. Secondly, the MDL method requires a large data set, with no clear definition of “large”

in this context, whereas our consistency approach makes no such demands. Of course, with a larger sample of data to estimate models from, we would get more confident results in both cases.

Figure 3.21 shows the consistency results using the meat data. This is less clear than the other results: the error drops monotonically, as does the consistency. This consistency suggests that only one or two clusters exist, or else that this consistency test is not powerful enough, in this case, to indicate the true number of clusters. One problem is that models with few clusters tend to be consistent whether or not their size reflects the inherent structure in the data. This is why Figures 3.19–3.21 all show a Rand index very close to one when we search for  $k = 2$  clusters. Thus, when there are very few clusters present in the data, the local maximum that we hope to find may get swamped by the high Rand index scores, leaving a monotonically decreasing consistency score. Note that the MDL approach gave an estimate of just three clusters for the meat data (Figure 3.8), where as the graph shows a sharp drop in consistency when using more than four clusters.

As part of a review of cluster validation techniques, Halkidi et al. [60] mention a similar process where they search for a “knee” in the cluster error, as the number of clusters sought is varied. However, they only present results on a very simple data set, consisting of six points in two dimensions, whereas in this work, we have analysed the technique in more detail, and applied it to real-world data sets.

### 3.9 Usefulness of predictions

As well as asking how many clusters exist, we should also ask how many clusters is it *useful* to find? Would it be useful for food manufacturers to identify hundreds of clusters of consumers? Probably not: the extra overheads associated with producing, distributing and marketing hundreds of brands that are only marginally different would outweigh the benefits of satisfying each consumer only marginally more.

Sainsbury’s home-shopping web site<sup>13</sup> lists approximately 60 different instant coffees, 42 different loaves of white bread, 21 different beef burgers and 9 croissants<sup>14</sup>. A cynic might suggest that segmentation and even product design are hardly necessary: someone will buy almost anything put on a supermarket shelf. However, this assumes that each of these products is distinct, and aimed at a distinct market segment. Factors such as advertising, packaging and price will allow very similar products to survive in the same market. Even a marginal difference in sensory qualities will tend to make consumers prefer one product to another, even if neither is actually very close to each consumer’s optimum product profile.

---

<sup>13</sup>[www.sainsburys.co.uk](http://www.sainsburys.co.uk)

<sup>14</sup>Ignoring different sized packs, different thickness of sliced bread etc.

When considering the accuracy or consistency of clustering solutions, we must not lose sight of the reasons for attempting segmentation in the first place. We are trying to put consumers into groups in order to understand why each group likes the foods they like. Thus we can define a useful clustering solution as being one that allows us to model the preferences of the resultant clusters, and hence predict the preferences of groups of consumers.

If we cluster the data, and then build a regression model for each cluster to predict preferences from the sensory scores, then we want ultimately to minimise the regression error. We can therefore use regression as a wrapper (see Section 2.3, p. 40) around the clustering algorithm, to select between clustering solutions generated by one algorithm, and also to select between clustering algorithms.

Several options present themselves. Firstly, we could use regression accuracy as a way of selecting one clustering solution from many. We could, for example, use the  $k$ -means algorithm to build 100 segmentations, then calculate the regression accuracy for each segmentation, and select the segmentation with the lowest error as being the most useful.

Secondly, we could perform a stochastic search across cluster solutions, with a fitness function based on both cluster dispersion error and regression error. Genetic algorithms have been used as an alternative to more conventional clustering methods. Murthy and Chowdhury [89] used a simple binary representation, with each bit indicating whether a data point belonged to a given cluster or not. They show that their genetic clustering algorithm could equal or better standard  $k$ -means. However, their method requires considerably greater computational effort, and they use extremely simple data sets in the experiments described, making it difficult to come to general conclusions. Demiriz et al. [34] use a similar method combined with an “impurity measure” to form a semi-supervised genetic clustering algorithm. The fitness function is a weighted sum of the two sources of error, namely the total cluster dispersion and the cluster purity, measured using a subset of labelled data. In our work, we could define a fitness function as a weighted sum of dispersion error and regression error. The question of how to determine a suitable weight for the error term is left unanswered (by Demiriz et al. and by us).

Thirdly, we could analytically combine clustering and regression into a single hybrid algorithm. However, this is likely to produce an unstable algorithm, because the clustering component will re-estimate the centres of each cluster, and it is these centres which form the target of the regression component.

These approaches all suffer from the same flaw: the more clusters we use, the fewer members each cluster will contain, and so each individual will be better represented within the cluster by the regression model. Thus as with conventional clustering, the more clusters (and hence regression models) we use, the lower the apparent error will

be. For this approach to be useful, we need some complexity penalisation method, such as MDL, or a more reliable error estimate, such as cross validation error. We now propose one such approach.

### 3.9.1 Weighted cluster-regression

We now compare  $k$ -means and Gaussian mixture models (GMMs), with either spherical or full (i.e. variable-orientation elliptical) components, and use a weighted regression error to distinguish between them. We segment the preference data using one of these algorithms, and calculate the mean vector (i.e. the centre) of each cluster. This vector represents the preference scores of a “virtual consumer”, and approximates the preferences of every member of the cluster. We then use the sensory data to predict these mean preference scores for each cluster, estimating the generalization performance using leave one out cross validation (Section 2.5.1) for a simple linear regression model (Section 2.2.1). We select the sensory features for the model using forward sequential selection (Section 2.4.2). The final error of each model is the regression error for each cluster, weighted by the proportion of the data assigned to the cluster, summed over all the clusters. Thus the final model error  $m_e$  is given by:

$$m_e = \sum_{i=1}^k \pi_i l_i,$$

where  $\pi_i$  is the proportion of data assigned to the  $i^{\text{th}}$  cluster,  $l_i$  is the cross validation error of the  $i^{\text{th}}$  regression model, and there are  $k$  clusters in the model. This weighting is used to avoid being biased towards models that are very accurate, but account for very few consumers.

Note that the feature selection is performed independently for each cluster model. Although some features were used more often than others, we found that different feature sets tended to be selected for each cluster.

Figure 3.22 shows the resultant weighted error, when segmenting the beverage data into between one and 15 clusters. The results shown are the average of 50 experiments. Whether we use  $k$ -means or GMM, the error clearly rises as we increase the number of clusters. Both versions of the GMM are at least as good as the  $k$ -means (solid line) on this data set. According to independent  $t$ -tests, the spherical GMM (dashed line) is significantly better (at the 1% level) than  $k$ -means on 10 out of 15 values of  $k$ . The fully elliptical GMM (dotted line) is significantly better than  $k$ -means in 14 cases. The elliptical GMM is significantly better than the spherical GMM 9 times. These  $t$ -test results (and those below) have been confirmed as significant by the equivalent Wilcoxon signed ranks tests, which makes weaker assumptions about the distribution of the samples.

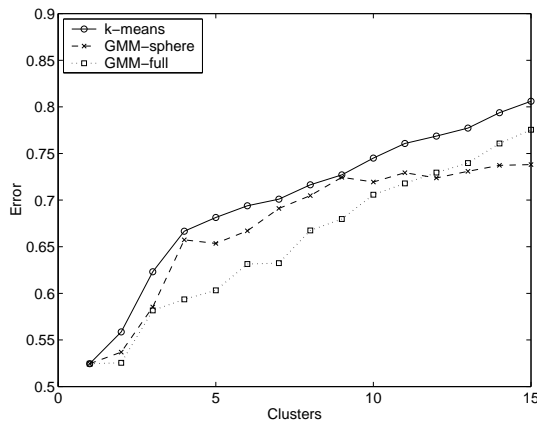


Figure 3.22: Clustering with regression: beverage data

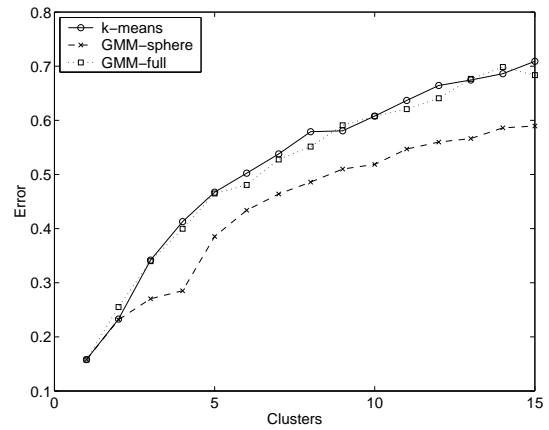


Figure 3.23: Clustering with regression: meat data

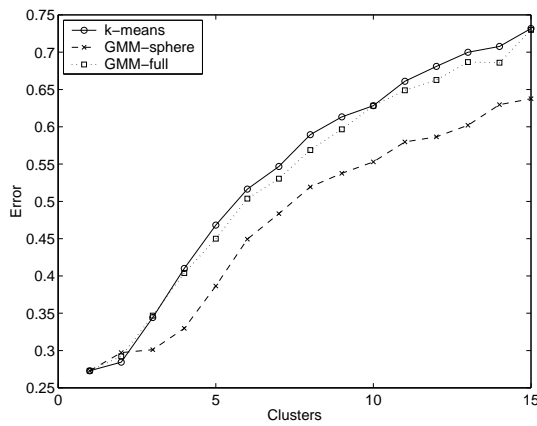


Figure 3.24: Clustering with regression: vegetable data

Figure 3.23 shows the equivalent results for the meat data set, where the spherical GMM produces significantly more accurate regression models than both  $k$ -means and non-spherical GMM in 13 out of the 15 cases. The non-spherical GMM is significantly better than  $k$ -means in just two cases.

Figure 3.24 shows the equivalent graph for the vegetable data, where the spherical GMM produces significantly more accurate regression models 14 out of 15 times, compared with both elliptical GMM and  $k$ -means. The elliptical GMM is significantly better than  $k$ -means just once.

To summarise these results, Table 3.5 shows the ratio between each the regression error of the three clustering methods, and the lowest regression error found, averaged across the 15 cases. A value of exactly one shows that the method was never beaten by the others. Thus we can see that for the beverage data, the fully elliptical GMM was

	Beverage	Meat	Vegetable
<i>k</i> -means	1.076	1.178	1.135
GMM sphere	1.035	1.000	1.003
GMM full	1.007	1.167	1.117

Table 3.5: Accuracy of weighted cluster regression with respect to best result found

best, although the other two methods were not much worse. For the other two food data sets, the spherical GMM was better than the two alternatives.

From these results, GMMs appear to produce more useful cluster models than *k*-means, irrespective of the value of *k* used, or the inherent number of clusters present. Unfortunately, all three clustering methods produce regression errors that rise as we increase the number of clusters. This suggests that just one cluster is present, which contradicts both the minimum description length results (Section 3.6.2) and the consistency results (Section 3.8), and provides little useful information. By dividing the consumers into groups before attempting regression, the data we are using to estimate the regression parameters contain less information, as the mean preference is the mean of a smaller set of consumers. This could explain why the error increases as we increase the number of clusters.

The poorer performance of the full elliptical GMMs on two of the data sets is presumably due to poor parameter estimation during clustering, owing to a lack of data. For example, when splitting the vegetable preference data into 15 clusters, each cluster contained between one and 27 consumers, so each regression model is based on very few consumers' preferences. For the beverage data set, the non-spherical GMMs were best, suggesting that the clusters present are indeed non-spherical. However, this data also has more records (450, compared with 240 and 210 for the meat and vegetable sets respectively), and it could be that given more consumers, the non-spherical GMM would have produced more accurate regression models from the other sets as well.

### 3.10 Conclusions

We can never determine with absolute certainty the intrinsic number of clusters in a distribution given a finite sample of data. Moreover, the degree of confidence we have in the results depends largely on the number of records: given a small amount of data, as we have here, any decision about the number of clusters should be made with caution.

As with regression, the limited amount of information available in a small sample biases solutions towards simpler models, whether or not this is true in the population as a whole. For example, there appear to be three or four clusters in meat preferences, if we assume spherical clusters — with more complex models, there may appear to be

fewer clusters (Section 3.6.3).

We have shown that our proposed use of consistency to estimate the number of clusters present in a sample is useful — but only when several clusters exist. This method assumes that the data is not pathological, in the sense of having a solution that is locally optimal and easy to find, but globally sub-optimal. By relying on fewer assumptions, this technique is more widely applicable than the minimum description length principle for estimating the number of clusters present.

The results of weighted clustering regression suggest that Gaussian mixture models produce more accurate models than  $k$ -means clustering (Section 3.9.1), with the costs of extra computational expense and of requiring more parameters to be estimated (Section 3.4.6). The principal differences between  $k$ -means and spherical GMMs are 1) that the GMM allows clusters to vary in size, in terms of both their radius and their prior probability; and 2) that GMM allows clusters to overlap. The relative performance of spherical and non-spherical GMMs gives some indication of the shape of the intrinsic clusters, and hence of the nature of sensory experience. A non-spherical model means that preferences expressed for one product are not independent of those expressed for other related products, and points to the existence of factors shared between products.

We can conclude that the clusters of consumers are different sizes, because GMMs tend to out-perform  $k$ -means clustering. Furthermore, we can conclude that the clusters are spherical, given that spherical GMMs tend to outperform elliptical GMMs. However, this last result could be a result of using small data sets: for the largest preference set, elliptical GMMs outperform spherical GMMs, and we have shown that using complex models with small data sets can produce misleading results (Section 3.4.6). Finally, we can conclude that the clusters of consumers overlap to some extent, because GMMs tend to outperform  $k$ -means models. Following Section 3.4.6, we suggest that we should regard consumers' preferences as being generated by some underlying processes, common to the population.