Chapter 6
K-Means and Related Clustering Methods

6.1 General

Clustering is a set of methods for finding and describing cohesive groups in data, typically, as "compact" clusters of entities in the feature space.

Consider data patterns on Fig. 6.1: a clear-cut cluster structure on part (a), a blob on (b), and an ambiguous "cloud" on (c).

Some argue that term "clustering" applies only to structures of the type presented on Fig. 6.1a, c, moreover, depending on the resolution, one may distinguish 3 or 7 clusters on (c). Yet there are no "natural" clusters in the other two cases, Fig. 6.1a and 6.1b. Indeed, initially the term was used to express a clear-cut clustering. But currently clustering has become synonymous to building a classification over empirical data, and as such it embraces all the situations in which data is structured into cohesive chunks.

To serve as models of natural classes and categories, clusters need to be not only found but conceptually described as well. A class always expresses a concept embedded into a fragment of knowledge – this is what is referred to, in logics, as the class' "intension", in contrast to empirical instances of the class constituting what is referred to as the class' "extension", e.g., the concept of "tree" versus real plants growing here and there. Therefore, two dual intelligent activities – cluster finding and cluster describing – should be exercised both when clustering.

As Fig. 6.2 illustrates, a cluster is rather easy to describe by combining corresponding feature intervals when it is clear-cut. This knowledge-driven data analysis perspective can be reflected in dividing all cluster finding techniques in the following categories:

(a) clusters are to be found directly in terms of features – this is frequently referred to as conceptual clustering;
(b) clusters are to be found simultaneously with a transformation of the feature space making them clear-cut – this direction only started very recently and is not well shaped yet;
(c) clusters are to be found as subsets of entities first, so that the description comes as a follow-up stage – this is the genuine clustering which covers most of the clustering activities so far.

B. Mirkin, *Core Concepts in Data Analysis: Summarization, Correlation and Visualization*, Undergraduate Topics in Computer Science,
DOI 10.1007/978-0-85729-287-2_6, © Springer-Verlag London Limited 2011

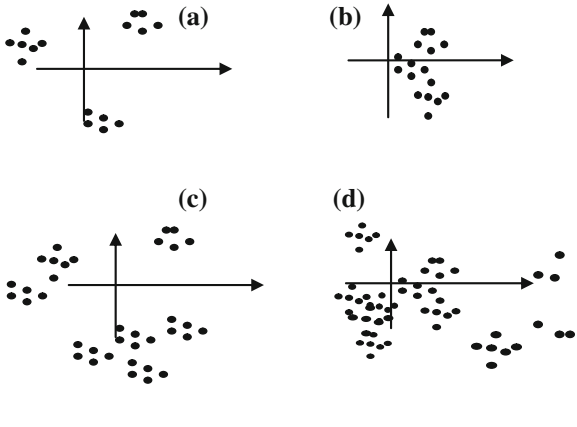

Fig. 6.1 Clear-cut cluster structures at (**a**) and (**c**); data clouds with no clear structure at (**b**) and (**d**)

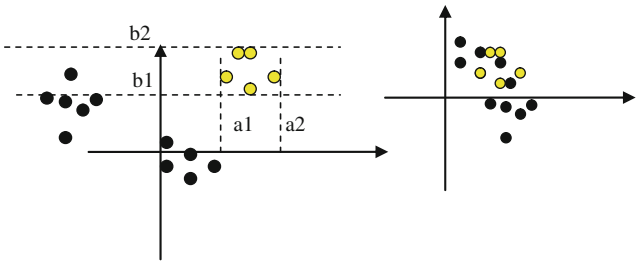

Fig. 6.2 Cluster of *blank circles* on the right is well described by the predicate a1<x<a2 & b1<y<b2. A similar cluster on the right cannot be accurately described by interval predicates without false positive and false negative errors

6.2 K-Means Clustering

P6.2.1 Batch K-Means Partitioning

K-Means is a major clustering technique, of type (c), that is present, in various forms, in all major statistical packages such as SPSS and SAS, as well as data mining packages such as Clementine, iDA tool and DBMiner. It is very popular in many application areas such as image analysis, marketing research, bioinformatics, and medical informatics.

In general, the cluster finding process according to K-Means starts from K tentative centroids and repeatedly applies two steps:

(a) collecting clusters around centroids,
(b) updating centroids as within cluster means,

– until convergence.

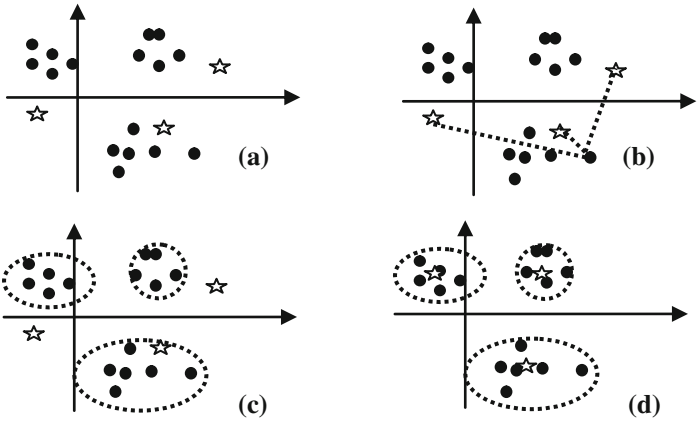

Fig. 6.3 Main steps of Batch K-Means: (**a**) initialization of centroids, (**b**) cluster update using Minimum distance rule (the pointed lines show distances from an entity to all centroids), (**c**) cluster update completed, (**d**) centroid update completed

This makes much sense – whichever centroids are suggested first, as hypothetical cluster tendencies, they are checked then against the real data and moved to the areas of higher density.

In its generic, so-called Batch mode, K-Means can be formulated as comprising the following steps 0–3 illustrated on Fig. 6.3 for $K = 3$ and entity set I:

0. *Initialization*: the user chooses the number K of clusters and puts K hypothetic cluster centroids among the entity points, see Fig. 6.3a;
1. *Cluster update*: Given K centroids c_k ($k = 1, 2, \ldots, K$), each of the entities $i \in I$ is assigned to one of the centroids according to Minimum distance rule: distances between i and each c_k are calculated, and i is assigned to the nearest c_k, see Fig. 6.3b. For each centroid c_k, the entities assigned to it form cluster S_k ($k = 1, 2, \ldots, K$), see Fig. 6.3c.
2. *Centroid update*: At each of the given K clusters S_k, its gravity center is computed and set as the new centroid c_k' ($k = 1, 2, \ldots, K$), see Fig. 6.3d.
3. *Halting test*: New centroids c_k' are compared with those from the previous iteration. If $c_k' = c_k$ for all $k = 1, 2, \ldots, K$, stop and output both c_k' and S_k for all $k = 1, 2, \ldots, K$. Otherwise, set c_k' as c_k and go to "1. Cluster update step".

The algorithm is appealing in several aspects. Conceptually it may be considered a model for the human process of typology making, with types represented by clusters S_k and centroids c_k. Also, it has nice mathematical properties. This method is computationally easy, fast and memory-efficient. However, researchers and practitioners point to some less desirable properties of K-Means. Specifically, they refer to lack of advice with respect to

(a) the initial setting, i.e. the number of clusters K and initial positioning of centroids,

(b) instability of clustering results with respect to the initial setting and data standardization, and
(c) insufficient interpretation aids.

These issues can be alleviated, to an extent, as will be explained later in this section.

A decoder based summarization model underlying the method is that the entities are assigned to clusters in such a way that each cluster is represented by its centroid, sometimes referred to as the cluster's standard point or prototype. This point expresses, intensionally, the typical tendencies of the cluster.

Worked example 6.1. K-Means clustering of Company data

Consider the standardized Company data in Table 5.9 copied here as Table 6.1.

This data set can be visualized with two principal components as presented on Fig. 6.4 (copied from Section 5.2.2).

For example, let entities An, Br and Ci be suggested centroids of three clusters. Now we can computationally compare each of the entities with each of the centroids to decide which centroid better represents an entity. To compare two points, Euclidean squared distance is a natural choice (see Table 6.2).

According to the Minimum distance rule, an entity is assigned to its nearest centroid (see Table 6.3 in which all distances between the entities and centroids are presented; those chosen according to the Minimum distance rule are highlighted in bold.) Entities assigned to the same centroid form a tentative cluster around it. Clusters found at Table 6.3 are

S1 = {Av,An,As}, S2 = {Ba,Br,Bu}, and S3 = {Ci,Cy}. These are the product classes already, but this is irrelevant to the computation. K-Means procedure has its own logic that needs to ensure that the new tentative clusters lead to the same centroids.

Table 6.1 The company data standardized by: (i) shifting to the feature averages, (ii) dividing by the feature ranges, and (iii) further dividing the category based three columns by 3. Contributions of the features to the data scatter are presented in the bottom

Av	−0.20	0.23	−0.33	−0.63	0.36	−0.22	−0.14
An	0.40	0.05	0	−0.63	0.36	−0.22	−0.14
As	0.08	0.09	0	−0.63	−0.22	0.36	−0.14
Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14
Br	0.19	−0.29	0	0.38	−0.22	0.36	−0.14
Bu	−0.60	−0.42	−0.33	0.38	−0.22	0.36	−0.14
Ci	0.08	−0.10	0.33	0.38	−0.22	−0.22	0.43
Cy	0.27	0.58	0.67	0.38	−0.22	−0.22	0.43
Cnt	0.74	0.69	0.89	1.88	0.62	0.62	0.50
Cnt %	12.42	11.66	14.95	31.54	10.51	10.51	8.41

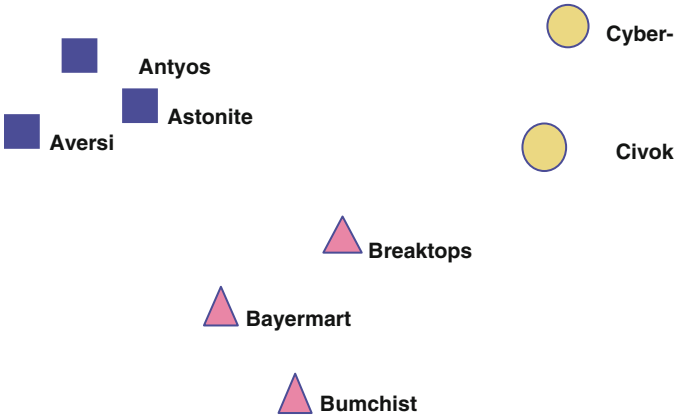

Fig. 6.4 Table 6.1 rows on the plane of two first principal components: it should not be difficult to discern clusters formed by products: distances within A, B and C groups are smaller than between them

Table 6.2 Computation of squared Euclidean distance between rows Av and An in Table 6.1 as the sum of squared differences between corresponding components

Points	Coordinates							d(An,Av)
An	0.40	0.05	0.00	−0.63	0.36	−0.22	−0.14	
Av	−0.20	0.23	−0.33	−0.63	0.36	−0.22	−0.14	
An-Av	0.60	−0.18	0.33	0	0	0	0	
Squares	0.36	0.03	0.11	0	0	0	0	0.50

Table 6.3 Distances between three company entities chosen as centroids and all the companies; each company column shows three distances between the company and centroids – the highlighted minima present best matches between centroids and companies

Point	Av	An	As	Ba	Br	Bu	Ci	Cy
An	**0.50**	**0.00**	**0.77**	1.55	1.82	2.99	1.90	2.41
Br	2.20	1.82	1.16	**0.97**	**0.00**	**0.75**	0.83	1.87
Ci	2.30	1.90	1.81	1.22	0.83	1.68	**0.00**	**0.61**

One needs to proceed further on and update centroids by using the information of the assigned clusters. New centroids are defined as centers of the tentative clusters, whose components are the averages of the corresponding components within the clusters; these are presented in Table 6.4.

The updated centroids differ from the previous ones. Thus we must update their cluster lists by using the distances between updated centroids and entities; the distances are presented in Table 6.5. As it is easy to see, the Minimum distance rule assigns centroids again with the same entity lists. Therefore, the process has stabilized – if we repeat it all over again, nothing new would ever come – the same

Table 6.4 Tentative clusters from Table 6.3 and their centroids

Av	−0.20	0.23	−0.33	−0.63	0.36	−0.22	−0.14
An	0.40	0.05	0	−0.63	0.36	−0.22	−0.14
As	0.08	0.09	0	−0.63	−0.22	0.36	−0.14
Centroid1	0.10	0.12	−0.11	−0.63	0.17	−0.02	−0.14
Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14
Br	0.19	−0.29	0	0.38	−0.22	0.36	−0.14
Bu	−0.60	−0.42	−0.33	0.38	−0.22	0.36	−0.14
Centroid2	−0.21	−0.29	−0.22	0.38	−0.02	0.17	−0.14
Ci	0.08	−0.10	0.33	0.38	−0.22	−0.22	0.43
Cy	0.27	0.58	0.67	0.38	−0.22	−0.22	0.43
Centroid3	0.18	0.24	0.50	0.38	−0.22	−0.22	0.43

Table 6.5 Distances between the three updated centroids and all the companies; the highlighted column minima present best matches between centroids and companies

Point	Av	An	As	Ba	Br	Bu	Ci	Cy
Centroid1	**0.22**	**0.19**	**0.31**	1.31	1.49	2.12	1.76	2.36
Centroid2	1.58	1.84	1.36	**0.33**	**0.29**	**0.25**	0.95	2.30
Centroid3	2.50	2.00	1.95	1.69	1.20	2.40	**0.15**	**0.15**

centroids and the same assignments. The process stops at this point, and the found clusters along with their centroids are returned (they are, in the standardized format, in Table 6.4).

The result obviously depends on the standardization of the data performed beforehand, as the method heavily relies on the squared Euclidean distance and, thus, on the relative weighting of the features, just like PCA.

Case Study 6.1. Dependence of K-Means on Initialization: A Drawback and Advantage

The bad news is that the result of K-Means depends on initialization – the choice of the initial tentative centroids, even if we know, or have guessed correctly, the number of clusters K. Indeed, if we start from wrong entities as the tentative centroids, the result can be rather disappointing.

In some packages, such as SPSS (Green and Salkind 2003), K first entities are taken as the initial centroids. Why not start from rows for Av, An and As then? Taking these three as the initial tentative centroids will stabilize the process at wrong clusters S1 = {Av, Ba, Br}, S2 = {An}, and S3 = {As, Bu, Ci, Cy} (See Q.6.5). But what else can be expected if all centroids are taken from the same cluster?

However, even if centroids are taken from right clusters, this would not necessarily guarantee good results either. Start, for example, from Av, Ba and Ci (note, these produce different products!); the final result still will be rather disappointing – S1={Av,An,As}, S2={Ba, Bu}, and S3={Br, Ci, Cy} (see Q.6.6).

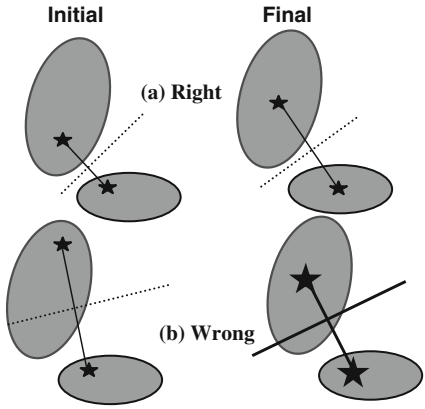

Fig. 6.5 Case of two clear-cut clusters and two different initializations: (**a**) and (**b**). Case (**a**) results in a correct separation of the clusters, case (**b**) does not

Figure 6.5 illustrates the fact that such instability is not because of a specially designed example but rather an ordinary phenomenon. There are two clear-cut clusters on Fig. 6.5, that can be thought of as uniformly distributed sets of points, and two different initializations, symmetric one on the (a) part and not symmetric one on (b) part. The Minimum distance rule at $K=2$ amounts to drawing a hyperplane that orthogonally cuts through the middle of line between the two centroids; the hyperplane is shown on Fig. 6.5 as the line separating the centroids. In Fig. 6.5, the case (a) presents initial centroids that are more or less symmetric so that the line through the middle separates the clusters indeed. In the case (b), initial centroids are highly asymmetric so that the separating line cuts through one of the clusters, thus distorting the position of the further centroids; the final separation still cuts through one of the clusters and, therefore, is utterly wrong.

There is one more property of K-Means clusters illustrated by Fig. 6.5: they are convex. Indeed, the Minimum Distance rule assigns each centroid with the intersection of half-spaces formed by the orthogonal cutting hyperplanes.

Another example of non-optimality of K-Means is presented on Fig. 6.6 which involves four points only.

Yet non-optimality of K-Means can be of an advantage, too – in those cases when K-Means criterion leads to solutions that are counterintuitive such as those in which the fact that K-Means favors equal cluster sizes brings good results.

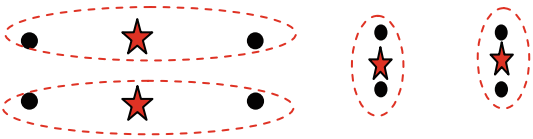

Fig. 6.6 An example of K-Means failure: two clusterings at a four-point-set with K-Means – that intuitive on the *right* and that counter-intuitive on the *left*, with *stars* as centroids

Case Study 6.2. Uniform Clusters Can Be Too Costly

Here is an example when the square-error clustering criterion leads to a solution which is at odds with intuition, however that cannot be reached with batch K-Means algorithm because of its local nature.

Consider the case of Fig. 6.7 that presents three sets of points, two consisting of big clumps of say 100 entities each, around points A and B, and a small one around point C, consisting say of just one entity located at that point. Assume that the distance between A and B is 2, and between B and C, 10. There can be only two 2-cluster partitions possible: (I) 200 of A and B entities together in one cluster while the second cluster consists of just one entity in C; (II) 100 A entities for one cluster while 101 entities in B and C for the other. The third partition, consisting of cluster B and cluster A+C, cannot be optimal because cluster A+C is more outstretched than a similar cluster B+C in (II).

Let us compare the values of K-Means criterion using the squared Euclidean distance between entities and their centroids.

In case (I), centroid of cluster A+B will be located in the middle of the interval between A and B, thus on the distance 1 from each, leading to the total squared Euclidean distance $200^*1 = 200$. Since cluster C contains just one entity, it adds 0 to the value of K-Means criterion, which is 200 in this case.

In case (II), cluster B+C has centroid, which is the gravity center, between B and C distanced from B by $d = 10/101$. Thus, the total value of K-Means criterion here is $100^*d^2 + (10\text{-}d)^2$ which is less than $100^*(1/10)^2 + 10^2 = 101$ because d<1/10 and 10-d<10. Cluster A contributes 0 because all 100 entities are located in A which is, therefore, the centroid of this cluster.

Case (II) wins by a great margin: K-Means criterion, in this case, favors more equal distribution of entities between clusters in spite of the fact that case (I) is intuitive and case (II) is not: A and B are much closer together than B and C.

Yet Batch K-means algorithm leads to non-optimal, but intuitive, case (I) rather than optimal, but odd, case (II), if started from the most distant points A and C as initial centroids – an intuitively appealing option. Indeed, according to Minimal distance rule, all entities in B will join A, thus resulting in (I) clustering.

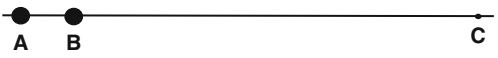

Fig. 6.7 Three sets of points subject to 2-Means clustering: which two will join together, A and B or B and C?

Case Study 6.3. Robustness of K-Means Criterion with Data Normalization

Let us generate two 2D clusters, of 100 and 200 elements. First cluster – a Gaussian spherical distribution with the mean in point (1,1) and the standard deviation 0.5. The second cluster, of 200 elements, is uniformly randomly distributed in the

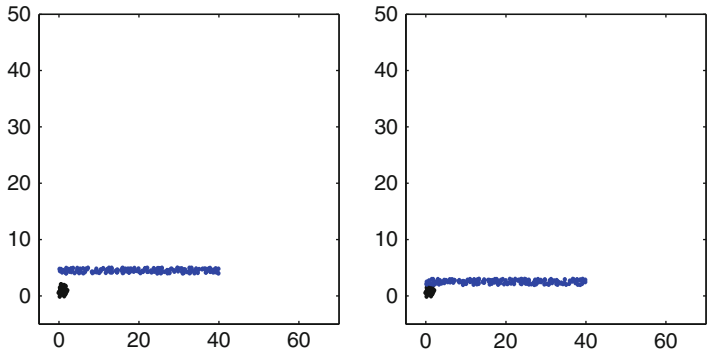

Fig. 6.8 A set of two differently shaped clusters, a *circle* and *rectangle*; the *y*-coordinates of their centers are 1 (*circle*) and 5.5 (*rectangle*) on the *left*, and 1 and 3.5, on the *right*

rectangle of the length 40 and width 1, put over axis x either at $y = 5$ (Fig. 6.8, on the left) or at $y = 3$ (Fig. 6.8, on the right). K-Means criterion of course cannot separate these two if applied in the original space; its criterion value will be minimized by dividing the set somewhere closer to one fourth of the strip of rectangular cluster so that the split parts have approximately 150 points each.

Yet after the data standardization, with the grand means subtracted and range-normalized, the clusters on the left part of Fig. 6.8 are perfectly separable with K-Means criterion, that does attain its minimum value, at $K = 2$, on the cluster-based partition of the set. This holds with z-scoring, too.

This tendency changes, though, at a less structured case on the right of Fig. 6.8. The best split indeed holds at about $x = 10$ in this case. At a random sample, 32 points of the rectangular cluster join circular cluster in the optimal split. Curiously, the z-scoring standardization, in this case, works towards a better recovery of the structure so that the optimal 2-cluster partition, at the same data sample, merges only 5 rectangular cluster elements into the circular cluster, thus splitting the rectangular cluster over a mark $x = 5$.

These results do not much change when we go to Gaussian similarities (affinity data) defined by formula $G(x, y) = e^{-d(x,y)/2\sigma^2}$ where $d(x,y)$ is the squared Euclidean distance between x and y if $x \neq y$, and σ^2 is equal to 10 at the original data and 0.5 at the standardized data, in the manner of spectral clustering (see Section 8.2) and apply algorithm AddRem from Section 8.3.

F6.2.2 Batch K-Means and Its Criterion: Formulation

F6.2.2.1 Batch K-Means as Alternating Minimization

The cluster structure in K-Means is specified by a partition S of the entity set in K non-overlapping clusters, $S = \{S_1, S_2, \ldots, S_K\}$ represented by lists of entities S_k, and cluster centroids $c_k = (c_{k1}, c_{k2}, \ldots, c_{kV})$, $k = 1, 2, \ldots, K$.

There is a model that can be thought of as that driving K-Means algorithm. According to this model, each entity, represented by the corresponding row of Y matrix as $y_i = (y_{i1}, y_{i2}, \ldots, y_{iV})$, belongs to a cluster, say S_k, and is equal, up to small residuals, or errors, to the cluster's centroid:

$$y_{iv} = c_{kv} + e_{iv} \text{ for all } i \in S_k \text{ and all } v = 1, 2, \ldots, V \tag{6.1}$$

Equations (6.1) define as simple a decoder as possible: whatever entity belongs to cluster S_k, its data point in the feature space is decoded as centroid c_k.

The problem is to find such a partition $S = \{S_1, S_2, \ldots, S_K\}$ and cluster centroids $c_k = (c_{k1}, c_{k2}, \ldots, c_{kV})$, $k = 1, 2, \ldots, K$, that minimize the square error of decoding

$$L^2 = \sum_{i \in I} \sum_{v \in V} e^2{}_{iv} = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v \in V} (y_{iv} - c_{kv})^2 \tag{6.2}$$

Criterion (6.2) can be equivalently reformulated in terms of the squared Euclidean distances as the summary distance between entities and their cluster centroids (see (6.3)) (see Fig. 6.9). Please note that the number of distances in the sum is N and does not depend on the number of clusters.

$$L^2 = W(S, c) = \sum_{k=1}^{K} \sum_{i \in S_k} d(y_i, c_k) \tag{6.3}$$

This is because the distance referred to as squared Euclidean distance is defined, for any V-dimensional $x = (x_v)$ and $y = (y_v)$ as $d(x, y) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_2 - y_2)^2 + \ldots + (x_V - y_V)^2$ so that the rightmost summation symbol in (6.2) leads to $d(y_i, c_k)$ indeed.

This criterion depends on two groups of variables, S and c, and thus can be minimized by the alternating minimization method which proceeds by repetitively applying the same minimization step: Given one group of variables, optimize criterion over the other group, and so forth, until convergence.

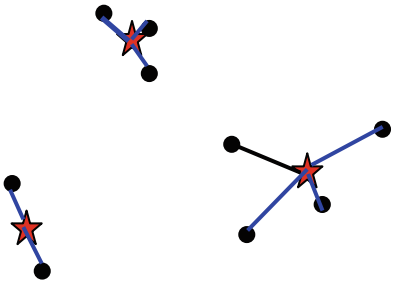

Fig. 6.9 The distances – intervals connecting centroids with entity points – in criterion *W(S,c)*

Specifically, given centroids $c = (c_1, c_2, \ldots, c_K)$, find a partition S minimizing the summary distance (6.3). Obviously, to choose a partition S, one should choose, for each entity $i \in I$, one of K distances $d(y_i, c_1), d(y_i, c_2), \ldots, d(y_i, c_K)$. The choice to minimize (6.3) is according to the Minimum distance rule: for each $i \in I$, choose the minimum of $d(y_i, c_k)$, $k = 1, \ldots, V$, that is, assign any entity to its nearest centroid. When there are several nearest centroids, the assignment is taken among them arbitrarily. In general, some centroids may be assigned no entity at all with this rule.

The other step in the alternating minimization would be minimizing (6.3) over c at a given S. The solution to this problem comes from the additive format of criterion (6.2) that provides for the independence of cluster centroid components from each other. As was indicated in Section 2.2.2, it is the mean that minimizes the square error, and thus the within-cluster mean vectors minimize (6.3) over c at given S.

Thus, starting from an initial set of centroids, $c = (c_1, c_2, \ldots, c_K)$, the alternating minimization method for criterion (6.3) will consist of a series of repeated applications of two steps: (a) clusters update – find clusters S according to the Minimum distance rule, (b) centroids update – make centroids equal to within cluster mean vectors. The computation stops when new clusters coincide with those on the previous step. This is exactly the K-Means in its Batch mode.

The convergence of the method follows from two facts: (i) at each step, criterion (6.3) can only decrease, and (ii) the number of different partitions S is finite.

F6.2.2.2 Various Formulations of K-Means Criterion

Let us consider any S_k and define c_k being within-cluster means ($k = 1, 2, \ldots, K$) so that $c_{kv} = \sum_{i \in Sk} y_{kv}/N_k$ where N_k is the number of entities in S_k. Multiply then each equation in (6.1) by itself and sum the resulting equations over both entities and features. The result will be the following equation:

$$\sum_{i \in I} \sum_{v \in V} {y^2}_{iv} = \sum_{k=1}^{K} \sum_{v \in V} {c_{kv}}^2 N_k + \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{v \in V} {e_{iv}}^2 \tag{6.4}$$

The three items in (6.4) come from summation of the products of the equations in (6.1) by themselves. The remaining sum $2\Sigma_v \Sigma_k c_{kv} \Sigma_{i \in Sk} e_{kv}$ is zero because $\Sigma_{i \in Sk} e_{kv} = \Sigma_{i \in Sk}(y_{iv} - c_{kv}) = c_{kv} N_k - c_{kv} N_k = 0$. This proves (6.4). Note that the item on the left in (6.4) is just the data scatter *T(Y),* whereas the right-hand item is the least-squares criterion of K-Means (6.2). Therefore, Equation (6.4) can be reformulated as

$$T(Y) = B(S, c) + W(S, c) \tag{6.5}$$

where *T(Y)* is data scatter, *W(S,c)* the least-squares clustering criterion expressed as the summary within cluster distance (6.3) and *B(S,c)* is clustering's contribution to the data scatter:

$$B(S,c) = \sum_{k=1}^{K} \sum_{v \in V} {c_{kv}}^2 N_k \tag{6.6}$$

Pythagorean Equation (6.5) decomposes data scatter *T(Y)* in two parts: that one explained by the cluster structure (*S,c*), which is *B(S,c)*, and the unexplained part which is *W(S,c)*. The larger the explained part, the smaller the unexplained part, and the better the match between clustering *(S,c)* and data. Equation (6.5) is well known in the analysis of variance in statistics; items *B(S,c)* and *W(S,c)* are referred to in that other context as between-group and within-group variance.

Criteria *W(S,c)* and *B(S,c)* admit different equivalent reformulations that could lead to different systems of neighborhoods and local algorithms for minimization of *W(S,c)* which may have been never attempted yet. Take, for example, the criterion of maximization of clustering's contribution to the data scatter (6.6). Since the sum of ${c_{kv}}^2$ over v is but the squared Euclidean distance between 0 and c_k, one has

$$B(S,c) = \sum_{k=1}^{K} \sum_{v \in V} {c_{kv}}^2 N_k = \sum_{k=1}^{K} N_k d(0, c_k) \tag{6.7}$$

The criterion on the right in formula (6.7) was first mentioned, under the name of "criterion of distant centers", by Mirkin (1996, p. 292). To maximize the criterion on the right in formula (6.7), the clusters should be as far away from 0 as possible. This idea may lead to a "parallel" version of the Anomalous Pattern method described later in Section 6.2.7.

Another expression of the cluster-explained part of the data scatter is

$$B(S,c) = \sum_{k=1}^{K} \sum_{i \in S_k} < y_i, c_k > \tag{6.8}$$

which can be derived from (6.6) by taking into account that the internal sum $\Sigma_v {c_{kv}}^2$ in (6.6) is in fact the inner square $< c_k c_k >$ and substituting instead of one its expression as within cluster average $< c_k, c_k > = < c_k, \Sigma_{i \in Sk} y_{kv}/N_k > = \Sigma_{i \in Sk} < c_k, c_k > /N_k$. This expression shows that the K-Means criterion of minimizing within-cluster distances to centroids is equivalent to criterion of maximizing within-cluster inner products with centroids – they sum to the data scatter which does not depend on the clustering. Note that the distance based criterion makes sense at any set of centroids whereas the inner product based criterion makes sense only when centroids are within-cluster averages. As is well known, the distance does not depend on the location of the space origin whereas the inner product heavily depends on that – only special arrangements are suitable for the latter.

In this regard, it deserves to be mentioned that *W(S,c)* can be reformulated in terms of entity-to-entity distances or similarities only – without any reference to centroids at all. One can prove that minimization of K-Means criterion is equivalent to minimization of *D(S)* or maximization of *C(S)* defined by

$$D(S) = \sum_{k=1}^{K} \sum_{i,j \in S_k} d(y_i, y_j)/N_k \tag{6.9}$$

$$C(S) = \sum_{k=1}^{K} \sum_{i,j \in S_k} a_{ij}/N_k \tag{6.10}$$

where $d(y_i, y_j)$ is the squared Euclidean distance between i and j's rows and $a_{ij} = <y_i, y_j>$ is the inner product of them. Both follow from the expression $d(y_i, y_j) = <y_i - y_j, y_i - y_j> = <y_i, y_i> + <y_j, y_j> - 2<y_i, y_j>$ and the definition of the centroid of S_k as $\Sigma_{i \in Sk} y_i/N_k$. These formulations suggest algorithms for optimization based on exchanges and mergers between clusters.

A most unusual reformulation can be stated as a criterion of consensus among the features. Consider a measure of association $\zeta(S,v)$ between a partition S of the entity set I and a feature v, $v = 1, 2, \ldots, V$. Consider, for the sake of simplicity, that all features are quantitative and have been standardized by z-scoring, then $\zeta(S,v)$ is the correlation ratio η^2 defined by formula (3.15). Then maximizing the summary association

$$\zeta(S) = \Sigma_{v \in V} \eta^2(S, v) \tag{6.11}$$

is equivalent to minimization of the K-Means least squares criterion $W(S,c)$ involving the squared Euclidean distances. A very similar equivalent consensus criterion can be formulated for the case when feature set consists of categorical or even mixed scale features. For the case of all features being categorical so that the categories are represented by dummy variables the total contribution to the data scatter in (6.4) can be formulated as

$$\varphi(S) = \Sigma_{v \in V} \varphi(S, v) \tag{6.11'}$$

where $\varphi(S,v)$ can be an association measure such as Pearson chi-squared or Gini index – see more detail in Section 4.5.3.

Since clusters are not overlapping, model in (6.1) can be rewritten differently in such a way that no explicit references are made over individual clusters. To do that, let us introduce N-dimensional membership vectors $z_k = (z_{ik})$ such that $z_{ik} = 1$ if $i \in S_k$ and $z_{ik} = 0$, otherwise. Using this notation allows us to use the following reformulation of the model. For any data entry, the following equation holds:

$$y_{iv} = \sum_{k=1}^{K} c_{kv} z_{ik} + e_{iv}, \tag{6.12}$$

Indeed, since any entity $i \in I$ belongs to one and only one cluster S_k, only one of $z_{i1}, z_{i2}, \ldots, z_{iK}$ can be non-zero, that is, equal to 1, at any given i, which makes (6.12) equivalent to (6.1).

Yet (6.12) makes the clustering model similar to that of PCA in (5.14) except that z_{ik} in PCA are arbitrary values to score hidden factors, whereas in (6.12) z_{ik} are to be 1/0 binary values: it is clusters, not factors, that are of concern here. That is, clusters in model (6.12) correspond to factors in model (5.14).

The decomposition (6.4) of data scatter into explained and unexplained parts is similar to that in (5.17) making the contributions of individual clusters $\sum_{v \in V} c_{kv}^2 N_k$ akin to contributions μ_k^2 of individual principal components. More precisely, μ_k^2 in (5.17) are eigen-values of YY^T, that can be expressed thus with the analogous formula,

$$\mu_k^2 = z_k^T Y Y^T z_k / z_k^T z_k = \Sigma_v c_{kv}^2 |S_k| \tag{6.13}$$

in which the latter equation is due to the fact that vector z_k here consists of binary 1/0 entries.

Q.6.1. How many distances are summed up in *W(S, c)*? (**A:** This is equal to the number of entities *N*.) Does this number depend on the number of clusters *K*? (**A**: No.) Does this imply: the greater the *K*, the less the *W(S, c)*? (**A:** Yes.) Why?

Q.6.2. What is the difference between PCA model (5.14) and clustering model (6.12)? **A.** The scores z are constrained in (6.12) to take only binary values – either 1 or 0.

Q.6.3. Why is convergence guaranteed for K-Means? **A.** Because K-Means is alternating minimization process at which criterion *W(S,c)* may only decrease at each step. Convergence follows from the fact that there are only a finite number of different partitions on *I*.

Q.6.4. Assume that $d(y_i, c_k)$ in *W(S, c)* is city-block distance rather than Euclidean squared. Could K-Means be adjusted to make it alternating minimization algorithm for the modified *W(S,c)*? **A.** Yes, just use the city-block distance through, as well as within cluster median points rather than gravity centers.) Would this make any difference? (Yes, it will; especially at skewed distributions of the variables.)

Q.6.5. Demonstrate that, at Companies data, value *W(S,c)* at product-based partition {1-2-3, 4-5-6, 7-8} is lower than at partition {1-4-6, 2, 3-5-7-8} found at seeds 1, 2 and 3. **A.** Indeed the sums of within-cluster distances to cluster centroids in the product based clusters are 0.7193, 0.8701, 0.3070, respectively, totaling to 1.8964, whereas the sums ot the second partition are 1.4411, 0, 2.1789 and sum to 3.62.

Q.6.6. Demonstrate that, at Companies data, value *W(S,c)* at product-based partition {1-2-3, 4-5-6, 7-8} is lower than at partition {1-2-3, 4-6, 5-7-8} found at seeds 1, 4 and 7. **A.** Indeed the sums of within-cluster distances to cluster centroids in the product based clusters are 0.7193, 0.8701, 0.3070, respectively, totaling to 1.8964, whereas the sums ot the second partition are 0.7193, 0.4413, 1.1020 that total to 2.2626.

Q.6.7. Can example of Fig. 6.6 or its modification lead to a similar effect for the case of least-modules criterion related to the city-block distance and median rather than average centroids? Can it be further extended to PAM method which uses city-block distance and median entities rather than coordinates?

Q.6.8. Formulate a version of K-Means to alternatingly maximize criterion (6.8) rather than to minimize (6.3) as the generic version.

Q.6.9. Formulate a version of K-Means to alternatingly maximize criterion (6.7) rather than to minimize (6.3) as the generic version (a "parallel" version of the Anomalous Pattern method). Take care of starting from a most distant set of centroids.

C6.2.3 A Pseudo-Code for Batch K-Means: Computation

To summarize, an application of K-Means clustering involves the following steps:

0. Select a data set.
1. Standardize the data.
2. Choose number of clusters K.
3. Define K hypothetical centroids (seeds).
4. Clusters update: Assign entities to the centroids according to Minimum distance rule.
5. Centroids update: define centroids as the gravity centers of thus obtained clusters.
6. Iterate 4. and 5. until convergence.

MatLab codes for the items 4 and 5 can be put as follows.

4. *Clusters update*: Assign points to the centroids according to Minimum distance rule:

   Given data matrix X and a KxV array of centroids cent, produce an N-dimensional array of cluster labels for the entities and the summary within cluster distance to centroids, wc:

```
function [labelc,wc]=clusterupdate(X,cent)
    [K,m]=size(cent);
    [N,m]=size(X);
    for k=1:K
            cc=cent(k,:); %centroid of cluster k
            Ck=repmat(cc,N,1);
            dif=X-Ck;
            ddif=dif.*dif; %Nxm matrix of squares
            dist(k,:)=sum(ddif');
    %distances from entities to cluster centroid
    end
```

```
        [aa,bb]=min(dist); %Minimum distance rule
        wc=sum(aa);
        labelc=bb;
        return
```

5. *Centroids update*: Put centroids in gravity centres of clusters defined by the array of cluster labels *labelc* according to data in matrix X, to produce KxV array centres of the centroids:

```
function  centres=ceupdate(X,labelc)
        K=max(labelc);
        for k=1:K
      clk=find(labelc==k);
      elemk=X(clk,:);
      centres(k,:)=mean(elemk);
   end
   return
```

Batch K-Means with MatLab, therefore, is to embrace steps 3–6 above and output a clustering in cell array termed, say, Clusters, along with the proportion of unexplained data scatter found by using preliminarily standardized matrix X and set of initial centroids, cent, as input. This can be put like this:

```
function [Clusters,uds]=k_means(X,cent)
 [N,m]=size(X);
 [K,m1]=size(cent);
 flag=0; %-- stop-condition
 membership=zeros(N,1);
 dd=sum(sum(X.*X)); %-- data scatter
 %--- clusters and centroids updates
 while flag==0
     [labelc,wc]=clusterupdate(Y,cent);
     if isequal(labelc,membership)
    %--stop-condition's working
             flag=1;
             centre=cent;
             w=wc;
       else
             cent=ceupdate(Y,labelc);
             membership=labelc;
       end
 end
 %-----preparing the output --------------
  uds=w*100/dd;
  Clusters{1}=membership;
  Clusters{2}=centre;
  return
```

Q.6.10. Check the values of criterion (6.3) at each initial setting considered for Company data above. Find out which is the best among them.

Q.6.11. Prove that the square-error criterion (6.2) can be reformulated as the sum of within cluster variances $\sigma_{kv}{}^2 = \sum_{i \in Sk} (y_{iv} - c_{kv})^2 / N_k$ weighted by the cluster cardinalities N_k.

Q.6.12. Prove that reformulation (6.9) of criterion (6.3) in terms of the squared

Euclidean distances is correct.

Q.6.13. Prove that if Batch K-Means is applied to Iris data mean-range normalized with $K = 3$ and specimens 1, 51, and 101 taken as the initial centroids, the resulting clustering cross-classified with the prior three classes forms contingency table presented in Table 6.6.

In the following two sections we describe two approaches at reaching deeper minima of K-Means criterion (6.3): (a) an incremental version and (b) nature inspired versions.

6.2.4 Incremental K-Means

P6.2.4.1 Incremental K-Means: Presentation

An incremental version of K-Means uses the Minimum distance rule not for all of the entities but for one of them only. There can be two different reasons for doing so:

(**Ri**) The user is not able to operate over the entire data set and takes in the entities one by one, because of the data protocol, so that entities are to be clustered on the fly as, for instance, in an on-line application process.

(**Rii**) The user operates with the entire data set, but wants to smooth the action of the algorithm so that no drastic changes in the cluster contents may occur. To do this, the user may specify an order of the entities and run entities one-by-one in this order for a number of epochs like it is done in a neural network learning process.

The result of such a one-by-one entity processing may differ from that of Batch K-Means because each version finds a locally optimal solution on a different structure of locality neighborhoods.

Table 6.6 Cross classification of the original Iris taxa and 3-cluster clustering found starting from entities 1, 51 and 101 as initial seeds. The clustering does separate Iris Setosa but misplaces 14+3=17 specimens between two other taxa

Cluster	Setosa	Versicolor	Virginica	Total
S1	50	0	0	50
S2	0	47	14	61
S3	0	3	36	39
Total	50	50	50	150

Q.6.14. What is the difference in neighborhoods between Batch and incremental versions of K-Means?

Q.6.15. Consider a run of incremental K-Means at situation **Rii** on the Companies data, at which the order of entities follows the order of their distances to nearest centroids. Let $K = 3$ and entities Av, Ba and Ci initial centroids. **A.** Sequential steps of the incremental computation are presented in Table 6.7. In this table, cluster updates are provided as well as their centroids after each single update. The column on the right presents squared Euclidean distances between centroids and entities yet unclustered, with the minima highlighted in bold. The minimum distance determines, in this version, which of the entities joins the clustering next. One can see that on iteration 2 company Br switches to centroid Ba after centroid Ci of the third cluster had been updated to the mean of Ci and Cy – because its distance to the new centroid increased from the minimum 0.83–1.20. This leads to correct, product-based, clusters.

Q.6.16. Prove that the same initialization leads to wrong, that is, non-product based, clusters with Batch K-Means.

F6.2.4.2 Incremental K-Means: Formulation

When an entity y_i joins cluster S_k whose cardinality is N_k, centroid c_k changes to c'_k to follow the within cluster means, according to the following formula:

$$c'_k = N_k c_k/(N_k + 1) + y_i/(N_k + 1)$$

When y_i moves out of cluster S_k, the formula remains valid if all pluses are changed for minuses. To extend the formula so that it holds for both cases, let us introduce variable z_i which is equal to +1 when y_i joins the cluster and -1 when it moves out of it. Then the extended formula is:

$$c'_k = N_k c_k/(N_k + z_i) + y_i z_i/(N_k + z_i)$$

Accordingly, the distances from other entities change to $d(y_j, c'_k)$.

Because of the incremental setting, the stopping rule of the straight version (reaching a stationary state) may be not necessarily applicable here. In **Ri** case, the natural stopping rule is to end when there are no new entities observed. In **Rii** case, the process of running through the entities one-by-one stops when all entities remain in their clusters. The process may be stopped as well when a pre-specified number of runs through the entity set, that is, epochs, is reached.

6.2.5 Nature Inspired Algorithms for K-Means

P6.2.5.1 Nature Inspired Algorithms: Presentation

In real-world applications, K-Means typically does not move far away from the initial setting of centroids. Considered in the perspective of minimization of criterion

Table 6.7 Iterations of incremental K-means on standardized company data starting with centroids Av, Ba and Ci

Incremental one-by-one entity clustering									Distances				
Iteration	Cumulative clusters	Centroids							An	As	Br	Bu	Cy
0	Av	−0.20	0.23	−0.33	−0.62	0.36	−0.22	−0.14	0.51	0.88	2.20	2.25	3.01
	Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14	1.55	1.94	0.97	0.87	2.46
	Ci	0.08	−0.10	0.33	0.38	−0.22	−0.22	0.43	1.90	1.81	0.83	1.68	0.61
1	Av, An	0.10	0.14	−0.17	−0.62	0.36	−0.22	−0.14		**0.70**	1.88	2.50	2.59
	Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14		1.94	0.97	0.87	2.46
	Ci	0.08	−0.10	0.33	0.38	−0.22	−0.22	0.43		1.81	0.83	1.68	**0.61**
2	Av, An	0.10	0.14	−0.17	−0.62	0.36	−0.22	−0.14		0.70	1.88	2.50	
	Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14		1.94	0.97	0.87	
	Ci, Cy	0.18	0.24	0.50	0.38	−0.22	−0.22	0.43		1.95	1.20	2.40	
3	Av, An, As	0.10	0.12	−0.11	−0.62	0.17	−0.02	−0.14			1.49	2.12	
	Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14			0.97	**0.87**	
	Ci, Cy	0.18	0.24	0.50	0.38	−0.22	−0.22	0.43			1.20	2.40	
4	Av, An, As	0.10	0.12	−0.11	−0.62	0.17	−0.02	−0.14			1.49		
	Ba, Bu	−0.42	−0.29	−0.33	0.38	0.07	0.07	−0.14			**0.64**		
	Ci, Cy	0.18	0.24	0.50	0.38	−0.22	−0.22	0.43			1.20		
5	Av, An, As	0.10	0.12	−0.11	−0.62	0.17	−0.02	−0.14					
	Ba, Bu, Br	−0.21	−0.29	−0.22	0.38	−0.02	0.17	−0.14					
	Ci, Cy	0.18	0.24	0.50	0.38	−0.22	−0.22	0.43					

(6.3), this leads to the strategy of multiple runs of K-Means starting from randomly generated sets of centroids to reach as deep a minimum of (6.3) as possible. This strategy works well on illustrative small data sets but it may fail when the data set is large because in this case random settings cannot cover the space of solutions in a reasonable time. Nature inspired approach provides a well-defined framework for using random centroids in parallel, rather than in sequence, to channel them to deeper minima as an evolving population of admissible solutions. The main difference of the nature inspired optimization from the classical optimization is that the latter reaches for a single solution, provably optimal, whereas the former runs a population of solutions and does not much care for the provability.

A nature inspired algorithm mimics some natural process to set rules for the population behavior and/or evolution. Among the nature inspired approaches, the following are especially popular:

A. Genetic
B. Evolutionary
C. Particle swarm optimization

A K-Means method according to each of these will be described in this section.

A nature inspired algorithm proceeds as a sequence of steps of evolution for a population of possible solutions, that is, clusterings represented by specific data structures. A K-Means clustering comprises two items: a partition S of the entity set I in K clusters and a set of clusters' K centroids $c = \{c_1, c_2, \ldots, c_K\}$. Typically, only one of them is carried out in a nature-inspired algorithm. The other is easily recovered according to K-Means rules. Given a partition S, centroids c_k are found as vectors of within cluster means. Given a set of centroids, each cluster S_k is defined as the set of points nearest to its centroid c_k, according to the Minimum distance rule ($k = 1, 2, \ldots, K$). Respectively, the following two representations are most popular in nature inspired algorithms:

(i) Partition as a string,
(ii) Centroids as a string.

Consider them in turn.

(i) Partition as a string

Having pre-specified an order of entities, a partition S can be represented as a string of cluster labels $k = 1, 2, \ldots, K$ of the entities thus ordered. If, for instance, there are eight entities ordered as e1, e2, e3, e4, e5, e6, e7, e8, then the string 12333112 represents partition S with three classes according to the labels, S1 = {e1, e6, e7}, S2 = {e2, e8}, and S3 = {e3, e4, e5}, which can be easily seen from the diagram relating the entities and labels:

e1	e2	e3	e4	e5	e6	e7	e8
1	2	3	3	3	1	1	2

A string of N integers from 1 to K is considered not admissible, if some integer between *1* and K is absent from it (so that the corresponding cluster is empty). Such a not admissible string for the entity set above would be 11333111, because it lacks label 2 and, therefore, makes class S2 empty.

(ii) Centroids as a string

Consider the same partition as in (i) on the set of eight objects, the companies in Company data Table 6.1 in their order. Clusters S1 = {e1, e6, e7}, S2 = {e2, e8}, and S3 = {e3, e4, e5}, as well as their centroids, are presented in Table 6.8. The three centroids form a sequence of $7 \times 3 = 21$ numbers $c = (-0.24, -0.09, -0.11, 0.04, -0.02, -0.02, 0.05, 0.34, 0.31, 0.33, -0.12, 0.07, -0.22, 0.14, 0.01, -0.12, -0.11, 0.04, -0.02, 0.17, -0.14)$, which suffices for representing the clustering: the sequence can be easily converted back in three 7-dimensional centroid vectors to recover then clusters with the Minimum distance rule. It should be pointed out that the original clusters may be somewhat weird and not recoverable in this way. For example, entity e4, which is Ba, appears to be nearer to centroid 1 rather than to centroid 3 so that the Minimum distance rule would produce clusters S1 = {e1, e4, e6, e7}, S2 = {e2, e8}, and S3 = {e3, e5} rather than those original ones, but such a loss makes no difference, because K-Means clusters necessarily satisfy the Minimum distance rule so that all the entities are nearest to their cluster's centroids.

What is important is that any 21-dimensional sequence of real values can be treated as the clustering code for its centroids.

GA for K-Means Clustering

Genetic algorithms work over a population of strings, each representing an admissible solution and referred to as a chromosome. The optimized function is referred to as the fitness function. Let us use the string represenatation for partitions $S = \{S_1, \ldots, S_K\}$ of the entity set. The minimized fitness function is the summary within-cluster distance to centroids, the function *W(S,c)* in (6.3):

0. *Initial setting*. Specify an even integer P for the population size (no rules exist for this), and randomly generate P chromosomes, that is, strings $s_1, \ldots, s_P$ of

Table 6.8 Centroids of clusters S1 = {e1, e6, e7}, S2 = {e2, e8}, and S3 = {e3, e4, e5} according to data in Table 6.1

Av	−0.20	0.23	−0.33	−0.63	0.36	−0.22	−0.14
Bu	−0.60	−0.42	−0.33	0.38	−0.22	0.36	−0.14
Ci	0.08	−0.10	0.33	0.38	−0.22	−0.22	0.43
Centroid1	−0.24	−0.09	−0.11	0.04	−0.02	−0.02	0.05
An	0.40	0.05	0.00	−0.62	0.36	−0.22	−0.14
Cy	0.27	0.58	0.67	0.38	−0.22	−0.22	0.43
Centroid2	0.34	0.31	0.33	−0.12	0.07	−0.22	0.14
As	0.08	0.09	0.00	−0.62	−0.22	0.36	−0.14
Ba	−0.23	−0.15	−0.33	0.38	0.36	−0.22	−0.14
Br	0.10	−0.29	0.00	0.38	−0.22	0.36	−0.14
Centroid3	0.01	−0.12	−0.11	0.04	−0.02	0.17	−0.14

K integers 1, ..., K in such a way that all K integers 1, 2, ..., K are present within each chromosome.. For each of the strings, define corresponding clusters, calculate their centroids as gravity centres and the value of criterion, $W(s_1)$, ..., $W(s_P)$, according to formula (6.3).

1. *Mating selection*. Choose $P/2$ pairs of strings to mate and produce two "children" strings. The mating pairs usually are selected randomly (with replacement, so that the same string may appear in several pairs and, moreover, can form both parents in a pair). To mimic Darwin's "survival of the fittest" law, the probability of selection of string s_t ($t = 1,...,P$) should reflect its fitness value $W(s_t)$. Since the fitness is greater for the smaller W value, some make the probability inversely proportional to $W(s_t)$ (see Murthy and Chowdhury 1996) and some to the difference between a rather large number and $W(s_t)$ (see Yi Lu et al. 2004). This latter approach can be taken further with the probability proportional to the explained part of the data scatter – in this case "the rather large number" is the data scatter rather than an arbitrary value.
2. *Cross-over*. For each of the mating pairs, generate a random number r between 0 and 1. If r is smaller than a pre-specified probability p (typically, p is taken about 0.7–0.8), then perform a crossover; otherwise the mates themselves are considered the result. A (single-point) crossover of string chromosomes $a = a_1a_2 \ldots a_N$ and $b = b_1b_2 \ldots b_N$ is performed as follows. A random number n between 1 and N-1 is selected and the strings are crossed over to produce children $a_1a_2 \ldots a_nb_{n+1} \ldots b_N$ and $b_1b_2 \ldots b_na_{n+1} \ldots a_N$. If a child is not admissible (like, for instance, strings $a = 11133222$ and $b = 32123311$ crossed over at $n = 4$ would produce $a' = 11133311$ and $b' = 32123222$ so that a' is inadmissible because of absent 2), then various policies can be applied. Some authors suggest the crossover operation to be repeated until an admissible pair is produced. Some say inadmissible chromosomes are ok, just they must be assigned with a smaller probability of selection.
3. *Mutation*. Mutation is a random alteration of a character in a chromosome. This provides a mechanism for jumping to different "ravines" of the minimized fitness function. Every character in every string is subject to the mutation process, with a low probability q which can be constant or inversely proportional to the distance between the corresponding entity and corresponding centroid.
4. *Elitist survival*. This strategy suggests keeping the best fitting chromosome(s) stored separately. After the crossover and mutations have been completed, find fitness values for the new generation of chromosomes. Check whether the worst of them is better than the record or not. If not, put the record chromosome instead of the worst one into the population. Then find the record for thus obtained population.
5. *Halt condition*. Check the stop condition (typically, a limit on the number of iterations). If this doesn't hold, go to 1; otherwise, halt.

Y. Lu et al. (2004) note that such a GA works much faster if after step 3. Mutation the labels are changed according to the Minimum distance rule. They apply this instead of the elitist survival.

Thus, a GA algorithm operates with a population of chromosomes representing admissible solutions. To update the population, mates are selected, undergone a cross-over process generating offspring which then is subjected to mutation process. Elite maintenance completes the update. In the end, the elite is output as the best solution.

A computational shortcoming of the GA algorithm is that the length of the chromosomes is the size of the entity set N, which may run in millions in contemporary applications. Can this be overcome? Sure, by using centroid, not partition, strings to represent a clustering. Centroid string sizes depend on the number of features and number of clusters, not the number of entities. Another advantage of centroid strings is in the mutation process. Rather than an abrupt swap between literals, they can be changed softly, in a quantitative manner by adding or subtracting a small change. This is utilized in evolutionary and particle swarm algorithms.

Evolutionary K-Means

Here, the chromosome is represented by a set of K centroids $c = (c_1, c_2, \ldots c_K)$ which can be considered a string of KV real ("float") numbers. In contrast to the partition-as-string representation, the length of the string here does not depend on the number of entities that can be of advantage when the number of entities is massive. Each centroid in the string is analogous to a gene in the chromosome.

The crossover of two centroid strings c and c', each of the length KV, is performed at a randomly selected place n, $1 \leq n < KV$, exactly as it is in the genetic algorithm above. Chromosomes c and c' exchange the portions lying to the right of n-th component to produce two offspring. This means that, a number of centroids in c is substituted by corresponding centroids in c'. Moreover, if n cuts across a centroid, its components change in each of the offspring chromosomes.

The process of mutation, according to Bandyopadhyay and Maulik (2002), can be organized as follows. Given the fitness W values of all the chromosomes, let $minW$ and $maxW$ denote their minimum and maximum respectively. For each chromosome, its radius R is defined as a proportion of $maxW$ reached at it: $R = (W\text{-}minW)/(maxW\text{-}minW)$. When the denominator is 0, that is, if $minW = maxW$, define $R = 1$ in all chromosomes. Here, W is the fitness value of the chromosome under consideration. Then the mutation intensity δ is generated randomly in the interval between $-R$ and $+R$.

Let $minv$ and $maxv$ denote the minimum and maximum values in the data set along feature v ($v = 1,\ldots, V$). Then every v-th component xv of each centroid c_k in the chromosome changes to

$$xv + \delta^*(maxv - xv) \text{ if } \delta \leq 0 \text{ (increase), or}$$
$$xv + \delta^*(xv - minv), \text{ otherwise (decrease).}$$

The perturbation leaves chromosomes within the hyper-rectangle defined by boundaries $minv$ and $maxv$. Please note that the best chromosome, at which $W = minW$, does not change in this process because its $R = 0$.

Elitism is maintained in the process as well.

The algorithm follows the scheme outlined for the genetic algorithm.
Based on little experimentation, this algorithm is said to outperform the previous one, GA, many times in terms of the speed of convergence.

The evolutionary approach can be further modified such as, for example, the so-called Differential evolution (see Paterlini and Krink 2006 who claim that this method outperforms the others in K-Means). In Differential evolution, the crossover, mutation and elite maintenance are merged together by removing the mating stage and changing those for the following. An offspring chromosome is created for every chromosome t in the population ($t = 1, \ldots, P$) as follows. Three other chromosomes, k, l and m, are taken randomly from the population. Then, for every component (gene) $x.t$ of the chromosome t, a uniformly random value r between 0 and 1 is drawn. This value is compared to the pre-specified probability p (somewhat between 0.5 and 0.8). If $r > p$ then the component goes to the offspring unchanged. Otherwise, this component is substituted by the linear combination of the same component in the three other chromosomes: $x.m + \alpha^*(x.k - x.l)$ where α is a small scaling parameter. After the offspring's fitness is evaluated, it substitutes chromosome t if it is better; otherwise, t remains as is and the process applies to the next chromosome.

Particle Swarm Optimization for K-Means

Particle swarm mimics a drift of a bee population so that the population members here are not crossbred, nor they mutate. They just move randomly by drifting in random directions having an eye on the best places visited so far, individually and socially. This can be done because they are vectors of real numbers. Because of the change, the genetic metaphor is abandoned here, and the elements are referred to as particles rather than chromosomes, and the set of them as a swarm rather than a population.

Each particle comprises:

- a position vector x that is an admissible solution to the problem in question (such as the *KV* centroid vector in K-Means),
- the evaluation of its fitness *f(x)* (such as the summary distance W in (6.3)),
- a velocity vector z of the same dimension as x, and
- the record of the best position b reached by the particle so far.

The swarm best position bg is determined as the best among all the individual best positions b.

At iteration t ($t = 0, 1, \ldots$) the next iteration's position is defined as the current position shifted by the velocity vector:

$$x(t+1) = x(t) + z(t+1)$$

where $z(t+1)$ is computed as a change in the direction of personal and population's best positions:

$$z(t+1) = z(t) + \alpha(b - x(t)) + \beta(bg - x(t))$$

where

– α and β are uniformly distributed random numbers (typically, within the interval between 0 and 2, so that they are around unity),
– item $\alpha(b - x(t))$ is referred to as the cognitive component and
– item $\beta(bg - x(t))$ as the social component of the process.

Initial values *x(0)* and *z(0)* are generated randomly within the manifold of admissible values.

In some implementations, the group best position *bg* is changed for that of local best position *bl* that is defined by the particle's neighbors only so that some predefined neighborhood topology makes its effect. There is a report that the local best position works especially well, in terms of the depth of the minimum reached, when it is based on just two Euclidean neighbors.

Q.6.17. Formulate a particle swarm optimization algorithm for K-Means clustering.

6.2.6 Partition Around Medoids PAM

K-Means centroids are average points rather than individual entities, which may be considered artificial in contexts in which the user may wish to involve but only genuinely occurring real world entities rather the "synthetic" averages. Estates or art objects or countries are examples of entities for which this makes sense. To implement the idea, let us change the concept of cluster prototype from centroid to *medoid* (Kaufman and Rousseeuw 1990). An entity in a cluster S, $i^* \in S$, is referred to as its medoid if it is the nearest in S to all other elements of S, that is, if i^* minimizes the sum of distances $D(i) = \Sigma_{j \in S} d(i,j)$ over all $i \in S$. The symbol *d(i,j)* is used here to denote any dissimilarity function, which may or may not be squared Euclidean distance, between observed entities $i, j \in I$.

The method of partitioning around medoids PAM (Kaufman and Rousseeuw 1990) works exactly as Batch K-Means with the only difference that medoids, not centroids, are used as cluster prototypes. It starts, as usual, with choosing the number of clusters K and initial medoids $c = (c_1, c_2, \ldots, c_K)$ that are not just V dimensional points but individual entities. Given medoids c, clusters S_k are collected according to the Minimum distance rule – as sets of entities that are nearest to entity c_k for all $k = 1, 2, \ldots, K$. Given clusters S_k, medoids are updated according to the definition. This process reiterates again and again, and halts when no change of the clustering occurs. It obviously will never leave a cluster S_k empty. If the size of the data set is not large, all computations can be done over the entity-to-entity distance matrix without ever changing it.

Worked example 6.2. PAM applied to Company data

Let us apply PAM to the Company data displayed in Table 6.1 with $K = 3$ and entities Av, Br and Cy as initial medoids. We can operate over the distance matrix, presented in Table 6.9, because there are only eight entities.

Table 6.9 Distances between standardized company entities. For the sake of convenience, those smaller than 1, are highlighted in bold

Entities	Av	An	Ast	Ba	Br	Bu	Ci	Cy
Av	**0.00**	**0.51**	**0.88**	1.15	2.20	2.25	2.30	3.01
An	**0.51**	**0.00**	**0.77**	1.55	1.82	2.99	1.90	2.41
As	**0.88**	**0.77**	**0.00**	1.94	1.16	1.84	1.81	2.38
Ba	1.15	1.55	1.94	**0.00**	**0.97**	**0.87**	1.22	2.46
Br	2.20	1.82	1.16	**0.97**	**0.00**	**0.75**	**0.83**	1.87
Bu	2.25	2.99	1.84	**0.87**	**0.75**	**0.00**	1.68	3.43
Ci	2.30	1.90	1.81	1.22	**0.83**	1.68	**0.00**	**0.61**
Cy	3.01	2.41	2.38	2.46	1.87	3.43	**0.61**	**0.00**

With seeds Av, Br, Cy, the Minimum distance rule would obviously produce the product-based clusters A, B, and C. At the next iteration, clusters' medoids are computed: they are obviously An in A cluster, Bu in B cluster and either of the two entities in C cluster – leave it thus at the less controversial Cy. With the set of medoids changed to An, Bu and Cy, we apply the Minimum distance rule again, leading us to the product-based clusters again. This halts the process.

Note that PAM can lead to instability in results because the assignment depends on distances to just a single entity.

Q.6.18. Why Cy is less controversial than Ci in Table 6.9? **A.** Because Cy unequivocally relates to Ci only, whereas Ci is close to Br as well.

Q.6.19. Assume that the distance d(Br, Bu) in Table 6.9 is 0.85 rather than 0.75. Show that then if one chooses Ci to be medoid of C cluster, then the Minimum distance rule would assign to Ci not only Cy but also Br, because its distance to Ci, 0.83, would be less than its distance to Bu, 0.85. Show that this cluster, {Ci, Cy, Br} will remain stable over successive iterations.

6.2.7 Initialization of K-Means

To initialize K-Means, one needs to specify:

(i) the number of clusters, K, and
(ii) initial centroids, $c = (c_1, c_2, \ldots, c_K)$.

Each of these can be of an issue in practical computations. Both depend on the user's expectations related to the level of granularity and typological attitudes, which remain beyond the scope of the theory of K-Means. This is why some suggest relying on the user's view of the substantive domain to specify the number and positions of initial centroids as hypothetical prototypes. There have been however a number of approaches for specifying the number and location of the initial centroids by exploring the structure of the data, of which we describe the following three:

(a) multiple runs of K-Means;
(b) distant representatives;
(c) anomalous patterns.

(a) Multiple runs of K-Means

According to this approach, at a given K, a number of K-Means' runs R is pre-specified; each run starts with K randomly selected entities as the initial seeds (randomly generated points within the feature ranges have proven to give inferior results in experiments reported by several authors). Then the best result in terms of the square-error criterion $W(S,c)$ (6.3) is output. This can be further extended to choosing the "right" number of clusters K. Let us denote by W_K the minimum value of $W(S,c)$ found after R runs of K-Means over random initializations. Then the series W_K found at different K, from a pre-specified range say between 2 and 20, is usually taken to see which K would lead to the best W_K over the range. Unfortunately, the best W_K is not necessarily minimum W_K, because the minimum value of the square-error criterion cannot increase when K grows, which should be reflected in the empirically found W_K's. In the literature, a number of stop criteria utilizing W_K have been suggested based on some simplified data models and intuition such as "gap" or "jump" statistics. Unfortunately, they all may fail even in the relatively simple situations of controlled computation experiments (see Chiang and Mirkin 2010 for a review).

A relatively simple heuristic rule is based on the intuition that when there are K^* well separated clusters, then for $K<K^*$ a $(K+1)$-cluster partition should be the K-cluster partition with one of its clusters split in two, which would drastically decrease W_{K+1} from W_K. On the other hand, at $K>K^*$, both K- and $(K+1)$-cluster partitions are to be the "right" K^*-cluster partition with some of the "right" clusters split randomly, so that W_K and W_{K+1} are not that different. Therefore, as "a crude rule of thumb", Hartigan (1975, p. 91) proposed calculating index

$$H_K = (W_K/W_{K+1} - 1)(N - K - 1),$$

where N is the number of entities, while increasing K, so that the very first K at which H_K becomes smaller than 10 is to be taken as the estimate of K^*. It should be noted that, in the experiments by Chiang and Mirkin (2010), this rule came as the best of a set of nine different criteria and, moreover, the threshold 10 in the rule appears to be not very sensitive to 10–20% changes.

Case Study 6.4. Hartigan's Index for Choosing the Number of Clusters

Consider values of H_K for Iris and Town datasets computed after the results of 100 runs of Batch K-Means using the mean/range standardization starting from random K entities taken as seeds (Table 6.10). Each of the computations has been

Table 6.10 Values of Hartigan's H_K index for two data sets at K ranging from 2 to 11 as based on two different sets of 100 clusterings from random K entities as initial centroids

Dataset		K = 2	3	4	5	6	7	8	9	10	11
Iris	1st set	108.3	38.8	29.6	24.1	18.6	15.0	16.1	15.4	15.4	**9.4**
	2nd set	108.3	38.8	29.6	24.1	18.7	15.4	15.6	15.7	16.0	**7.2**
Town	1st set	13.2	10.5	**9.3**	5.0	4.7	3.1	3.0	3.2	3.2	1.6
	2nd set	13.2	10.5	**9.3**	5.8	4.1	2.5	3.0	7.2	−0.2	1.8

repeated twice (see 1st and 2nd sets in Table 6.10) to illustrate typical variations of H_K values due to the fact that empirical values of W_K may be not optimal. In particular, at the 2nd set of K-Means over Town data we can see a break of the rule that H_K is positive because of the monotonic relation between K and the optimal W_K that are to decrease when K grows. The monotonic relation here is broken because the values of W_K after 100 runs are not necessarily minimal indeed.

The "natural" number of clusters in Iris data, according to Hartigan's criterion is not 3 as claimed because of substantive considerations but much greater, 11! In Town data set, the criterion would indicate 4 naturally occurring clusters. However, one should argue that the exact value of 10 in Hartigan's rule does not bear much credibility – it should be accompanied by a significant drop in H_K value. We can see such a drop at $K = 5$, which should be taken, thus, as the "natural" number of clusters in Town data. Similarly, a substantial drop of H_K on Iris data occurs at $K = 3$, which is the number of natural clusters, taxa, in this set.

Altogether, making multiple runs of K-Means seems a sensible strategy, especially when the number of entities is not that high. With the number of entities growing into thousands, the number of tries needed to reach a representative value of W_K may become prohibitively large. Deeper minima can be sought by using the evolutionary schemes described above. On the other hand, the criterion *W(S,c)* has some intrinsic flaws and should be used only along some domain-knowledge or data-structure based strategy.

Two data-driven approaches, (b) and (c) above, to defining initial centroids are described in the next two sections. They both employ the idea that clusters should represent some anomalous yet typical tendencies.

(b) "Build" algorithm for a pre-specified *K* (Kaufman and Rousseeuw 1990)

This process involves only actual entities. It starts with choosing the medoid of set I, that is, the entity whose summary distance to the others is minimum, and takes it as the first medoid c_1. Assume that a subset of m initial seeds have been selected already ($K > m \geq 1$) and proceed to selecting c_{m+1}. Denote the set of already selected seeds by c and consider all remaining entities $i \in I - c$. Define distance *d(i,c)* as the minimum of the distances $d(i, c_k)(k = 1, \ldots, m)$ and form an auxiliary

cluster A_i consisting of such j that are closer to i than to c so that $E_{ij} = d(j, c) - d_{ij} > 0$. The summary value $E_i = \Sigma_{j \in Ai} E_{ij}$ reflects both the number of points in A_i and their remoteness from c. That $i \in I - c$ for which E_i is maximum is taken as the next seed c_{m+1}.

Worked example 6.3. Selection of initial medoids in Company data

Let us apply Build algorithm to the matrix of entity-to-entity distances for Company data displayed in Table 6.9, at $K = 3$. First, we calculate the summary distances from all the entities to the others, see Table 6.11, and notice that Br is the medoid of the entire set I, because its total distance to the others, 9.60, is the minimum of total distances in Table 6.11. Thus, we set Bre as the first initial seed.

Now we build auxiliary clusters A_i around all other entities. To form A_{Av} , we take the distance between Av and Br, 2.20, and see, in Table 6.10, that distances from Av to entities An, As, and Ba are smaller than that, which makes them Av's auxiliary cluster with $E_{Av} = 4.06$. Similarly, A_{An} is set to consist of the same entities, but it is less remote than Av because $E_{An} = 2.98$ is less than E_{Av}. Auxiliary cluster A_{As} consists of Av and An with even smaller $E_{As} = 0.67$. Auxiliary clusters for Ba, Ci and Cy consist of one entity each (Bu, Cy and Ci, respectively) and have much smaller the levels of remoteness; cluster A_{Bu} is empty because Br is its nearest. This makes the most remote entity Ave the next selected seed. Now, we can start building auxiliary clusters on the remaining six entities again. Of them, clusters A_{An} and A_{Bu} are empty and the others are singletons, of which A_{Cy} consisting of Ci is the remotest, with $E_{Cy} = 1.87 - 0.61 = 1.26$. This completes the set of initial seeds: Br, Av, and Cy. Note, these are companies producing different products. It is this set that was used to illustrate PAM in Section 6.2.6.

(c) **Anomalous patterns** (Mirkin 2005)

This method involves remote clusters, as Build does, too, but it does not discard them after finding, which allows for obtaining the number of clusters K as well. Besides, it is less computationally intensive. The method employs the concept of reference point. A reference point is chosen to exemplify an "average" or "normal" entity, not necessarily among the dataset. For example, when analyzing student marks over different subjects, one might choose a "normal student" point which

Table 6.11 Summary distances for entities according to Table 6.9

Entity	Av	An	As	Ba	Br	Bu	Ci	Cy
Distance to others	12.30	11.95	10.78	10.16	9.60	13.81	10.35	16.17

would indicate levels of marks in tests and work in projects that are considered normal for the contingent of students under consideration, and then see what patterns of observed behavior deviate from this. Or, a bank manager may set as his reference point, a customer having specific assets and backgrounds, to see what patterns of customers deviate from this. In engineering, a moving robotic device should be able to segment the view into homogeneous chunks according to the robot's location as its reference point, with objects that are nearer to it having finer resolution than objects that are farther away. In many cases the gravity center of the entire entity set, its "grand mean", can be taken as a reference point of choice.

Using the chosen reference point allows for the comparison of entities with it, not with each other, which drastically reduces computations: instead of mulling over all the pair-wise distances, one may focus on entity-to-reference-point distances only – a reduction to the order of N from the order of N^2.

An anomalous pattern is found by building a cluster which is most distant from the reference point. To do this, the cluster's seed is defined as the entity farthest away from the reference point. Now a version of K-Means at $K = 2$ is applied with two seeds: the reference point which is never changed in the process and the cluster's seed, which is updated according to the standard procedure. In fact, only the anomalous cluster is of interest here. Given a centroid, the cluster is defined as the set of entities that are closer to it than to the reference point. Given a cluster, its centroid is found as the gravity center, by averaging all the cluster entities. The procedure is reiterated until convergence (see Fig. 6.10).

Assuming the reference point has been shifted into the origin, the Anomalous pattern method is a version of K-Means in which:

(i) the number of clusters K is 2;
(ii) centroid of one of the clusters is 0, which is forcibly kept there through all the iterations;
(iii) the initial centroid of the anomalous cluster is taken as an entity far thest away from 0.

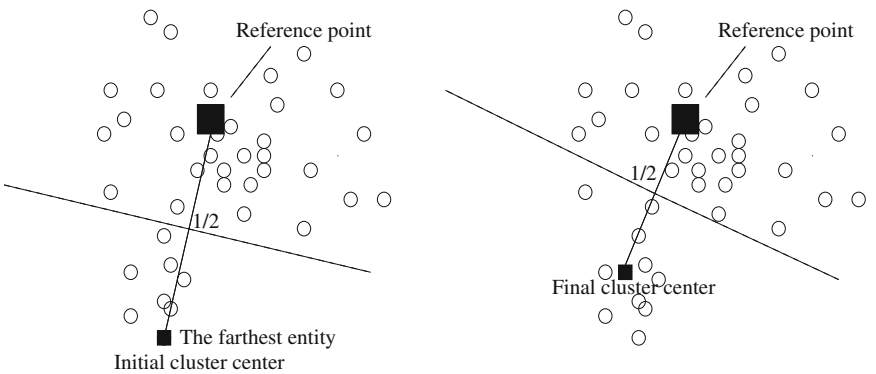

Fig. 6.10 Extracting an Anomalous pattern cluster with the reference point in the gravity center: the first iteration is on the *left* and the final one on the *right*

Property (iii) mitigates the issue of determining appropriate initial seeds. This provides for using Anomalous pattern algorithm iteratively to obtain an initial setting for K-Means.

There is a certain similarity between selecting initial centroids in iK-Means and initial medoids with Build. But there are certain differences as well:

– K must be pre-specified in Build and not necessarily in iK-Means;
– The central point of the entire set I is taken as an initial seed in Build and is not in iK-Means;
– Addition of a new seed is based on different criteria in the two methods.

A clustering algorithm should present the user with a comfortable set of options. The iK-Means or PAM with Build/AP can be easily extended so that some entities can be removed from the data set because they are either (i) "deviant" or (ii) "intermediate" or (iii) "trivial". These can be defined as the contents of small AP or Build clusters, for the case (i), or entities that are far away from their centroids/medoids, for the case (ii), or entities that are close to the grand mean, the center of gravity of the entire data set, for the case (iii).

Worked example 6.4. Anomalous pattern in Market towns

Let us apply the Anomalous pattern method to Town data assuming the grand mean as the reference point and scaling by range. That means that after mean-range standardization the reference point is 0.

The point farthest from 0, to be taken as the initial "anomalous" centroid, appears to be entity 35 (St Austell) whose distance from zero (remember – after standardization!) is 4.33, the maximum. There are only three entities, 26, 29 and 44 (Newton Abbot, Penzance and Truro) that are closer to the seed than to 0, thus forming the cluster along with the original seed, at this stage. After one more iteration, the anomalous pattern cluster stabilizes with 8 entities 4, 9, 25, 26, 29, 35, 41, 44. Its centroid is displayed in Table 6.12.

As follows from the fact that all the standardized entries are positive and mostly fall within the range of 0.3–0.5, the anomalous cluster, according to Table 6.12, consists of better off towns – all the centroid values are larger than the grand mean by 30–50% of the feature ranges. This probably relates to the fact that they comprise eight out of the eleven towns that have a resident population greater than 10,000.

Table 6.12 Centroid of the anomalous pattern cluster of town data in real and standardized forms

Centroid	P	PS	Do	Ho	Ba	Sm	Pe	DIY	Sp	Po	CAB	FM
Real	18484	7.6	3.6	1.1	11.6	4.6	4.1	1.0	1.4	6.4	1.2	4.0
Std'zed	0.51	0.38	0.56	0.36	0.38	0.38	0.30	0.26	0.44	0.47	0.30	0.18

The other three largest towns have not made it into the cluster because of their deficiencies in services such as Hospitals and Farmers' Markets. The fact that the scale of population is by far the largest in the original table doesn't much affect the computation here as it runs with the range standardized scales at which the total contribution of this feature is not high, just about 8.5% only. It is rather the concerted action of all the features associated with a greater population which makes the cluster.

This process is illustrated on Fig. 6.11. The stars show the origin and the anomalous seed at the beginning of the iteration. Curiously, this picture does not fit well into the concept of the anomalous pattern cluster, as illustrated on the previous Fig. 6.10 – the anomalous pattern is dispersed here across the plane, which is at odds with the property that the entities in it must be closer to the seed than to the origin. The cause is not an error, but the fact that this plane represents all 12 original variables and presents them rather selectively. It is not that the plane makes too little of the data scatter – on the contrary, it makes a decent 76% of the data scatter. The issue here is the second axis in which the last feature FM expressing whether there is a Farmers market or not takes a lion share – thus stratifying the entire image over y axis.

The Fig. 6.11 has been produced with commands:

≫ subplot(1,2,1);plot(x1,x2,′k.′, 0,0,′kp′,x1(35),x2(35),′kp′);text(x1(fir),x2(fir),ftm);
≫ subplot(1,2,2);plot(x1,x2,′k.′, 0,0,′kp′,x1a,x2a,′kp′);text(x1(sec),x2(sec),fsm);

Here fir and sec are lists of indices of towns belonging to the pattern after the first and second iterations, respectively, while ftm and fsm refer to lists of their names.

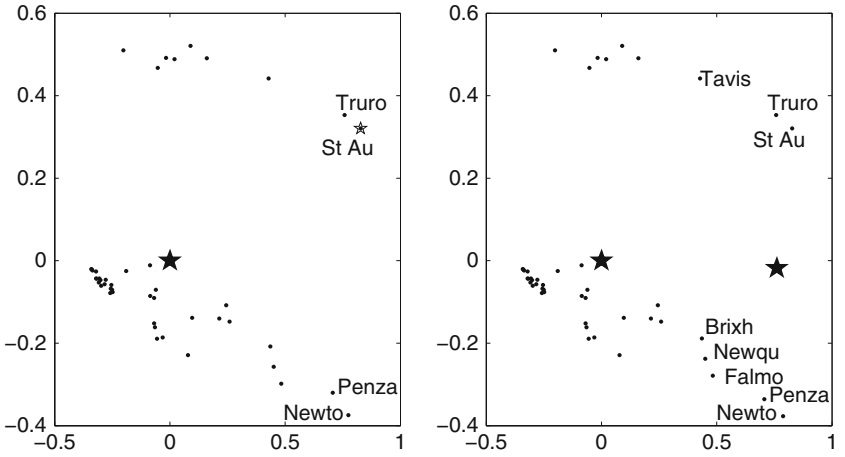

Fig. 6.11 The first and second iterations of Anomalous pattern cluster on the principal component plane; the visual separation of the pattern over y axis is due to a very high loading of the presence (*top*) or absence (*bottom*) of a farmer's market

6.2.8 Anomalous Pattern and Intelligent K-Means

P6.2.8.1 Anomalous Pattern and iK-Means: Presentation

The Anomalous pattern method can be used as a procedure to automatically determine both the number of clusters and initial seeds for K-Means. Preceded by this option, K-Means is referred to as intelligent K-Means, iK-Means for brevity, because it relieves the user from the task of specifying the initial setting.

In iK-Means method, the user is required to specify an integer, *t,* the threshold of resolution, to be used to discard all the Anomalous patterns consisting of t or less entities. When $t = 0$, nothing is discarded. At $t = 1$ – the default option, singleton anomalous patterns are considered a nuisance and put back to the data set. If $t = 10$, all patterns with 10 or less entities are discarded as too small to deserve any attention at all – the level of resolution which may be justified at larger datasets and coarser details needed.

In our experiments, the entities comprising singleton Anomalous pattern clusters are frequently erroneous, that is, errors are in some of their features such as, for instance, the human age of 5,000 years. That means, that Anomalous pattern clustering can be used as a device for checking against huge errors in data entries.

The iK-Means method is flexible with regard to outliers and the "swamp" of inexpressive – normal or ordinary – entities around the grand mean. For example, at its step 4, K-Means can be applied to either the entire dataset or to the set from which the smaller APs have been removed. This may depend on the domain: in some problems, such as structuring of a set of settlements for better planning or monitoring or analysis of climate changes, no entity should be dropped out of the consideration, whereas in other problems, such as developing synoptic descriptions for text corpora, some "deviant" texts could be left out of the coverage at all.

In a series of experiments with overlapping Gaussian clusters described by Chiang and Mirkin (2010), iK-Means has performed rather well and appeared superior to many other options for choosing K. These options included approaches based on post-processing of results of multiple runs of K-Means and then treating them according to either of the following:

(a) Variance based approach: using intuitive or model based functions of criterion (6.3) which should get extreme or "elbow" values at a correct K such as Hatigan's rule above;
(b) Structural approach: comparing within-cluster cohesion versus between-cluster separation at different K;
(c) Consensus distribution approach: choosing K according to the distribution of the consensus matrix for sets of K-Means clusterings at different K.

Some other approaches rely on different ideas for choosing K such as

(d) using results of a divisive or agglomerative clustering procedure or
(e) using the similarity of K-Means clustering results on randomly perturbed or sampled data.

Worked example 6.5. Iterated Anomalous patterns in Market towns

Applied to the range-standardized Market town data, AP algorithm, iterated until no unclustered entities remained, has produced 12 clusters of which 5 are singletons. These singletons have strange patterns of facilities indeed. For example, entity 19 (Liskeard, 7,044 residents) has an unusually large number of Hospitals (6) and CABs (2), which makes it a singleton cluster. Lists of seven non-singleton clusters are in Table 6.13, in the order of their extraction in the iterated AP.

This cluster structure doesn't much change when, according to the iK-Means algorithm, Batch K-Means is applied to the seven centroids (with the five singletons put back into the data). Moreover, similar results have been observed with clustering of the original all-England list of about thirteen hundred Market towns described by a wider list of eighteen characteristics of their development: the number of non-singleton clusters was the same, with very similar descriptions.

Q.6.20. Why is the contribution of AP 4, 18.6%, greater than that of the preceding AP3, 10.0%? **A.** Because of a much larger number of entities, 18 against 6 in AP 3. Even if the centroid of AP 3 is further away from 0 than centroid of AP 4, which is the cause that AP 3 is extracted first, the contribution takes into account the number of entities as well!

FC6.2.8.2 Anomalous Pattern and iK-Means: Formulation and Computation

Before substantiating AP algorithm, let us give it a more explicit formulation.

Anomalous Pattern (AP) Algorithm

1. *Pre-processing*. Specify a reference point $a = (a_1, \ldots, a_V)$ (when in doubt, take a to be the data grand mean) and standardize the original data table by shifting the origin to $a = (a_1, \ldots, a_V)$.
2. *Initial setting*. Put a tentative centroid, c, as the entity farthest away from the origin, 0.

Table 6.13 Iterated AP market town non-singleton clusters

Cluster #	Size	Contents	Contribution (%)
1	8	4, 9, 25, 26, 29, 35, 41, 44	35.1
3	6	5, 8 , 12, 16, 21, 43	10.0
4	18	2, 6, 7, 10,13, 14, 17, 22, 23, 24,27, 30, 31, 33, 34, 37, 38, 40	18.6
5	2	3, 32	2.4
6	2	1,11	1.6
8	2	39, 42	1.7
11	2	20,45	1.2

3. *Cluster update*. Determine cluster list S around c against the only other "centroid" 0, so that entity y_i is assigned to S if $d(y_i, c) < d(y_i, 0)$.
4. *Centroid update*. Calculate the within S mean c' and check whether it differs from the previous centroid c. If c' and c do differ, update the centroid by assigning $c \Leftarrow c'$ and go to Step 3. Otherwise, go to 5.
5. *Output*. Output list S and centroid c, with accompanying interpretation aids (as advised in the next section), as the anomalous pattern.

It is not difficult to prove that, like K-Means itself, the Anomalous pattern alternately minimizes a specific version of K-Means general criterion *W(S,c)* (6.3),

$$W(S, c) = \sum_{i \in S} d(y_i, c) + \sum_{i \notin S} d(y_i, 0) \tag{6.14}$$

where S is a subset of I rather than partition and c its centroid. Yet AP differs from 2-Means in the following aspect: there is only one centroid, c, which is updated in AP; the other centroid, 0, never changes and serves only to attract not-anomalous entities. This is why 2-Means produces two clusters whereas AP – only one, that is farthest away from the reference point, 0.

In fact, criterion (6.14) can be equivalently rephrased using Equations (6.6) and (6.7) representing the complimentary criterion *B(S,c)*. When (6.7) applies to the situation of two clusters, one with centroid in c, the other in *0*, it becomes of finding a cluster S maximizing its contribution to the data scatter *T(Y)*:

$$\mu^2 = z^T Y Y^T z / z^T z = {c_v}^2 |S| = d(0, c) |S| \tag{6.15}$$

This means that AP algorithm straightforwardly follows the Principal Component Analysis one-by-one extraction strategy extended to binary scoring vectors. That is, the model behind AP is a version of the PCA Equation (5.10) in which the scoring values ${z^*}_i$ are but zeros or ones:

$$y_{iv} = \begin{cases} c_v + e_v, i \in S \\ 0 + e_v, i \notin S \end{cases} \tag{6.16}$$

where S is the cluster list of the anomalous pattern to be found.

In spite of the rather simplistic assumption presented in (6.16), AP clusters fare well with real data. They can be extracted one-by-one, along with their contributions to the data scatter (6.15) showing cluster saliencies. These saliencies can be used to halt the process when the contribution of the next cluster drops decisively, thus leading to an incomplete clustering when needed.

Here are steps of iK-Means(t) where t is the cluster discharge threshold – the minimum number of entities in a pattern that can be considered a cluster on its own. In most applications dealing with moderately sized data (up to a few hundred entities) t can be put to be equal to 1.

iK-Means(t) Algorithm

0. *Setting*. Preprocess and standardize the data set. Take t as the threshold of resolution. Put $k = 1$ and $I_k = I$, the original entity set.
1. *Anomalous pattern*. Apply AP to I_k to find k-th anomalous pattern S_k and its centroid c_k.
2. *Test*. If Stop-condition (see below) does not hold, remove S_k from I_k to make $k \Leftarrow k + 1$ and $I_k \Leftarrow I_k - S_k$, after which step 1 is executed again. If it does, go to 3.
3. *Discarding small clusters.* Remove all of the found clusters containing t entities or less. Denote the number of remaining clusters by K and re-label them so that their centroids are $c_1, c_2, \ldots, c_K$.
4. *K-Means*. Do Batch K-Means using $c_1, c_2, \ldots, c_K$ as initial seeds.

Case Study 6.5. iK-Means Clustering of a Normally Distributed 1D Dataset

Let us generate a one dimensional set X of 280 points according to Gaussian N(0,10) distribution (see Fig. 6.12). This data set is attached in the appendix as Table A5.2. Many would say that this sample constitutes a single, Gaussian, cluster. Yet the idea of applying a clustering algorithm seems attractive as a litmus paper to capture the pattern of clustering embedded in iK-Means algorithm.

In spite of the symmetry in the generating model, the sample is slightly biased to the negative side; its mean is -0.89 rather than 0, and its median is about -1.27. Thus the maximum distance from the mean is at the maximum of 32.02 rather than at the minimum of -30.27.

The Anomalous pattern starting from the furthest away value of maximum comprises 83 entities between the maximum and 5.28. Such a stripping goes along

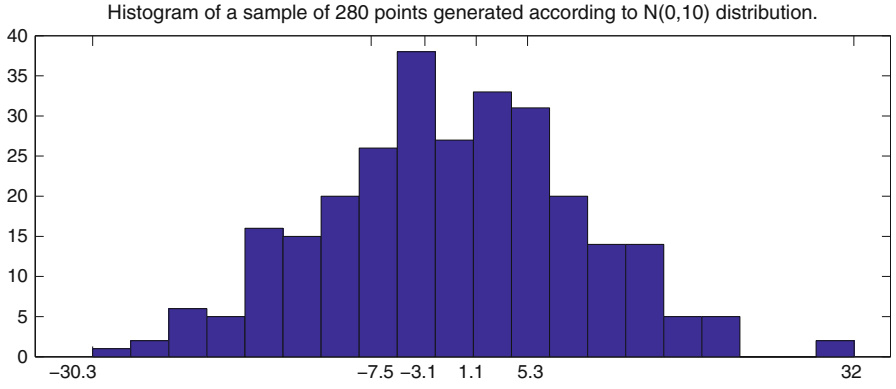

Fig. 6.12 Histogram of the sample of 280 values generated by Matlab's randn command from the Gaussian distribution N(0,10)

Table 6.14 A summary of the iterative Anomalous pattern clustering results for the sample of Gaussian distribution in Table A5.2. Clusters are shown in the extraction order, along with their sizes, left and right boundary entity indices, means and contributions to the data scatter

Order of extraction	Size	Left index	Right index	Mean	Contrib (%)
1	83	198	280	11.35	34.28
2	70	1	70	−14.32	46.03
3	47	71	117	−5.40	4.39
4	41	157	197	2.90	1.11
5	18	118	135	−2.54	0.38
6	10	147	156	0.27	0.002
7	6	136	141	−1.42	0.039
8	2	145	146	−0.49	0.002
9	3	142	144	−0.77	0.006

real-world conventional procedures. For example, consider the heights of a sample of young males to be drafted for a military action whose histogram is known to be bell shaped like Gaussian. Those on the either side of the bell shaped height histogram are not quite fitting for action: those too short cannot accomplish many a specific task whereas those too tall may have problems in closed spaces such as submarines or aircraft.

The iterative Anomalous pattern clustering would sequentially strip the remaining margins off too. The set of fragments of the sorted sequence in Table 6.14 that have been found by the Anomalous pattern clustering algorithm in the order of their forming, including the cluster means and contributions to the data scatter.

The last extracted clusters are all around the mean and, predictably, small in size. One also can see that the contribution of a following cluster can be greater than that of the preceding cluster thus reflecting the local nature of the Anomalous pattern algorithm which intends to find the maximally contributing cluster each time. The total contribution of the nine clusters is about 86% to which the last five clusters contribute next to nothing.

Project 6.1. Using contributions to determine the number of clusters

The question of determining the number K can be addressed with the model (6.16) itself, applied to the cluster contribution values as the raw data. Assume the contributions are sorted in the descending order and denoted by h_k so that $h_1 \geq h_2 \geq \ldots$ $(k = 1, 2, \ldots)$.

If one assumes that the first K values are all approximately equal to each other, whereas the rest approximate zero, then the optimal K can be derived as follows.

Denote the average of the first K contributions as $h(K)$. Then criterion (6.16) to maximize is the product $Kh^2(K)$. The optimal K obviously satisfies inequality $Kh^2(K) > (K+1)h^2(K+1)$. Since the average $h(K+1)$ can be expressed as $h(K+1) = (K^*h(K) + h_{K+1})/(K+1)$, the inequality can be easily transformed to $h^2(K) -$

$2\,h(K)h_{K+1} + {h_{K+1}}^2/K > 0$ which can be further presented as $(h(K) - h_{K+1})^2 > {h_{K+1}}^2(1 - 1/K)$. Since $h(K) \geq h_{K+1}$, this inequality can be further simplified to $h(K) - h_{K+1} > h_{K+1}\sqrt{(1 - 1/K)}$, that is,

$$h(K) > h_{K+1}\left(1 + \sqrt{1 - 1/K}\right) \tag{6.17}$$

which is, roughly, $h_{K+1} < h(K)/2$. This has an advantage that the threshold is not pre-specified but rather determined according to the structure of gaps between the numbers h_k in their sorted order. The value of K at which (6.17) holds can be considered as a candidate for the right number of clusters or components or, in fact, anything evaluated by contributions.

Similar inequalities can be derived at different models for the chosen contribution values. One may try, for example, the power law assumption that $h(k) = ak^{-b}$ for $k = 1, \ldots, K$ and $h(k) = 0$ for $k>K$ (see also Cangelosi and Goriely 2007).

Method iK-Means utilizes a slightly different strategy for choosing the right K. This strategy involves (i) all the anomalous patterns rather than those most contributing, thus involving the patterns close to the reference points too, and (2) a different scoring device – the intuitively clear number of entities rather than a purely geometric contribution whose intuitive value is unclear.

Project 6.2. Does PCA clean the data structure indeed: K-Means after PCA

There is a wide-spread opinion that in a situation of many features, the data structure can come less noisy if the features are first "cleaned off" by applying PCA and using a few principal components instead of the original features. Although strongly debated by specialists (see, for example, Kettenring 2006), the opinion is wide-spread among the practitioners. A voice of dissent was raised by late A. Kryshtanowski (2008) who provided an example of data structure that "becomes less pronounced in the space of principal components".

The example refers to data of two Gaussian clusters, each containing 500 of 15-dimensional entities. The first cluster can be generated by the following MatLab commands:

```
≫b(1:500,1)=10*randn(500,1);
≫b(1:500,2:15)=repmat(b(1:500,1),1,14)+20*randn(500, 14);
```

The first variable in the cluster is Gaussian with the mean 0 and standard deviation 10, whereas the other fourteen variables add to that another Gaussian variable whose mean and standard deviation are 0 and 20, respectively. That is, this set is a sample from a 15-dimensional Gaussian with a diagonal covariance matrix, whose center is in or near the origin of the space, with the standard deviations of all features at 22.36, the square root of 10^2+20^2, except for the first one that has the standard deviation of 10.

The entities in the second cluster are generated as the next 500 rows in the same matrix in a similar manner:

```
≫b(501:1000,1)=20+10*randn(500,1);
≫b(501:1000,2:15)=repmat(b(501:1000,1),1,14)+20*randn(500,14)+10;
```

The first variable now is centered at 20, and the other variables, at 30. The standard deviations follow the pattern of the first cluster.

Since the standard deviations by far exceed the distance between centroids, these clusters are not easy to distinguish: see Fig. 6.13 illustrating the data cloud, after centering, on the plane of the first principal components.

When applying iK-Means to this data, preliminarily centered and range-normalized, the algorithm finds indeed much more clusters, 13 of them, at the discarding threshold $t = 1$. However, when the discarding threshold is set to $t = 200$, to remove any less populated anomalous patterns, the method arrives at just two clusters that differ from those generated by 96 entities (see the very first resulting column in Table 6.15 presenting the results of the computation) constituting the total error of 9.6%. The same method applied to the data z-score standardized, that is, centered and normalized by the standard deviations, arrives at 99 errors; a rather modest increase, probably due to specifics of the data generation.

Since Kryshtanowsky (2008) operated with the four most contributing principal components, we also take the first four principal components, after centering by the means and normalizing data by the range:

```
≫n=1000; br=(b−repmat(mean(b),n,1))./ repmat(max(b)−min(b),n,1);
≫[zr,mr,cr] =svd(br);
≫ zr4=zr(:,1:4);
```

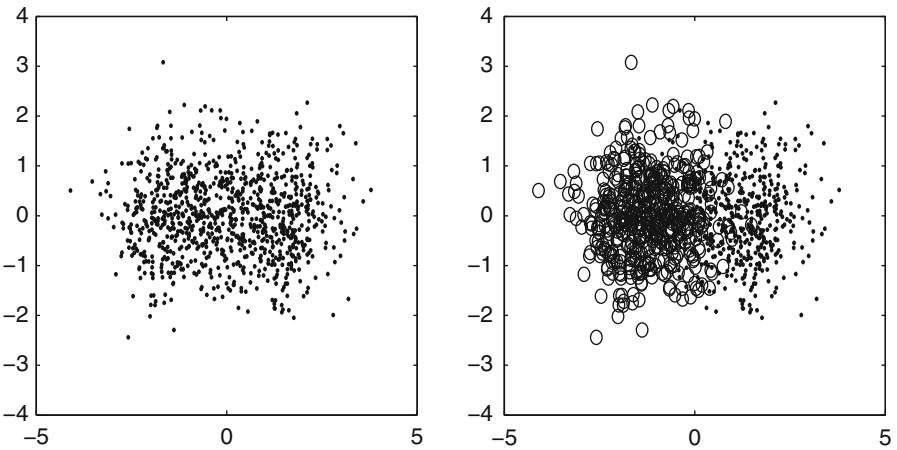

Fig. 6.13 Data of two clusters generated as described above, after centering, are represented by points on the plane of the two first principal components (on the *left*); the second cluster is represented by *circles* on the *right*

Table 6.15 The numbers of errors of iK-Means clustering at different data transformations: over the original data differently normalized and over four principal components derived at different data normalizations

Data	Original 15 features		Four principal components		
Normalized by	Range	St. deviation	No normalization	Range	Standard deviation
Cluster 1	44	43	37	51	47
Cluster 2	52	56	47	47	45
Total	96	99	84	98	92

These four first components bear 66% of the data scatter. We also derived four principal components from the data non-normalized (yet centered) and from the data normalized by the standard deviations (z-score standardized). The latter is especially important in this context, because Kryshtanowsky (2008) used the conventional form of PCA based on the correlation matrix between the variables, which is equivalent to the model-based PCA applied to the data after z-score standardization. The iK-Means method applied to each of these data sets at the discarding threshold of 200, has shown rather consistent results (see Table 6.15).

Overall, these results seem to support the idea of better structuring under principal components rather than to refute it. The negative results of using principal components by Kryshtanowsky (2008) probably could be attributed to his indiscriminate usage of Batch K-Means method with random initializations that failed to find a "right" pair of initial centroids, in contrast to iK-Means.

6.3 Cluster Interpretation Aids

P6.3.1 Cluster Interpretation Aids: Presentation

Results of K-Means clustering, as well as any other method resulting in the list of clusters $S = \{S_1, S_2, \ldots, S_K\}$ and their centroids, $c = \{c_1, c_2, \ldots, c_K\}$, can be interpreted by using

(a) cluster centroids versus grand means (feature averages on the entire data set)
(b) cluster representatives
(c) cluster-feature contributions to the data scatter
(d) conceptual descriptions of clusters

One should not forget that, under the zero-one coding system for categories, cluster-to-category cross-classification frequencies are, in fact, cluster centroids – therefore, (a) includes looking at cross-classifications between S and categorical features although this is conventionally considered a separate interpretation device.

Consider these in turn.

(a) Cluster centroids versus grand means

These should be utilized in both, original and standardized, formats. To express a standardized centroid value c_{kv} of feature v in cluster S_k resulting from a K-Means run, in the original scale of feature v, one needs to invert the scale transformation by multiplying over rescaling factor b_v with the follow up adding the shift value a_v, so that this becomes $b_v c_{kv} + a_v$.

Worked example 6.6. Centroids of Market town clusters

Let us take a look at centroids of the seven clusters of Market towns data both in real and range standardized scales in Table 6.16.

These show some tendencies rather clearly. For instance, the first cluster appears to be a set of larger towns that score 30–50% higher than the average on almost all of the 12 features. Similarly, cluster 3 obviously relates to smaller than average towns. However, in other cases, it is not always clear what features caused a cluster to separate. For instance, both clusters 6 and 7 seem too close to the average to make any real difference at all.

(b) Cluster representative

A cluster is typically characterized by its centroid consisting of the within-cluster feature means. Sometimes, the means make no sense – like the number of suppliers 4.5 above. In such a case, it is more intuitive to characterize a cluster by its

Table 6.16 Patterns of Market towns in the cluster structure found with iK-Means (see Table 6.13). For each of the clusters, real values are on the top line and the standardized values are in the bottom; GM is the grand mean

#	Pop	PS	Do	Ho	Ba	Su	Pe	DIY	SP	PO	CAB	FM
1	18484	7.63	3.63	1.13	11.63	4.63	4.13	1.00	1.38	6.38	1.25	0.38
	0.51	0.38	0.56	0.36	0.38	0.38	0.30	0.26	0.44	0.47	0.30	0.17
2	5268.00	2.17	0.83	0.50	4.67	1.83	1.67	0.00	0.50	1.67	0.67	1.00
	−0.10	−0.07	−0.14	0.05	0.02	−0.01	−0.05	−0.07	0.01	−0.12	0.01	0.80
3	2597	1.17	0.50	0.00	1.22	0.61	0.89	0.00	0.06	1.44	0.11	0.00
	−0.22	−0.15	−0.22	−0.20	−0.16	−0.19	−0.17	0.07	−0.22	−0.15	−0.27	−0.20
4	11245	3.67	2.00	1.33	5.33	2.33	3.67	0.67	1.00	2.33	1.33	0.00
	0.18	0.05	0.16	0.47	0.05	0.06	0.23	0.15	0.26	−0.04	0.34	−0.20
5	5347	2.50	0.00	1.00	2.00	1.50	2.00	0.00	0.50	1.50	1.00	0.00
	−0.09	−0.04	−0.34	0.30	−0.12	−0.06	−0.01	−0.07	0.01	−0.14	0.18	−0.20
6	8675	3.80	2.00	0.00	3.20	2.00	2.40	0.00	0.00	2.80	0.80	0.00
	0.06	0.06	0.16	−0.20	−0.06	0.01	0.05	−0.07	−0.24	0.02	0.08	−0.20
7	5593	2.00	1.00	0.00	5.00	2.67	2.00	0.00	1.00	2.33	1.00	0.00
	−0.08	−0.09	−0.09	−0.20	0.04	0.10	−0.01	−0.07	0.26	−0.04	0.18	−0.20
GM	7351.4	3.02	1.38	0.40	4.31	1.93	2.04	0.22	0.49	2.62	0.64	0.20

"typical" representative. This is especially appealing when the representative is a well known object. Such an object can give much better intuition to a cluster than a logical description in situations in which entities are complex and the features are superficial. This is the case, for instance, in mineralogy where a class of minerals can be represented by its "stratotype" mineral, or in art studies where a general concept such as "surrealism" can be represented by an art object such as a painting by S. Dali.

A cluster representative must be the nearest to its cluster's centroid. An issue is that two different expressions for K-Means lead to two different measures. The sum of entity-to-centroid distances W(S,c) in (6.3) leads to the strategy that can be referred to as "the nearest in distance." The sum of entity-to-centroid inner products for B(S,c) in (6.8) leads to the strategy "the nearest in inner product". Intuitively, the choice according to the inner product follows tendencies represented in c_k towards the whole of the data expressed in grand mean position whereas the distance follows just c_k itself. These two principles usually lead to similar choices, though sometimes rather not.

Worked example 6.7. Representatives of Company clusters

Consider, for example, A product cluster in Company data as presented in Table 6.17: The nearest to centroid in distance is Ant and nearest in inner product is Ave.

To see why is that, let us take a closer look at the two companies. Ant and Ave are similar on all four binary features. Each is at odds with the centroid's tendency on one feature only: Ant is zero on NSup while centroid is negative, and Ave is negative on Income while centroid is positive on that. The difference, however, is in feature contributions to the cluster; that of Income is less than that of NSup, which makes Ave to win, as a follower of NSup, over the inner product expressing contributions of entities to the data scatter. With the distance measure, the cluster tendency by itself does not matter at all because it is expressed in the signs of the standardized centroid.

Q.6.21. Find representatives of Company clusters B and C.

Table 6.17 Standardized entities and centroid of cluster A in company data. The nearest to centroid are: Ant, in distance, and Ave, in inner product (both are in thousandth)

Cluster	Income	SharP	NSup	EC	Util	Indu	Retail	Distance	InnerPr
Centroid	0.10	0.12	−0.11	−0.63	0.17	−0.02	−0.14		
Ave	−0.20	0.23	−0.33	−0.63	0.36	−0.22	−0.14	222	**524**
Ant	0.40	0.05	0.00	−0.63	0.36	−0.22	−0.14	**186**	521
Ast	0.08	0.09	0.00	−0.63	−0.22	0.36	−0.14	310	386

(c) Feature-cluster contributions to the data scatter

To see what features do matter in each of the clusters, the contributions of feature-cluster pairs to the data scatter are to be invoked. The feature-cluster contribution is equal to the product of the squared feature (standardized) centroid component and the cluster size. In fact, this is proportional to the squared difference between the feature's grand mean and its within-cluster mean: the further away the latter from the former, the greater the contribution! This is illustrated on Fig. 6.14:

Worked example 6.8. Contributions of features to Market town clusters

The cluster-specific feature contributions are presented in Table 6.18, along with their total contributions to the data scatter in row Total. The intermediate rows Exp and Unexp show the explained and unexplained parts of the totals, with Exp being the sum of all cluster-feature contributions and Unexp the difference between the Total and Exp rows.

The columns on the right show the total contributions of clusters to the data scatter, both as is and per cent. The cluster structure in total accounts for 73.3%

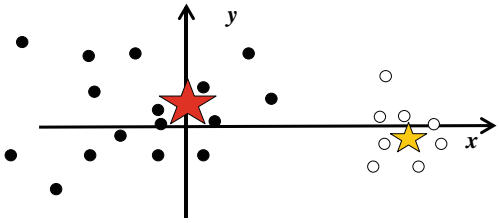

Fig. 6.14 Contributions of features *x* and *y* in the group of *blank-circled* points are proportional to the squared differences between their values at the grand mean (*large star*) and within-group centroid (*small star*)

Table 6.18 Decomposition of the data scatter over clusters and features at market town data; row Exp sums all the cluster contributions, row Total gives the feature contributions to the data scatter, and row Unexp is the difference, Total-Exp

#	P	PS	Do	Ho	Ba	Su	Pe	DIY	SP	PO	CAB	FM	Total	Total (%)
1	**2.09**	1.18	**2.53**	1.05	1.19	1.18	0.71	0.54	1.57	1.76	0.73	0.24	14.77	35.13
2	0.06	0.03	0.11	0.01	0.00	0.00	0.02	0.03	0.00	0.09	0.00	**3.84**	4.19	9.97
3	0.86	0.43	0.87	0.72	0.48	0.64	0.49	0.10	0.85	0.39	**1.28**	0.72	7.82	18.60
4	0.10	0.01	0.07	**0.65**	0.01	0.01	0.16	0.07	0.20	0.00	0.36	0.12	1.75	4.17
5	0.02	0.00	**0.24**	0.18	0.03	0.01	0.00	0.01	0.00	0.04	0.06	0.08	0.67	1.59
6	0.02	0.02	0.12	0.20	0.02	0.00	0.01	0.03	**0.30**	0.00	0.03	**0.20**	0.95	2.26
7	0.02	0.02	0.03	0.12	0.00	0.03	0.00	0.02	**0.20**	0.00	0.09	0.12	0.66	1.56
Exp	3.16	1.69	3.96	2.94	1.72	1.88	1.39	0.79	3.11	2.29	2.56	5.33	30.81	73.28
Unexp	0.40	0.59	0.70	0.76	0.62	0.79	1.02	0.96	1.20	0.79	1.52	1.88	11.23	26.72
Total	3.56	2.28	4.66	3.70	2.34	2.67	2.41	1.75	4.31	3.07	4.08	7.20	42.04	100.00

of the data scatter, a rather high proportion. Of the total contributions, three first clusters have the largest ones totaling to 26.78, or 87% of the explained part of the data scatter, 30.81. Among the variables, FM gives the maximum contribution to the data scatter, 5.33. This can be attributed to the fact that FM is a binary variable on the data set – binary variables, in general, have the largest total contributions because they are bimodal. Indeed, FM's total contribution to the data scatter is 7.20 so that its explained part amounts to $5.33/7.20 = 0.74$ which is not as much as that of, say, the Population resident feature, $3.16/3.56 = 0.89$, which means that overall Population resident better explains the clusters than FM.

Worked example 6.9. Contributions and relative contributions of features at Company clusters

Consider the clustering of Companies data according to their main product, A or B or C, to find out what features can be associated with each of the clusters. The cluster centroids as well as feature-cluster contributions are presented in Table 6.19. The summary contributions over clusters are presented in the last column of Table 6.19, and over features, in the first line of third row of Table 6.19 termed "Explain". The feature contributions to the data scatter, that is, the sums of squares of the feature's column entries, are in the second line of the third row – these allow us to express the explained feature contributions per cent, in the third line.

Now we can take a look at the most contributing feature-to-cluster pairs. This can be done by considering relative contributions within individual lines (clusters) or within individual columns (features). For example, in the third row of Table 6.19, each contribution that covers half or more of the explained contribution by the feature is highlighted in bold. Obviously, the within-line maxima do not necessarily

Table 6.19 Centroids and feature-to-cluster contributions for product clusters in company data

Item		Income	SharP	NSup	EC	Util	Indu	Retail	Total
Cluster	A	0.10	0.12	−0.11	−0.63	0.17	−0.02	−0.14	
centroids	B	−0.21	−0.29	−0.22	0.38	−0.02	0.17	−0.14	
standardized	C	0.18	0.24	0.50	0.38	−0.22	−0.22	0.43	
Cluster	A	0.03	0.05	0.04	**1.17**	0.09	0.00	0.06	1.43
contributions	B	**0.14**	**0.25**	0.15	0.42	0.00	0.09	0.06	1.10
	C	0.06	0.12	**0.50**	0.28	0.09	**0.09**	**0.38**	1.53
Total	Ex	0.23	0.41	0.69	1.88	0.18	0.18	0.50	4.06
contributions	Data	0.74	0.69	0.89	1.88	0.63	0.63	0.50	5.95
	Ex%	31.1	59.4	77.5	100.0	28.6	28.6	100.0	68.3
Relative	A	16.7	29.5	18.5	**258.1**	59.9	0.0	49.5	
contribution	B	101.2	**191.1**	90.2	120.2	0.0	77.7	64.2	
indexes, %	C	31.7	67.0	**219.5**	58.5	56.7	56.7	**297.0**	
Cluster		24.10	39.23	2.67	**0.00**	0.67	0.33	0.00	
centroids real		18.73	**22.37**	2.33	1.00	0.33	0.67	0.00	
		25.55	44.10	**4.50**	1.00	0.00	0.00	**1.00**	

match those within columns. The relative contribution indexes in the fourth row of Table 6.19 combine these two perspectives: they are ratios of two relative contributions: the relative explained feature contribution within a cluster to the relative feature contribution to the data scatter. For example, relative contribution index of feature ShareP to cluster B, 1.911, is found by relating its relative explained contribution 0.25/1.10 to its relative contribution to the data scatter, 0.69/5.95. Those of the relative contribution indexes that are greater than 150%, so that the feature contribution to the cluster structure is at least 50% greater than its contribution to the data scatter, are highlighted.

The most contributing features are those that make the clusters different. To see this, Table 6.19 is supplemented with the real values of the within-cluster feature means, in its last row. The values corresponding to the outstanding contributions are highlighted in bold. Cluster A differs by feature EC – A-listed companies do not use e-commerce; cluster B differs by the relatively low Share Prices; and cluster C differs by either the fact that it all falls within Retail sector or the fact that its companies have relatively high numbers of suppliers, 4 or 5. It is easy to see that each of these statements not only points to a tendency but distinctly describes the cluster as a whole.

Case-Study 6.6. 2D Analysis of Most Contributing Features

Consider 2D analysis of the relationship between the Company data partition in three product classes, A, B, and C, and the most contributing of the quantitative features in Table 6.19 – the Number of suppliers (77.5%) as illustrated in Table 6.20.

To calculate the correlation ratio of the NSup feature according to formula ( ), let us first calculate the average within-class variance $\sigma_u^2 = (3^*0.22 + 3^*0.022 + 2^*0.25/8 = 0.23$; the correlation ratio then will be equal to $\eta^2 = (\sigma^2 - \sigma_u^2)/\sigma^2 = (1.00 - 0.23)/1.00 = 0.77$. According to (6.13), this, multiplied by $N = 8$ and $\sigma^2 = 1$, must be equal to the total explained contribution of feature NSup to the data scatter in Table 6.21, 0.69 – which is clearly not! Why? Because the correlation ratio in (6.13) refers to the standardized, not original feature, and to make up for this, one needs to divide the result by the squared scaling parameter, the range r, which is 3 in this case. Now we get things right: $\eta^{2*}N/r^2 = 0.77^*8/9 = 0.69$ indeed!

Table 6.20 Tabular regression of NSup feature over the product-based classes in the company dataset in Table 5.2

Classes	#	NSup mean	NSup variance
A	3	2.67	0.22
B	3	2.33	0.22
C	2	4.50	0.25
Total	8	3.00	1.00

Table 6.21 Contingency table between the product-based classes and nominal feature Sector in the company dataset according to Table 5.1

Category Class	Utility	Industrial	Retail	Total
A	2	1	0	3
B	1	2	0	3
C	0	0	2	2
Total	3	3	2	8

Table 6.22 Relative frequencies together with absolute and relative Quetélet indexes for contingency Table 6.21

Cat. class	Utility	Indust	Retail	Total	Utility	Indust	Retail	Utility	Indust	Retail
	Relative frequencies				Absolute Quetélet ind.			Relative Quetélet ind.		
A	0.25	0.12	0.00	0.37	0.29	−0.04	−0.25	0.78	−0.11	−1.00
B	0.12	0.25	0.00	0.37	−0.04	0.29	−0.25	−0.11	0.78	−1.00
C	0.00	0.00	0.25	0.25	−0.38	−0.38	0.75	−1.00	−1.00	3.00
Total	0.37	0.37	0.25	1.00						

The case of a nominal feature can be analyzed similarly. Consider contingency table between the product based partition S and feature Sector in Company data (Table 6.19).

In contrast to the classical statistics perspective, the small and even zero values are not of an issue here. Table 6.22 presents, on the left, the same data in the relative format; the other two parts present absolute and relative Quetelet indexes as described in Section 3.4.

These indexes have something to do with the cluster-feature contributions in Table 6.19. Given that the categories have been normalized by unities as well as the other features, the absolute Quetélet indexes are involved. [To use the relative Quetélet indexes, the categories have to be normalized by the square roots of their frequencies, as explained in the Formulation part.] Their squares multiplied by the cluster cardinalities and additionally divided by the squared rescaling parameter, 3 in this case, are the contributions, according to formula (6.14), as presented in the Table 6.23.

Table 6.23 Absolute Quetélet indexes from Table 6.22 and their squares factored according to formula (6.14)

Cat. class	Size	Utility	Industrial	Retail	Utility	Industrial	Retail	Total
		Absolute Quetélet indexes			Contributions			
A	3	0.29	−0.04	−0.25	**0.085**	0.002	0.062	0.149
B	3	−0.04	0.29	−0.25	0.002	**0.085**	0.062	0.149
C	2	−0.38	−0.38	0.75	0.094	0.094	**0.375**	0.563
Total	8				0.181	0.181	0.500	0.862

Obviously, all entries in the right part of Table 6.23 are items in the total Proportional prediction index (6.16) divided by 3 – because of the specifics of the data normalization with the additional normalization by the square root of the number of categories.

(d) Conceptual description of clusters

If a contribution is high, then, as can be seen on Fig. 6.14, it is likely that the corresponding feature can be utilized for conceptual description of the corresponding class.

Worked example 6.10. Describing Market town clusters conceptually

Consider, for example, Table 6.18 of contributions of the clusters found at Market towns data. Several entries in Table 6.18 are highlighted in bold as those most contributing to the data scatter parts explained by clusters, the columns on the right. Take a look at them, cluster-wise.

Cluster 1 is indeed characterized by its two most contributing features, Population resident (P, contribution 2.09) and the number of doctor surgeries (Do, contribution 2.53). It can be described as a "set of towns with the population resident P not less than 10,200 and number of doctor surgeries Do not less than 3" – this description perfectly fits the cluster with no errors, be it false positive or false negative. Cluster 2 is blessed with an unusually high relative contribution of FM, 3.84 of the total 4.19; this may be seen as the driving force of the cluster's separation: it comprises all the towns with a Farmers market that have not been included in cluster 1! Other clusters can be described similarly. Let us note the difference between clusters 6 and 7, underlined by the high contributions of swimming pools (SW) to both, though by different reasons: every town in cluster 7 has a swimming pool whereas any town in cluster 6 has none.

Worked example 6.11. Describing Company clusters conceptually

Conceptual descriptions can be drawn for the product clusters in Company data according to Table 6.19. This table shows that feature EC is the most contributing to the Product A cluster, feature ShaP to the Product B cluster, and features SupN and Retail to the Product C cluster. The relatively high contribution of ShaP to B cluster is not that obvious because that of EC, 0.42, is even higher. It becomes clear only on the level of relative contributions when the contributions are related to their respective Total counterparts, 0.25/0.69 and 0.42/1.88 – the former prevails indeed. Clusters A, B, and C can be distinctively described by the statements "EC==0", "ShaP < 28", and "SupN >3" (or "Sector is Retail"), respectively.

Unfortunately, high feature contributions not always lead to clear-cut conceptual descriptions. The former are based on the averages whereas the latter on clear-cut divisions, and division boundaries can be at odds with the averages.

F6.3.2 Cluster Interpretation Aids: Formulation

According to (6.4) and (6.5), clustering *(S,c)* decomposes the scatter $T(Y) = \Sigma_{i,v} y_{iv}^2$ of data matrix Y in the explained and unexplained parts, *B(S,c)* and *W(S,c)*, respectively. The latter is the square-error K-Means criterion, whereas the explained part *B(S,c)* is clustering's contribution to the data scatter, which is equal, according to (6.6), to

$$B(S,c) = \sum_{k=1}^{K} \sum_{v \in V} c_{kv}^2 N_k$$

This is the sum of additive items $B_{kv} = N_k c_{kv}^2$, each accounting for the contribution of feature-cluster pair, $v \in V$ and $S_k (k = 1, 2, \ldots, K)$.

Since the total contribution of feature v to the data scatter is $T_v = \sum_{i \in I} y_{iv}^2$, its unexplained part can be expressed as $W_v = T_v - B_{+v}$ where $B_{+v} = \sum_{k=1}^{K} B_{kv}$ is feature's v explained part, the total contribution of v to the cluster structure. This can be displayed as a Scatter Decomposition (ScaD) table whose rows correspond to clusters, columns to variables and entries to the contributions B_{kv} (see Table 6.24).

The summary rows, Explained, Unexplained and Total, as well as column Total can be expressed as percentages of the data scatter *T(Y)*. The contributions highlight relative roles of features both at individual clusters and in total.

The explained part *B(S,c)* is, ccording to (6.6), the sum of contributions of individual feature-to-cluster pairs $B_{kv} = c_{kv}^2 N_k$ which can be used for interpretation of the clustering results. The sums of B_{kv}'s over features or clusters

Table 6.24 ScaD: Data scatter decomposed over clusters and features using notation introduced above

Feature				
Cluster	f_1	f_2	f_M	Total
S_1	B_{11}	B_{12}	B_{1V}	B_{1+}
S_2	B_{21}	B_{22}	B_{2V}	B_{2+}
S_K	B_{K1}	B_{K2}	B_{KV}	B_{K+}
Explained	B_{+1}	B_{+2}	B_{+V}	$B(S,c)$
Unexplained	W_{+1}	W_{+2}	W_{+V}	$W(S,c)$
Total	T_1	T_2	T_V	$T(Y)$

express total contributions of individual clusters or features into the explanations of clusters.

As has been shown in Section 4.5.3, summary contributions of individual data features to clustering *(S,c)* have something to do with statistical measures of association in bivariate data, such as correlation ratio η^2 (3.15) in Section 3.3 and chi-squared X^2 (3.13) in Section 3.4 (Mirkin 2005). In fact, the analysis in Section 4.5.3.2 applies in full to the case when target features are those used for building clustering S.

Specifically, for a quantitative feature v represented by the standardized column y_v, its summary contribution B_{+v} to the data scatter is equal to

$$B_{+v} = N\sigma_v^2\eta_v^2 \tag{6.18}$$

Note that the correlation ratio in (6.18) has been computed over the normalized feature y_v. The correlation ratio of the original non-standardized feature x_v differs from that by factor equal to the squared rescaling parameter b_v^2.

Consider now a nominal feature v represented by a set of binary columns, dummies, corresponding to individual categories $l \in v$. The grand mean of binary column for $v \in F$ is obviously the proportion of this category in the set, p_{+v}. To standardize the column, one needs to subtract the mean, p_{+v}, from all its entries and divide them by the scaling parameter, b_v. After the standardization, the centroid of cluster S_k can be expressed through co-occurrence proportions too, as expressed by the formula on p. 149:

$$c_{kv} = \left(\frac{p_{kv}}{p_k} - p_{+v}\right)/b_v$$

where p_{kv} is the proportion of entities falling in both category v and cluster S_k; the other symbols: p_{+v} is the frequency of v, p_k the proportion of entities in S_k, and b_v the normalizing scale parameter.

According to Equations (4.18) and (4.19), the summary contribution of all pairs category-cluster *(l,k)* is equal to

$$B(v/S) = N\sum_{l\in v}\sum_{k=1}^{K}\frac{(p_{kl} - p_k p_{+l})^2}{p_k b_l^2} \tag{6.19}$$

This is akin to several contingency table association measures considered in the literature including Pearson chi-squared X^2 in (3.15) and Gini impurity function, or summary absolute Quetelet index, in (3.22). To make *B(v/S)* equal to the chi-squared coefficient, the scaling of binary features must be done by using $b_l = \sqrt{p_l}$, which is the standard deviation of the Poisson probabilistic distribution that randomly throws $p_l N$ unities into an N-dimensional binary vector. To make *B(v/S)* equal to Gini impurity function, no normalization of the dummies is to be done, or rather the recommended option of normalization by ranges applies since the range of a dummy is 1.

One should not forget the additional normalization of the binary columns by the square root of the number of categories in a nominal feature v, $\sqrt{|v|}$ leading to both the individual contributions B_{kv} in (6.18) and the total contribution $B(v/S)$ in (6.19) divided by the number of categories $|v|$. When applied to Pearson chi-squared, the division by $|v|$ can be considered as another normalization of the coefficient. As mentioned in Section 3.4, the maximum of Pearson chi-squared (related to N) is $\min(|v|, K) - 1$. Therefore, when $|v| \leq K$, the division would lead to a normalized index whose values are between 0 and $1 - 1/|v|$. If, however, the number of categories is larger so that $K < |v|$, then the normalized index could be very near 0 indeed. In this regard, it should be of interest to mention that in the literature some other normalizations have been considered. Specifically, Pearson chi-squared is referred to as Cramer coefficient if related to $\min(|v|, K) - 1$, and as Tchouproff coefficient if related to $\sqrt{(|v| - 1)(K - 1)}$ (Kendall and Stewart 1973).

Q.6.22. Prove that, for any cluster k in K-Means clustering, $\sum_{i \in S_k} y_{iv}^2 = |S_k|(c_{kv}^2 + \sigma_{kv}^2)$.

Q.6.23. How one should interpret the normalization of a category by the $\sqrt{p_v}$? What category gets a greater contribution: that more frequent or that less frequent?

Comment 6.1. When the chi-squared contingency coefficient or related indexes are applied in the traditional statistics context, the presence of zeros in a contingency table becomes an issue because it contradicts the hypothesis of statistical independence. In the context of data recovery clustering, zeros are treated as any other numbers and create no problems at all because the coefficients are measures of contributions and bear no other statistical meaning in this context.

Comment 6.2. K-Means advantages: The method

(i) Models typology building activity
(ii) Computationally effective both in memory and time
(iii) Can be utilized incrementally, "on-line"
(iv) Straightforwardly associates feature salience weights with feature scales
(v) Applicable to both quantitative and categorical data and mixed data provided that care has been taken of the relative feature scaling
(vi) Provides a number of interpretation aids including cluster prototypes and features and entities most contributing to cluster specificity.

K-Means issues:

(vii) Simple convex spherical shape of clusters.
(viii) Choosing the number of clusters and initial seeds.
(ix) Instability of results with respect to initial seeds.

Although conventionally considered as shortcomings, issues vii-ix can be beneficial too. To cope with issue vii, the feature set should be chosen carefully. Then the simple shape of a cluster will provide for a simpler conceptual description of it. To cope with issue viii, the initial seeds should be selected not randomly but rather based

Table 6.25 Decomposition of the data scatter over product clusters in Company data; notations are similar to those in Table 6.18

Product	Income	ShaP	SupN	EC	Util	Indu	Retail	Total	Total %
A	0.03	0.05	0.04	**1.17**	0.00	0.09	0.06	1.43	24.08
B	0.14	**0.25**	0.15	0.42	0.09	0.00	0.06	1.10	18.56
C	0.06	0.12	**0.50**	0.28	0.09	0.09	**0.38**	1.53	25.66
Exp	0.23	0.41	0.69	1.88	0.18	0.18	0.50	4.06	68.30
Unexp	0.51	0.28	0.20	0.00	0.44	0.44	0.00	1.88	31.70
Total	0.74	0.69	0.89	1.88	0.63	0.63	0.50	5.95	100.00

on preliminary analysis of the substantive domain or using anomalous approaches Build or AP. Another side of issue ix is that solutions are close to pre-specified centroids, which is good when the centroids have been chosen carefully.

Q.6.24. Find SCAD decomposition for the product clusters in Company data. **A.** This is in Table 6.25. Table 6.25 shows feature EC as the one most contributing to the Product A cluster, feature ShaP to the Product B cluster, and features SupN and Retail to the Product C cluster. The relatively high contribution of ShaP to B cluster is not that obvious because that of EC, 0.42, is higher. It becomes clear only on the level of relative contributions relating the absolute values to their respective Exp counterparts, 0.25/0.41 and 0.42/1.88 – the former prevails indeed. Clusters A, B, and C can be distinctively described by statements "EC==0", "ShaP < 28", and "SupN >3" (or "Sector is Retail"), respectively.

6.4 Extension of K-Means to Different Cluster Structures

So far the clustering was to encode a data set with a number of clusters forming a partition. Yet there can be differing partition-like clustering structures of which, arguably, the most popular are:

I *Fuzzy*: Cluster membership of entities may be not necessarily confined to one cluster only but shared among several clusters;
II *Probabilistic*: Clusters can be represented by probabilistic distributions rather than manifolds;
III *Self-Organizing Map (SOM)*: Capturing clusters within cells of a plane grid along with the grid's neighborhood structure.

Further on in this section extensions of K-Means to these structures are presented.

6.4.1 Fuzzy K-Means Clustering

A fuzzy cluster is represented by its membership function $z = (z_i),\ i \in I$, in which z_i $(0 \leq z_i \leq 1)$ is interpreted as the degree of membership of entity i to the cluster.

This extends the concept of conventional, hard (crisp) cluster, which can be considered a special case of the fuzzy cluster corresponding to membership z_i restricted to only 1 or 0 values.

A conventional (crisp) cluster k ($k = 1, \ldots, K$) can be thought of as a pair consisting of centroid $c_k = (c_{k1}, \ldots, c_{kv}, \ldots, c_{kV})$ in the V feature space and membership vector $z_k = (z_{1k}, \ldots, z_{ik}, \ldots, z_{Nk})$ over N entities so that $z_{ik} = 1$ means that i belongs to cluster k, and $z_{ik} = 0$ means that i does not. Moreover, clusters form a partition of the entity set so that every i belongs to one and only one cluster if and only if $\Sigma_k z_{ik} = 1$ for every $i \in I$.

These are extended to the case of fuzzy clusters, so that fuzzy cluster k ($k = 1, \ldots, K$) is a pair comprising centroid $c_k = (c_{k1}, \ldots, c_{kv}, \ldots, c_{kV})$, a point in the feature space, and membership vector $z_k = (z_{1k}, \ldots, z_{ik}, \ldots, z_{Nk})$ such that all its components are between 0 and 1, $0 \leq z_{ik} \leq 1$, expressing the extent of belongingness of i to each of the clusters k. Fuzzy clusters form what is referred to as a fuzzy partition of the entity set, if the summary membership value of every entity $i \in I$ is unity, that is, $\Sigma_k z_{ik} = 1$ for each $i \in I$. One may think of the total membership of any entity i as a substance that can be differently distributed among the centroids.

These concepts are especially easy to grasp if membership value z_{ik} is considered as the probability of belongingness. However, in many cases fuzzy partitions have nothing to do with probabilities. For instance, dividing all people by their height may involve fuzzy categories "short," "medium" and "tall" with fuzzy meanings such as those shown in Fig. 6.15.

Fuzzy clustering can be of interest in applications related with natural fuzziness of cluster boundaries such as in image analysis, robot planning, geography, etc.

If fuzzy cluster membership values are put into the PCA model, as K-Means crisp memberships have been (see formula (6.12) in Section 6.2.2.2), they make a rather weird structure in which centroids are not average but rather extreme points in their clusters, which can be relaxed in a certain way and make clusters appealing, if somewhat unusual (Nascimento 2005).

An empirically convenient criterion (6.20) below differently extends that of (6.3) by using the Euclidean squared distance d and factoring in an exponent of the membership, z^α. The value α affects the fuzziness of the optimal solution: at $\alpha = 1$,

$$F(\{c_k, z_k\}) = \sum_{k=1}^{K} \sum_{i=1}^{N} z_{ik}{}^{\alpha} d(y_i, c_k) \tag{6.20}$$

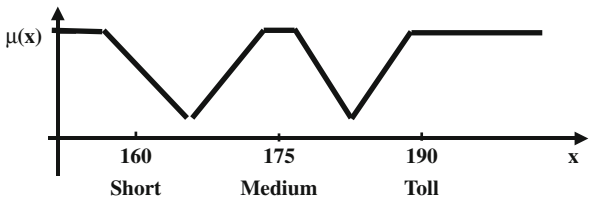

Fig. 6.15 Possible trapezoid fuzzy sets corresponding to fuzzy concept of man's height: short, regular, and toll

the optimal membership values are proven to be crisp, the larger the α the "smoother" the membership values. Usually α is taken to be $\alpha = 2$.

Globally minimizing criterion (6.20) is a difficult task. Yet the alternating minimization of it appears rather easy. As usual, this works in iterations starting, from somehow initialized centroids. Each iteration proceeds in two steps: (1) given cluster centroids, cluster membership values are updated; (2) given membership values, centroids are updated – after which everything is ready for the next iteration. The process stops when the updated centroids are close enough to the previous ones. Updating formulas are derived from the first-order optimality conditions. They require the partial derivatives of the criterion over the optimized variables to be set to 0.

Membership values update formula:

$$z_{ik} = 1/\sum_{k'=1}^{K} [d(y_i, c_k)/d(y_i, c_{k'})]^{\frac{1}{\alpha-1}} \tag{6.21}$$

Centroids update formula:

$$c_{kv} = \sum_{i=1}^{N} {z_{ik}}^{\alpha} y_i / \sum_{i'=1}^{N} {z_{i'k}}^{\alpha} \tag{6.22}$$

Since Equations (6.21) and (6.22) are the first-order optimality conditions for criterion (6.20) leading to unique solutions, convergence of the method, usually referred to as fuzzy K-Means (*c*-means, too, assuming *c* is the number of clusters, see Bezdek et al. 1999), is guaranteed.

Yet the meaning of criterion (6.20) has not been paid much attention to until recently. It appears, criterion F in (6.20) can be presented as $F = \Sigma_i F(i)$, the sum of weighted distances $F(i)$ between points $i \in I$ and cluster centroids, so that $F(i)$ is equal to the harmonic average of the individual memberships at $\alpha = 2$ (see Stanforth, Mirkin and Kolossov 2007, where this fact is used for the analysis of domain of applicability for predicting toxicity of chemical compounds). Figure 6.16 presents the indifference contours of the averaged F values versus those of the nearest centroids. The former look much smoother.

The Anomalous pattern method is applicable as a tool for initializing Fuzzy K-Means as well as crisp K-Means, leading to reasonable results as reported by Stanforth, Mirkin and Kolossov 2007. Nascimento and Franco (2009) applied this method for segmentation of sea surface temperature maps; found fuzzy clusters closely follow the expert-identified regions of the so-called coastal upwelling, that are relatively cold, and nutrient rich, water masses. In contrast, the conventional fuzzy K-Means, with user defined K, under- or over-segments the images.

Q.6.25. Regression-wise clustering. In general, centroids c_k can be defined in a space which is different from that of the entity points y_i ($i \in I$). Such is the case of regression-wise clustering. Recall that a regression function $x_V =$

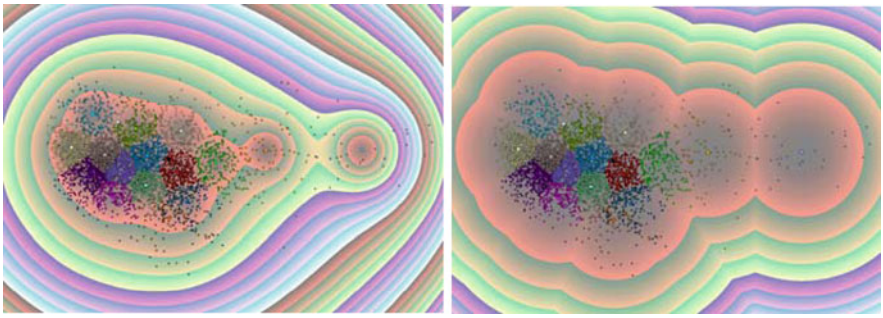

Fig. 6.16 Maps of the indifference levels for the membership function *F(i)* at about 14,000 chemical compounds clustered with iK-Means in 41 clusters (**a**); (**b**) scores membership using only the nearest cluster's centroid

$f(x_1, x_2, \ldots, x_{V-1})$ may relate a target feature, x_V, to (some of the) other features $x_1, x_2, \ldots, x_{V-1}$ as, for example, the price of a product to its consumer value and production cost attributes. In regression-wise clustering, entities are grouped together according to the degree of their correspondence to a regression function rather than according to their closeness to the gravity center. That means that regression functions play the role of centroids in regression-wise clustering (see Fig. 6.17).

Consider a version of Straight K-Means for regression-wise clustering to involve linear regression functions relating standardized variable y_V to other standardized variables, $y_1, y_2, \ldots, y_{V-1}$, in each cluster. Such a function is defined by the equation $y_V = a_1 y_1 + a_2 y_2 + \ldots + a_{V-1} y_{V-1} + a_0$ for some coefficients $a_0, a_1, \ldots, a_{V-1}$. These coefficients form a vector, $a = (a_0, a_1, \ldots, a_{V-1})$, which can be referred to as a regression-wise centroid.

When a regression-wise centroid is given, its distance to an entity point $y_i = (y_{i1}, \ldots, y_{iV})$ is defined as $r(i, a) = (y_{iV} - a_1 y_{i1} - a_2 y_{i2} - \ldots - a_{V-1} y_{i,V-1} - a_0)^2$, the squared difference between the observed value of y_V and that calculated from the regression equation. To determine the regression-wise centroid *a(S)*, given a

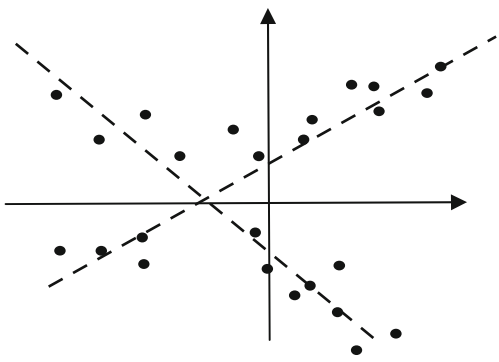

Fig. 6.17 Two regression-wise clusters with their regression *lines* as centroids

cluster list $S \subseteq I$, the standard technique of multivariate linear regression analysis is applied, which is but minimizing the within cluster summary residual $\sum_{i \in S} r(i,a)$ over all possible a.

Formulate a version of the Straight K-Means for this situation.

Hint: Same as Batch K-Means, except that:

(1) centroids must be regression-wise centroids and
(2) the entity-to-centroid distance must be *r(i,a)*.

6.4.2 Mixture of Distributions and EM Algorithm

Data of financial transactions or astronomic observations can be considered as a random sample from a (potentially) infinite population. In such cases, the data structure can be analyzed with probabilistic approaches of which arguably the most radical is the mixture of distributions approach.

According to this approach, each of the yet unknown clusters k is modeled by a density function $f(x, \alpha_k)$ which represents a family of density functions over x defined up to a parameter vector α_k. Consider a one-dimensional density function *f(x)*, that, for any x and very small change dx, assigns its probability *f(x)dx* to the interval between x *and* $x+dx$, so that the probability of any interval *(a,b)* is integral $\int_a^b f(x)dx$, which is the area between x-axis and *f(x)* within *(a,b)* as illustrated on Fig. 6.18 for interval (6,8). Multidimensional density functions have a similar nterpretation.

Usually, the cluster density $f(x, \alpha_k)$ is considered uni-modal with the mode corresponding to the cluster standard point. Such is the normal, or Gaussian, density function defined by α_k consisting of its mean vector m_k and covariance matrix $\sum_k$:

$$f(x, m_k, \Sigma_k) = \exp(-(x - m_k)^T \Sigma_k^{-1} (x - m_k)/2) \Big/ \sqrt{(2\pi)^V |\Sigma_k|} \tag{6.23}$$

The shape of Gaussian clusters is ellipsoidal because any surface at which $f(x, \alpha_k)$ is constant satisfies equation $(x - m_k)^T \Sigma_k^{-1} (x - m_k) = c$, where c is any constant, that defines an ellipsoid. This is why the PCA representation is highly compatible with the assumption of the underlying distribution being Gaussian. The mean vector m_k specifies the k-th cluster's location.

The mixture of distributions clustering model can be set as follows. The row points $y_1, y_2, \ldots, y_N$ are considered a random sample of $|V|$-dimensional observations from a population with density function *f(x)* which is a mixture of individual cluster density functions $f(x, \alpha_k)(k = 1, 2, \ldots, K)$ so that $f(x) = \sum_{k=1}^{K} p_k f(x, \alpha_k)$, where $p_k \geq 0$ are the mixture probabilities such that $\sum_{k=1}^{K} p_k = 1$.

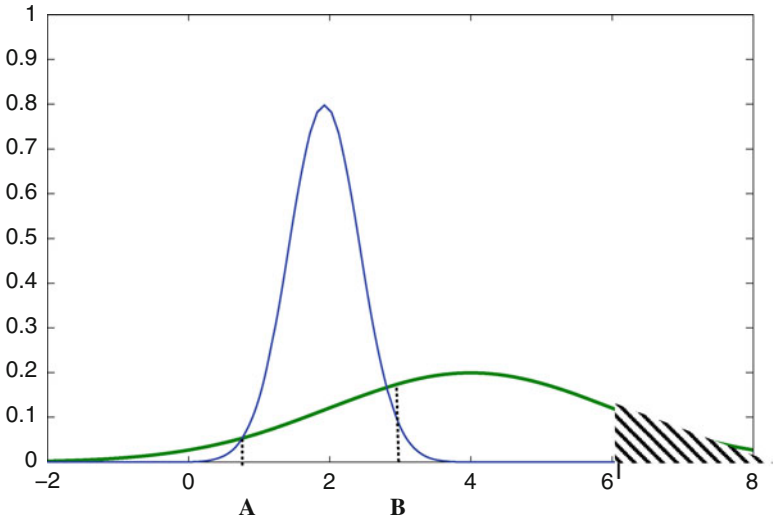

Fig. 6.18 Two Gaussian clusters represented by their density functions drawn with a thin and *bold lines*, respectively. The probability of interval (6,8) in the *bold line* cluster is shown by the area with diagonal filling. The interval (A,B) is the only place in which the thin *line* cluster is more likely than the *bold line* cluster

To estimate the individual cluster parameters, the principle of maximum likelihood, one of the main approaches in mathematical statistics, applies. The approach is based on the postulate that the events that have really occurred are those that are most likely. In general, this is not correct – everybody can recall a situation in which a less likely event has occurred. But the principle, applied for parameter estimation, is as much effective as a similarly wrong principle of the maximum parsimony, and even more. In its simplest version, the approach requires to find the mixture probabilities p_k and cluster parameters a_k, $k = 1, 2, \ldots, K$ by maximizing the likelihood of the observed data under the assumption that the observations come independently from a mixture of distributions. It is not difficult to show, under the assumption that the observations come independenly of each other, that the likelihood is the product of the density values, $P = \prod_{i=1}^{N} \sum_{k=1}^{K} p_k f(y_i, a_k)$. To computationally handle the maximization problem for P with respect to the unknown parameter values, its logarithm, $L = log(P)$, is maximized in the form of the following expression:

$$L = \sum_{i=1}^{N} \sum_{k=1}^{K} g_{ik}[\log(p_k) + \log(f(y_i, \alpha_k) - \log(g_{ik})], \tag{6.24}$$

where g_{ik} is the posterior density of cluster k defined as $g_{ik} = p_k f(y_i, \alpha_k) / \Sigma_k p_k f(y_i, \alpha_k)$.

Criterion L can be considered a function of two groups of variables:

(1) the mixture probabilities p_k and cluster parameters α_k, and
(2) posterior densities g_{ik},

to apply the method of alternating optimization. The alternating maximization algorithm for this criterion is referred to as EM-algorithm since computations are performed as a sequence of Expectation (E) and Maximization (M) steps. As usual, to start the process, the variables must be initialized. Then E-step is executed: Given p_k and α_k, optimal g_{ik} are found. Given g_{ik}, M-step finds the optimal p_k and α_k. This brings the process to an E-step again to follow by an M-step. And so forth. The computation stops when the current parameter values approximately coincide with the previous ones. This algorithm has been developed, in various versions, for Gaussian density functions as well as for some other parametric families of probability distributions. It should be noted that developing a fitting algorithm is not that simple, and not only because there are too many parameters here to estimate. One should take into consideration that there is a tradeoff between the complexity of the probabilistic model and the number of clusters: a more complex model may fit to a smaller number of clusters. To select a better model one can utilize the likelihood criterion penalized for the complexity of the model. A popular penalized log-likelihood criterion is referred to as Bayesian Information Criterion (BIC) and is defined, in this case, as

$$\text{BIC} = 2\ log\ p(X/p_k,\ \alpha_k) - \lambda log(N), \tag{6.25}$$

where X is the observed data matrix, λ the number of parameters to be fitted, and N the number of observations, that is, rows in X. The greater the value, the better. BIC analysis has been shown to be useful, for example, in assessing the number of clusters K for the mixture of Gaussians model.

The goal of EM algorithm is determining the density functions rather than assigning entities to clusters. If the user needs to see the "actual clusters", the posterior probabilities g_{ik} can be utilized: i is assigned to that k for which g_{ik} is the maximum. Since this "optimal assignment" rule deviates from the distribution of g_{ik}, the proportions of entities in clusters obtained in this way will deviate from the mixture probabilities p_k. This is why it is advisable to consider the relative values of g_{ik} as fuzzy membership values.

The situation, in which all Gaussian clusters have their covariance matrices constant diagonal and equal to each other, so that $\Sigma_k = \sigma^2 E$, where E is identity matrix and σ^2 the variance, is of a theoretical interest. In this case, all clusters have uniformly spherical distributions of the same radius. The maximum likelihood criterion P in this case is equivalent to the criterion of K-Means and, moreover, there is a certain homology between the EM and Batch K-Means algorithms in this case.

To see what is going on here, consider feature vectors corresponding to entities $x_i,\ i \in I$, as randomly and independently sampled from the population, with an

unknown assignment of the entities to clusters S_k. The likelihood of this sample is determined by the following equation:

$$P = C \prod_{k=1}^{K} \prod_{i \in S_k} \sigma^{-V} \exp\{-(x_i - m_k)^T \sigma^{-2} (x_i - m_k)/2\},$$

because in this case the determinant in (6.23) is equal to $|\Sigma_k| = \sigma^{2V}$. and the inverse covariance matrix is σ^{-2}E. The logarithm of the likelihood is proportional to

$$L = -2V \log(\sigma) - \sum_{k=1}^{K} \sum_{i \in S_k} (x_i - m_k)^T (x_i - m_k)/\sigma^2$$

It is not difficult to see from the first-order optimality conditions for L that, given partition $S = \{S_1, S_2, \ldots, S_K\}$, the optimal values of m_k and σ are determined according to the usual formulas for the mean and the standard deviation. Moreover, given m_k and σ, the partition $S = \{S_1, S_2, \ldots, S_K\}$ maximizing L will simultaneously minimize the double sum in the right part of its expression above, which is exactly the summary squared Euclidean distance from all entities to their centroids, that is, criterion *W(S,m)* for K-Means in (6.3) except for notation: the cluster gravity centers are denoted here by m_k rather than by c_k, which is not a big deal after all.

Thus the mixture model leads to the conventional K-Means method as a method for fitting the model, under the condition that all clusters have spherical Gaussian distribution of the same variance. This leads some authors to conclude that K-Means is applicable only under the assumption of such a model. However, this conclusion is wrong because it involves a logic trap: it is well known that the fact that A implies B does not necessarily mean that B implies A – there are plenty of examples to the opposite. Note however that the K-Means data recovery model, also leading to K-Means, assumes no restricting hypotheses on the mechanism of data generation. It also implies, through the data scatter decomposition, that useful data standardization options should involve dividing by range or similar range-related indexes rather than by the standard deviation, associated with the spherical Gaussian model. In general, the situation here is similar to that of the linear regression, which is a good method to apply when there is a Gaussian distribution of all variables involved, but it can and should be applied under any other distribution of observations if they tend to lie around a straight line.

6.4.3 Kohonen's Self-Organizing Maps SOM

Kohonen's Self-Organizing Map is an approach to visualize the data cluster structure by explicitly mapping it onto a plane grid (Kohonen 1995). Typically, the grid is rectangular and its size is determined by the user-specified numbers of its rows and columns, r and c, respectively, so that there are $r \times c$ nodes on the grid. Each of the grid nodes, $g_k (k = 1, 2, \ldots, rc)$, is one-to-one associated with the so-called

model, or reference, vector m_k which is of the same dimension as the entity points y_i, $i \in I$.

The grid has a neighborhood structure which is to be set by the user. In a typical case, the neighborhood G_k of node g_k is defined as the set of all the grid nodes whose path distance from g_k is less than a pre-selected threshold value (see Fig. 6.19).

Then each m_k is associated with some data points – a process that can be reiterated. In the end, data points associated at each m_k are visualized at the grid point ($k = 1, \ldots, rc$) (see Fig. 6.20). Historically, all SOM algorithms have been set in an incremental manner as neuron networks do, but later, after some theoretical investigation, straight/ batch versions appeared, such as the following.

Initially, vectors m_k are initialized in the data space either randomly or according to an assumption of the data structure such as, for instance, centroids of K-Means clusters found at $K = rc$. Given vectors m_k, entity points y_i are partitioned into "neighborhood" sets I_k. For each $k = 1, 2, \ldots, rc$, the neighborhood set I_k is defined as consisting of those y_i that are assigned to m_k according to the Minimum distance rule. Given sets I_k, model vectors m_k are updated as centers of gravity of all entities y_i assigned to grid nodes in the neighborhood of g_k, that is, such y_i that $i \in I_t$ for some $g_t \in G_k$. Then a new iteration of building I_k with the follow-up updating

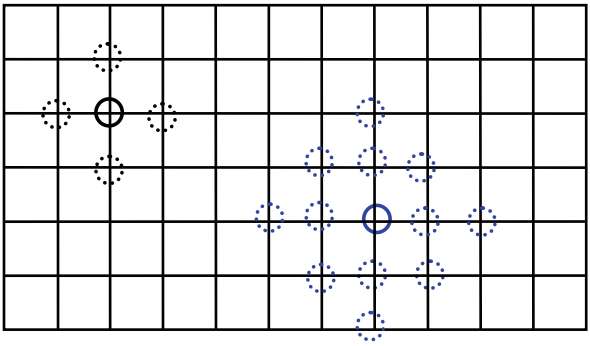

Fig. 6.19 A 7×12 SOM grid on which nodes *g1* and *g2* are shown along with their neighborhoods defined by thresholds 1 and 2, respectively

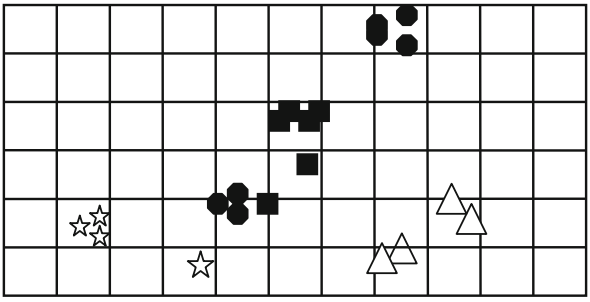

Fig. 6.20 A pattern of final SOM structure using entity labels of geometrical shapes

m_k's, is run. The computation stops when new m_k are close enough to the previous ones or after a pre-specified number of iterations.

As one can see, SOM in this version is much similar to Straight/Batch K-Means except for the following:

(a) number $K = rc$ of model vectors is large and has nothing to do with the number of final clusters – this comes visually as the number of grid clusters;
(b) data points are averaged over the grid neighbourhood, not the feature space neighborhood;
(c) there are no interpretation rules except according to positioning of points on the grid.

Item (a) results in the fact that many of final I_k's are empty, so that relatively very few of grid nodes are populated, which may create a powerful image of a cluster structure that may go to a deeper – or more interesting – minimum than K-Means, because of (b).

6.5 Summary

This Chapter is devoted to K-Means, arguably the most popular clustering method. The method partitions the entity set into clusters along with centroids representing them. It is very intuitive and usually does not require that much space to get presented, except of course its various versions such as incremental or nature inspired or medoid based algorithms. This text also includes less popular subjects that are important when using K-Means for real-world data analysis:

- Presentation and analysis of examples of its failures
- Innate tools for interpretation of clusters
- Reformulations of the criterion that could yield different algorithms for K-Means
- Initialization – the choice of K and location of centroids

Three modifications of K-Means onto different cluster structures are presented as well. These are: Fuzzy K-Means for finding fuzzy clusters, Expectation-Maximization (EM) for finding probabilistic clusters as items of a mixture of distributions, and Kohonen self-organizing maps (SOM) that tie up the sought clusters to a visually comfortable two-dimensional grid.

References

Bandyopadhyay, S., Maulik, U.: An evolutionary technique based on K-means algorithm for optimal clustering in R^N. Inf. Sci. **146**, 221–237 (2002).

Bezdek, J., Keller, J., Krisnapuram, R., Pal, M.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing. Kluwer Academic Publishers, Dordrecht (1999).

Cangelosi, R., Goriely, A.: Component retention in principal component analysis with application to cDNA microarray data. Biol. Direct. **2**, 2 (2007). http://www.biolgy-direct.com/con-tent/2/1/2.

Green, S.B., Salkind, N.J.: Using SPSS for the Windows and Mackintosh: Analyzing and Understanding Data. Prentice Hall, Upper Saddle River, NJ (2003).

Hartigan, J.A.: Clustering Algorithms. Wiley, New York (1975).

Kaufman. L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (1990).

Kendall, M.G., Stewart, A.: Advanced Statistics: Inference and Relationship (3d edition). Griffin, London (1973). ISBN: 0852642156.

Kettenring, J.: The practice of cluster analysis. J. Classific. **23**, 3–30 (2006).

Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1995).

Kryshtanowski, A.: Analysis of Sociology Data with SPSS. Higher School of Economics Publishers, Moscow (in Russian) (2008).

Lu, Y., Lu, S., Fotouhi, F., Deng, Y., Brown, S.: Incremental genetic algorithm and its application in gene expression data analysis. BMC Bioinform. **5**,172 (2004).

Ming-Tso Chiang, M., Mirkin, B.: Intelligent choice of the number of clusters in K-Means clustering: an experimental study with different cluster spreads. J. Classif. **27**(1), 3–40 (2010).

Mirkin, B.: Clustering for Data Mining: A Data Recovery Approach. Chapman & Hall/CRC, Roca Baton, FL (2005). ISBN 1-58488-534-3.

Mirkin, B.: Mathematical Classification and Clustering. Kluwer Academic Press, Boston-Dordrecht (1996).

Murthy, C.A., Chowdhury, N.: In search of optimal clusters using genetic algorithms. Pattern Recognit. Lett. **17**, 825–832 (1996).

Nascimento, S., Franco, P.: Unsupervised fuzzy clustering for the segmentation and annotation of upwelling regions in sea surface temperature images. In: Gama, J. (ed.) Discovery Science, LNCS 5808, pp. 212–226. Springer (2009).

Nascimento, S.: Fuzzy Clustering via Proportional Membership Model. ISO Press, Amsterdam (2005).

Paterlini, S., Krink, T.: Differential evolution and PSO in partitional clustering. Comput. Stat. Data Anal. **50**, 1220–1247 (2006).

Stanforth, R., Mirkin, B., Kolossov, E.: A measure of domain of applicability for QSAR modelling based on intelligent K-Means clustering. QSAR Comb. Sci. **26**(7), 837–844 (2007).