CHAPTER 19

HIERARCHICAL AND k-MEANS CLUSTERING

19.1 THE CLUSTERING TASK

Clustering refers to the grouping of records, observations, or cases into classes of similar objects. A *cluster* is a collection of records that are similar to one another and dissimilar to records in other clusters. Clustering differs from classification in that there is no target variable for clustering. The clustering task does not try to classify, estimate, or predict the value of a target variable. Instead, clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or clusters, where the similarity of the records within the cluster is maximized, and the similarity to records outside this cluster is minimized.

For example, the Nielsen PRIZM segments, developed by Claritas Inc., represent demographic profiles of each geographic area in the United States, in terms of distinct lifestyle types, as defined by zip code. For example, the clusters identified for zip code 90210, Beverly Hills, California, are as follows:

- Cluster # 01: Upper Crust Estates
- Cluster # 03: Movers and Shakers
- Cluster # 04: Young Digerati
- Cluster # 07: Money and Brains
- Cluster # 16: Bohemian Mix.

The description for Cluster # 01: Upper Crust is "The nation's most exclusive address, Upper Crust is the wealthiest lifestyle in America, a haven for empty-nesting couples between the ages of 45 and 64. No segment has a higher concentration of residents earning over $100,000 a year and possessing a postgraduate degree. And none has a more opulent standard of living."

Examples of clustering tasks in business and research include the following:

- Target marketing of a niche product for a small-capitalization business that does not have a large marketing budget.

Data Mining and Predictive Analytics, First Edition. Daniel T. Larose and Chantal D. Larose.
© 2015 John Wiley & Sons, Inc. Published 2015 by John Wiley & Sons, Inc.

- For accounting auditing purposes, to segment financial behavior into benign and suspicious categories.
- As a dimension-reduction tool when a data set has hundreds of attributes.
- For gene expression clustering, where very large quantities of genes may exhibit similar behavior.

Clustering is often performed as a preliminary step in a data mining process, with the resulting clusters being used as further inputs into a different technique downstream, such as neural networks. Owing to the enormous size of many present-day databases, it is often helpful to apply clustering analysis first, to reduce the search space for the downstream algorithms. In this chapter, after a brief look at hierarchical clustering methods, we discuss in detail k-means clustering; in Chapter 20, we examine clustering using Kohonen networks, a structure related to neural networks.

Cluster analysis encounters many of the same issues that we dealt with in the chapters on classification. For example, we shall need to determine

- how to measure similarity;
- how to recode categorical variables;
- how to standardize or normalize numerical variables;
- how many clusters we expect to uncover.

For simplicity, in this book, we concentrate on Euclidean distance between records:

$$d_{\text{Euclidean}}(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

where $x = x_1, x_2, \ldots, x_m$, and $y = y_1, y_2, \ldots, y_m$ represent the m attribute values of two records. Of course, many other metrics exist, such as *city-block distance:*

$$d_{\text{city-block}}(x, y) = \sum_i |x_i - y_i|$$

or *Minkowski distance*, which represents the general case of the foregoing two metrics for a general exponent q:

$$d_{\text{Minkowski}}(x, y) = \left(\sum_i |x_i - y_i|^q \right)^{1/q}$$

For categorical variables, we may again define the "different from" function for comparing the ith attribute values of a pair of records:

$$\text{different}(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

where x_i and y_i are categorical values. We may then substitute different (x_i, y_i) for the ith term in the Euclidean distance metric above.

For optimal performance, clustering algorithms, just like algorithms for classification, require the data to be normalized so that no particular variable or subset of

variables dominates the analysis. Analysts may use either the *min–max normalization* or *Z-score standardization*, discussed in earlier chapters:

$$\text{Min} - \text{max normalization: } X^* = \frac{X - \min(X)}{\text{Range}(X)}$$

$$Z\text{-score standardization: } X^* = \frac{X - \text{mean}(X)}{\text{SD}(X)}$$

All clustering methods have as their goal the identification of groups of records such that similarity within a group is very high while the similarity to records in other groups is very low. In other words, as shown in Figure 19.1, clustering algorithms seek to construct clusters of records such that the *between-cluster variation* is large compared to the *within-cluster variation*. This is somewhat analogous to the concept behind analysis of variance.

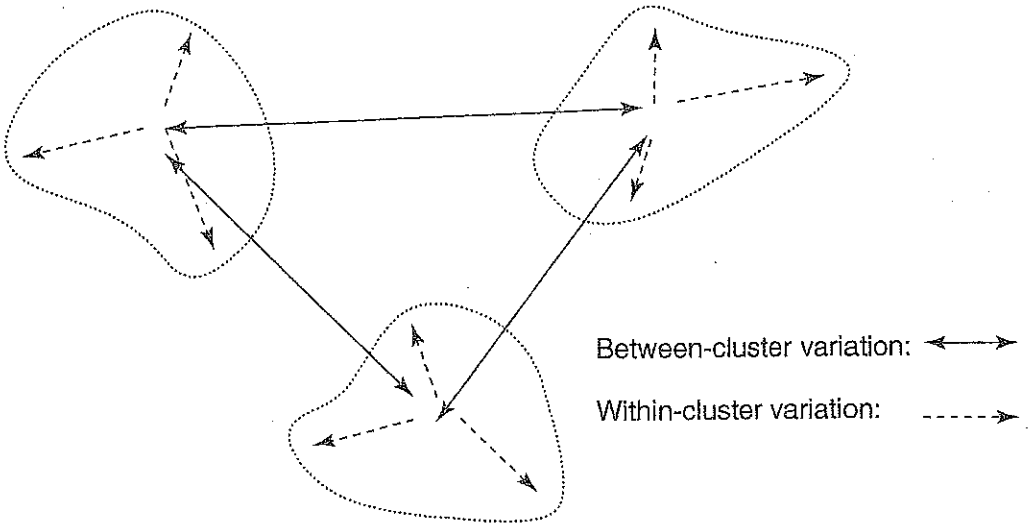

Figure 19.1 Clusters should have small within-cluster variation compared to the between–cluster variation.

19.2 HIERARCHICAL CLUSTERING METHODS

Clustering algorithms are either hierarchical or nonhierarchical. In *hierarchical clustering*, a treelike cluster structure (*dendrogram*) is created through recursive partitioning (divisive methods) or combining (agglomerative) of existing clusters. *Agglomerative clustering methods* initialize each observation to be a tiny cluster of its own. Then, in succeeding steps, the two closest clusters are aggregated into a new combined cluster. In this way, the number of clusters in the data set is reduced by one at each step. Eventually, all records are combined into a single huge cluster. *Divisive clustering methods* begin with all the records in one big cluster, with the most dissimilar records being split off recursively, into a separate cluster, until each record represents its own cluster. Because most computer programs that apply hierarchical clustering use agglomerative methods, we focus on those.

Distance between records is rather straightforward once appropriate recoding and normalization has taken place. But how do we determine *distance between clusters* of records? Should we consider two clusters to be close if their nearest neighbors are close or if their farthest neighbors are close? How about criteria that average out these extremes?

We examine several criteria for determining distance between arbitrary clusters A and B:

- *Single linkage*, sometimes termed the *nearest-neighbor approach*, is based on the minimum distance between any record in cluster A and any record in cluster B. In other words, cluster similarity is based on the similarity of the most similar members from each cluster. Single linkage tends to form long, slender clusters, which may sometimes lead to heterogeneous records being clustered together.
- *Complete linkage*, sometimes termed the *farthest-neighbor approach*, is based on the maximum distance between any record in cluster A and any record in cluster B. In other words, cluster similarity is based on the similarity of the most dissimilar members from each cluster. Complete linkage tends to form more compact, spherelike clusters.
- *Average linkage* is designed to reduce the dependence of the cluster-linkage criterion on extreme values, such as the most similar or dissimilar records. In average linkage, the criterion is the average distance of all the records in cluster A from all the records in cluster B. The resulting clusters tend to have approximately equal within-cluster variability.

Let us examine how these linkage methods work, using the following small, one-dimensional data set:

2	5	9	15	16	18	25	33	33	45

19.3 SINGLE-LINKAGE CLUSTERING

Suppose that we are interested in using *single-linkage* agglomerative clustering on this data set. Agglomerative methods start by assigning each record to its own cluster. Then, single linkage seeks the minimum distance between any records in two clusters. Figure 19.2 illustrates how this is accomplished for this data set. The minimum cluster distance is clearly between the single-record clusters where each contains the value 33, for which the distance must be 0 for any valid metric. Thus, these two clusters are combined into a new cluster of two records, both of value 33, as shown in Figure 19.2. Note that, after step 1, only nine ($n - 1$) clusters remain. Next, in step 2, the clusters containing values 15 and 16 are combined into a new cluster, because their distance of 1 is the minimum between any two clusters remaining.

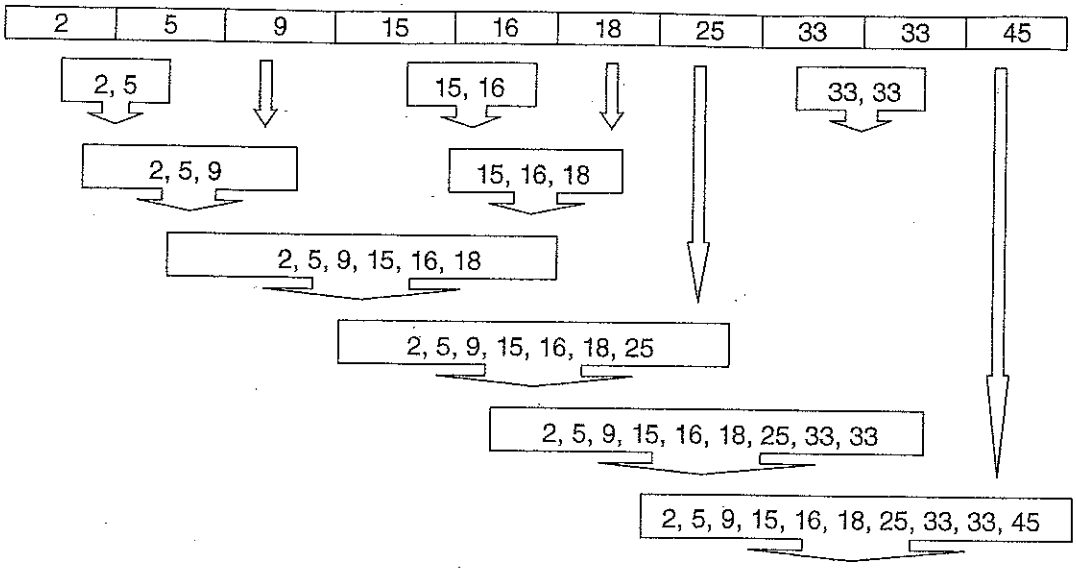

Figure 19.2 Single-linkage agglomerative clustering on the sample data set.

Here are the remaining steps:

- *Step 3:* The cluster containing values 15 and 16 (cluster {15,16}) is combined with cluster {18}, because the distance between 16 and 18 (the closest records in each cluster) is 2, the minimum among remaining clusters.
- *Step 4:* Clusters {2} and {5} are combined.
- *Step 5:* Cluster {2,5} is combined with cluster {9}, because the distance between 5 and 9 (the closest records in each cluster) is 4, the minimum among remaining clusters.
- *Step 6:* Cluster {2,5,9} is combined with cluster {15,16,18}, because the distance between 9 and 15 is 6, the minimum among remaining clusters.
- *Step 7:* Cluster {2,5,9,15,16,18} is combined with cluster {25}, because the distance between 18 and 25 is 7, the minimum among remaining clusters.
- *Step 8:* Cluster {2,5,9,15,16,18,25} is combined with cluster {33,33}, because the distance between 25 and 33 is 8, the minimum among remaining clusters.
- *Step 9:* Cluster {2,5,9,15,16,18,25,33,33} is combined with cluster {45}. This last cluster now contains all the records in the data set.

19.4 COMPLETE-LINKAGE CLUSTERING

Next, let us examine whether using the complete-linkage criterion would result in a different clustering of this sample data set. Complete linkage seeks to minimize the distance among the records in two clusters that are farthest from each other. Figure 19.3 illustrates complete-linkage clustering for this data set.

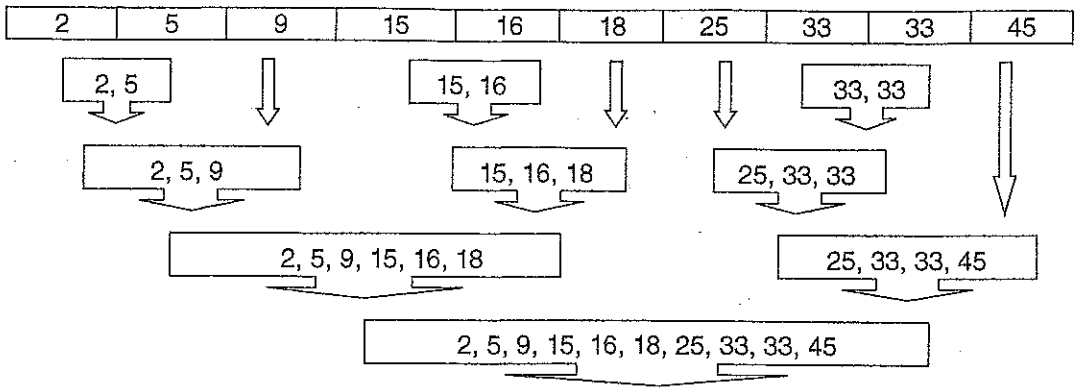

Figure 19.3 Complete-linkage agglomerative clustering on the sample data set.

- *Step 1:* As each cluster contains a single record only, there is no difference between single linkage and complete linkage at step 1. The two clusters each containing 33 are again combined.
- *Step 2:* Just as for single linkage, the clusters containing values 15 and 16 are combined into a new cluster. Again, this is because there is no difference in the two criteria for single-record clusters.
- *Step 3:* At this point, complete linkage begins to diverge from its predecessor. In single linkage, cluster {15,16} was at this point combined with cluster {18}. But complete linkage looks at the farthest neighbors, not the nearest neighbors. The farthest neighbors for these two clusters are 15 and 18, for a distance of 3. This is the same distance separating clusters {2} and {5}. The complete-linkage criterion is silent regarding ties, so we arbitrarily select the first such combination found, therefore combining the clusters {2} and {5} into a new cluster.
- *Step 4:* Now cluster {15,16} is combined with cluster {18}.
- *Step 5:* Cluster {2,5} is combined with cluster {9}, because the complete-linkage distance is 7, the smallest among remaining clusters.
- *Step 6:* Cluster {25} is combined with cluster {33,33}, with a complete-linkage distance of 8.
- *Step 7:* Cluster {2,5,9} is combined with cluster {15,16,18}, with a complete-linkage distance of 16.
- *Step 8:* Cluster {25,33,33} is combined with cluster {45}, with a complete-linkage distance of 20.
- *Step 9:* Cluster {2,5,9,15,16,18} is combined with cluster {25,33,33,45}. All records are now contained in this last large cluster.

Finally, with average linkage, the criterion is the average distance of all the records in cluster A from all the records in cluster B. As the average of a single record is the record's value itself, this method does not differ from the earlier methods in the early stages, where single-record clusters are being combined. At step 3, average linkage would be faced with the choice of combining clusters {2} and {5}, or combining the {15,16} cluster with the single-record {18} cluster. The average

distance between the {15,16} cluster and the {18} cluster is the average of |18 − 15| and |18 − 16|, which is 2.5, while the average distance between clusters {2} and {5} is of course 3. Therefore, average linkage would combine the {15,16} cluster with cluster {18} at this step, followed by combining cluster {2} with cluster {5}. The reader may verify that the average-linkage criterion leads to the same hierarchical structure for this example as the complete-linkage criterion. In general, average linkage leads to clusters more similar in shape to complete linkage than does single linkage.

19.5 k-MEANS CLUSTERING

The k-means clustering algorithm[1] is a straightforward and effective algorithm for finding clusters in data. The algorithm proceeds as follows:

- *Step 1:* Ask the user how many clusters k the data set should be partitioned into.
- *Step 2:* Randomly assign k records to be the initial cluster center locations.
- *Step 3:* For each record, find the nearest cluster center. Thus, in a sense, each cluster center "owns" a subset of the records, thereby representing a partition of the data set. We therefore have k clusters, $C_1, C_2, \ldots, C_k$.
- *Step 4:* For each of the k clusters, find the cluster *centroid*, and update the location of each cluster center to the new value of the centroid.
- *Step 5:* Repeat steps 3–5 until convergence or termination.

The "nearest" criterion in step 3 is usually Euclidean distance, although other criteria may be applied as well. The cluster centroid in step 4 is found as follows. Suppose that we have n data points $(a_1, b_1, c_1), (a_2, b_2, c_2), \ldots, (a_n, b_n, c_n)$, the *centroid* of these points is the center of gravity of these points and is located at point $\left(\sum a_i/n, \sum b_i/n, \sum c_i/n,\right)$. For example, the points (1,1,1), (1,2,1), (1,3,1), and (2,1,1) would have centroid

$$\left(\frac{1+1+1+2}{4}, \frac{1+2+3+1}{4}, \frac{1+1+1+1}{4}\right) = (1.25, 1.75, 1.00)$$

The algorithm terminates when the centroids no longer change. In other words, the algorithm terminates when for all clusters $C_1, C_2, \ldots, C_k$, all the records "owned" by each cluster center remain in that cluster. Alternatively, the algorithm may terminate when some convergence criterion is met, such as no significant shrinkage in the *mean squared error* (MSE):

$$\text{MSE} = \frac{\text{SSE}}{N-k} = \frac{\sum_{i=1}^{k} \sum_{p \in C_i} d(p, m_i)^2}{N-k}$$

where SSE represents the *sum of squares error*, $p \in C_i$ represents each data point in cluster i, m_i represents the centroid (cluster center) of cluster i, N is the total sample

[1] J. MacQueen, Some methods for classification and analysis of multivariate observations, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, pp. 281–297, University of California Press, Berkeley, CA, 1967.

size, and k is the number of clusters. Recall that clustering algorithms seek to construct clusters of records such that the between-cluster variation is large compared to the within-cluster variation. Because this concept is analogous to the analysis of variance, we may define a *pseudo-F statistic* as follows:

$$F_{k-1,N-k} = \frac{\text{MSB}}{\text{MSE}} = \frac{\text{SSB}/k-1}{\text{SSE}/N-k}$$

where SSE is defined as above, MSB is the *mean square between*, and SSB is the *sum of squares between* clusters, defined as

$$\text{SSB} = \sum_{i=1}^{k} n_i \cdot d(m_i, M)^2$$

where n_i is the number of records in cluster i, m_i is the centroid (cluster center) for cluster i, and M is the grand mean of all the data.

MSB represents the between-cluster variation and MSE represents the within-cluster variation. Thus, a "good" cluster would have a large value of the pseudo-F statistic, representing a situation where the between-cluster variation is large compared to the within-cluster variation. Hence, as the k-means algorithm proceeds, and the quality of the clusters increases, we would expect MSB to increase, MSE to decrease, and F to increase.

19.6 EXAMPLE OF k-MEANS CLUSTERING AT WORK

Let us examine an example of how the k-means algorithm works. Suppose that we have the eight data points in two-dimensional space shown in Table 19.1 and plotted in Figure 19.4 and are interested in uncovering $k=2$ clusters.

Let us apply the k-means algorithm step by step.

- *Step 1*: Ask the user how many clusters k the data set should be partitioned into. We have already indicated that we are interested in $k=2$ clusters.
- *Step 2*: Randomly assign k records to be the initial cluster center locations. For this example, we assign the cluster centers to be $m_1=(1,1)$ and $m_2=(2,1)$.
- *Step 3 (first pass)*: For each record, find the nearest cluster center. Table 19.2 contains the (rounded) Euclidean distances between each point and each cluster center $m_1=(1,1)$ and $m_2=(2,1)$, along with an indication of which cluster center the point is nearest to. Therefore, cluster 1 contains points $\{a,e,g\}$, and cluster 2 contains points $\{b,c,d,f,h\}$.
- *Step 4 (first pass)*: For each of the k clusters find the cluster *centroid* and update the location of each cluster center to the new value of the centroid. The centroid

TABLE 19.1 Data points for k-means example

a	b	c	d	e	f	g	h
(1,3)	(3,3)	(4,3)	(5,3)	(1,2)	(4,2)	(1,1)	(2,1)