2.1 Feature Scaling

Before covering clustering algorithms, it is appropriate to discuss what is known as *feature scaling*. This is also referred to as the normalization or *standardization* of data. It is a necessary first step to for many machine learning algorithms, including the *k*-means algorithm. The purpose of feature scaling is to ensure that the features are given equal importance in an algorithm. Suppose for example that we are clustering men according to two features: height in inches and weight in pounds. Heights might range from 60 to 80 inches while weights range from 100 to 350 pounds. Without feature scaling, the two features will not be treated with equal importance because the range of heights is much less than the range of weights (20 inches vs 250 pounds).

One approach to feature scaling is to calculate the mean and standard deviation of the observations on each feature and scale observations on the feature by subtracting the mean and dividing by the standard deviation. If V is a feature value for a particular observation,

$$\text{Scaled Feature Value} = \frac{V - \mu}{\sigma}$$

where μ and σ are the mean and standard deviation of observations for the feature. This method of feature scaling is referred to as *Z-score scaling* or *Z-score normalization*. The scaled feature values have means equal to zero and standard deviations equal to one.

An alternative approach to feature scaling is to subtract the minimum feature value and divide by the difference between the maximum and minimum values so that:

$$\text{Scaled Feature Value} = \frac{V - \min}{\max - \min}$$

where max and min denote the maximum and minimum feature values. This is referred to as *min-max scaling*. The scaled feature values lie between zero and one.

Z-score scaling is usually preferred because it is less sensitive to extreme values, but it can make sense to use min-max scaling when features have been measured on bounded scales. In our description of the *k*-means algorithm in the rest of this chapter, we assume that feature values have been scaled using one of the two methods we have described.

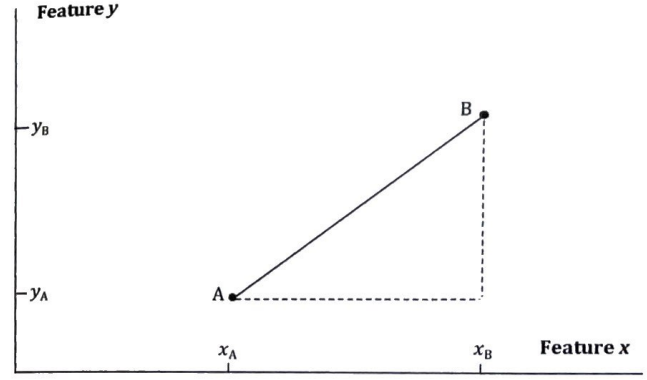
The usual approach is to use the training data set to define the scaling parameters (i.e., the means and standard deviations of features or their minimums and maximums). The scaling defined by the training set is then applied to the validation set and the test set as well to new data.

2.2 The *k*-Means Algorithm

To cluster observations, we need a distance measure. Suppose first that there are only two features, x and y, so that we can plot the observations on a two-dimensional chart. Consider the two observations, A and B, in Figure 2.1. A natural distance measure is the Euclidean distance. This is the length of the line AB. Suppose that for observation A, $x = x_A$ and $y = y_A$, while for observation B, $x = x_B$ and $y = y_B$. The Euclidean distance between A and B (using Pythagoras' theorem) is

$$\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Figure 2.1 The Euclidean distance between observations A and B, with co-ordinates (x_A, y_A) and (x_B, y_B), is the length of the line AB.



This distance measure can be extended to many dimensions. Suppose we have observations on m features and that the value of the jth feature for the ith observation is v_{ij}. The distance between the pth observation and the qth observation is

$$\sqrt{\sum_{j=1}^{m}\left(v_{pj}-v_{qj}\right)^2}$$

The extension from two features to three features is fairly easy to understand. It involves measuring the Euclidean distance in three dimensions rather than two. Imagining distances when $m > 3$ is not so easy, but the formula is a natural extension of that for one, two, and three dimensions.

Another concept we need in order to understand the *k*-means algorithm is the center of a cluster (sometimes referred to as the cluster's *centroid*). Suppose that a certain set of observations is regarded as a cluster. The center is calculated by averaging the values of each of the features for the observations in the cluster. Suppose there are four features and the five observations in Table 2.1 are a cluster. The center of the cluster is a point that has values of 0.914, 0.990, 0.316, and 0.330 for features 1, 2, 3, and 4, respectively. (For example, 0.914 is the average of 1.00, 0.80, 0.82, 1.10, and 0.85.) The distance between each observation and the center of the cluster (shown in the final column of Table 2.1) is calculated in the same way as the distance between A and B in Figure 2.1. For example, the distance of the first observation from the center of the cluster is

$$\sqrt{(1.00-0.914)^2+(1.00-0.990)^2+(0.40-0.316)^2+(0.25-0.330)^2}$$

which equals 0.145.

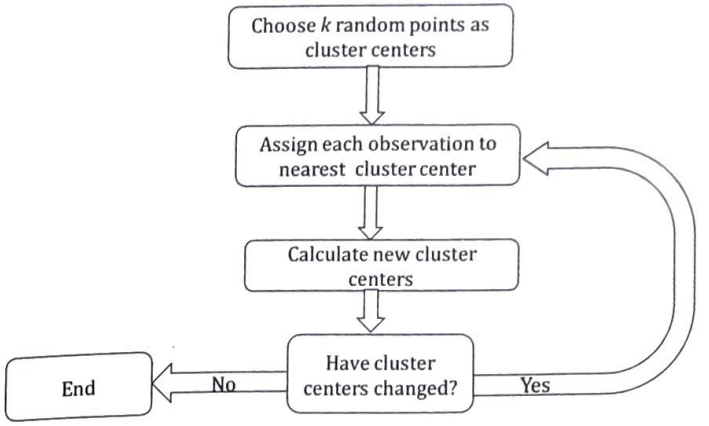
Table 2.1 Calculation of the center of a cluster of five observations on four features.

Observation	*Feature 1*	*Feature 2*	*Feature 3*	*Feature 4*	*Distance to center*
1	1.00	1.00	0.40	0.25	0.145
2	0.80	1.20	0.25	0.40	0.258
3	0.82	1.05	0.35	0.50	0.206
4	1.10	0.80	0.21	0.23	0.303
5	0.85	0.90	0.37	0.27	0.137
Center	0.914	0.990	0.316	0.330	

Figure 2.2 illustrates how the *k*-means algorithm works. The first step is to choose *k*, the number of clusters (more on this later). We then

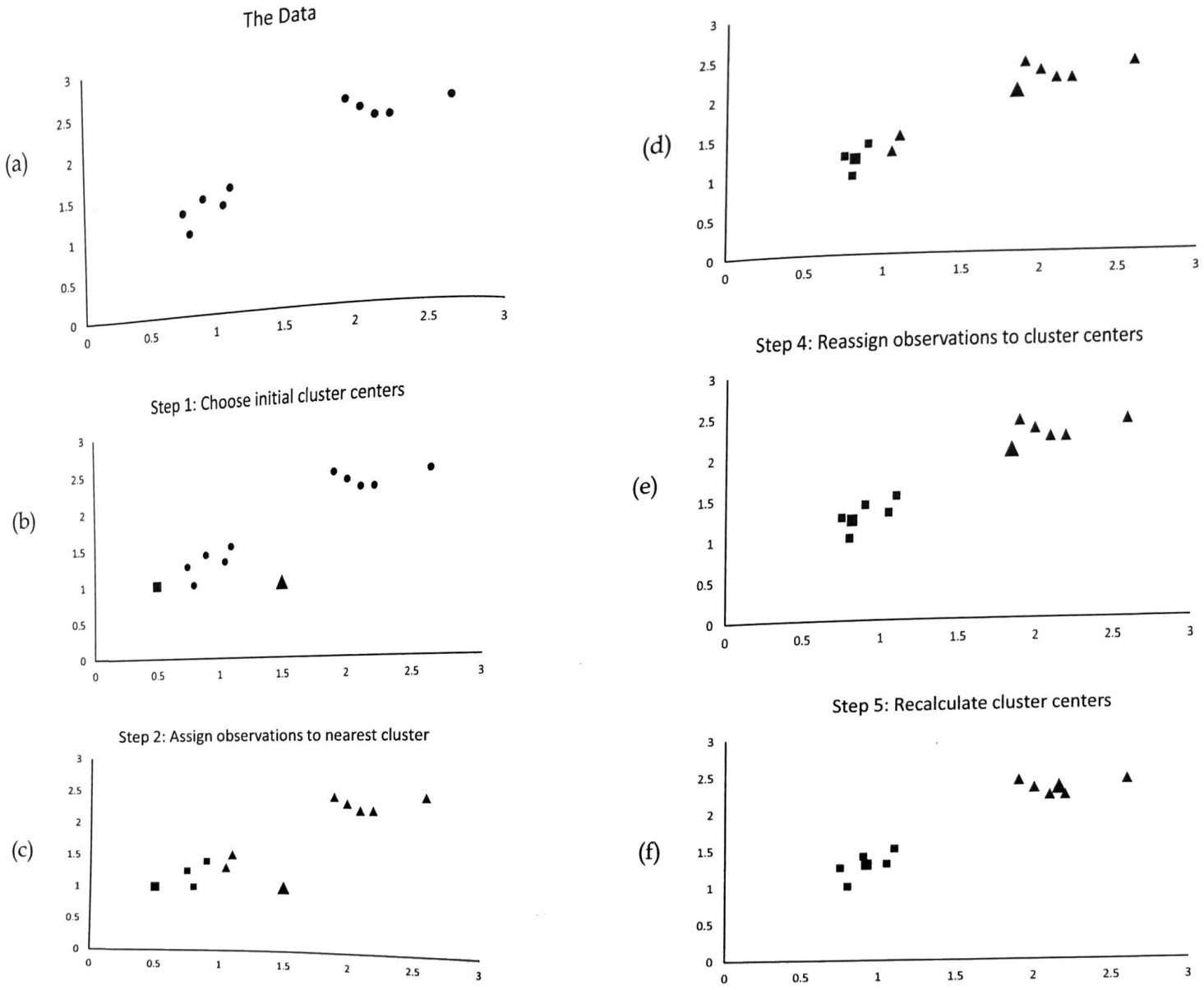
randomly choose *k* points for the centers of the clusters. The distance of each observation from each cluster center is calculated as indicated above and observations are assigned to the nearest cluster center. This produces a first division of the observations into *k* clusters. We then compute new centers for each of the clusters, as indicated in Figure 2.2. The distances of each observation from the new cluster centers is then computed and the observations are re-assigned to the nearest cluster center. We then compute new centers for each of the clusters and continue in this fashion until the clusters do not change.

Figure 2.2 The *k*-means algorithm



A baby example illustrating how the *k*-means algorithm works is in Figure 2.3. We suppose $k = 2$ and there are two features. The best fit two clusters is clear from the data in Figure 2.3a. Let's see how *k*-means would find them. We first choose (randomly) two initial cluster centers. We suppose that these are represented by the square and triangle in Step 1. In Step 2, the observations are allocated to the nearest cluster centers. (Those allocated to the square are shown as squares, while those allocated to the triangle are shown as triangles). In Step 3, the cluster centers are recalculated. In Step 4, the observations are reassigned to cluster centers. There are now five observations in each cluster. Finally, in Step 5 the cluster centers are recalculated. No new allocation of observations to cluster centers takes place after this and so the algorithm stops.

Figure 2.3 A simple example of the k-means algorithm



A measure of the performance of a clustering algorithm is the within-cluster sum of squares, also known as the *inertia*. Define d_i as the distance of the ith observation from the center of the cluster to which it belongs. Then:

$$\text{Inertia} = \sum_{i=1}^{n} d_i^2$$

where n is the number of observations.

For any given value of k, an objective of the k-means algorithm is usually to minimize the inertia. The results from one run of the algorithm may depend on the initial cluster centers that are chosen. It is therefore usual to re-run the algorithm several times with different initial cluster centers. The best result across all runs is the one for which the inertia is least.

Generally, the inertia decreases as k increases. In the limit, when k equals the number of observations, there is one cluster for each observation and the inertia is zero.

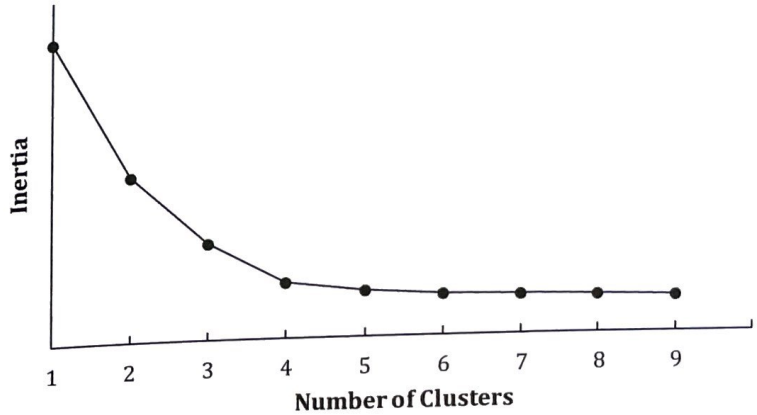
2.3 Choosing k

In some cases, the choice of k may depend on the objective of the clustering. For example, a company that is planning to produce small, medium, large, and extra-large sweaters for men might collect data on various relevant features (arm length, shoulder width, chest measurement, etc.) for a random sample of men and then create four clusters to help with product design. In other situations, the user of the algorithm may not have any preconceived ideas about k and just want to find a natural grouping of observations.

The *elbow method* is one approach for determining the number of clusters. The k-means algorithm is carried out for a range of values of k (e.g., all values between 1 and 9). The best inertia obtained for each value of k is then plotted against the number of clusters as indicated in Figure 2.4. The slope of the line in this chart indicates how the within-cluster sum of squares declines as the number of clusters increases. In Figure 2.4, the decline is quite large when we move from one to two, two to three, and three to four clusters. After four clusters, the decline is much smaller. We would therefore conclude that a good choice for the number of clusters is four.

In addition to the within-cluster sum of squares, we are likely to be interested in how distinct the clusters are. If two clusters are very close together, we might reasonably conclude that not much is gained by keeping them separate. Analysts therefore often monitor the distance between cluster centers. If changing the number of clusters from k to $k + 1$ leads to two clusters with centers that are very close to each other, it might be considered best not to make the change.

Figure 2.4 Application of the elbow method. The inertia (within-cluster sum of squares) is plotted against the number of clusters



A less subjective way of choosing the number of clusters is the *silhouette method*. Again, we carry out the k-means algorithm for a range of values of k. For each value of k, we calculate for each observation, i, the average distance between the observation and the other observations in the cluster to which it belongs. Define this as $a(i)$. We also calculate, for each of the other clusters, the average distance between the observation and the observations in that cluster. We define $b(i)$ as the minimum value of these average distances across all the other clusters. We expect $b(i)$ to be greater than $a(i)$ most of the time. The *silhouette* of an observation measures the extent to which $b(i)$ is greater than $a(i)$. It is defined as[1]

[1] See L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley 1990.

$$s(i) = \frac{b(i) - a(i)}{\max[a(i), b(i)]}$$

The value of $s(i)$ lies between −1 and +1. As it becomes closer to +1, the observation more clearly belongs to the group to which it has been assigned. The average of $s(i)$ over all observations in a cluster is a measure of the tightness of the grouping of those observations. The average of $s(i)$ over all observations in all clusters is an overall measure of the appropriateness of the clustering and is referred to as the *average silhouette score*. If for a particular data set the average silhouette scores are 0.40, 0.23, 0.35, 0.22, and 0.15 for k = 2, 3, 4, 5, and 6, respectively, we would conclude that k = 2 and 4 are better choices for the number of clusters than k = 3, 5, and 6.

Yet another approach for choosing k, known as the *gap statistic*, was suggested by Tibshirani et al (2001).[2] In this, the within-cluster sum of squares is compared with the value we would expect under the null hypothesis that the observations are created randomly. We create N sets of random points and, for each value of k that is considered, we cluster each set, calculating the within-cluster sum of squares. (N = 500 usually works well.) Define

- m_k: the mean of the within-cluster sum of squares for randomly created data when there are k clusters
- s_k: the standard deviation of the within-cluster sum of squares for randomly created data when there are k clusters
- w_k: the within-cluster sum of squares for the data we are considering when there are k clusters

We set

$$\text{Gap}(k) = m_k - w_k$$

This is the difference between the within-cluster sum of squares statistic for the random data and the data of interest. It is argued that the best choice for k is the smallest value such that Gap(k) is within s_{k+1} of Gap(k + 1).

[2] See R. Tibshirani, G. Walther, and T. Hastie (2001), "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society*, B, 63, Part 2: 411-423.

2.4 The Curse of Dimensionality

As the number of features increases, the k-means algorithm becomes affected by what is known as the "curse of dimensionality." Distances between observations increase. Consider the Euclidean distance between a point where all features equal 1.0 and a point where all features equal 0.0. When there is one feature the distance is 1.0; when there are two features the distance is $\sqrt{2}$ or 1.4; when there are three features, it is $\sqrt{3}$ or 1.7; when then are 100 features it is 10; and when there are 1,000 features it is 31.6. One consequence of this is that we cannot compare a within-cluster sum of squares given by data with a small number of features to one given by data with a large number of features.

Another problem is that, as the number of features increases, the distance measure that we have defined does not always differentiate well between observations that are close and those that are far apart. As a result, the k-means algorithm works less well. This has led some users of the algorithm to search for alternatives to the Euclidean distance measure.

The Euclidean distance between an observation where feature j is x_j and another observation where feature j is y_j can be written

$$\sqrt{\sum_{j=1}^{m} (x_j - y_j)^2}$$

An alternative, based on the cosine similarity function, is[3]

$$1 - \frac{\sum_{j=1}^{m} x_j y_j}{\sqrt{\sum_{j=1}^{m} x_j^2 \sum_{j=1}^{m} y_j^2}} \qquad (2.1)$$

This always lies between 0 and 2.

[3] The cosine similarity function is the second term in equation (2.1) and lies between −1 and +1. It measures the similarity of the directions in which two vectors point. It is +1 when they point in exactly the same direction and −1 when they point in opposite directions.