7.3.1 K-Means Basics

A k-means algorithm is outlined in Fig. 7.7. There are several ways to select the initial k points that represent the clusters, and we shall discuss them in Section 7.3.2. The heart of the algorithm is the for-loop, in which we consider each point other than the k selected points and assign it to the closest cluster, where "closest" means closest to the centroid of the cluster. Note that the centroid of a cluster can migrate as points are assigned to it. However, since only points near the cluster are likely to be assigned, the centroid tends not to move too much.

```
Initially choose k points that are likely to be in
    different clusters;
Make these points the centroids of their clusters;
FOR each remaining point p DO
    find the centroid to which p is closest;
    Add p to the cluster of that centroid;
    Adjust the centroid of that cluster to account for p;
END;
```

Figure 7.7: Outline of k-means algorithms

An optional step at the end is to fix the centroids of the clusters and to reassign each point, including the k initial points, to the k clusters. Usually, a point p will be assigned to the same cluster in which it was placed on the first pass. However, there are cases where the centroid of p's original cluster moved quite far from p after p was placed there, and p is assigned to a different cluster on the second pass. In fact, even some of the original k points could wind up being reassigned. As these examples are unusual, we shall not dwell on the subject.

7.3.2 Initializing Clusters for K-Means

We want to pick points that have a good chance of lying in different clusters. There are two approaches.

1. Pick points that are as far away from one another as possible.
2. Cluster a sample of the data, perhaps hierarchically, so there are k clusters. Pick a point from each cluster, perhaps that point closest to the centroid of the cluster.

The second approach requires little elaboration. For the first approach, there are variations. One good choice is:

```
Pick the first point at random;
```

```
WHILE there are fewer than k points DO
    Add the point whose minimum distance from the selected
      points is as large as possible;
END;
```

Example 7.8: Let us consider the twelve points of Fig. 7.2, which we reproduce here as Fig. 7.8. In the worst case, our initial choice of a point is near the center, say (6,8). The furthest point from (6,8) is (12,3), so that point is chosen next.

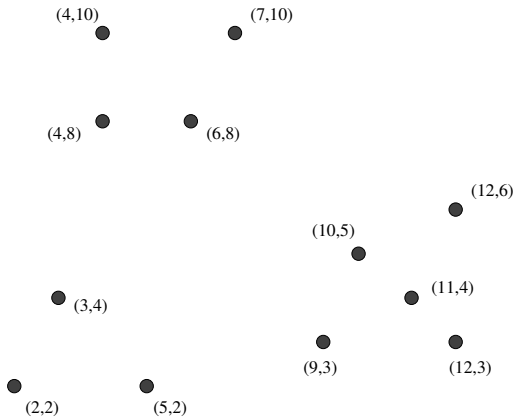

Figure 7.8: Repeat of Fig. 7.2

Among the remaining ten points, the one whose minimum distance to either (6,8) or (12,3) is a maximum is (2,2). That point has distance $\sqrt{52} = 7.21$ from (6,8) and distance $\sqrt{101} = 10.05$ to (12,3); thus its "score" is 7.21. You can check easily that any other point is less than distance 7.21 from at least one of (6,8) and (12,3). Our selection of three starting points is thus (6,8), (12,3), and (2,2). Notice that these three belong to different clusters.

Had we started with a different point, say (10,5), we would get a different set of three initial points. In this case, the starting points would be (10,5), (2,2), and (4,10). Again, these points belong to the three different clusters. □

7.3.3 Picking the Right Value of k

We may not know the correct value of k to use in a k-means clustering. However, if we can measure the quality of the clustering for various values of k, we can usually guess what the right value of k is. Recall the discussion in Section 7.2.3, especially Example 7.5, where we observed that if we take a measure of appropriateness for clusters, such as average radius or diameter, that value will grow slowly, as long as the number of clusters we assume remains at or above the true number of clusters. However, as soon as we try to form fewer

clusters than there really are, the measure will rise precipitously. The idea is expressed by the diagram of Fig. 7.9.

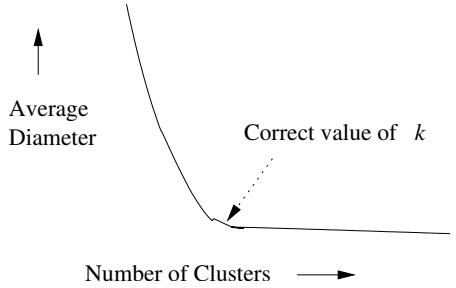

Figure 7.9: Average diameter or another measure of diffuseness rises quickly as soon as the number of clusters falls below the true number present in the data

If we have no idea what the correct value of k is, we can find a good value in a number of clustering operations that grows only logarithmically with the true number. Begin by running the k-means algorithm for $k = 1, 2, 4, 8, \ldots$. Eventually, you will find two values v and $2v$ between which there is very little decrease in the average diameter, or whatever measure of cluster cohesion you are using. We may conclude that the value of k that is justified by the data lies between $v/2$ and v. If you use a binary search (discussed below) in that range, you can find the best value for k in another $\log_2 v$ clustering operations, for a total of $2 \log_2 v$ clusterings. Since the true value of k is at least $v/2$, we have used a number of clusterings that is logarithmic in k.

Since the notion of "not much change" is imprecise, we cannot say exactly how much change is too much. However, the binary search can be conducted as follows, assuming the notion of "not much change" is made precise by some formula. We know that there is too much change between $v/2$ and v, or else we would not have gone on to run a clustering for $2v$ clusters. Suppose at some point we have narrowed the range of k to between x and y. Let $z = (x + y)/2$. Run a clustering with z as the target number of clusters. If there is not too much change between z and y, then the true value of k lies between x and z. So recursively narrow that range to find the correct value of k. On the other hand, if there is too much change between z and y, then use binary search in the range between z and y instead.

7.3.4 The Algorithm of Bradley, Fayyad, and Reina

This algorithm, which we shall refer to as *BFR* after its authors, is a variant of k-means that is designed to cluster data in a high-dimensional Euclidean space. It makes a very strong assumption about the shape of clusters: they must be normally distributed about a centroid. The mean and standard deviation for a cluster may differ for different dimensions, but the dimensions must be