Chapter 10

Clustering and the k-means algorithm

In the clustering problem, we are given a training set $\{x^{(1)}, \ldots, x^{(n)}\}$, and want to group the data into a few cohesive "clusters." Here, $x^{(i)} \in \mathbb{R}^d$ as usual; but no labels $y^{(i)}$ are given. So, this is an unsupervised learning problem.

The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^d$ randomly.

2. Repeat until convergence: {

   For every i, set
   $$c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$$

   For each j, set
   $$\mu_j := \frac{\sum_{i=1}^n 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^n 1\{c^{(i)} = j\}}.$$

   }

In the algorithm above, k (a parameter of the algorithm) is the number of clusters we want to find; and the cluster centroids μ_j represent our current guesses for the positions of the centers of the clusters. To initialize the cluster centroids (in step 1 of the algorithm above), we could choose k training examples randomly, and set the cluster centroids to be equal to the values of these k examples. (Other initialization methods are also possible.)

The inner-loop of the algorithm repeatedly carries out two steps: (i) "Assigning" each training example $x^{(i)}$ to the closest cluster centroid μ_j, and

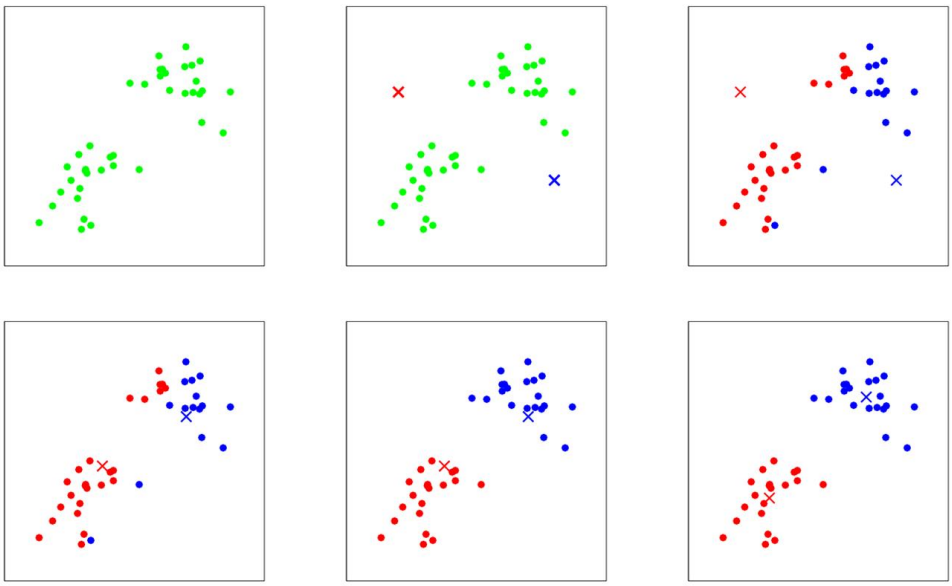

Figure 10.1: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids (in this instance, not chosen to be equal to two training examples). (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster centroid (shown by "painting" the training examples the same color as the cluster centroid to which is assigned); then we move each cluster centroid to the mean of the points assigned to it. (Best viewed in color.) Images courtesy Michael Jordan.

(ii) Moving each cluster centroid μ_j to the mean of the points assigned to it. Figure 10.1 shows an illustration of running k-means.

Is the k-means algorithm guaranteed to converge? Yes it is, in a certain sense. In particular, let us define the **distortion function** to be:

$$J(c, \mu) = \sum_{i=1}^{n} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

Thus, J measures the sum of squared distances between each training example $x^{(i)}$ and the cluster centroid $\mu_{c^{(i)}}$ to which it has been assigned. It can be shown that k-means is exactly coordinate descent on J. Specifically, the inner-loop of k-means repeatedly minimizes J with respect to c while holding μ fixed, and then minimizes J with respect to μ while holding c fixed. Thus,

J must monotonically decrease, and the value of J must converge. (Usually, this implies that c and μ will converge too. In theory, it is possible for k-means to oscillate between a few different clusterings—i.e., a few different values for c and/or μ—that have exactly the same value of J, but this almost never happens in practice.)

The distortion function J is a non-convex function, and so coordinate descent on J is not guaranteed to converge to the global minimum. In other words, k-means can be susceptible to local optima. Very often k-means will work fine and come up with very good clusterings despite this. But if you are worried about getting stuck in bad local minima, one common thing to do is run k-means many times (using different random initial values for the cluster centroids μ_j). Then, out of all the different clusterings found, pick the one that gives the lowest distortion $J(c, \mu)$.