CS229 Lecture notes

Andrew Ng

The k-means clustering algorithm

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

没有标签
非监督学习

The k-means clustering algorithm is as follows:

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

假设聚成 K 类，初始化每个类别的中心

2. Repeat until convergence: {

   For every i, set

遍历所有点，给每个点划分类别

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

   For each j, set

重新计算各类中心

$$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)} = j\}}.$$

   }

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 (ii) Moving each cluster centroid μ_j to the mean of the points assigned to it. Figure 1 shows an illustration of running k-means.

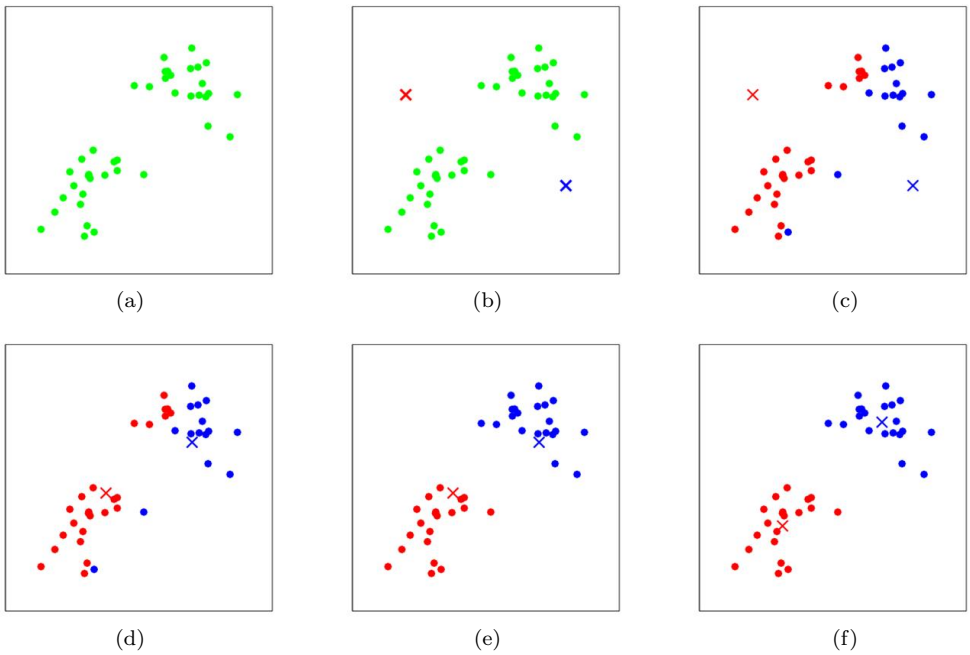

Figure 1: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids (in this instance, not chosen to be equal to two training examples). (c-f) Illustration of running two iterations of k-means. In each 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.

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:

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

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算法一定会收敛，k-means算法可以看成目标函数J(c, u)的坐标下降法。我们重复的进行2.1，2.2两步（已经在上一页标出），执行2.1的时候，相当于固定u，改变c（注意c是类别，改变的规则是点到那哪类的中心最小就将点改为哪类，所以J(c,u)一定会减少），执行2.2的时候相当于固定c，改变u，重新计算各个类别的中心，J(c,u)也会减少，综上，k-means相当于J(c,u)的坐标上升法，一定会收敛

问题一：怎么确定中心数目。有很多自动选择聚类中心的算法

问题二：J(c,u)不是一个凸函数，因此k-means算法能保证收敛到一个局部极值，但不能保证收敛到全局极值最优解，一个简单的解决办法就是随机初始化多次，以最优的聚类结果为最终的结果。

问题三：聚类结束后，如果一个中心没有任何相关的样本，那么这个中心应该去掉。或者重新初始化

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 $J(c, \mu)$.

飞机零件的例子：
振动，发热