Data Scientist Review

Ce Peng

June 22 2021

Latex code ref https://www.overleaf.com/learn
Algorithm content ref https://stanford.edu/ cpiech/cs221/handouts/kmeans.html

0.1 Interview Questions

0.1.1 Question 1

What are drawback of K-means clustering?

- Outliers need to be removed before clustering as it can tend to include outliers into clusters.

0.1.2 Outlier

In Wikipedia,it says that in statistics, an outlier is a data point that differs significantly from other observations.[1][2] An outlier may be due to variability in In statistics, an outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set. An outlier can cause serious problems in statistical analyses.

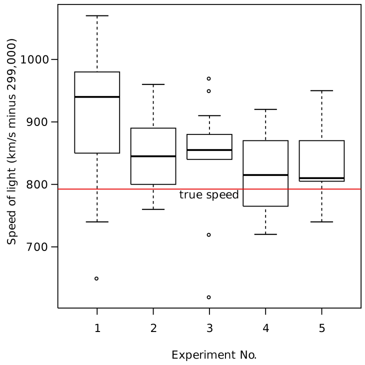

Figure 1: Box plot of data from the Michelson–Morley experiment displaying four outliers in the middle column, as well as one outlier in the first column.

1 K-means

This chapter is written by Chris Piech. Based on a handout by Andrew Ng.

1.1 Basic Idea

Say you are given a data set where each observed example has a set of features, but has no labels. Labels are an essential ingredient to a supervised algorithm like Support Vector Machines, which learns a hypothesis function to predict labels given features. So we can't run supervised learning. What can we do?

One of the most straightforward tasks we can perform on a data set without labels is to find groups of data in our dataset which are similar to one another – what we call clusters.

K-Means is one of the most popular "clustering" algorithms. K-means stores k centroids that it uses to define clusters. A point is considered to be in a particular cluster if it is closer to that cluster's centroid than any other centroid.

K-Means finds the best centroids by alternating between (1) assigning data points to clusters based on the current centroids (2) chosing centroids (points which are the center of a cluster) based on the current assignment of data points to clusters.

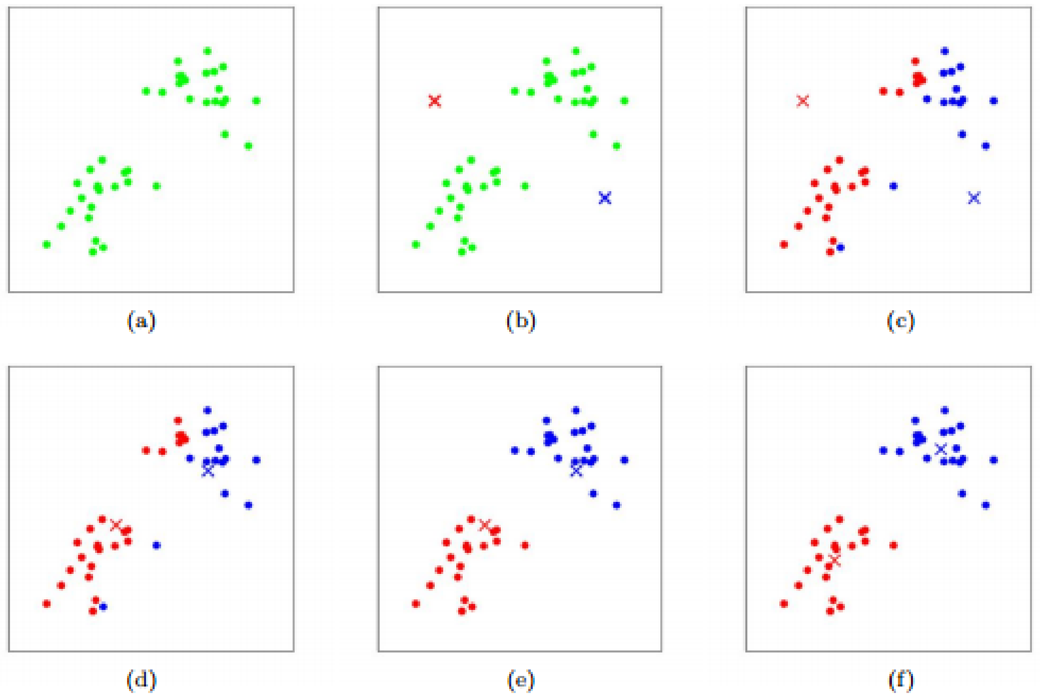

Figure 2: K-means algorithm. Training examples are shown as dots, and cluster centroids are shown as crosses. (a) Original dataset. (b) Random initial cluster centroids. (c-f) Illustration of running two iterations of k-means. In each iteration, we assign each training example to the closest cluster 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. Images courtesy of Michael Jordan.

1.2 Algorithm

In the clustering problem, we are given a training set $x^{(1)},...,x^{(m)}$, and want to group the data into a few cohesive "clusters." Here, we are given feature vectors for each data point $x^{(i)} \in \mathbb{R}^n$ as usual; but no labels $y^{(i)}$ (making this an unsupervised learning problem). Our goal is to predict k centroids and a label $c^{(i)}$ for each datapoint. The k-means clustering algorithm is as follows:

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

2 Repeat until convergence:$\Big\{$

For every i, set

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

For every j, set

$$\mu_j := \frac{\Sigma_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\Sigma_1^m 1\{c^{(i)=j}\}}.$$

$\Big\}$

1.3 Implementation

Here is pseudo-python code which runs k-means on a dataset. It is a short algorithm made longer by verbose commenting.

```
# Function : K-means
# -----------------
# K-means is an algorithm that take in a dataset and a constant
# K and returns K centroid (Which define clusters of data in the
# dataset which are similar to one another)
def kmeans(dataSet, k):
  # Initialize centroid randomly
  numFeatures = dataSet.getNumFeatures()
  centroids = getRandomCentroids(numFeatures, k)
  # Initialize book keeping vars
  iterations = 0
  oldCentroids = None
    # Run the main k-means algorithm
  while not shouldStop(oldCentroids, centroids, iterations):
    # Save old centroids for convergence test. Book keeping.
    oldCentroids = centroids
    iterations += 1

    # Assign labels to each datapoint based on centroids
    labels = getLabels(dataSet, centroids)

    # Assign centroids based on datapoint labels
    centroids = getCentroids(dataSet, labels, k)

  # We can get the labels too by calling getLabels(dataSet, centroids)
  return centroids
```

Listing 1: K-means Pseudo

```
# Function: Should Stop
# -------------
# Returns True or False if k-means is done. K-means terminates either
# because it has run a maximum number of iterations OR the centroids
# stop changing.
def shouldStop(oldCentroids, centroids, iterations):
  if iterations > MAX_ITERATIONS: return True
  return oldCentroids == centroids
```

Listing 2: Should Stop Function Pseudo

```
# Function: Get Labels
# -------------
# Returns a label for each piece of data in the dataset.
def getLabels(dataSet, centroids):
  # For each element in the dataset, chose the closest centroid.
  # Make that centroid the element's label.
```

Listing 3: Get Labels Function

```
# Function: Get Centroids
# -------------
# Returns k random centroids, each of dimension n.
def getCentroids(dataSet, labels, k):
  # Each centroid is the geometric mean of the points that
  # have that centroid's label. Important: If a centroid is empty (no points have
  # that centroid's label) you should randomly re-initialize it.
```

Listing 4: Get Centroid Function

Important note: You might be tempted to calculate the distance between two points manually, by looping over values. This will work, but it will lead to a slow k-means! And a slow k-means will mean that you have to wait longer to test and debug your solution.

Let's define three vectors:

```
x = np.array([1,2,3,4,5])
y = np.array([8,8,8,8,8])
z = np.ones((5,9))
```

To calculate the distance between x and y we can use:

```
np.sqrt(sum(x - y)**2)
```

To calculate the distance between all the length 5 vectors in z and x we can use:

```
np.sqrt(((z-x)**2).sum(axis=0))
```

1.4 Expectation Maximization

K-Means is really just the EM (Expectation Maximization) algorithm applied to a particular naive bayes model.

To demonstrate this remarkable claim, consider the classic naive bayes model with a class variable which can take on discrete values (with domain size k) and a set of feature variables, each of which can take on a continuous value (see figure 2). The conditional probability distributions for $P(f_i = x|C = c)$ is going to be slightly different than usual. Instead of storing this conditional probability as a table, we are going to store it as a single normal (gaussian) distribution, with it's own mean and a standard deviation of 1. Specifically, this means that: $P(f_i = x|C = c) \sim \mathcal{N}(\mu_{c,i}, 1)$

Learning the values of $\mu_{c,i}$ given a dataset with assigned values to the features but not the class variables is the provably identical to running k-means on that dataset.

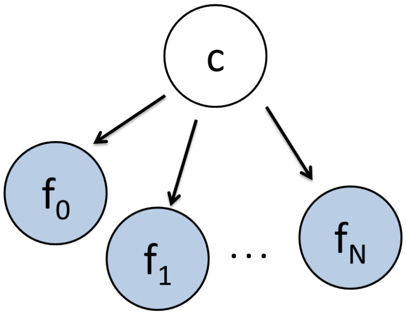

Figure 3: K-Means algorithm is the EM algorithm applied to this Bayes Net.

If we know that this is the strcuture of our bayes net, but we don't know any of the conditional probability distributions then we have to run Parameter Learning before we can run Inference.

In the dataset we are given, all the feature variables are observed (for each data point) but the class variable is hidden. Since we are running Parameter Learning on a bayes net where some variables are unobserved, we should use EM.

Lets review EM. In EM, you randomly initialize your model parameters, then you alternate between (E) assigning values to hidden variables, based on parameters and (M) computing parameters based on fully observed data.

E-Step: Coming up with values to hidden variables, based on parameters. If you work out the math of chosing the best values for the class variable based on the features of a given piece of data in your data set, it comes out to ”for each data-point, chose the centroid that it is closest to, by euclidean distance, and assign that centroid's label.” The proof of this is within your grasp! See lecture.

M-Step: Coming up with parameters, based on full assignments. If you work out the math of chosing the best parameter values based on the features of a given piece of data in your data set, it comes out to ”take the mean of all the data-points that were labeled as c.”

So what? Well this gives you an idea of the qualities of k-means. Like EM, it is provably going to find a local optimum. Like EM, it is not necessarily going to find a global optimum. It turns out those random initial values do matter.

1.5 Intuition

Figure 1 shows k-means with a 2-dimensional feature vector (each point has two dimensions, an x and a y). In your applications, will probably be working with data that has a lot of features. In fact each data-point may be hundreds of dimensions. We can visualize clusters in up to 3 dimensions (see figure 3) but beyond that you have to rely on a more mathematical understanding.

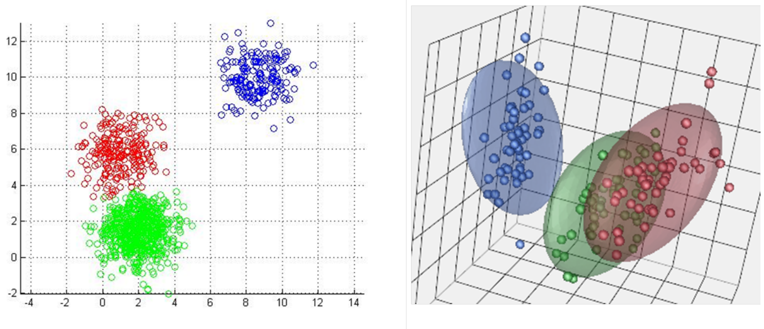

Figure 4: KMeans in other dimensions. (left) K-means in 2d. (right) K-means in 3d. You have to imagine k-means in 4d.

2 Decision Tree