Week6 Learning Reflection

Name: Xinyu Chang
Time: 2022.05.02-05.08
Class: CSE 416 Intro. Machine Learning
Lectures: 11 and 12

Summary

1. K-means++: Pros, improves the quality of local minima; faster convergence to local minima. Cons, Computationally more expensive at the beginning when compared to the simple random initialization.
2. K-means works well for well-separated circular clusters of the same size.
3. K-means is a special case of the EM algorithm.
4. For visualization, generally a smaller number of clusters is better.
5. For tasks like outlier detection, cut based on:
   Distance Threshold
   Or some other metric that tries to measure how big the distance increased after a merge.

Concepts

(Lecture 11)

Clustering: an automatic process of trying to find related groups within the given dataset; classification learns from minimizing the error between a prediction and an actual label.

- **Cluster:** defined by centroid and spread; Unsupervised learning.
- **Centroid/spread;** centroid: the location of its center, spread: shape and size
- **Cluster assignment:** the process of finding these clusters and assigning each example to a particular cluster.
  Close distance reflects the strong similarity between data points.

K-means algorithm:

Given a training dataset of n datapoints and a particular choice of k
Step 0: Initialize cluster centroids randomly
Repeat until convergence:
Step1: Assign each example to its closest cluster centroid
Step2: Update the centroids to be the average of all the points assigned to that cluster.
Stopping Condition:
Cluster assignments haven't changed
Centroids haven't changed
Some number of max iterations have been passed
Converge to *local optima*.

Objective function for k-means

k-means is trying to optimize the heterogeneity objective

$$argmin_{y,\mu} \sum_{j=1}^{k} \sum_{i=1}^{n} 1\{y^{(i)} = j\} \, ||\mu^{(i)} - x^{(i)}||_2^{\,2}$$

How to choose a good k

Better initialization with k-means++

Idea: Try to select a set of points farther away from each other.

(Lecture 12)

k-means failure modes:

Because we minimize Euclidean distance, k-means assumes all the clusters are spherical, if we don't meet the assumption of spherical clusters, we will get unexpected results.

Mixture Model: e.g. Gaussian Mixtures, allows for different cluster shapes and sizes.

- **Soft assignments:**
  example: A news article: 54% chance is about world news, 45% science, 1% conspiracy theory, 0% other.
- **Expectation-Maximization (EM):** converge to local minima

Hierarchical clustering

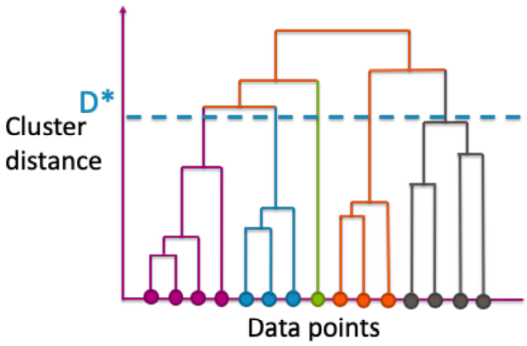
- **Dendrogram:**
  The tall links in the dendrogram show us we are merging vlusters that are far away from each other.
- **Divisive clustering:** top-down.
  Start with all the data in one big cluster and then recursively split the data into the smaller cluster.
  E.g.: Wikipedia
  - **Recursive k-means**

Gaussian Mixture Model

- **Expectation-Minimization algorithm**
  Start with k randomly placed Gaussian means and covariances that represent k clusters
  Repeat until convergence:
  For each point: Calculate the probability that each point belongs to a certain cluster
  Adjust the means and covariances based on the calculated probabilities.

Hierarchical clustering

- **Agglomerative clustering:** bottom-up.
  Start with each data point in its own cluster. Merge clusters until all points are in one big cluster.
  - **Algorithm**
    Initialize each point in its own cluster
    Define a distance metric between clusters
    While there is more than one cluster:
    Merge the two closest clusters
  - **Single Linkage distance metric**
    $distance(C^{(1)}, C^{(2)}) = min_{x^{(i)} \in C^{(1)}, x^{(j)} \in C^{(2)}} d(x^{(i)}, x^{(j)})$
    We are defining the distance between two clusters as the smallest distance between any pair of points between the clusters.
- **How to interpret dendrogram**
  The path shows you all clusters that a single point belongs and the order in which its clusters merged.
  - **Cut dendrogram**
    Choose a distance D* to cut the dendrogram
    Use the largest clusters with distance < D*
    Usually ignore the idea of the nested clusters after cutting
    Every branch that crosses D* becomes its own cluster



- **Agglomerative clustering vs K-means clustering**
  Agglomerative clustering cannot handle big data well, but K-means can.
- **Agglomerative clustering vs divisive clustering**
  Divisive is more complicated to implement. We have to specify different values of k in different recursive loops.
  If we do not generate a complete hierarchy all the way down to the end, divisive clustering is more efficient.
  Given a fixed # of top levels, using an efficient flat algorithm like K-Means, divisive algorithms are linear in the number of clusters k and number of data points n.

Uncertainties

1. How do we choose a good k? Is k the larger the better?