Clustering Theory and Spectral Clustering
Lecture 2

Sorin Istrail

Department of Computer Science
Brown University, Providence
sorin@cs.brown.edu

April 9, 2020

Outline

- **Ch. 5 Clustering Theory and Spectral Clustering:**
  The *k*-Mean Clustering Algorithm

Overview for Ch. 5

- Clustering spaces and distance measures
- The "curse of dimensionality"
- Classification of clustering algorithms
- Hierarchical Clustering Algorithms
- *k*-means Clustering Algorithms
- EM Clustering Algorithms
- Euclidean vs Non-Eulcidian Spaces for Clustering
- An Introduction to Spectral Graph Theory: eigenvalues and eigenvectors in graph theory
- Dimensionality Reduction: Principal Component Analysis
- Spectral Clustering Algorithms

Classification of Clustering Algoritms

- - Machine Learning: **Classification** is in Supervized Learning
  - Machine Learning: **Clustering** is in Unsupervized Learning, maybe the most important

- - Clustering Algorithms: Type 1 **Hierarchical Clustering** aka “tree construction” or “flat” clustering; **hard** clustering
  - Clustering Algorithms: Type 2 k-**Means Clustering** aka “point assignment” clustering; **hard** clustering
  - Clustering Algorithms: Type 3 **Model-based EM Clustering**; **soft** clustering

k-means Clustering Algorithms

- In the class of **point assignment clsutering** algorithms, the best known family of algorithm is the k-**means Clustering** algorithms family
- Two assimptions are in place:
  - The Clustering Space is Euclidean, and
  - The number of clusters k is know in advance, i.e., part of the input

A generic k-means clsutering algorithm

```
GENERIC k-MEANS CLUSTERING ALGORITHM
INPUT: N points of a space S, and k the number of clusters

While termination criterion is not met
BEGIN Choose k points in different clusters;
            Make these points centroids of their clusters;

            FOR each remaining points  p in the input 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
 OUTPUT: the k clusters C1, ..., Ck
```

- The algorithm initializes the k clusters by placing one input point in each cluster
- Then it places each of the remaining points into the clusters one at a time
- For each point, it places it in the cluster whose centroid is closest to the point
- A centroid of a cluster can move around, as points are assigned to that cluster, but not too much
- One further step could be that at the end of the algorithm to fix the centroids and start again the algorithm assigning all to points to the centroids for robustness

k-Means Clustering Theory

- We would like to show that the k-means algorithm iterations converges, by proving that RSS monotonically decreases (in fact decreases or no change) in each iteration.

- The k-means is the most important "flat clustering" (flat meaning non hierarchical) algorithm
- its optimization objective is to minimize the average Euclidean L_2 distance between the points and their centroids
- The **centroid** for cluster C is defined by

$$\mu(C) = \frac{1}{\mid C \mid} \sum_{x \in C} x$$

- The **residual sum of squares** or **RSS** is the square distance of each vector from its centroid summed over all points

$$RSS_r = \sum_{x \in C_r} (x - \mu(C_r))^2$$

- $$RSS = \sum_{r=1}^{k} RSS_r$$

- RSS is objective function of the k-means clustering minimization
- Since the number the points N is fixed, RSS is equivalent to minimizing the **average square distance**, a measure of how well the centroids represent their points in their clusters

- First, RSS deacreases in the reassignment step: each point p is assigned to its closest centroid, so the distance it contributes to RSS decreases
- Second, it decreases in the recomputation step because the new centroid is the minimum of the RSS_r where point p was reassigned to cluster C_r

- $$RSS_r = \sum_{x \in C_r} (x - \mu_r)^2$$
- For finding the minimum we set the derivative to 0:

$$\frac{\partial RSS_r(\mu)}{\partial \mu_r} = \sum_{x \in C_r} 2(x - \mu_r) = 0$$

- $$\sum_{x \in C_r} 2(x - \mu_r) = 0$$

  implies

  $$\mu_r = \frac{1}{\mid C_r \mid} \sum_{x \in C_r}$$

  which is exactly of centroid formula!

- In conclusion, we minimize RSS_r, when the old centroid is replaced with the new controid. RSS, the sum of the RSS_r, must also decrease during recomputation
- Because there are only a finite number of possible clusterings, a monotonically decreasing algorithm will eventually arrive at a local minimum. A note about breaking the ties when ties exist: one can pick among the ties the smallest index of the point in the input order (or other order on the N input points); otherwise, if not careful, the algorithm might cycle forever.
- There is, of course, no guarantee for the global minimum, just reaching a local minimum

Time complexity of the k-means clustering algorithm $=$ $O(N)$ a linear time algorithm

- Most time is computing distances between a point and a centroid, such a computation takes $O(1)$
- The reassignement of a point to one of the k centroids takes constant time as k is a constant
- Overall we caompute kN pairwise distances
- If we perform I iterations (one iteration is reassignement of all the points) then the overall time is $O(IkN)$ which is $O(N)$ as Ii and k are constants

Initializing Clusters for k-Means

- We want to pick initially k "seeds" points that will be in different clusters. Two approaches are used:
- 1. We k pick points that are as far away from one another as possible. We can cluster the sample data hierachically into k clusters. Pick from each clusters a point closer to the cluster centroid
  2. We can also use another approach for the selection set of the first k points to initialiize the k clusters: at $t = 1$ pick the first point at random from the input set; then we add one point to the selection set at time t : for each point not in the selection set yet, compute all the distances to the points in the selection set; then pick at time t the point with the maximum of the minimum distances to the points in the selection set of the $t - 1$ points. Stop after the $t = k$ step.

Outliers

- Outliers present problems for the k-Means clustering
- If an outlier is picked as a seed, the algorithm may end up with a cluster with only one element in that cluset, the outlier element, a singleton cluster; avoiding ouliers from the seed selection phase is important

Picking the value of k for the k-means clustering

- We can use a measure for quality of clustering based on such measures of "diffuseness" as average diameter size or average radius size, and and use the value of k for which e.g., the averge diameter size increases moderately from step to step; if we use a "wrong" k, e.g., fewer clusters that they really are, such monotone increases of the average diameter will go up abruptly at some value of k; it seems that the best such k is the last for the curve is "not bending" up

- If we have no correct value of of what k is, we can find a good value in a number of clustering operations that grows only logarithmically with the true number
- we can run the k-means algorithm for $k = 1, 2, 4, 8, \ldots$ and eventually we will find that somewhere between two values b and $2b$ there is very small difference of the measure of "cohesion" of "diffuseness" that we use; we could conclude that the value of k that is witnessed by the data is bewteen $\frac{b}{2}$ and b
- If we use a binary search in that range we can find the best value of k in another $\log_2 b$ clustering operations, for a total of $2 \log_2 b$ clusterings; since that "true value" of k is at least $\frac{b}{2}$ we have used a number of clusterings that is logarithmic in k