Week-3 | Summary

Karthik Thiagarajan

1. Common

1.1. Notation

Scalars:

$$x_1, x_2, y_1, y_2, z_2, z_2, a, b, \alpha, \beta$$

Column vector:

$$\mathbf{x} \in \mathbb{R}^d$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Row vector:

$$\mathbf{x} \in \mathbb{R}^d$$

$$\mathbf{x}^T = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}$$

Matrix:

$$\mathbf{X} \in \mathbb{R}^{d \times n}$$

1.2. Dataset

$$D = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$$

1.3. Data-matrix

$$\mathbf{X} \in \mathbb{R}^{d \times n}$$

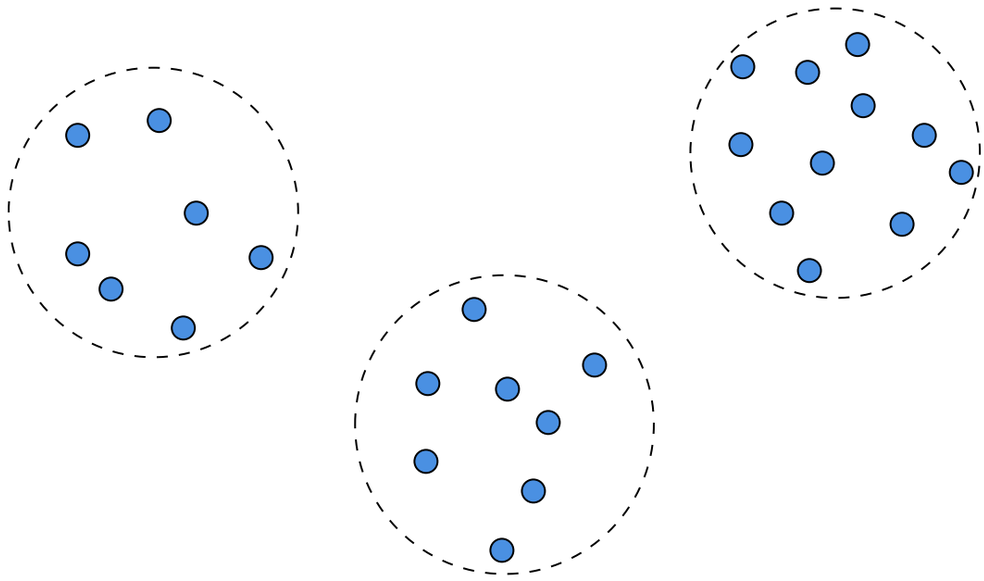
- $d \rightarrow$ number of features
- $n \rightarrow$ number of data-points

$$X = \begin{bmatrix} | & & | \\ \mathbf{x}_1 & \cdots & \mathbf{x}_n \\ | & & | \end{bmatrix}$$

1.4. Data-point

$$\mathbf{x}_i \in \mathbb{R}^d$$

2. Clustering



Cluster membership

Assuming K clusters:

$$\mathbf{z} \in \{1, \cdots, K\}^n$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

Total number of cluster assignments possible

$$\underbrace{k \cdots k}_{n \text{ points}} = k^n$$

Indicator function

$$\mathbf{1}\big(\text{cond}\big) = \begin{cases} 1, & \text{cond is true} \\ 0, & \text{cond is false} \end{cases}$$

Cluster means

For cluster-k:

$$\boldsymbol{\mu}_k \in \mathbb{R}^d$$

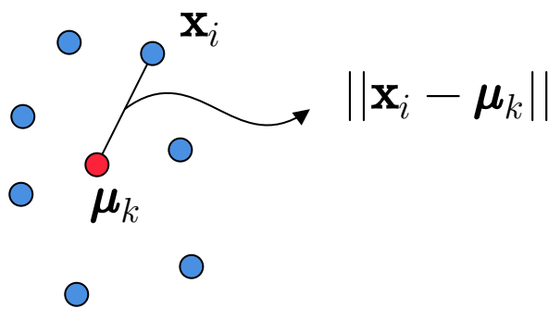
$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} \mathbf{1}(z_i = k) \cdot \mathbf{x}_i}{\sum_{i=1}^{n} \mathbf{1}(z_i = k)}$$

Cluster mean is the mean of the points assigned to it.

3. Optimization problem

Distance of $\mathbf{x}_i$ to the mean of the cluster to which it is assigned, $\boldsymbol{\mu}_{z_i}$:

$$||\mathbf{x}_i - \boldsymbol{\mu}_{z_i}||$$



Objective function

Data-point view

$$f(D, \mathbf{z}) = \sum_{i=1}^{n} ||\mathbf{x}_i - \boldsymbol{\mu}_{z_i}||^2$$

Cluster-view

$$f(D, \mathbf{z}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbf{1}(z_i = k) \cdot ||\mathbf{x}_i - \boldsymbol{\mu}_k||^2$$

Solve

$$\min_{\mathbf{z} \in \{1, \cdots, K\}^n} f(D, \mathbf{z})$$

The search space is combinatorial and hence this is can be very hard to optimize exactly.

4. LLoyd's Algorithm

This is an iterative algorithm that tries to arrive at a reasonably good solution. It is also called the k-means algorithm. The algorithm always convergence and the criterion is given below.

Convergence criterion

$$\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)}$$

Cluster assignment at the end of iteration-t

$$\mathbf{z}^{(t)} \in \{1, \cdots, K\}^n$$

Mean of cluster k at the end of iteration-t

$$\boldsymbol{\mu}_k^{(t)} \in \mathbb{R}^d$$

Initialization

$$\mathbf{z}^{(0)} \rightarrow \text{random vector}$$

$$\boldsymbol{\mu}_k^{(0)} = \frac{\sum_{i=1}^{n} \mathbf{1}\Big(z_i^{(0)} = k\Big)\mathbf{x}_i}{\sum_{i=1}^{n} \mathbf{1}\Big(z_i^{(0)} = k\Big)}$$

Until convergence, update

Cluster membership

$$z_i^{(t+1)} = \underset{k \in \{1,\cdots,K\}}{\arg\min} \; ||\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)}||^2$$

In the case of a tie between multiple clusters for $\mathbf{x}_i$, do not shift allegiance, that is, retain the cluster assignment at time-step t by setting $z_i^{(t+1)} := z_i^{(t)}$.

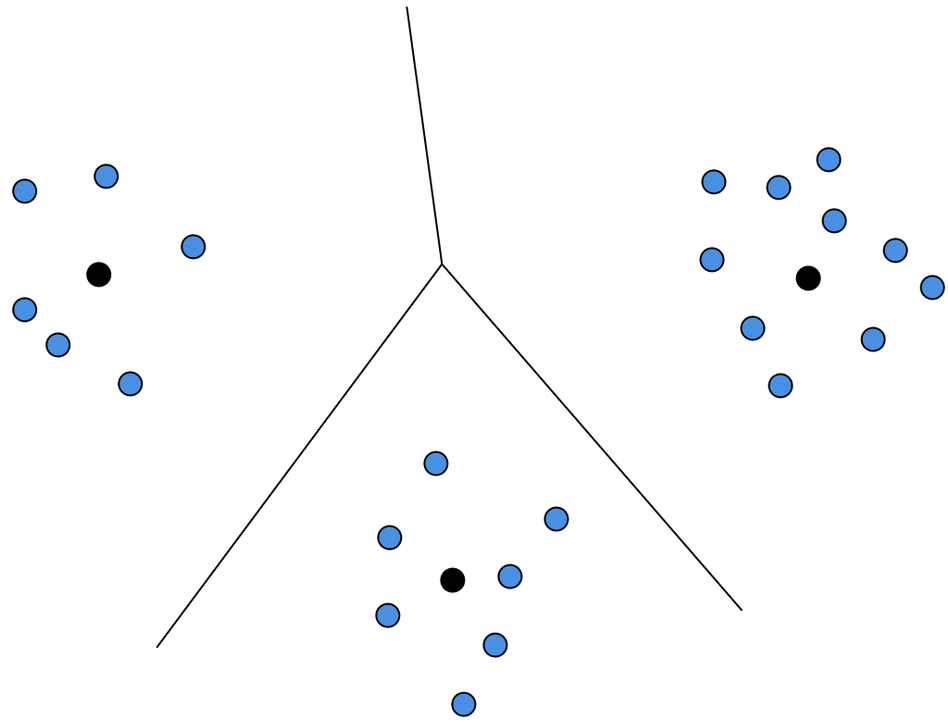
Cluster means

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{n} \mathbf{1}\Big(z_i^{(t+1)} = k\Big)\mathbf{x}_i}{\sum_{i=1}^{n} \mathbf{1}\Big(z_i^{(t+1)} = k\Big)}$$

If it takes T time steps for the algorithm to converge, we can make the following observations:

- $\mathbf{z}^{(0)}, \cdots, \mathbf{z}^{(T)}$ are the sequence of assignments
- $\mathbf{z}^{(T)} = \mathbf{z}^{(T-1)}$ for convergence
- All cluster assignments from $\mathbf{z}^{(0)}$ to $\mathbf{z}^{(T-1)}$ are distinct. That is, cluster assignments never repeat.
- $f\big(D, \mathbf{z}^{(T)}\big) = f\big(D, \mathbf{z}^{(T-1)}\big) <\cdots< f\big(D, \mathbf{z}^{(t)}\big) <\cdots< f\big(D, \mathbf{z}^{(0)}\big)$

5. Voronoi Regions



- Given two clusters indexed by r and s, the set of all points closer to cluster s than cluster r is given by the following half-plane:

$$\Big\{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x} - \boldsymbol{\mu}_s||^2 \leqslant ||\mathbf{x} - \boldsymbol{\mu}_r||^2\Big\}$$

- This separating plane passes through the mid-point of the line segment joining the means, $\dfrac{\boldsymbol{\mu}_s + \boldsymbol{\mu}_r}{2}$, and is perpendicular to $\boldsymbol{\mu}_r - \boldsymbol{\mu}_s$ [this is the perpendicular bisector (a line) in $\mathbb{R}^2$]
- Each cell is formed by the intersection of $K-1$ such half-planes, obtained by comparing cluster s with the remaining $K-1$ clusters.
- The cell that is formed in this manner is convex since:
  - half-planes are convex sets

- the intersection of convex sets is convex

6. Other considerations

- Lloyd's algorithm is deterministic. Given an initial cluster assignment, it will always return the same final clusters.
- k is a hyperparameter and must be chosen appropriately. A hyperparameter is different from a parameter in that it is not "learnt from data" but is chosen before the learning begins. One heuristic to choose a value of k is the elbow method.
- Initialization plays a key role in obtaining good clusters. Pathological initialization could lead to very poor clusters. In practice, for a given k, multiple runs of K-means with different initializations are performed. The run which yields the smallest objective function value is chosen.

7. K-means++

K-means++ is an algorithm that provides a sounder initialization which results in some convergence guarantees. The algorithm is probabilistic in nature. We will describe the algorithm for $K = 3$.

Step-1: Choose any of the n data-points as the first mean $\boldsymbol{\mu}_1$ by sampling uniformly at random.

For convenience, we will relabel the sampled mean as $\mathbf{x}_1$. That is, if $\mathbf{x}_5$ is sampled, swap the labels for $\mathbf{x}_1$ and $\mathbf{x}_5$.

$$P(\mu_1 = \mathbf{x}_1) = \frac{1}{n}$$

Step-2: Choose the second mean as one of the remaining data-points using this process

$$d(\mathbf{x}_i, \boldsymbol{\mu}_1) = \text{distance of } \mathbf{x}_i \text{ from mean } \boldsymbol{\mu}_1$$

The score for each data-point is the squared distance:

$$s(\mathbf{x}_i) = d(\mathbf{x}_i, \boldsymbol{\mu}_1)^2$$

A probability distribution over the $n-1$ points. Recall that the summation starts from 2 since $\mathbf{x}_1$ has been chosen.

$$P(\mathbf{x}_i) = \frac{s(\mathbf{x}_i)}{\sum_{j=2}^{n} s(\mathbf{x}_j)}$$

Sample a data-point using this distribution and assign it to $\boldsymbol{\mu}_2$. For convenience, relabel the sampled point $\mathbf{x}_2$. That is, if $\mathbf{x}_5$ is sampled, swap the labels and the scores for $\mathbf{x}_2$ and $\mathbf{x}_5$.

$$P(\boldsymbol{\mu}_2 = \mathbf{x}_2 \mid \boldsymbol{\mu}_1 = \mathbf{x}_1) = \frac{s(\mathbf{x}_2)}{\sum_{j=2}^{n} s(\mathbf{x}_j)}$$

Step-3: Choose the third mean as one of the remaining data-points using this process

$$d(\mathbf{x}_i, \boldsymbol{\mu}_j) = \text{distance of } \mathbf{x}_i \text{ from mean } \boldsymbol{\mu}_j$$

The score for each data-point is the squared distance of the distance of $\mathbf{x}_i$ to the closest mean:

$$s(\mathbf{x}_i) = \min\left(d(\mathbf{x}_i, \boldsymbol{\mu}_1), d(\mathbf{x}_i, \boldsymbol{\mu}_2)\right)^2$$

A probability distribution over the $n-2$ points. Recall that the summation starts from 3 since $\mathbf{x}_1$ and $\mathbf{x}_2$ have been chosen.

$$P(\mathbf{x}_i) = \frac{s(\mathbf{x}_i)}{\sum_{j=3}^{n} s(\mathbf{x}_j)}$$

Sample a data-point using this distribution and assign it to $\boldsymbol{\mu}_3$. For convenience, relabel the sampled point $\mathbf{x}_3$. That is, if $\mathbf{x}_5$ is sampled, swap the labels and the scores for $\mathbf{x}_3$ and $\mathbf{x}_5$.

$$P(\boldsymbol{\mu}_3 = \mathbf{x}_3 \mid \boldsymbol{\mu}_1 = \mathbf{x}_1, \boldsymbol{\mu}_2 = \mathbf{x}_2) = \frac{s(\mathbf{x}_3)}{\sum_{j=3}^{n} s(\mathbf{x}_j)}$$

The overall probability of choosing these three means in this order is the probability of the three probabilities we have computed so far. For $K > 3$, the algorithm can be extended in a similar fashion.