

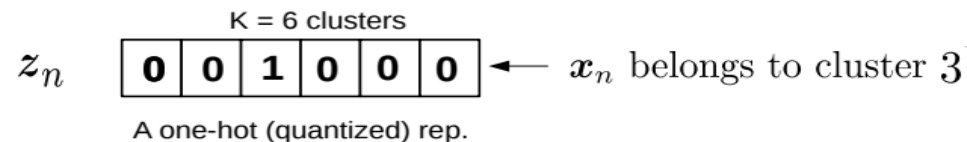
# Unsupervised Learning: Clustering

CS771: Introduction to Machine Learning

# Unsupervised Learning

- It's about learning interesting/useful structures in the data (unsupervisedly!)
- There is no supervision (no labels/responses), only inputs  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
- Some examples of unsupervised learning
  - **Clustering**: Grouping similar inputs together (and dissimilar ones far apart)
  - **Dimensionality Reduction**: Reducing the dimensionality of inputs
  - **Estimating the probability density of inputs** (which distribution  $p(\mathbf{x}|\theta)$  “generated” the inputs)
- Most unsup. learning algos also learn a **new feature representation** of inputs, e.g.,
  - Clustering gives a **one-hot “quantized” representation**  $\mathbf{z}_n$  for each input  $\mathbf{x}_n$

Some clustering algos learn a **soft/probabilistic clustering** in which  $\mathbf{z}_n$  will be a probability vector that sums to 1 (will see later)



Assuming each input belongs deterministically to a single cluster

- Dim-red gives  $\mathbf{z}_n$ , a lower-dim representation of  $\mathbf{x}_n$



# Unsupervised Learning as Representation Learning<sup>3</sup>

- Unsupervised Learning can be seen as learning a function  $f$  s.t.

$$\mathbf{x}_n \approx f(\mathbf{z}_n) \quad \forall n$$

- $\mathbf{z}_n$  denotes a new representation of  $\mathbf{x}_n$  (note:  $\mathbf{z}_n$  can have lower/higher dim than  $\mathbf{x}_n$ )
- Different unsupervised learning algos differ in terms of
  - The nature of the function  $f$
  - The nature of the representation  $\mathbf{z}_n$
- The goal is to learn  $f$  and  $\mathbf{z}_n$  by minimizing some distortion function

$$(\hat{f}, \hat{\mathbf{Z}}) = \operatorname{argmin}_{f, \mathbf{z}} \sum_{n=1}^N \ell(\mathbf{x}_n, f(\mathbf{z}_n))$$

Can additionally also have regularizers/constraints on  $f$  and  $\mathbf{z}_n$

How well we can reconstruct  $\mathbf{x}_n$  from  $f$  and  $\mathbf{z}_n$

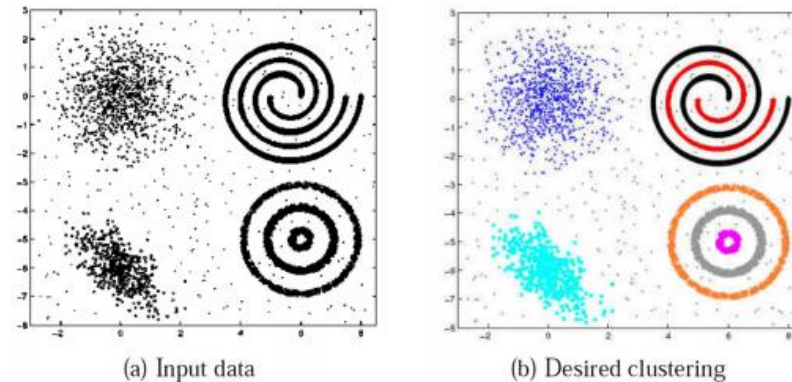


# Clustering

4  
In some cases, we may not know the right number of clusters in the data and may want to learn that (technique exists for doing this but beyond the scope)

- Given:  $N$  unlabeled inputs  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  ; desired no. of partitions  $K$
- Goal: Group the examples into  $K$  “homogeneous” partitions

In addition to partitioning these  $N$  inputs, we may also want to predict which partition a new test point belongs to



Picture courtesy: “Data Clustering: 50 Years Beyond K-Means”, A.K. Jain (2008)

- Loosely speaking, it is classification without ground truth labels of training data
- A good clustering is one that achieves
  - High within-cluster similarity
  - Low inter-cluster similarity



# Similarity can be Subjective

- Clustering only looks at similarities b/w inputs, since no labels are given
- Without labels, similarity can be hard to define

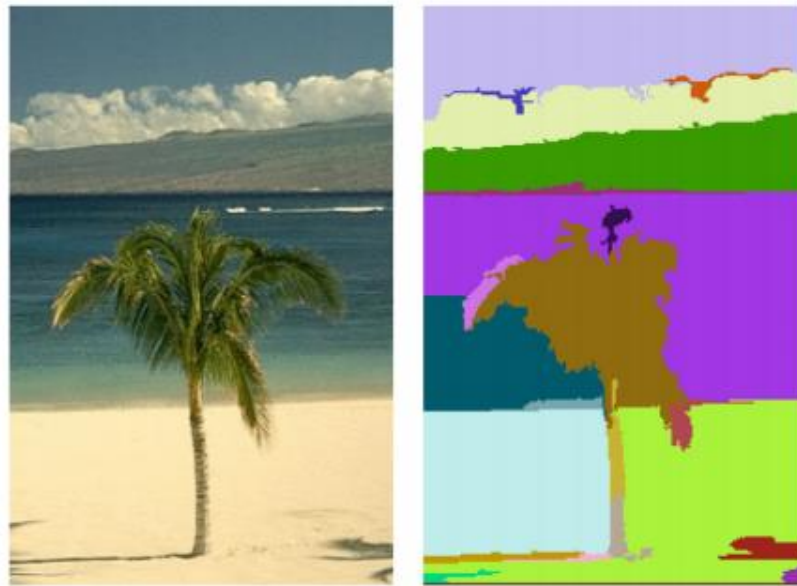


- Thus using the right distance/similarity is very important in clustering
- In some sense, related to asking: “Clustering based on what”?



# Clustering: Some Examples

- Document/Image/Webpage Clustering
- Image Segmentation (clustering pixels)

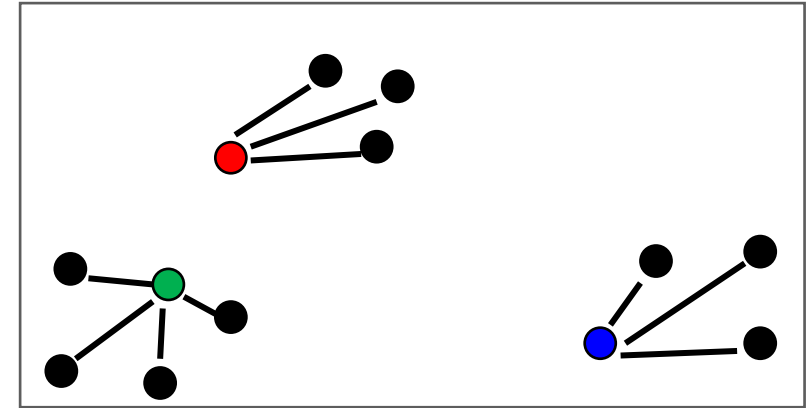


- Clustering web-search results
- Clustering (people) nodes in (social) networks/graphs
- .. and many more..

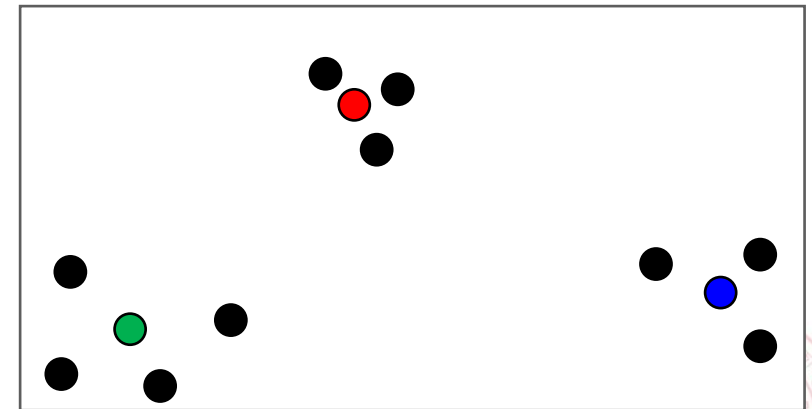
# K-means Clustering

- Based on the following two steps (applied in alternating fashion until not converged)

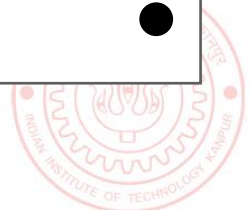
1. Assign each input to the current closest mean



1. Recompute (update) the means



- At the very beginning, the means need to be initialized (at random locations or using other schemes that we will see later)



# K-means Clustering

8

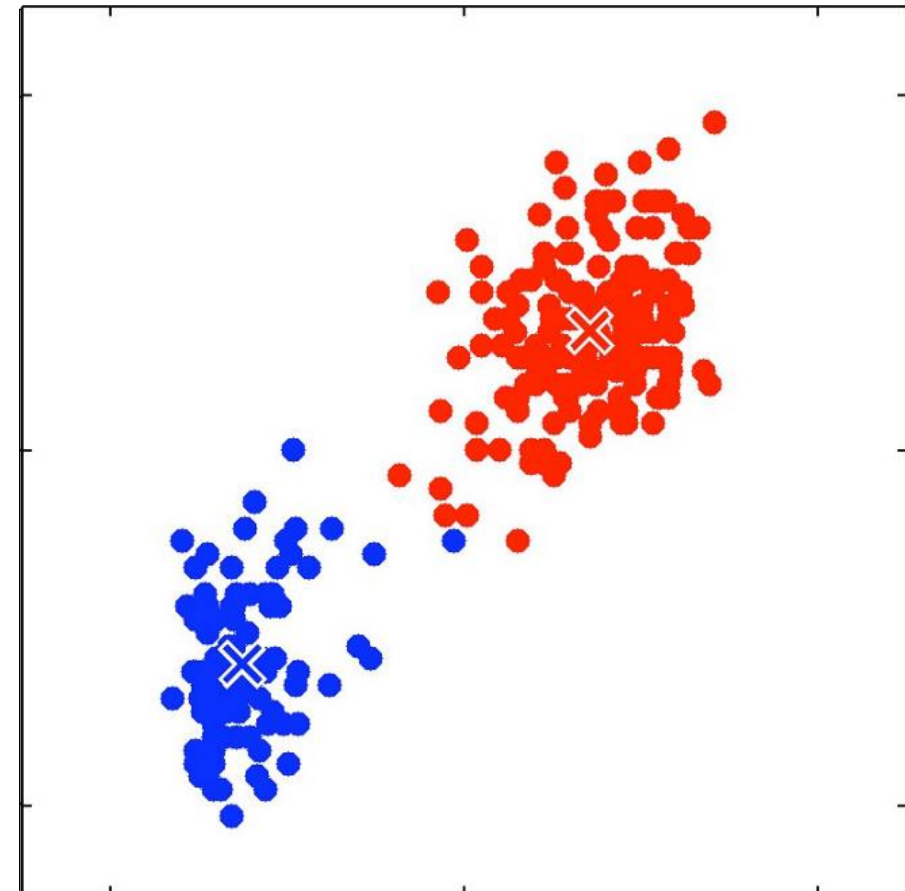
Good initialization matters (some methods exist to pick good initialization, e.g., *K*-means++)

Similar to LwP but in LwP the labels are known so the means can be computed in a single step without iterating multiple times



1. Randomly initialize  $K$  locations (the means)
2. Using the current means, predict the cluster id for each input (closest mean)
3. Recompute the means using the predicted cluster id of each input
4. Go to step 2 if not converged

$K$ -means clustering with  $K = 2$





# $K$ -means (a.k.a. Lloyd's) Algorithm: Formally

- Input:  $N$  inputs  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ ;  $\mathbf{x}_n \in \mathbb{R}^D$ ; desired no. of partitions  $K$
- Desired Output: Cluster ids of these  $N$  inputs and  $K$  cluster means  $\mu_1, \mu_2, \dots, \mu_K$

## $K$ -means Algorithm

- Initialize  $K$  cluster means  $\mu_1, \dots, \mu_K$
- For  $n = 1, \dots, N$ , assign each point  $\mathbf{x}_n$  to the **closest cluster**

$z_n = k$  means  $z_{nk} = 1$  in one-hot representation of  $z_n$

$$z_n = \arg \min_{k \in \{1, \dots, K\}} \|\mathbf{x}_n - \mu_k\|^2$$

- Suppose  $\mathcal{C}_k = \{\mathbf{x}_n : z_n = k\}$ . Re-compute the means

$$\mu_k = \text{mean}(\mathcal{C}_k), \quad k = 1, \dots, K$$

- Go to step 2 if not yet converged

$K$ -means algo can also be seen as doing a compression by "quantization": Representing each of the  $N$  inputs by one of the  $K < N$  means

Can be fixed by modeling each cluster by a probability distribution, such as Gaussian (e.g., Gaussian Mixture Model; will see later)

This basic  $K$ -means models each cluster by a single mean  $\mu_k$ . Ignores size/shape of clusters



# What Loss Function is $K$ -means Optimizing?

- Define the distortion or “loss” for the cluster assignment of a single input  $\mathbf{x}_n$

No labels here. We are measuring how much error we will incur if we assign  $\mathbf{x}_n$  to its nearest cluster mean

$$\ell(\mathbf{x}_n, \boldsymbol{\mu}, \mathbf{z}_n) = \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

$\mathbf{z}_n = [z_{n1}, z_{n2}, \dots, z_{nK}]$  denotes a length  $K$  one-hot encoding of  $\mathbf{x}_n$  and  $z_{nk} = 1$  if  $\mathbf{x}_n$  is assigned to cluster  $k$

- Notation:

- $\mathbf{X}$  is  $N \times D$  matrix of inputs,  $\mathbf{Z}$  is  $N \times K$  matrix denoting cluster ids (each row is a one-hot  $\mathbf{z}_n$ )
- $\boldsymbol{\mu}$  is  $K \times D$  (each row contains the mean  $\boldsymbol{\mu}_k$  of cluster  $k$ )

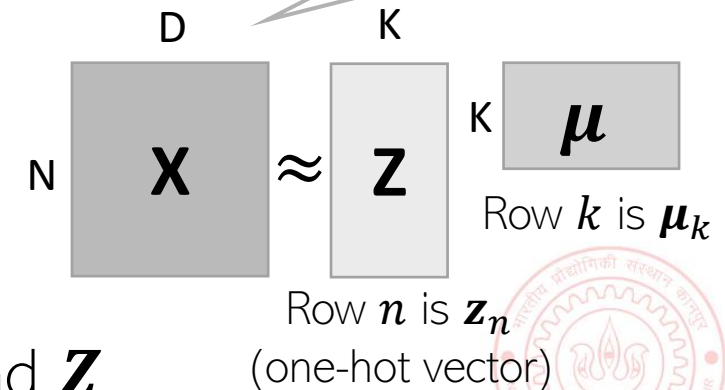
- Total distortion over all inputs defines the  $K$ -means “loss function”

$$\ell(\mathbf{X}, \boldsymbol{\mu}, \mathbf{Z}) = \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

$K$ -means problem viewed as a matrix factorization problem

$$= \|\mathbf{X} - \mathbf{Z}\boldsymbol{\mu}\|_F^2$$

$\|\cdot\|_F^2$  denotes the matrix Frobenius norm (squared norm for matrices)



- The  $K$ -means problem is to minimize this objective w.r.t.  $\boldsymbol{\mu}$  and  $\mathbf{Z}$ 
  - Alternating optimization on this loss would give the  $K$ -means (Lloyd's) algorithm we saw earlier!

# Optimizing the $K$ -means Loss Function

- The  $K$ -means problem is

$$\{\hat{\mathbf{Z}}, \hat{\boldsymbol{\mu}}\} = \arg \min_{\mathbf{Z}, \boldsymbol{\mu}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}) = \arg \min_{\mathbf{Z}, \boldsymbol{\mu}} \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- Can't optimize it jointly for  $\mathbf{Z}$  and  $\boldsymbol{\mu}$ . Let's try alternating optimization for  $\mathbf{Z}$  and  $\boldsymbol{\mu}$

## Alternating Optimization for $K$ -means Problem

- 1 Fix  $\boldsymbol{\mu}$  as  $\hat{\boldsymbol{\mu}}$  and find the optimal  $\mathbf{Z}$  as

Given the current estimates of the  $K$  means  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$ , find the optimal cluster ids  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N$  for all the inputs

$$\hat{\mathbf{Z}} = \arg \min_{\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \hat{\boldsymbol{\mu}}) \quad (\text{still not easy - next slide})$$

- 2 Fix  $\mathbf{Z}$  as  $\hat{\mathbf{Z}}$  and find the optimal  $\boldsymbol{\mu}$  as

Given the current estimates of the optimal cluster ids  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N$ , compute the  $K$  means  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$

$$\hat{\boldsymbol{\mu}} = \arg \min_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{X}, \hat{\mathbf{Z}}, \boldsymbol{\mu})$$

- 3 Go to step 1 if not yet converged

# Solving for $\mathbf{Z}$

- Solving for  $\mathbf{Z}$  with  $\boldsymbol{\mu}$  fixed at  $\hat{\boldsymbol{\mu}}$

$$\hat{\mathbf{Z}} = \arg \min_{\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \hat{\boldsymbol{\mu}}) = \arg \min_{\mathbf{Z}} \sum_{n=1}^N \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k\|^2$$

- Still not easy.  $\mathbf{Z}$  is discrete and above is an NP-hard problem
  - Combinatorial optimization:  $K^N$  possibilities for  $\mathbf{Z}$  ( $N \times K$  matrix with one-hot rows)
- Greedy approach: Optimize  $\mathbf{Z}$  one row ( $\mathbf{z}_n$ ) at a time keeping all others  $\mathbf{z}_n$ 's (and the cluster means  $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$ ) fixed

$$\hat{\mathbf{z}}_n = \arg \min_{\mathbf{z}_n} \sum_{k=1}^K z_{nk} \|\mathbf{x}_n - \hat{\boldsymbol{\mu}}_k\|^2 = \arg \min_{\mathbf{z}_n} \|\mathbf{x}_n - \hat{\boldsymbol{\mu}}_{\mathbf{z}_n}\|^2$$

- Easy to see that this is minimized by assigning  $\mathbf{x}_n$  to the closest mean
  - This is exactly what the  $K$ -means (Lloyd's) algo does!



# Solving for $\mu$

- Solving for  $\mu$  with  $\mathbf{Z}$  fixed at  $\hat{\mathbf{Z}}$

$$\hat{\mu} = \arg \min_{\mu} \mathcal{L}(\mathbf{X}, \hat{\mathbf{Z}}, \mu) = \arg \min_{\mu} \sum_{k=1}^K \sum_{n: \hat{z}_n=k} ||\mathbf{x}_n - \mu_k||^2$$

- Not difficult to solve (each  $\mu_k$  is a real-valued vector, can optimize easily)

$$\hat{\mu}_k = \arg \min_{\mu_k} \sum_{n: \hat{z}_n=k} ||\mathbf{x}_n - \mu_k||^2$$

- Note that each  $\mu_k$  can be optimized for independently
- (Verify) This is minimized by setting  $\hat{\mu}_k$  to be mean of points currently in cluster  $k$ 
  - This is exactly what the  $K$ -means (Lloyd's) algo does!



# Convergence of $K$ -means algorithm

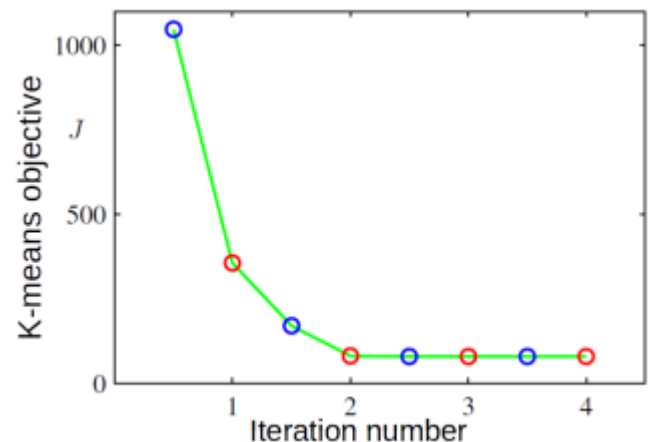
- Each step (updating  $\mathbf{Z}$  or  $\boldsymbol{\mu}$ ) can never increase the  $K$ -means loss
- When we update  $\mathbf{Z}$  from  $\mathbf{Z}^{(t-1)}$  to  $\mathbf{Z}^{(t)}$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu}^{(t-1)}) \leq \mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t-1)}, \boldsymbol{\mu}^{(t-1)}) \quad \text{because} \quad \mathbf{Z}^{(t)} = \arg \min_{\mathbf{Z}} \mathcal{L}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\mu}^{(t-1)})$$

- When we update  $\boldsymbol{\mu}$  from  $\boldsymbol{\mu}^{(t-1)}$  to  $\boldsymbol{\mu}^{(t)}$

$$\mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu}^{(t)}) \leq \mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu}^{(t-1)}) \quad \text{because} \quad \boldsymbol{\mu}^{(t)} = \arg \min_{\boldsymbol{\mu}} \mathcal{L}(\mathbf{X}, \mathbf{Z}^{(t)}, \boldsymbol{\mu})$$

- Thus the  $K$ -means algorithm monotonically decreases the objective



(blue: after  $\mathbf{Z}$  updated, red: after  $\boldsymbol{\mu}$  updated)



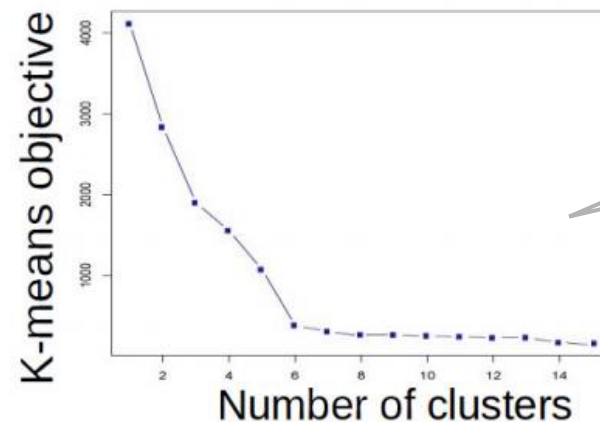
# K-means: Choosing $K$

- One way to select  $K$  for the  $K$ -means algorithm is to try different values of  $K$ , plot the  $K$ -means objective versus  $K$ , and look at the “elbow-point”

If we use  $K = N$  then  $K$ -means also can achieve smallest possible loss (zero!). Think why.



However, if we use  $K = N$  it isn't a useful clustering



$K=6$  is the elbow point

- Can also use information criterion such as AIC (Akaike Information Criterion)

$$AIC = 2\mathcal{L}(\hat{\mu}, \mathbf{X}, \hat{\mathbf{Z}}) + KD$$

$KD$  is the Total number of parameters (note that we have  $K$  means, each is a vector of size  $D$ )

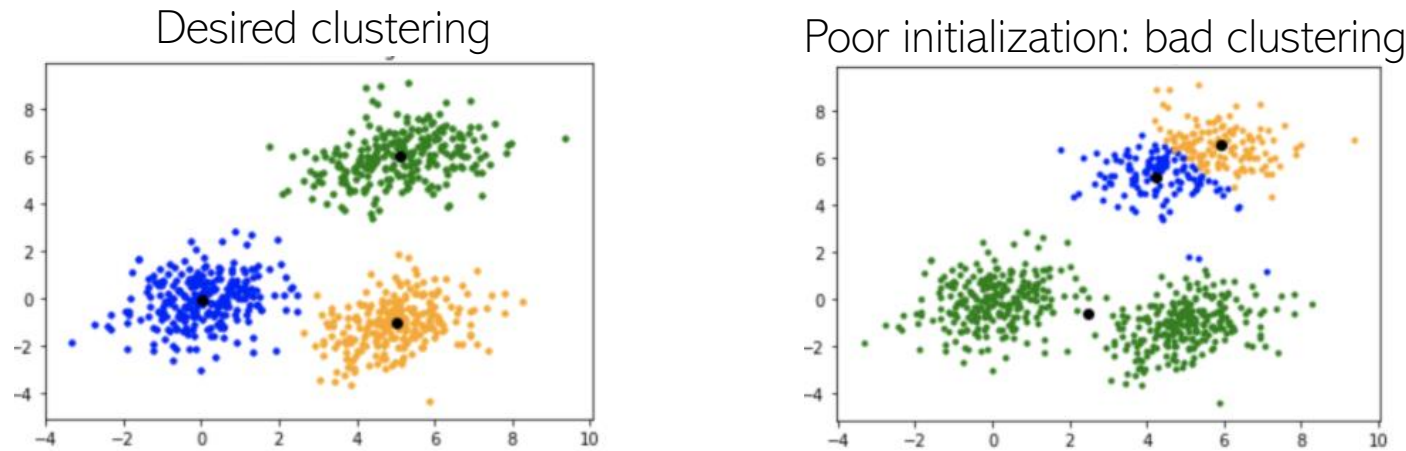
and choose  $K$  which gives the **smallest AIC** (small loss + large  $K$  values penalized)

- More advanced approaches, such as nonparametric Bayesian methods (Dirichlet Process mixture models also used, not within K-means but with other clustering algos)

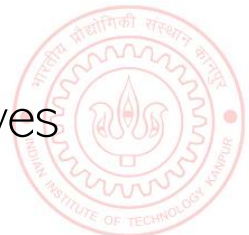


# $K$ -means++

- $K$ -means results can be sensitive to initialization



- $K$ -means++ (Arthur and Vassilvitskii, 2007) an improvement over  $K$ -means
  - Only difference is the way we initialize the cluster centers (rest of it is just  $K$ -means)
  - Basic idea: Initialize cluster centers such that they are reasonably far from each other
  - Note: In  $K$ -means++, the cluster centers are chosen to be  $K$  of the data points themselves





# K-means++

- K-means++ works as follows

- Choose the first cluster mean uniformly randomly to be one of the data points
- The subsequent  $K - 1$  cluster means are chosen as follows
  1. For each unselected point  $\mathbf{x}$ , compute its smallest distance  $D(\mathbf{x})$  from already initialized means
  2. Select the next cluster mean unif. rand. to be one of the unselected points based on probability prop. to  $D(\mathbf{x})^2$
  3. Repeat 1 and 2 until the  $K - 1$  cluster means are initialized

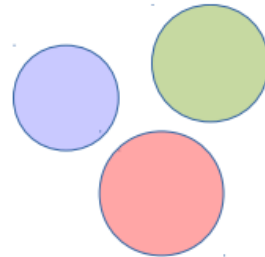
Thus farthest points are most likely to be selected as cluster means

- Now run standard  $K$ -means with these initial cluster means
- $K$ -means++ initialization scheme sort of ensures that the initial cluster means are located in different clusters



# K-means: Hard vs Soft Clustering

- $K$ -means makes hard assignments of points to clusters
  - **Hard assignment:** A point either completely belongs to a cluster or doesn't belong at all



Hard-assignment okay



Hard-assignment tricky

A more principled extension of  $K$ -means for doing soft-clustering is via probabilistic mixture models such as the [Gaussian Mixture Model](#)



- When clusters overlap, soft assignment is preferable (i.e., probability of being assigned to each cluster: say  $K = 3$  and for some point  $\mathbf{x}_n$ ,  $p_1 = 0.7, p_2 = 0.2, p_3 = 0.1$ )
- A heuristic to get **soft assignments**: Transform distances from clusters into prob.

$$\sum_{k=1}^K \gamma_{nk} = 1 \quad \gamma_{nk} = \frac{\exp(-\|\mathbf{x}_n - \mu_k\|^2)}{\sum_{\ell=1}^K \exp(-\|\mathbf{x}_n - \mu_{\ell}\|^2)} \quad (\text{prob. that } \mathbf{x}_n \text{ belongs to cluster } k)$$

- Each cluster mean updates changes:  $\mu_k = \frac{\sum_{n=1}^N \gamma_{nk} \mathbf{x}_n}{\sum_{n=1}^N \gamma_{nk}}$  (all points contribute fractionally)

