

# Big Data Computing

Master's Degree in Computer Science

2019-2020

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

[tolomei@di.uniroma1.it](mailto:tolomei@di.uniroma1.it)



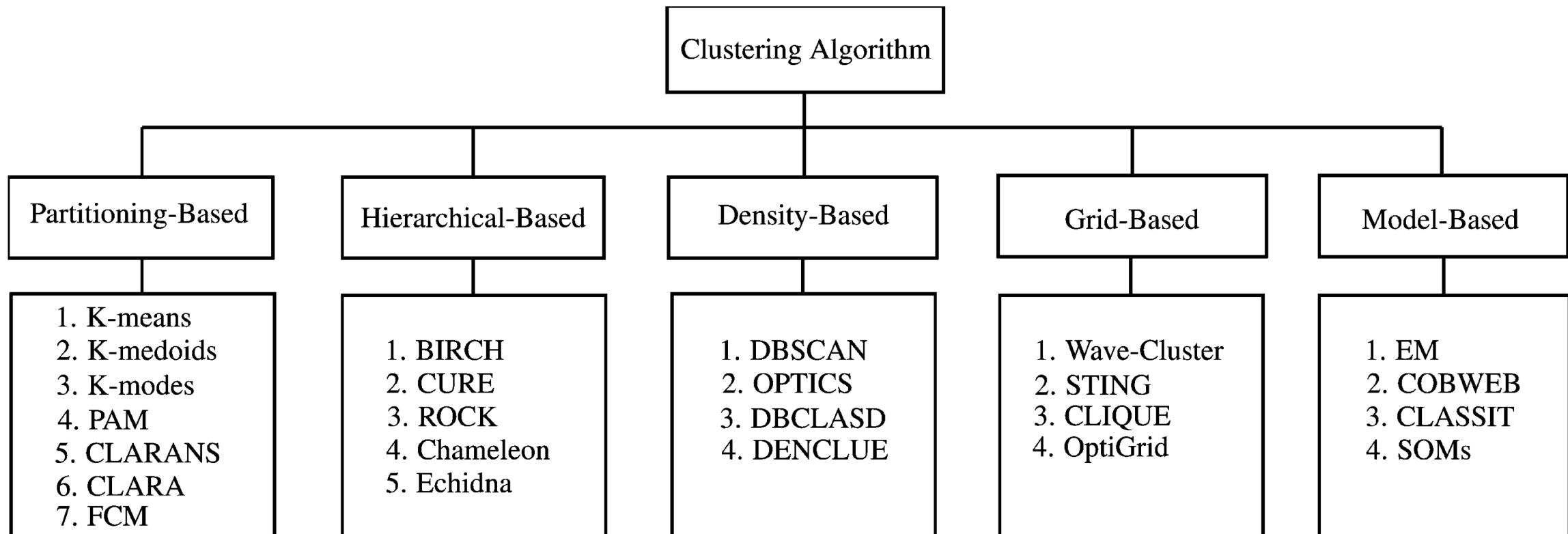
**SAPIENZA**  
UNIVERSITÀ DI ROMA

# Recap from Last Lecture(s)

- Clustering is an unsupervised learning technique to group "similar" data objects together
- Depends on:
  - **object representation**
  - **similarity measure**
- Harder when data dimensionality gets large (**curse of dimensionality**)
- Number of output clusters is part of the problem itself!

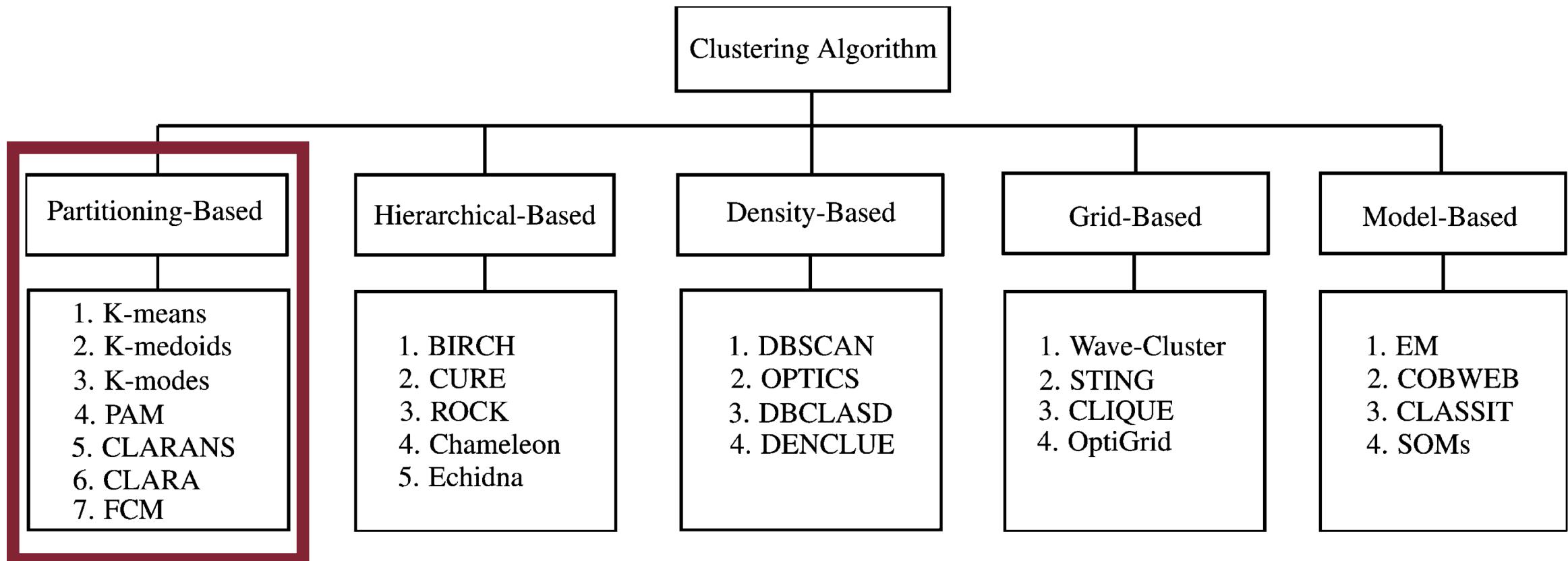
# Clustering Algorithms

# Clustering Algorithms: Taxonomy



source: <https://www.computer.org/csdl/journal/ec/2014/03/06832486/13rRUEgs2xB>

# Clustering Algorithms: Taxonomy



source: <https://www.computer.org/csdl/journal/ec/2014/03/06832486/13rRUEgs2xB>

# Partitioning: Hard Clustering

- **Input:** A set of  $N$  data points and a number  $K$  ( $K < N$ )

# Partitioning: Hard Clustering

- **Input:** A set of  $N$  data points and a number  $K$  ( $K < N$ )
- **Output:** A partition of the  $N$  data points into  $K$  clusters

# Partitioning: Hard Clustering

- **Input:** A set of  $N$  data points and a number  $K$  ( $K < N$ )
- **Output:** A partition of the  $N$  data points into  $K$  clusters
- **Goal:** Find the partition which optimizes a certain criterion
  - Global optimum → Intractable for many objective function (might require to enumerate all the possible partitions)\*
  - $S(K, N) \sim O(K^N)$  →  $K$ -way non-empty partitions of  $N$  elements Stirling partition number
  - Effective heuristics → K-means, K-medoids, K-means++, etc.

\*Kleinberg, J., "An Impossibility Theorem for Clustering" (NIPS 2002)

# Flat Hard Clustering: General Framework

- $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  the set of  $N$  input data points
- $\{C_1, \dots, C_K\}$  the set of  $K$  output clusters
- $C_k$  the generic  $k$ -th cluster
- $\theta_k$  is the *representative* of cluster  $C_k$

**Note:**

At this stage we haven't yet specified what a cluster representative actually is

# Objective Function

$$L(A, \Theta) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$
$$\sum_{k=1}^K \alpha_{n,k} = 1 \quad \boxed{\text{hard clustering}}$$

where:

- $A$  is an  $N \times K$  matrix s.t.  $\alpha_{n,k} = 1$  iff  $\mathbf{x}_n$  is assigned to cluster  $C_k$ , 0 otherwise
- $\Theta = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K\}$  are the cluster representatives
- $\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$  is a function measuring the distance between  $\mathbf{x}_n$  and  $\boldsymbol{\theta}_k$

# Objective Function

$$L(A, \Theta) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$
$$\sum_{k=1}^K \alpha_{n,k} = 1 \quad \boxed{\text{hard clustering}}$$

where:

- $A$  is an  $N \times K$  matrix s.t.  $\alpha_{n,k} = 1$  iff  $\mathbf{x}_n$  is assigned to cluster  $C_k$ , 0 otherwise
- $\Theta = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K\}$  are the cluster representatives
- $\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$  is a function measuring the distance between  $\mathbf{x}_n$  and  $\boldsymbol{\theta}_k$

$$A^*, \Theta^* = \operatorname{argmin}_{A, \Theta} \underbrace{\sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}_{L(A, \Theta)}$$

# Objective Function

$$A^*, \Theta^* = \operatorname{argmin}_{A, \Theta} \underbrace{\sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}_{L(A, \Theta)}$$

# Objective Function

$$A^*, \Theta^* = \operatorname{argmin}_{A, \Theta} \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$

$\underbrace{\qquad\qquad\qquad}_{L(A, \Theta)}$

**exact solution** must explore  
exponential search space  
 $S(K, N) \sim O(K^N)$



**NP-hard**

# Objective Function

$$A^*, \Theta^* = \operatorname{argmin}_{A, \Theta} \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \delta(\mathbf{x}_n, \boldsymbol{\theta}_k)$$

$\underbrace{\qquad\qquad\qquad}_{L(A, \Theta)}$

**exact solution** must explore  
exponential search space  
 $S(K, N) \sim O(K^N)$



**NP-hard**

**non-convex** due to the  
discrete assignment matrix  $A$



**multiple local  
minima**

# Iterative Solution: Lloyd-Forgy Algorithm

- NP-hardness doesn't allow us to compute the exact solution

# Iterative Solution: Lloyd-Forgy Algorithm

- NP-hardness doesn't allow us to compute the exact solution
- Non-convexity doesn't allow us to rely on nice property of convex optimization with numerical methods (unique global minimum)

# Iterative Solution: Lloyd-Forgy Algorithm

- NP-hardness doesn't allow us to compute the exact solution
- Non-convexity doesn't allow us to rely on nice property of convex optimization with numerical methods (unique global minimum)
- **Lloyd-Forgy Algorithm:** 2-step **iterative** approximated solution
  - Assignment step
  - Update step

Does not guarantee to find the global optimum as it may stuck to a local optimum or a saddle point

# 2-Step Optimization: Assignment Step

Minimize  $L$  w.r.t.  $A$  by fixing  $\Theta$

$L(\Theta; A)$  fixed  $\Theta$  parametrized by  $A$

# 2-Step Optimization: Assignment Step

Minimize  $L$  w.r.t.  $A$  by fixing  $\Theta$

$L(\Theta; A)$  fixed  $\Theta$  parametrized by  $A$

**Note:**

Can't take the gradient of  $L$  w.r.t.  $A$   
since  $A$  is discrete!

# 2-Step Optimization: Assignment Step

Minimize  $L$  w.r.t.  $A$  by fixing  $\Theta$

$L(\Theta; A)$  fixed  $\Theta$  parametrized by  $A$

Intuitively, given a set of fixed representatives,  $L$  is minimized if each data point is assigned to the closest centroid according to  $\delta$   
( $L$  is just the summation of all the distances from each data point to its assigned representative)

# 2-Step Optimization: Assignment Step

Minimize  $L$  w.r.t.  $A$  by fixing  $\Theta$

$L(\Theta; A)$  fixed  $\Theta$  parametrized by  $A$

Intuitively, given a set of fixed representatives,  $L$  is minimized if each data point is assigned to the closest centroid according to  $\delta$   
( $L$  is just the summation of all the distances from each data point to its assigned representative)

$$\alpha_{n,k} = \begin{cases} 1 & \text{if } \delta(\mathbf{x}_n, \boldsymbol{\theta}_k) = \min_{1 \leq j \leq K} \{\delta(\mathbf{x}_n, \boldsymbol{\theta}_j)\} \\ 0 & \text{otherwise} \end{cases}$$

# 2-Step Optimization: Update Step

Minimize  $L$  w.r.t.  $\Theta$  by fixing  $A$

$L(A; \Theta)$  fixed  $A$  parametrized by  $\Theta$

# 2-Step Optimization: Update Step

Minimize  $L$  w.r.t.  $\Theta$  by fixing  $A$

$L(A; \Theta)$  fixed  $A$  parametrized by  $\Theta$

We can minimize  $L$  by taking the gradient of  $L$  w.r.t  $\Theta$  (i.e., the vector of partial derivatives), set it to 0 and solve it for  $\Theta$

# 2-Step Optimization: Update Step

$$\nabla L(A; \Theta) = \left( \frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_K} \right)$$

$$\nabla L(A; \Theta) = 0 \Leftrightarrow \frac{\partial L}{\partial \theta_k} = 0 \quad \forall k \in \{1, \dots, K\}$$

# 2-Step Optimization: Update Step

$$\nabla L(A; \Theta) = \left( \frac{\partial L}{\partial \theta_1}, \dots, \frac{\partial L}{\partial \theta_K} \right)$$

$$\nabla L(A; \Theta) = 0 \Leftrightarrow \frac{\partial L}{\partial \theta_k} = 0 \quad \forall k \in \{1, \dots, K\}$$

$$\frac{\partial L}{\partial \theta_k} = \sum_{n=1}^N \alpha_{n,k} \delta(\mathbf{x}_n, \theta_k) = 0$$

Solve for each  $\theta_k$  independently

Depends on the distance function  $\delta$

# A Special Case: K-means

- Each cluster representative is its center of mass (i.e., **centroid**)

# A Special Case: K-means

- Each cluster representative is its center of mass (i.e., **centroid**)
- The centroid of a cluster is the **mean** of the instances assigned to that cluster

# A Special Case: K-means

- Each cluster representative is its center of mass (i.e., **centroid**)
- The centroid of a cluster is the **mean** of the instances assigned to that cluster
- (Re)Assignment of instances to clusters is based on distance/similarity to the current cluster centroids

# A Special Case: K-means

- Each cluster representative is its center of mass (i.e., **centroid**)
- The centroid of a cluster is the **mean** of the instances assigned to that cluster
- (Re)Assignment of instances to clusters is based on distance/similarity to the current cluster centroids
- The basic idea is constructing clusters so that the total within-cluster **Sum of Square Distances (SSD)** is minimized

# K-means: Setup

$\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  the set of  $N$  input data points  
 $\{C_1, \dots, C_K\}$  the set of  $K$  output clusters  
 $C_k$  the generic  $k$ -th cluster

$$\boldsymbol{\theta}_k = \frac{\sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n}{\sum_{n=1}^N \alpha_{n,k}} = \boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{n=1}^N \mathbf{x}_n$$

$$\text{where } |C_k| = \sum_{n=1}^N \alpha_{n,k}$$

# K-means: Objective Function

$$L(A, \Theta) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \underbrace{(||\mathbf{x}_n - \boldsymbol{\theta}_k||_2)^2}_{\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}$$

Euclidean space

# K-means: Objective Function

$$L(A, \Theta) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \underbrace{(||\mathbf{x}_n - \boldsymbol{\theta}_k||_2)^2}_{\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}$$

$$\begin{aligned}\delta(\mathbf{x}_n, \boldsymbol{\theta}_k) &= (||\mathbf{x}_n - \boldsymbol{\theta}_k||_2)^2 = \\ &= \left[ \sqrt{(\mathbf{x}_n - \boldsymbol{\theta}_k)^2} \right]^2 = (\mathbf{x}_n - \boldsymbol{\theta}_k)^2\end{aligned}$$

Sum of Square Distances  
(SSD)

# K-means: Objective Function

$$L(A, \Theta) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} \underbrace{(||\mathbf{x}_n - \boldsymbol{\theta}_k||_2)^2}_{\delta(\mathbf{x}_n, \boldsymbol{\theta}_k)}$$

$$\delta(\mathbf{x}_n, \boldsymbol{\theta}_k) = (||\mathbf{x}_n - \boldsymbol{\theta}_k||_2)^2 =$$

$$= \left[ \sqrt{(\mathbf{x}_n - \boldsymbol{\theta}_k)^2} \right]^2 = (\mathbf{x}_n - \boldsymbol{\theta}_k)^2$$

Sum of Square Distances  
(SSD)

$$L(A, \Theta) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)^2$$

# K-means: Assignment Step

Minimize  $L$  w.r.t.  $A$  by fixing  $\Theta$

Intuitively, given a set of fixed centroids,  $L$  is minimized if each data point is assigned to the centroid with the smallest SSD  
( $L$  is just the SSD from each data point to its assigned centroid)

$$\alpha_{n,k} = \begin{cases} 1 & \text{if } (\mathbf{x}_n - \boldsymbol{\theta}_k)^2 = \min_{1 \leq j \leq K} \{(\mathbf{x}_n - \boldsymbol{\theta}_j)^2\} \\ 0 & \text{otherwise} \end{cases}$$

# K-means: Update Step

Minimize  $L$  w.r.t.  $\Theta$  by fixing  $A$

$$\Theta^* = \operatorname{argmin}_{\Theta} \underbrace{\sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)^2}_{L(A, \Theta)}$$

Compute the gradient w.r.t.  $\Theta$ , set it to 0 and solve it for  $\Theta$

# K-means: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \sum_{n=1}^N \alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)^2 = 0 \quad \forall k \in \{1, \dots, K\}$$

# K-means: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \sum_{n=1}^N \alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)^2 = 0 \quad \forall k \in \{1, \dots, K\}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \sum_{n=1}^N -2\alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)$$

# K-means: Update Step

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \sum_{n=1}^N \alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)^2 = 0 \quad \forall k \in \{1, \dots, K\}$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}_k} = \sum_{n=1}^N -2\alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k)$$

$$\text{Find } \boldsymbol{\theta}_k^* \text{ s.t. } \sum_{n=1}^N -2\alpha_{n,k} (\mathbf{x}_n - \boldsymbol{\theta}_k^*) = 0$$

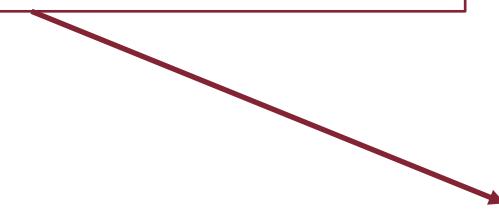
# K-means: Update Step

$$\sum_{n=1}^N -2\alpha_{n,k}(\mathbf{x}_n - \boldsymbol{\theta}_k^*) = 0 \Leftrightarrow$$
$$2 \sum_{n=1}^N \alpha_{n,k} \boldsymbol{\theta}_k^* = 2 \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$
$$\boldsymbol{\theta}_k^* \sum_{n=1}^N \alpha_{n,k} = \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$

# K-means: Update Step

$$\sum_{n=1}^N -2\alpha_{n,k}(\mathbf{x}_n - \boldsymbol{\theta}_k^*) = 0 \Leftrightarrow$$
$$2 \sum_{n=1}^N \alpha_{n,k} \boldsymbol{\theta}_k^* = 2 \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$
$$\boldsymbol{\theta}_k^* \sum_{n=1}^N \alpha_{n,k} = \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$

$\boldsymbol{\theta}_k^*$  does not depend on  $N$ ,  
therefore it can be factored out



# K-means: Update Step

$$\theta_k^* \sum_{n=1}^N \alpha_{n,k} = \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$

$$\theta_k^* = \frac{\sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n}{\sum_{n=1}^N \alpha_{n,k}} = \mu_k = \frac{1}{|C_k|} \sum_{n=1}^N \mathbf{x}_n$$

# K-means: Update Step

$$\theta_k^* \sum_{n=1}^N \alpha_{n,k} = \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$

$$\boxed{\theta_k^* = \frac{\sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n}{\sum_{n=1}^N \alpha_{n,k}} = \mu_k = \frac{1}{|C_k|} \sum_{n=1}^N \mathbf{x}_n}$$

# K-means: Update Step

$$\theta_k^* \sum_{n=1}^N \alpha_{n,k} = \sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n$$

$$\theta_k^* = \frac{\sum_{n=1}^N \alpha_{n,k} \mathbf{x}_n}{\sum_{n=1}^N \alpha_{n,k}} = \mu_k = \frac{1}{|C_k|} \sum_{n=1}^N \mathbf{x}_n$$

The cluster centroid (i.e., **mean**) minimizes the objective  
(for a fixed assignment  $A$ )

# K-means: Lloyd-Forgy Algorithm

I. Specify the number of output clusters K

# K-means: Lloyd-Forgy Algorithm

- I. Specify the number of output clusters K
2. Select K observations **at random** from the N data points as the initial cluster centroids

# K-means: Lloyd-Forgy Algorithm

- I. Specify the number of output clusters K
2. Select K observations **at random** from the N data points as the initial cluster centroids
3. **Assignment step:** Assign each observation to the closest centroid based on the distance measure chosen

# K-means: Lloyd-Forgy Algorithm

1. Specify the number of output clusters K
2. Select K observations **at random** from the N data points as the initial cluster centroids
3. **Assignment step:** Assign each observation to the closest centroid based on the distance measure chosen
4. **Update step:** For each of the K clusters update the centroid by computing the new mean values of all the data points now in the cluster

# K-means: Lloyd-Forgy Algorithm

1. Specify the number of output clusters K
2. Select K observations **at random** from the N data points as the initial cluster centroids
3. **Assignment step:** Assign each observation to the closest centroid based on the distance measure chosen
4. **Update step:** For each of the K clusters update the centroid by computing the new mean values of all the data points now in the cluster
5. Iteratively repeat steps 3-4 until a **stopping criterion** is met

# Stopping Criterion

- Several options to choose from:
  - Fixed number of iterations
  - Cluster assignments stop changing (beyond some threshold)
  - Centroid doesn't change (beyond some threshold)

# Lloyd-Forgy's Convergence

- How/Why are we guaranteed the K-means algorithm ever reaches a fixed point?
  - A state in which clusters do not change

# Lloyd-Forgy's Convergence

- How/Why are we guaranteed the K-means algorithm ever reaches a fixed point?
  - A state in which clusters do not change
- Intuitively, in both steps we either improve the objective or not

# Lloyd-Forgy's Convergence

- How/Why are we guaranteed the K-means algorithm ever reaches a fixed point?
  - A state in which clusters do not change
- Intuitively, in both steps we either improve the objective or not
- It is an instance of more general **Expectation Maximization (EM)**
  - EM is known to converge (although not necessarily to a global optimum)

# Lloyd-Forgy's Relationship with EM

- **E-step = Assignment step**
  - Each object is assigned to the closest centroid, i.e., to the most likely cluster
  - Monotonically decreases SSD

# Lloyd-Forgy's Relationship with EM

- **E-step = Assignment step**
  - Each object is assigned to the closest centroid, i.e., to the most likely cluster
  - Monotonically decreases SSD
- **M-step = Update step**
  - The model (i.e., centroids) are updated (i.e., SSD optimization)
  - Monotonically decreases each  $SSD_k$

# Lloyd-Forgy's Complexity Analysis

- Computing the distance between two  $d$ -dimensional data points takes  $O(d)$

# Lloyd-Forgy's Complexity Analysis

- Computing the distance between two  $d$ -dimensional data points takes  $O(d)$
- (Re-)Assigning clusters [E-step]:  $O(KN)$  distance computations or  $O(KNd)$

# Lloyd-Forgy's Complexity Analysis

- Computing the distance between two  $d$ -dimensional data points takes  $O(d)$
- (Re-)Assigning clusters [E-step]:  $O(KN)$  distance computations or  $O(KNd)$
- Computing centroids [M-step]:  $O(Nd)$  as there are  $O(N)$  average computations since each data point is added to a cluster exactly once *at each iteration*, each one taking  $O(d)$

# Lloyd-Forgy's Complexity Analysis

- Computing the distance between two  $d$ -dimensional data points takes  $O(d)$
- (Re-)Assigning clusters [E-step]:  $O(KN)$  distance computations or  $O(KNd)$
- Computing centroids [M-step]:  $O(Nd)$  as there are  $O(N)$  average computations since each data point is added to a cluster exactly once *at each iteration*, each one taking  $O(d)$
- Overall:  $O(RKNd)$  assuming the 2 steps above are repeated  $R$  times

# K-means: Seed Choice

- Convergence (rate) and clustering quality depends on the selection of **initial centroids**
  - Forgy method **randomly** chooses K data points as the initial means
  - Random Partition method **randomly** assigns a cluster to each observation

# K-means: Seed Choice

- Convergence (rate) and clustering quality depends on the selection of **initial centroids**
  - Forgy method **randomly** chooses K data points as the initial means
  - Random Partition method **randomly** assigns a cluster to each observation
- Randomness may result in convergence to **sub-optimal** clusterings

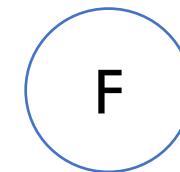
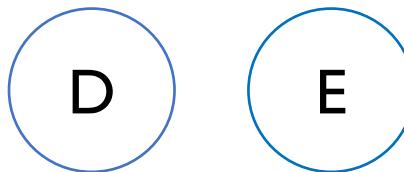
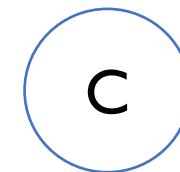
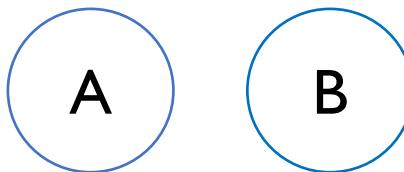
# K-means: Seed Choice

- Convergence (rate) and clustering quality depends on the selection of **initial centroids**
  - Forgy method **randomly** chooses K data points as the initial means
  - Random Partition method **randomly** assigns a cluster to each observation
- Randomness may result in convergence to **sub-optimal** clusterings

## Problem Mitigation:

Execute several runs of the Lloyd-Forgy algorithm  
with multiple random initialization seeds

# K-means: Seed Choice

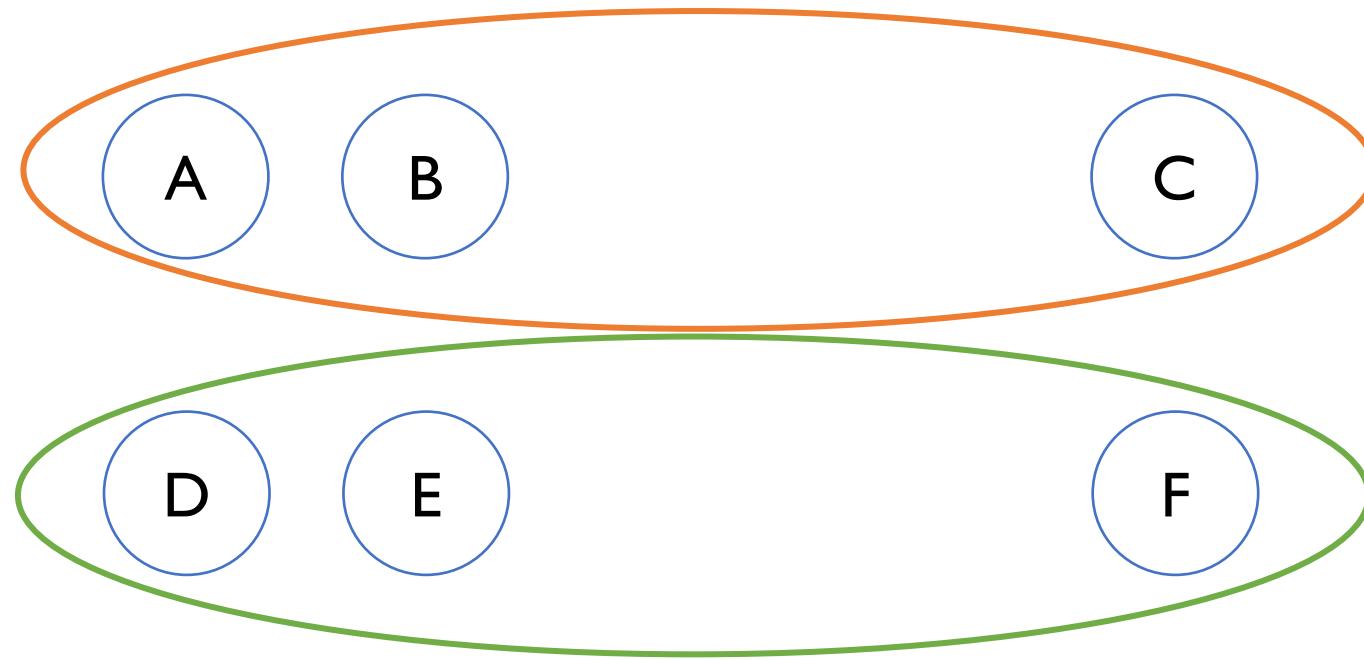


# K-means: Bad (Unlucky) Seed Choice



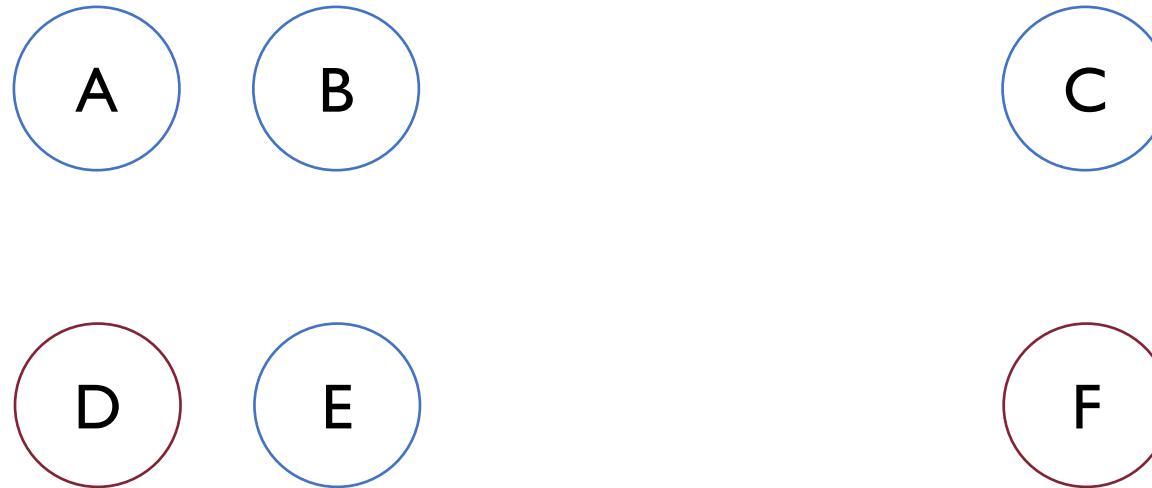
If B and E are randomly chosen as initial centroids...

# K-means: Bad (Unlucky) Seed Choice



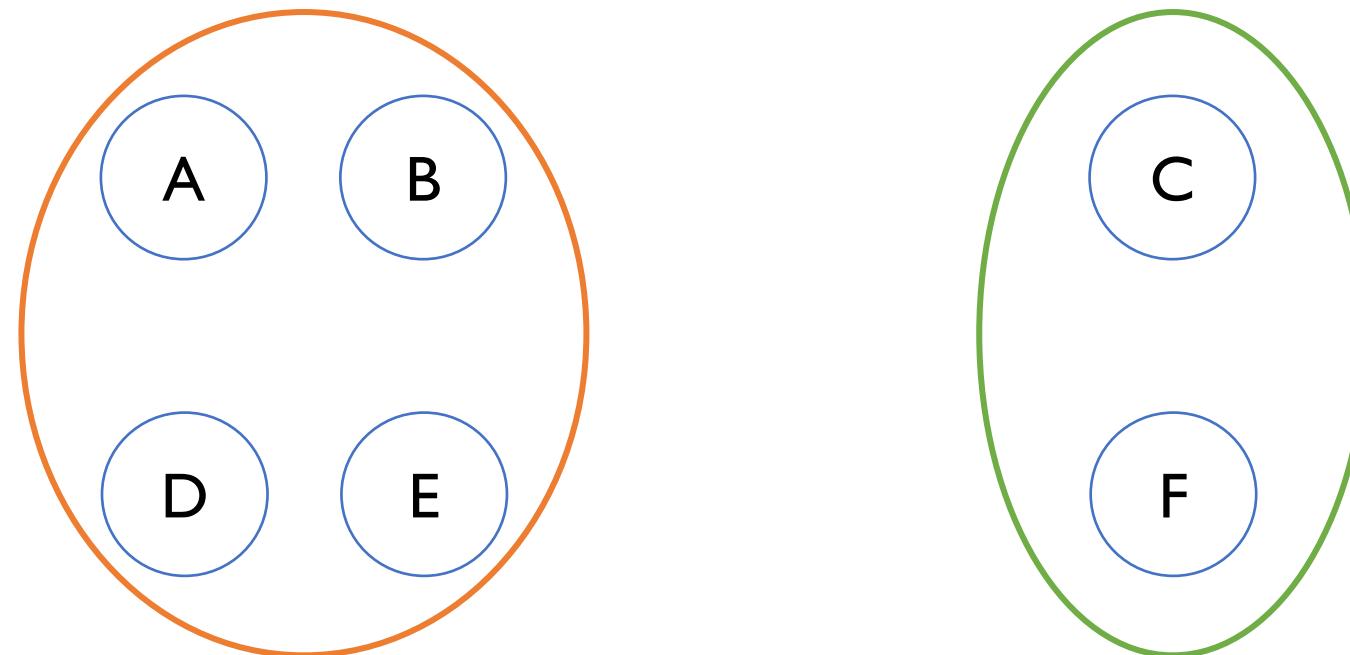
The algorithm converges to the sub-optimal clustering above

# K-means: Good (Lucky) Seed Choice



If D and F are randomly chosen as initial centroids instead...

# K-means: Good (Lucky) Seed Choice



The algorithm converges to a better clustering

# Alternative Seed Choice: K-means++

- A preliminary method to carefully select initial centroids proposed in 2007 by Arthur and Vassilvitskii [[paper](#)]

# Alternative Seed Choice: K-means++

- A preliminary method to carefully select initial centroids proposed in 2007 by Arthur and Vassilvitskii [[paper](#)]
- Intuition: spreading out the K initial cluster centers is a good thing

# Alternative Seed Choice: K-means++

- A preliminary method to carefully select initial centroids proposed in 2007 by Arthur and Vassilvitskii [[paper](#)]
- Intuition: spreading out the K initial cluster centers is a good thing
  - I. Choose one center uniformly at random from among initial data points

# Alternative Seed Choice: K-means++

- A preliminary method to carefully select initial centroids proposed in 2007 by Arthur and Vassilvitskii [[paper](#)]
- Intuition: spreading out the K initial cluster centers is a good thing
  - I. Choose one center uniformly at random from among initial data points
  2. For each data point  $\mathbf{x}$ , compute  $D(\mathbf{x})$  as the distance between  $\mathbf{x}$  and the nearest center that has already been chosen

# Alternative Seed Choice: K-means++

- A preliminary method to carefully select initial centroids proposed in 2007 by Arthur and Vassilvitskii [[paper](#)]
- Intuition: spreading out the K initial cluster centers is a good thing
  1. Choose one center uniformly at random from among initial data points
  2. For each data point  $\mathbf{x}$ , compute  $D(\mathbf{x})$  as the distance between  $\mathbf{x}$  and the nearest center that has already been chosen
  3. Choose one new data point at random as a new center with probability proportional to  $D(\mathbf{x})^2$

# Alternative Seed Choice: K-means++

- A preliminary method to carefully select initial centroids proposed in 2007 by Arthur and Vassilvitskii [[paper](#)]
- Intuition: spreading out the K initial cluster centers is a good thing
  1. Choose one center uniformly at random from among initial data points
  2. For each data point  $\mathbf{x}$ , compute  $D(\mathbf{x})$  as the distance between  $\mathbf{x}$  and the nearest center that has already been chosen
  3. Choose one new data point at random as a new center with probability proportional to  $D(\mathbf{x})^2$
  4. Repeat steps 2. and 3. until K centers are chosen, then run Lloyd-Forgy

# "Vanilla" K-means vs. K-means++

- Random initialization used with "vanilla" K-means may produce clusters that are **arbitrarily worse** than optimum

# "Vanilla" K-means vs. K-means++

- Random initialization used with "vanilla" K-means may produce clusters that are **arbitrarily worse** than optimum
- K-means++ provides an upper-bound to the approximation obtained w.r.t. the optimal solution

# "Vanilla" K-means vs. K-means++

- Random initialization used with "vanilla" K-means may produce clusters that are **arbitrarily worse** than optimum
- K-means++ provides an upper-bound to the approximation obtained w.r.t. the optimal solution
- At most, clusters obtained with K-means++ initialization are  $O(\log K)$  worse than the optimal partitioning

# K-means: How Many Clusters?

- Number of clusters K is given
  - Great! Partition N data points into a predetermined number K of clusters
  - Unfortunately, it is very uncommon to know K in advance

# K-means: How Many Clusters?

- Number of clusters  $K$  is given
  - Great! Partition  $N$  data points into a predetermined number  $K$  of clusters
  - Unfortunately, it is very uncommon to know  $K$  in advance
- Finding the “right” number  $K$  of clusters is part of the problem!
  - Trade-off between having too few and too many clusters
  - Total benefit vs. Total cost

# K-means: Total Benefit

- Given a clustering, define the benefit  $b_i$  for a data point  $x_i$  to be the similarity to its assigned centroid

# K-means: Total Benefit

- Given a clustering, define the benefit  $b_i$  for a data point  $x_i$  to be the similarity to its assigned centroid
- Define the total benefit  $B$  to be the sum of the individual benefits

# K-means: Total Benefit

- Given a clustering, define the benefit  $b_i$  for a data point  $x_i$  to be the similarity to its assigned centroid
- Define the total benefit  $B$  to be the sum of the individual benefits

## NOTE

There is always a clustering whose total benefit  $B=N$   
(where  $N$  is the number of data points)

Why?

# K-means: Total Cost

- Assign a cost  $p$  to each cluster, thereby a clustering with K clusters has a total cost  $P=Kp$

# K-means: Total Cost

- Assign a cost  $p$  to each cluster, thereby a clustering with K clusters has a total cost  $P=Kp$
- Define the value  $V$  of a clustering to be total benefit-total cost

$$V = B - P$$

# K-means: Total Cost

- Assign a cost  $p$  to each cluster, thereby a clustering with K clusters has a total cost  $P=Kp$
- Define the value  $V$  of a clustering to be total benefit-total cost

$$V = B - P$$

**Goal:**

Find the clustering which **maximizes**  $V$ , over all choices of K

# K-means: Total Cost

- Assign a cost  $p$  to each cluster, thereby a clustering with  $K$  clusters has a total cost  $P=Kp$
- Define the value  $V$  of a clustering to be total benefit-total cost

$$V = B - P$$

**Goal:**

Find the clustering which **maximizes**  $V$ , over all choices of  $K$

$B$  increases with larger values of  $K$ , but  $P$  allows to stop that

# K-means: "Elbow" Method

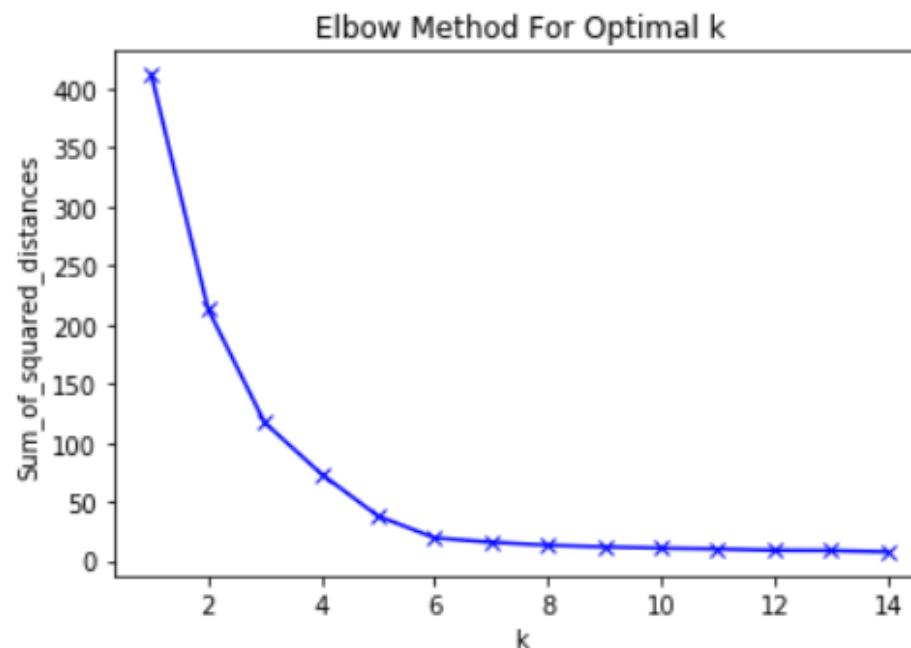
- Empirical method to figure out the right number K of clusters

# K-means: "Elbow" Method

- Empirical method to figure out the right number K of clusters
- Trade-off between total benefit and total cost

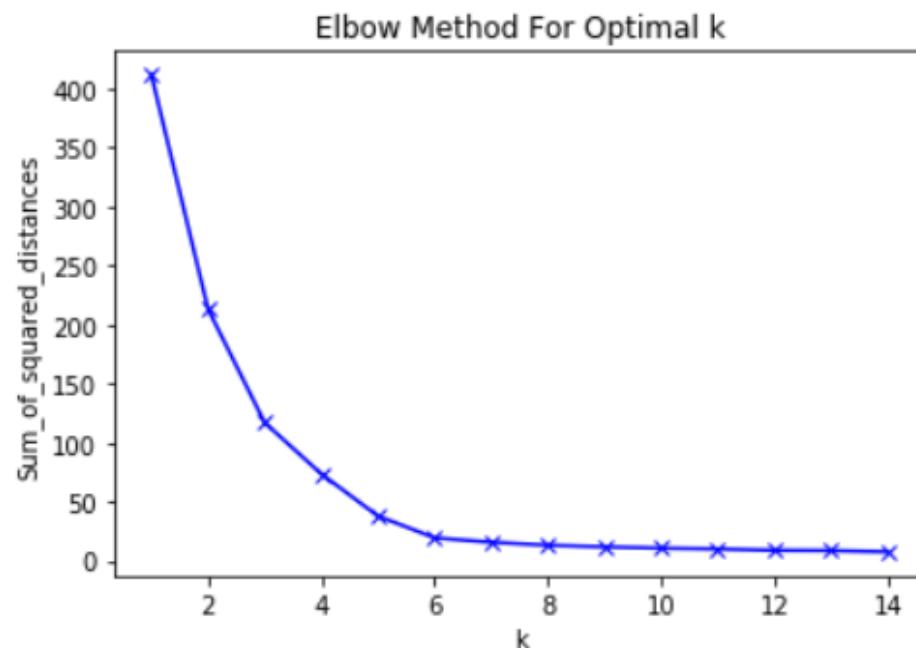
# K-means: "Elbow" Method

- Empirical method to figure out the right number K of clusters
- Trade-off between total benefit and total cost
- Try multiple values of K and look at the change of the SSD



# K-means: "Elbow" Method

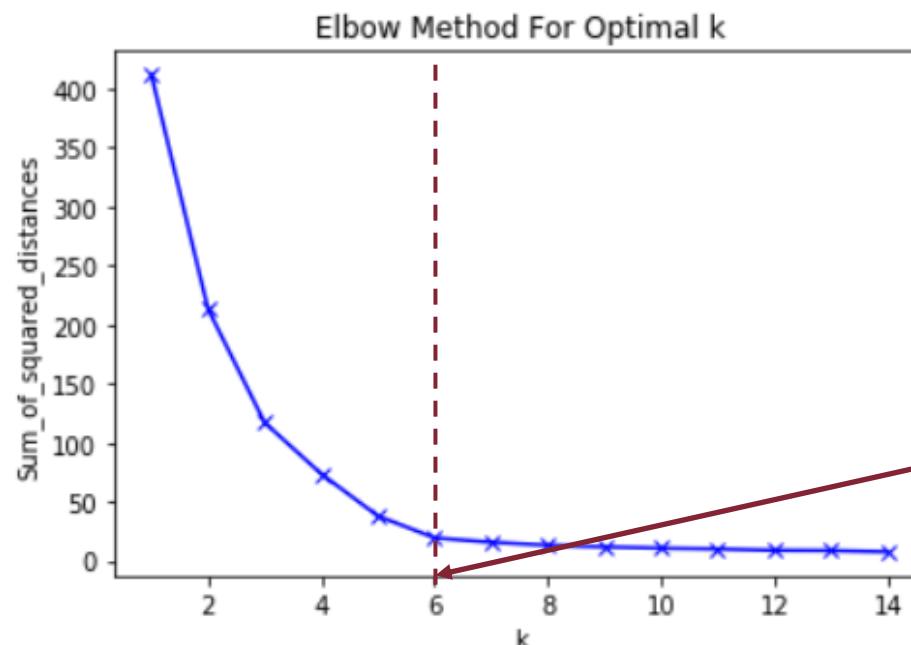
- Empirical method to figure out the right number K of clusters
- Trade-off between total benefit and total cost
- Try multiple values of K and look at the change of the SSD



As K increases, SSD sharply decreases

# K-means: "Elbow" Method

- Empirical method to figure out the right number K of clusters
- Trade-off between total benefit and total cost
- Try multiple values of K and look at the change of the SSD



As K increases, SSD sharply decreases

up to a certain value

# Non-Euclidean Distances

- So far, we have focused on **Euclidean distance** (i.e.,  $\delta = L^2$ -Norm)

# Non-Euclidean Distances

- So far, we have focused on **Euclidean distance** (i.e.,  $\delta = L^2$ -Norm)
- The same hard clustering framework can be used with other  $\delta$

# Non-Euclidean Distances

- So far, we have focused on **Euclidean distance** (i.e.,  $\delta = L^2$ -Norm)
- The same hard clustering framework can be used with other  $\delta$
- Some of them just resemble Euclidean distance, and centroids (i.e., means) still minimize those
  - $\delta = \text{Cosine distance}$  = Euclidean distance on normalized input points
  - $\delta = \text{Correlation}$  = Euclidean distance on standardized input points

# Non-Euclidean Distances

- So far, we have focused on **Euclidean distance** (i.e.,  $\delta = L^2$ -Norm)
- The same hard clustering framework can be used with other  $\delta$
- Some of them just resemble Euclidean distance, and centroids (i.e., means) still minimize those
  - $\delta = \text{Cosine distance}$  = Euclidean distance on normalized input points
  - $\delta = \text{Correlation}$  = Euclidean distance on standardized input points
- Others, require specific minimizers
  - $\delta = \text{Manhattan distance}$  ( $L^1$ -Norm)  $\rightarrow$  median is the minimizer (**K-medians**)

# Alternative Formulations: K-medoids

- Similar to K-means yet chooses input data points as centers (**medoids**)

# Alternative Formulations: K-medoids

- Similar to K-means yet chooses input data points as centers (**medoids**)
- A medoid is the closest object to any other point in the cluster

# Alternative Formulations: K-medoids

- Similar to K-means yet chooses input data points as centers (**medoids**)
- A medoid is the closest object to any other point in the cluster
- Works with **any arbitrary distance**  $\delta$

# Alternative Formulations: K-medoids

- Similar to K-means yet chooses input data points as centers (**medoids**)
- A medoid is the closest object to any other point in the cluster
- Works with **any arbitrary distance**  $\delta$
- **PAM** (**P**artitioning **A**round **M**edoids) greedy Algorithm, introduced by Kaufman and Rousseeuw in 1987 [[paper](#)] vs. Lloyd-Forgy

# Alternative Formulations: K-medoids

- Similar to K-means yet chooses input data points as centers (**medoids**)
- A medoid is the closest object to any other point in the cluster
- Works with **any arbitrary distance**  $\delta$
- **PAM** (**P**artitioning **A**round **M**edoids) greedy Algorithm, introduced by Kaufman and Rousseeuw in 1987 [[paper](#)] vs. Lloyd-Forgy
- Robust to outliers yet computationally expensive  $O(K(N-K)^2)$

# Bradley-Fayyad-Reina (BFR) K-means

- A variant of K-means explicitly thought for large datasets

# Bradley-Fayyad-Reina (BFR) K-means

- A variant of K-means explicitly thought for large datasets
- Works better in high-dimensional Euclidean space

# Bradley-Fayyad-Reina (BFR) K-means

- A variant of K-means explicitly thought for large datasets
- Works better in high-dimensional Euclidean space
- (Strong) Assumption on the shape of clusters:
  - Normally distributed around the centroid
  - Independence between data dimensions

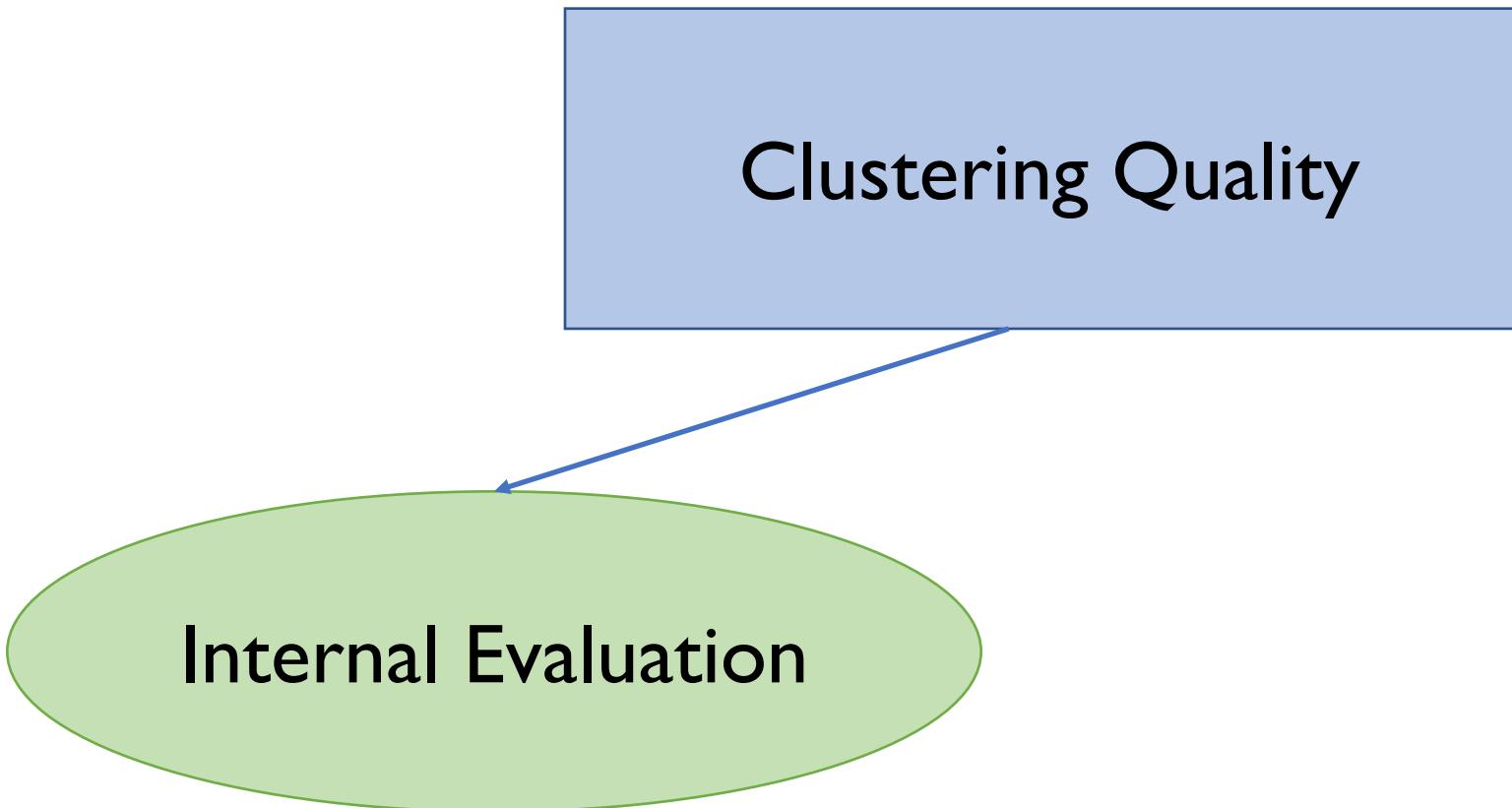
# Bradley-Fayyad-Reina (BFR) K-means

- A variant of K-means explicitly thought for large datasets
- Works better in high-dimensional Euclidean space
- (Strong) Assumption on the shape of clusters:
  - Normally distributed around the centroid
  - Independence between data dimensions
- Reference to the original [paper](#)

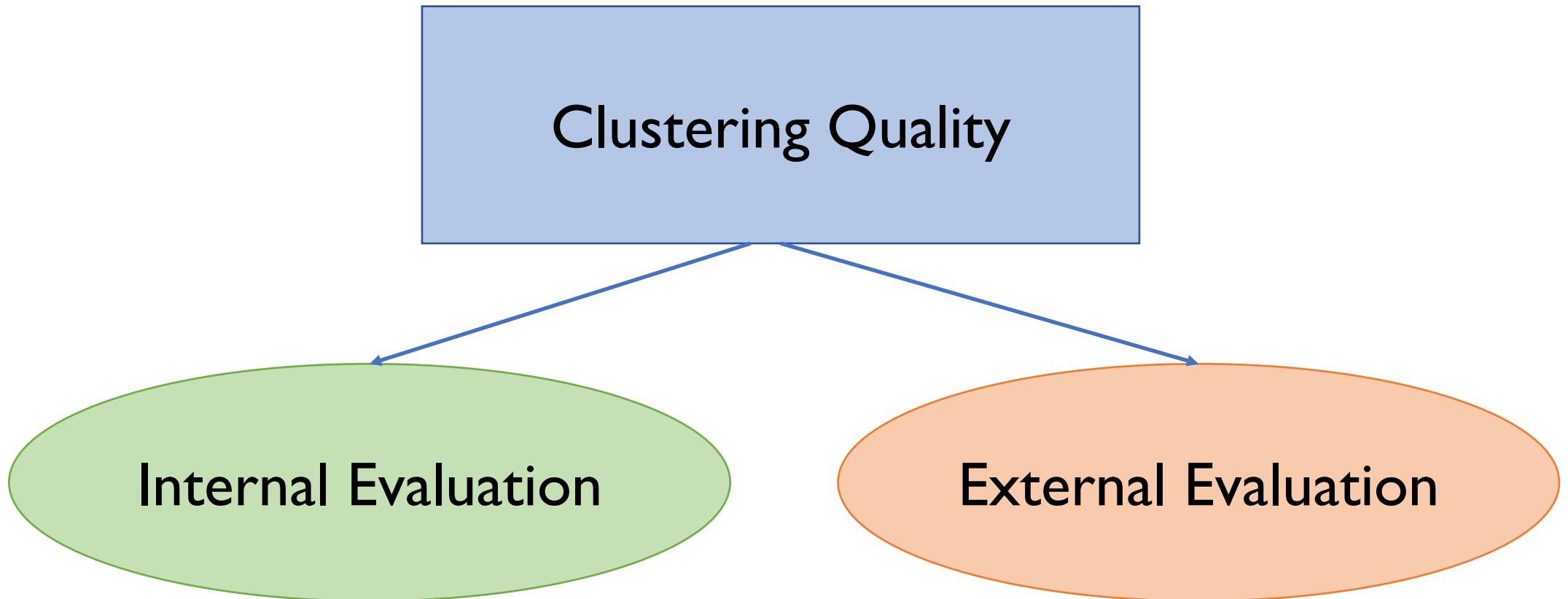
# Measures of Clustering Quality

Clustering Quality

# Measures of Clustering Quality



# Measures of Clustering Quality



# Internal Evaluation

- Clustering is evaluated based on the data that was clustered itself

# Internal Evaluation

- Clustering is evaluated based on the data that was clustered itself
- A good clustering will produce high quality clusters with:
  - **high intra-cluster similarity**
  - **low inter-cluster similarity**

# Internal Evaluation

- Clustering is evaluated based on the data that was clustered itself
- A good clustering will produce high quality clusters with:
  - **high intra-cluster similarity**
  - **low inter-cluster similarity**
- The measured quality of a clustering depends on
  - **data representation**
  - **similarity measure**

# Internal Evaluation: Davies-Bouldin Index

$$DB = \frac{1}{K} \sum_{i=1}^K \max_{j \neq i} \left( \frac{\sigma_i + \sigma_j}{\delta(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)} \right)$$

$K$  = number of clusters

$\boldsymbol{\mu}_k$  = centroid of cluster  $C_k$

$\sigma_k$  = avg. distance of all elements of cluster  $C_k$  from its centroid  $\boldsymbol{\mu}_k$

$\delta(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$  = distance between centroids of  $C_i$  and  $C_j$

The smaller the better

# Internal Evaluation: Dunn Index

$$D = \frac{\min_{1 \leq i < j \leq K} \delta(C_i, C_j)}{\max_{1 \leq k \leq K} \delta'(C_k)}$$

$K$  = number of clusters

$\delta(C_i, C_j)$  = distance between cluster  $C_i$  and  $C_j$

$\delta'(C_k)$  = intra-cluster distance of cluster  $C_k$

Distance between centroids

Max distance between any pair of objects

The higher the better

# Internal Evaluation: Silhouette Coefficient

mean distance between  $i$  and all other data points in the same cluster  $C_i$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, j \neq i} \delta(i, j)$$

smallest mean distance of  $i$  to all points in any other cluster  $C_k \neq C_i$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} \delta(i, j)$$

$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases}$$

The higher the better

# External Evaluation

- Clustering is evaluated based on data that was not used for clustering, yet pre-classified (**gold standard** data)

# External Evaluation

- Clustering is evaluated based on data that was not used for clustering, yet pre-classified (**gold standard** data)
- Quality measured by the ability to discover some or all of the hidden patterns in gold standard data

# External Evaluation

- Clustering is evaluated based on data that was not used for clustering, yet pre-classified (**gold standard** data)
- Quality measured by the ability to discover some or all of the hidden patterns in gold standard data
- Hard as it requires labeled data typically provided by human experts

# External Evaluation: Purity

$C_1 \dots, C_K$  = set of  $K$  clusters

$L_1 \dots, L_J$  = set of  $J$  labels

$n_{i,j}$  = number of items with label  $L_j$  clustered in  $C_i$

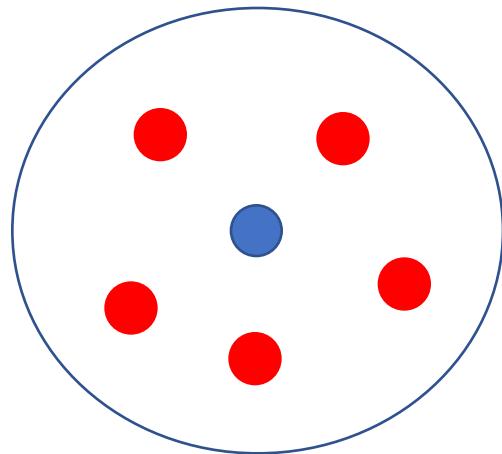
$n_i = \sum_{j=1}^J n_{i,j}$  number of items clustered in  $C_i$

$$\text{purity}(C_i) = \frac{1}{n_i} \max_{j \in \{1, \dots, J\}} n_{i,j}$$

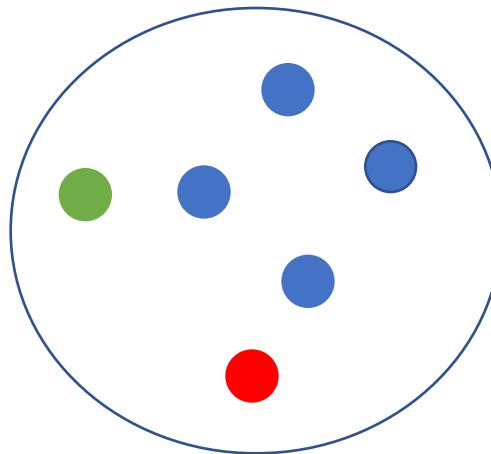
$$\text{purity} = \frac{1}{K} \sum_{i=1}^K \text{purity}(C_i)$$

Biased because having as many clusters as items maximizes purity

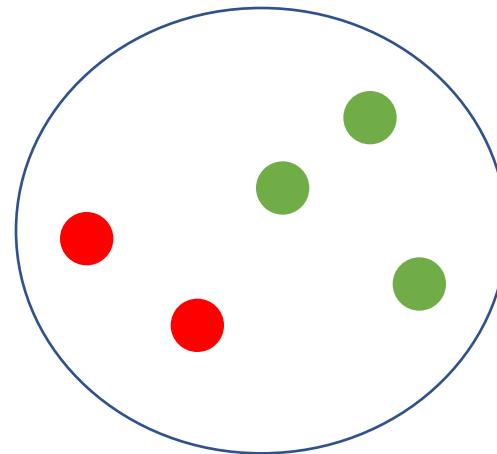
# External Evaluation: Purity Example



$C_1$



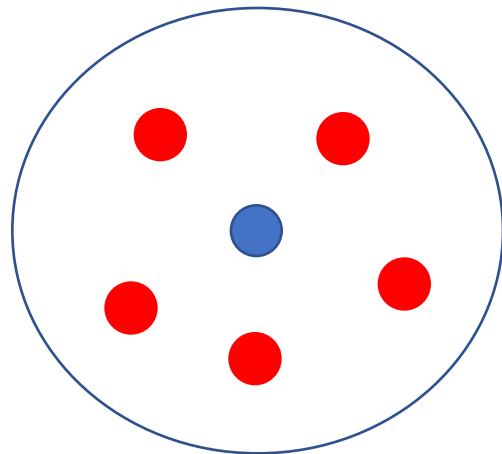
$C_2$



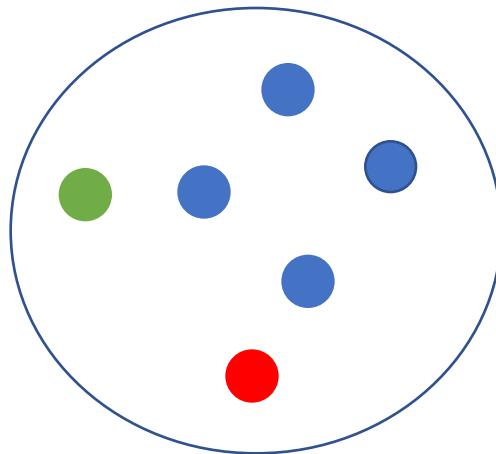
$C_3$

●  $L_1$  ●  $L_2$  ●  $L_3$

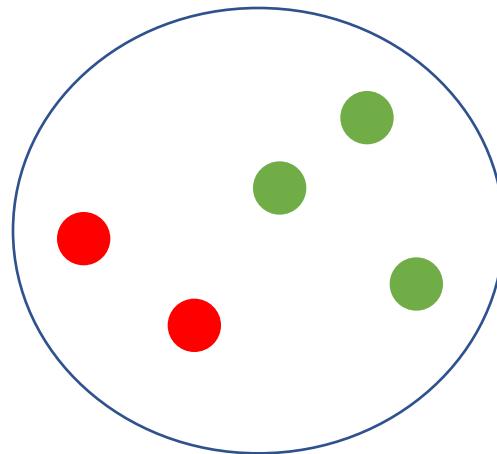
# External Evaluation: Purity Example



$C_1$



$C_2$

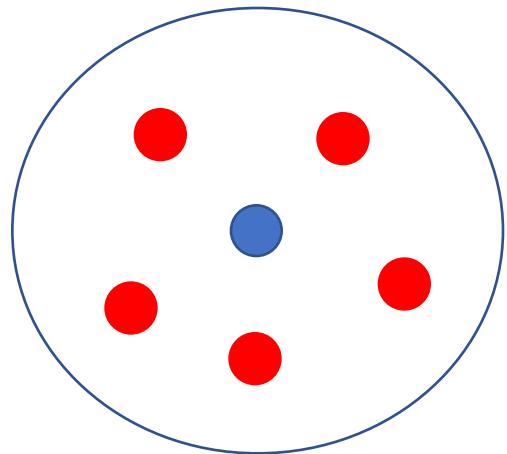


$C_3$

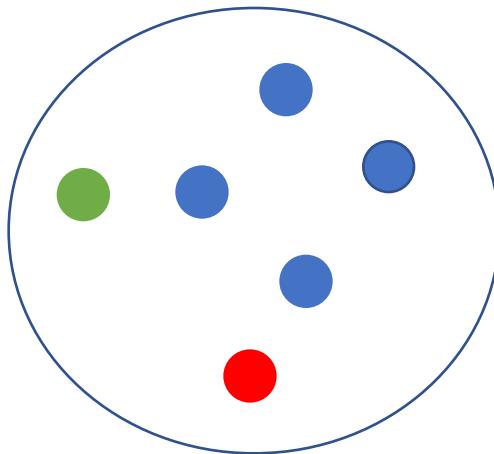


$$\text{purity}(C_1) = \frac{1}{6} * \max\{5, 1, 0\} = \frac{5}{6}$$

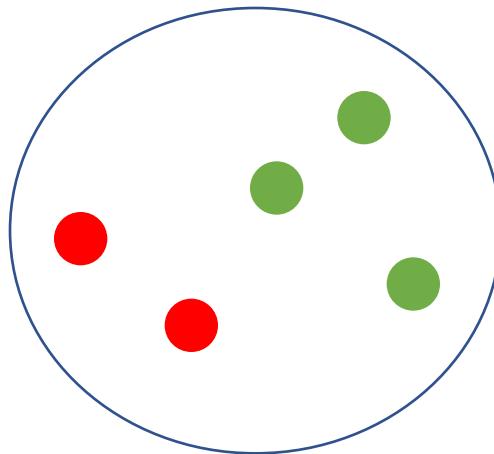
# External Evaluation: Purity Example



$C_1$



$C_2$



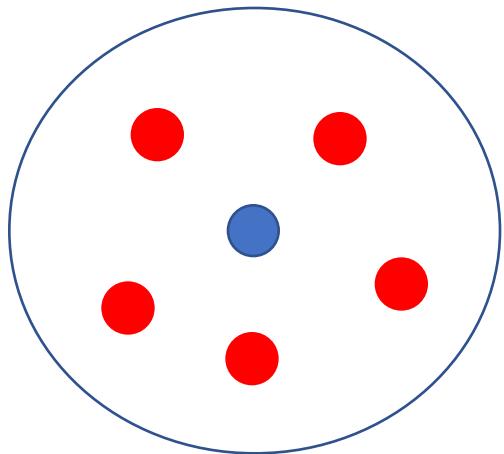
$C_3$



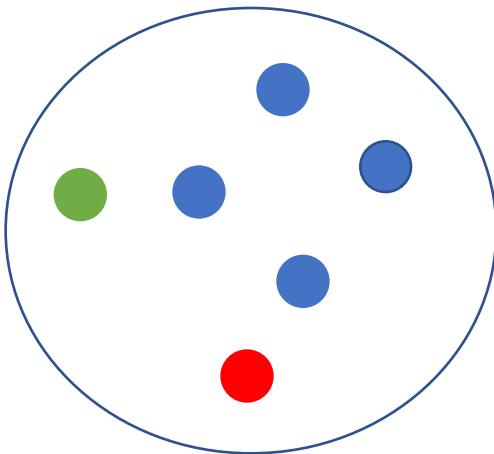
$$\text{purity}(C_1) = 1/6 * \max\{5, 1, 0\} = 5/6$$

$$\text{purity}(C_2) = 1/6 * \max\{1, 4, 1\} = 4/6 = 2/3$$

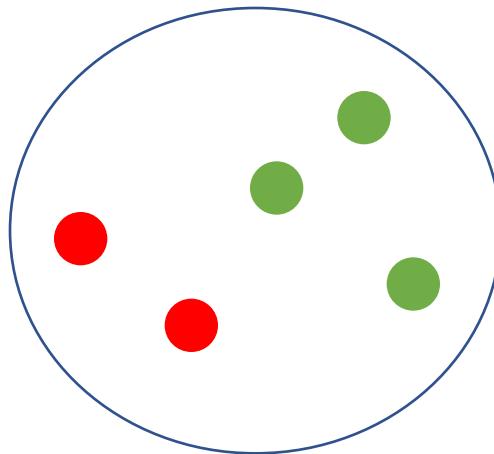
# External Evaluation: Purity Example



$C_1$



$C_2$



$C_3$

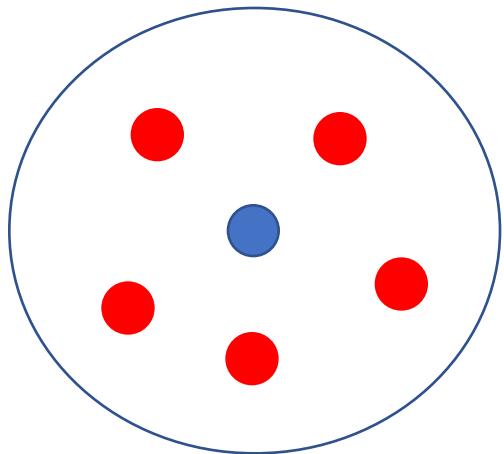


$$\text{purity}(C_1) = \frac{1}{6} * \max\{5, 1, 0\} = \frac{5}{6}$$

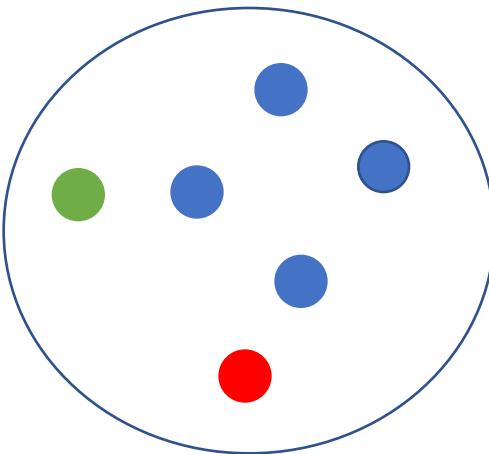
$$\text{purity}(C_2) = \frac{1}{6} * \max\{1, 4, 1\} = \frac{4}{6} = \frac{2}{3}$$

$$\text{purity}(C_3) = \frac{1}{5} * \max\{2, 0, 3\} = \frac{3}{5}$$

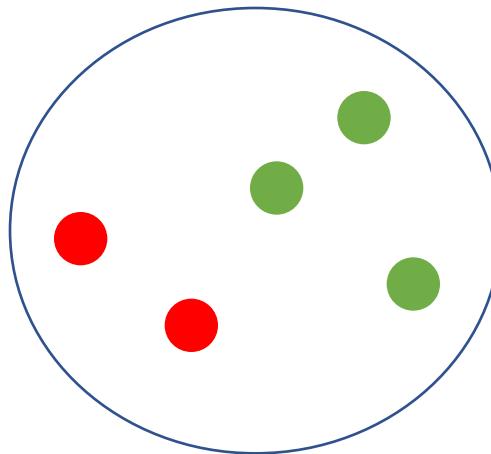
# External Evaluation: Purity Example



$C_1$



$C_2$



$C_3$

●  $L_1$  ●  $L_2$  ●  $L_3$

$$\text{purity}(C_1) = 1/6 * \max\{5, 1, 0\} = 5/6$$

$$\text{purity}(C_2) = 1/6 * \max\{1, 4, 1\} = 4/6 = 2/3$$

$$\text{purity}(C_3) = 1/5 * \max\{2, 0, 3\} = 3/5$$

**purity =  $1/3 * \text{purity}(C_1) + \text{purity}(C_2) + \text{purity}(C_3) = 7/10$**

# External Evaluation: Rand Index

$$\text{Rand} = \frac{TP + TN}{TP + TN + FP + FN}$$

$TP$  = number of *true positives*

$TN$  = number of *true negatives*

$FP$  = number of *false positives*

$FN$  = number of *false negatives*

All computed from **pairs** of elements

Measures the level of agreement between  
clustering and ground truth

# External Evaluation: Rand Index

n. of pairs	Same Cluster in Clustering	Different Clusters in Clustering
Same Cluster in Ground-Truth	TRUE POSITIVES (TP)	FALSE NEGATIVES (FN)
Different Clusters in Ground-Truth	FALSE POSITIVES (FP)	TRUE NEGATIVES (TN)

Confusion Matrix

# External Evaluation: Precision, Recall, F-measure

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN}$$

$$F_{\beta} = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R}$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

Balances the contribution of false negatives  
by weighting recall through a parameter  $\beta$

# External Evaluation: Many Other Measures

- Jaccard index
- Dice index
- Fowlkes-Mallows index
- Mutual information
- etc.

# Take-Home Message of Today

- Formulate clustering as a (**non-convex**) optimization problem
  - Focus on flat partitioning (hard)

# Take-Home Message of Today

- Formulate clustering as a (**non-convex**) optimization problem
  - Focus on flat partitioning (hard)
- Computing exact solution is **NP-hard** due to exponential search space

# Take-Home Message of Today

- Formulate clustering as a (**non-convex**) optimization problem
  - Focus on flat partitioning (hard)
- Computing exact solution is **NP-hard** due to exponential search space
- Iterative (**approximated**) methods converge to local minimum
  - Lloyd-Forgy Algorithm for **K-means** (to minimize Euclidean-related distances)
  - PAM Algorithm for **K-medoids** (to minimize any distance function)

# Take-Home Message of Today

- Formulate clustering as a (**non-convex**) optimization problem
  - Focus on flat partitioning (hard)
- Computing exact solution is **NP-hard** due to exponential search space
- Iterative (**approximated**) methods converge to local minimum
  - Lloyd-Forgy Algorithm for **K-means** (to minimize Euclidean-related distances)
  - PAM Algorithm for **K-medoids** (to minimize any distance function)
- Internal vs. External measures of **clustering quality**