50.007 Machine Learning

2026 Spring

6. Clustering: k-Medoids & Practical Issues

Na Zhao
Assistant Professor, ISTD

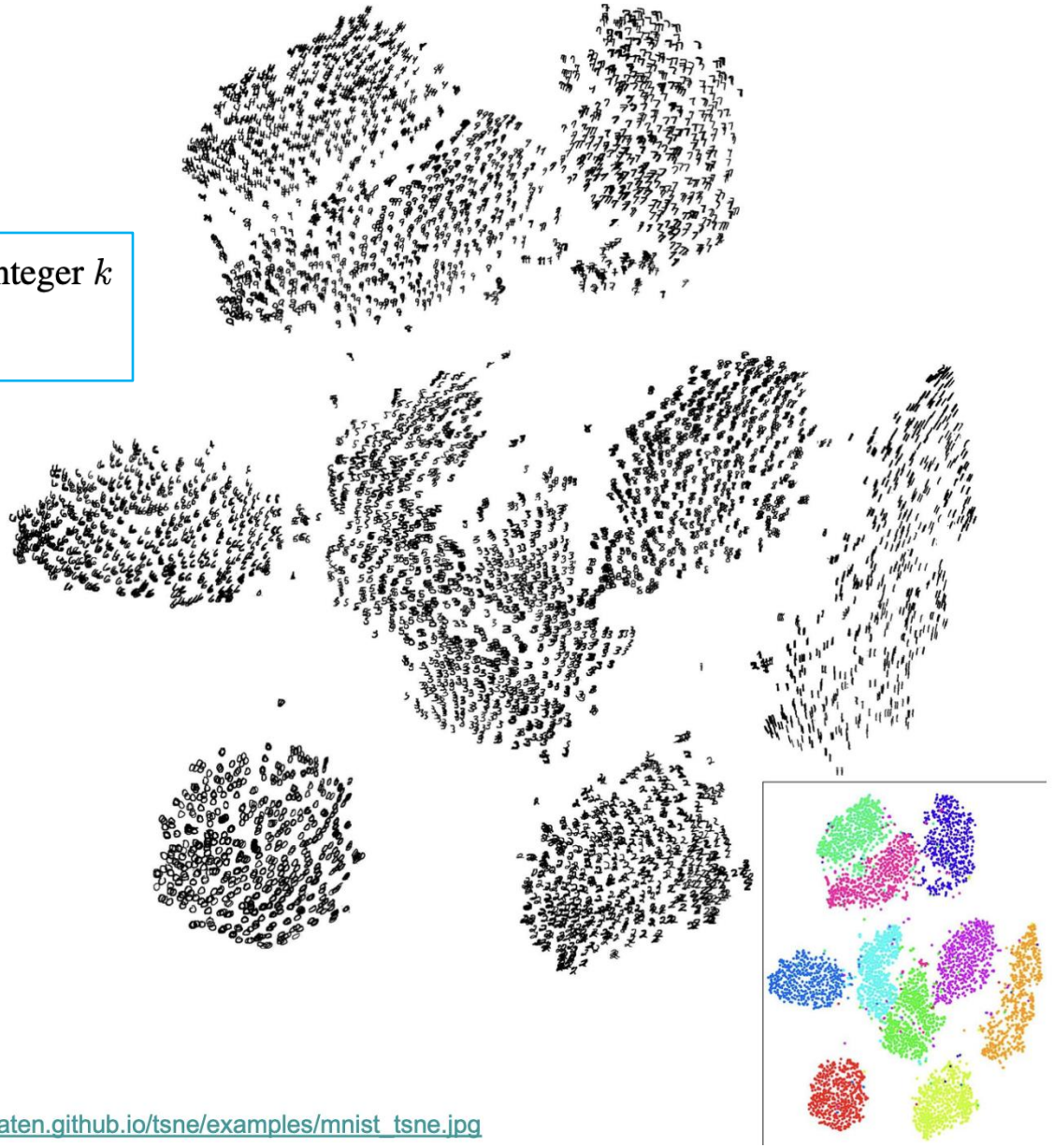
SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Recap - Clustering

Input: Training set $S_n = \{x^{(i)}; i = 1, \ldots, n\}$, where $x^{(i)} \in \mathcal{R}^d$, integer k

Output: A set of clusters $C_1, \ldots, C_k$.

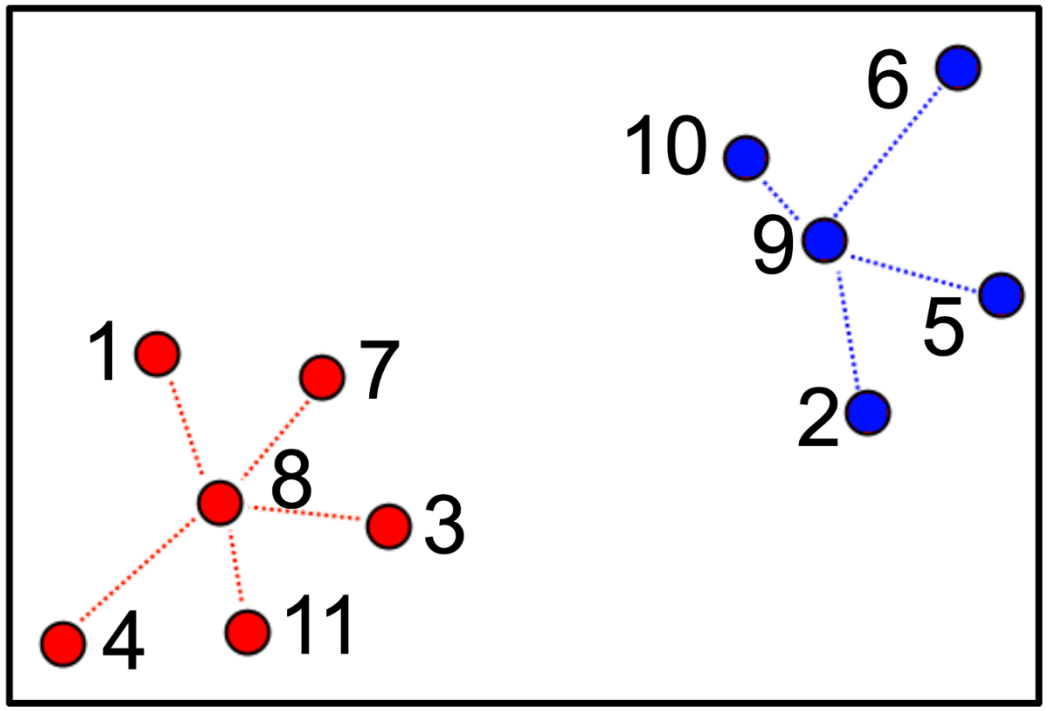
- **Clusters**:
  - – subgroups or subpopulations in the data
- **Goals**:
  - – Discovering the subgroups
  - – Estimating which subgroup a data point belongs to



http://lvdmaaten.github.io/tsne/examples/mnist_tsne.jpg

Recap - How to specify a Cluster

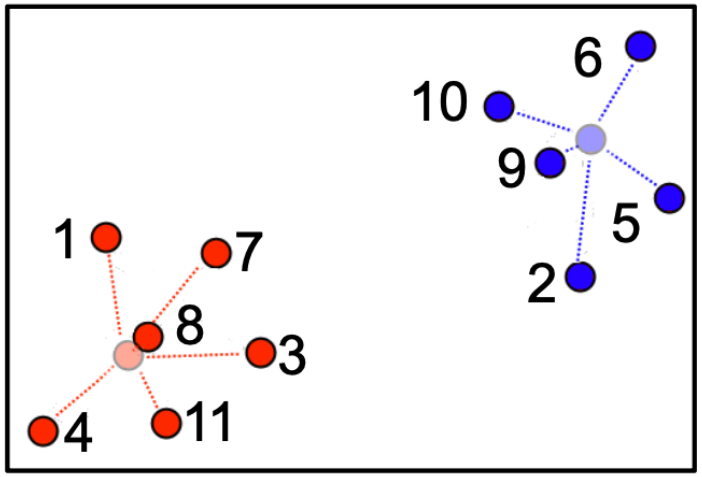
- By listing all its elements
  - $\mathcal{C}1$ = {1,3,4,7,8,11 1}
  - $\mathcal{C}2$ = {2,5,6,9,10}



Recap - How to specify a Cluster

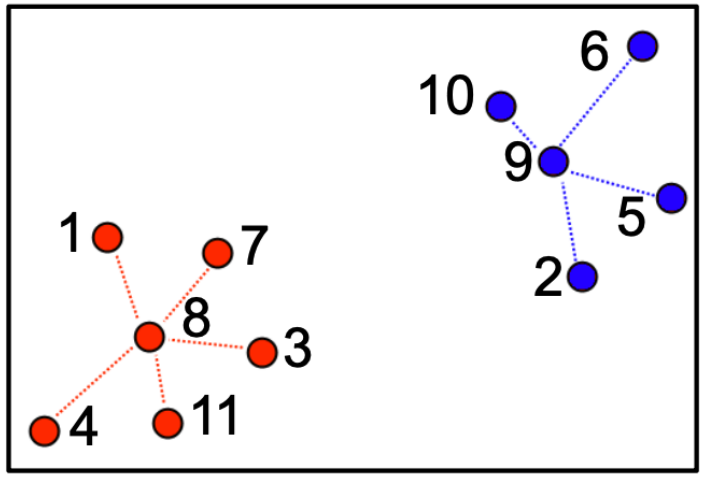
- Using a representative
  a) A point in center of cluster (e.g., centroid, median, mode)
  b) A point in the training data (exemplar)

$$z^{(1)} = \begin{pmatrix}1\\1\end{pmatrix}, z^{(2)} = \begin{pmatrix}5\\4\end{pmatrix}$$



centroid

$$z^{(1)} = 8, z^{(2)} = 9$$



exemplar

Recap - Cost Function

- Once we have the distance metric, we can specify an objective function for clustering:

$$Cost(C^1, \ldots, C^k, z^{(1)}, \ldots, z^{(k)}) = \sum_{j=1}^{k} \sum_{i \in C^j} d(x^{(i)}, z^{(j)})$$

  - k: the number of clusters (*pre-defined*)
  - C^j: the j-th cluster
  - z^j: the representative of a cluster
  - $d(x^i, z^j)$: the distance metric that measures the pairwise distance between i-th sample in C^j and z^j

Centroid: k-means

- The cost here depends on both the clusters and how the representatives are chosen for each cluster, but we can derive one from another.

Learning Objectives

You should be able to know:

1. What is k-medoids algorithm and how is it different from k-means?
2. Why does initialization of the k-means algorithm matter and what can you do?
3. How to determine the k used in the k-means algorithm?

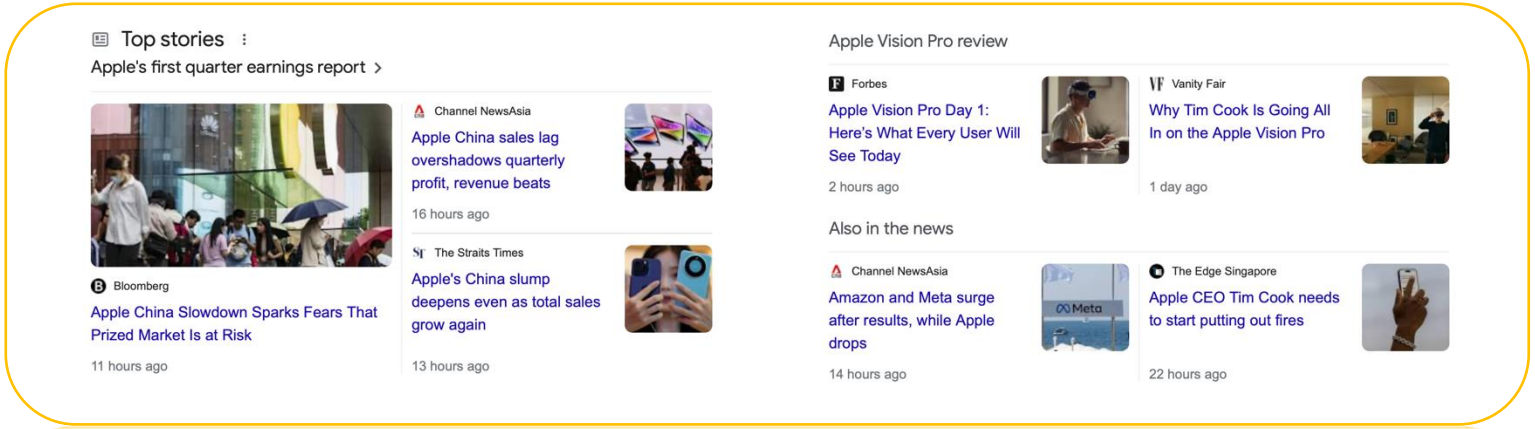
k-Medoids Clustering

k-Medoids Clustering

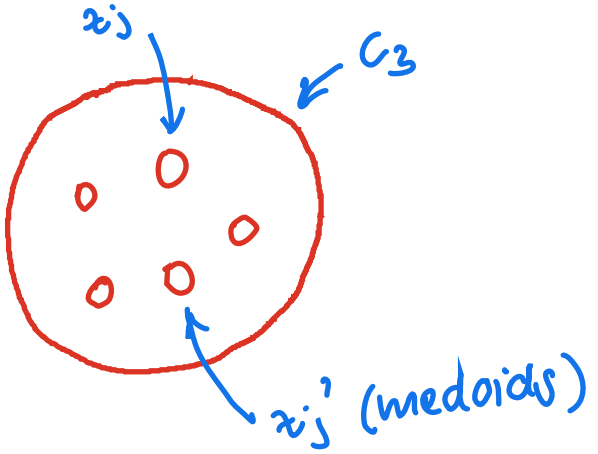
- ***k*-medoids** shares a similar objective as k-means but,
  - it chooses actual data points as representatives, known as **exemplars** or **medoids**.
  - It can be used with arbitrary dissimilarity measures.

- Why need k-medoids clustering?
  - Selecting exemplars as representatives can be *important* in some applications:

    E.g. **Google news**: a single article is used to represent a news cluster.

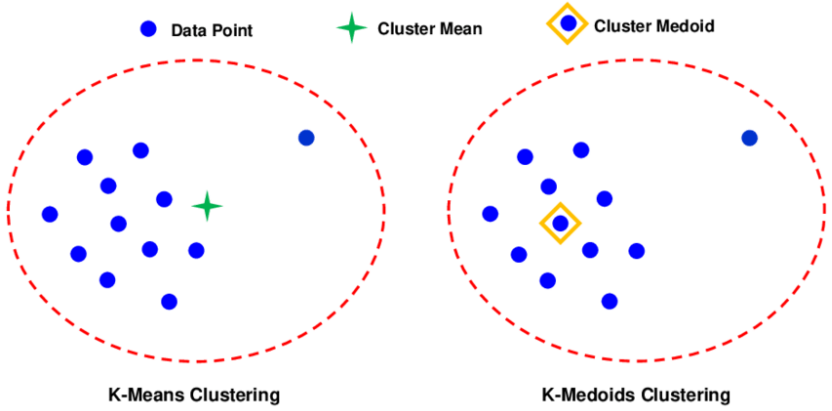


k-Medoids Clustering



- ***k*-medoids** shares a similar objective as k-means but,
  - it chooses actual data points as representatives, known as **exemplars** or **medoids**.
  - It can be used with arbitrary dissimilarity measures.

- Why need k-medoids clustering?
  - Selecting exemplars as representatives can be *important* in some applications:
    E.g. **Google news**: a single article is used to represent a news cluster.
  - We can also use non-Euclidean distance metrics in k-medoids
  - k-medoids is less sensitive to outliers than k-means
    - As the medoid is less influenced by individual data points than the cluster centroid



k-Medoids Solution-1: Lloyd's Algorithm (*alternate*)

- We can also use Lloyd's algorithm to solve ***k*-medoids** clustering, which alternatingly finds:
  - best clusters for exemplars (*assign data points to exemplars*)
  - and best exemplars for clusters (*find the data point results in a lower cost*)

1. Initialize exemplars: $\{z^{(1)}, \ldots, z^{(k)}\} \subseteq \{x^{(1)}, \ldots, x^{(n)}\}$ (exemplars are k points from the original dataset)
2. Repeat until there is no further change in cost:
   (a) for each $j : C^j = \{i : x^{(i)}$'s closest exemplar is $z^{(j)}\}$
   (b) for each j : set $z^{(j)}$ to be the point in C^j that minimizes $\sum_{i \in C^j} d(x^{(i)}, z^{(j)})$

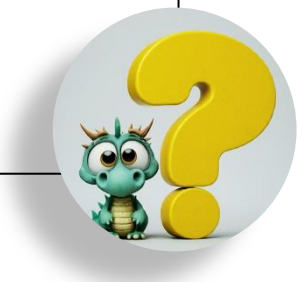
O(n*k)

K × |C_k|²

for each sample calculate diff

exemplar

We consider each point in turn as a candidate exemplar and compute the associated cost; the point that produces the minimum cost is chosen as the new exemplar.

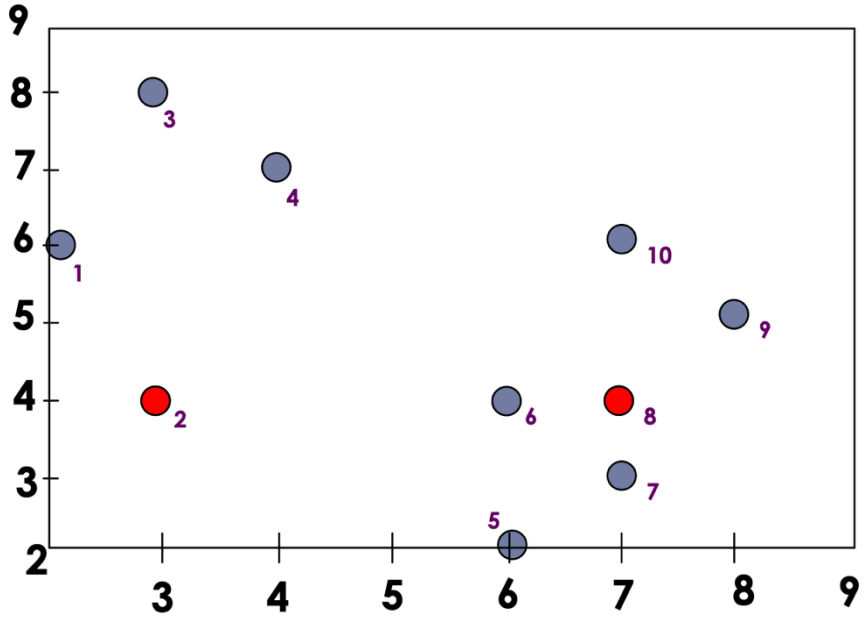


Time Complexity?
Any other problem?

Exercise - k-Medoids Clustering

$k = 2$

- Consider the following set of points



$x^{(1)}$	2	6
$x^{(2)}$	3	4
$x^{(3)}$	3	8
$x^{(4)}$	4	7
$x^{(5)}$	6	2
$x^{(6)}$	6	4
$x^{(7)}$	7	3
$x^{(8)}$	7	4
$x^{(9)}$	8	5
$x^{(10)}$	7	6

z_1

z_2

① randomly select point
② initialise z_1

- We will use L1 distance metric (Manhattan distance):

$$d\left(x^{(i)}, z^{(j)}\right) = |x^{(i)} - z^{(j)}|$$

manhattan dist

Exercise - k-Medoids Clustering

- Let the randomly selected 2 medoids/exemplars be

$$z^{(1)} = (3,4)$$
$$z^{(2)} = (7,4)$$

} random selection

- Calculate the cost of each non-medoid point with the medoids.

Data object		Distance to	
i	$x^{(i)}$	$z^{(1)} = (3,4)$	$z^{(2)} = (7,4)$
1	(2, 6)	3	7
2	(3, 4)	0	4
3	(3, 8)	4	8
4	(4, 7)	4	6
5	(6, 2)	5	3
6	(6, 4)	3	1
7	(7, 3)	5	1
8	(7, 4)	4	0
9	(8, 5)	6	2
10	(7, 6)	6	2
Cost			

Exercise - k-Medoids Clustering

- Let the randomly selected 2 medoids/exemplars be

$$z^{(1)} = (3,4)$$
$$z^{(2)} = (7,4)$$

- Calculate the cost of each non-medoid point with the medoids.

- Assign points to the the closest medoid and calculate the total loss of this clustering.

  Cluster 1: (3+0+4+4) = 11

  Cluster 2: (3+1+1+0+2+2) = 9

Data object		Distance to	
i	$x^{(i)}$	$z^{(1)} = (3,4)$	$z^{(2)} = (7,4)$
1	(2, 6)	**3**	7
2	(3, 4)	**0**	4
3	(3, 8)	**4**	8
4	(4, 7)	**4**	6
5	(6, 2)	5	**3**
6	(6, 4)	3	**1**
7	(7, 3)	5	**1**
8	(7, 4)	4	**0**
9	(8, 5)	6	**2**
10	(7, 6)	6	**2**
Cost		**11**	**9**

12022026

Exercise - k-Medoids Clustering

- For each cluster, we iteratively select one non-medoid point, and swap it with the medoid.
- For example, for cluster 2, we select non-medoid point $O'=x^{(7)}$, and swap it with $z^{(2)}$

$$z^{(1)} = (3,4)$$
$$O' = (7,3)$$

- The cost of cluster 2 with new medoid O' is:

  Cluster 2: (2+2+0+1+3+3) = 11

- Calculate cost change for cluster 2: $11 - 9 > 0$

  → **It is a bad idea to swap.**

Data object		Distance to	
i	$x^{(i)}$	$z^{(1)} = (3,4)$	$z^{(2)} = (7,4)$
1	(2, 6)	**3**	7
2	(3, 4)	**0**	4
3	(3, 8)	**4**	8
4	(4, 7)	**4**	6
5	(6, 2)	5	**3**
6	(6, 4)	3	**1**
7	(7, 3)	5	**1**
8	(7, 4)	4	**0**
9	(8, 5)	6	**2**
10	(7, 6)	6	**2**
Cost		**11**	**9**

i	O'		$x^{(i)}$		dist
5	7	3	6	2	**2**
6	7	3	6	4	**2**
8	7	3	7	4	**1**
9	7	3	8	5	**3**
10	7	3	7	6	**3**

k-Medoids Solution-2: **Partitioning Around Medoids**

- **Partitioning Around Medoids (PAM)** is also an ***iterative*** process but systematically search for optimal medoid candidates within the whole data points.
- PAM consists of two parts: initialization (BUILD) and iterative improvement (SWAP)

Algorithm: PAM BUILD: Find initial cluster centers

greedily select k of the n data points as the medoids to minimize the cost

greedy search => X random select

```
1 m_1 ← point with the smallest distance sum TD to all other points
2 for i=2...k do
3 |   m_i ← point which reduces TD most
4 return TD, {m_1, ..., m_k}
```

$$TD = \sum_{i=1}^{k} \sum_{x_j \in C_i} \text{dist}(x_j, m_i)$$

form cluster based on already existing

Algorithm: PAM SWAP: Improve the clustering (abstract)

```
1 repeat
2 |   for m_i ∈ {m_1, ..., m_k} do                    // each medoid
3 |   |   for x_j ∉ {m_1, ..., m_k} do                // each non-medoid
4 |   |   |   ΔTD ← change in TD with x_j medoid instead of m_i   // incremental, in O(n)
5 |   |   |   Remember (m_i, x_j, ΔTD) for the best ΔTD
6 |   break if the best ΔTD ≥ 0 (no improvement of the clustering)
7 |   swap (m_i, x_j) of the best ΔTD
8 return TD, M, C
```

$k(n-k)^2$

$O(k(n-k)^2)$

Source: Dr. Erich Schubert @ TU Dortmund University

k-Medoids vs k-Means

❑ **Similarities**:

- Both can be solved by *iterative* based algorithm:
  - Both k-medoids and k-means use an iterative approach (*e.g.*, Lloyd's algorithm) to refine the solution and find the optimal clustering solution.
- Both algorithms require the number of clusters to be *specified in advance*:
  - In both algorithms, the user must specify the number of clusters to be formed.
- Both algorithms can be used with *Euclidean distance metrics*:
  - Both k-medoids and k-means can use Euclidean distance metrics to calculate the similarity between data points.

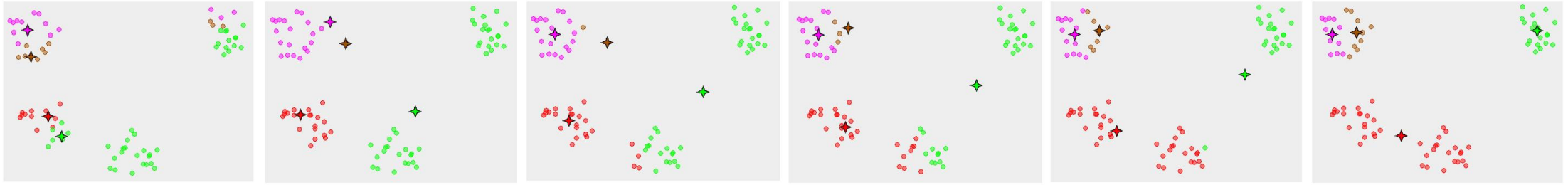
k-Medoids vs k-Means

❑ **Differences**:

- Representation of clusters:
  - k-medoids: **exemplars**; k-means: **centroids**
- Robustness to outliers:
  - k-medoids is less sensitive to outliers than k-means
- Computational cost:
  - k-medoids is computationally more expensive than k-means (thus it does not scale well for large data sets)
- Distance metric:
  - k-medoids: either Euclidean or non-Euclidean distance metrics;
  - k-means: restricted to Euclidean distance metrics

Practical Issues in k-Means

Initialization



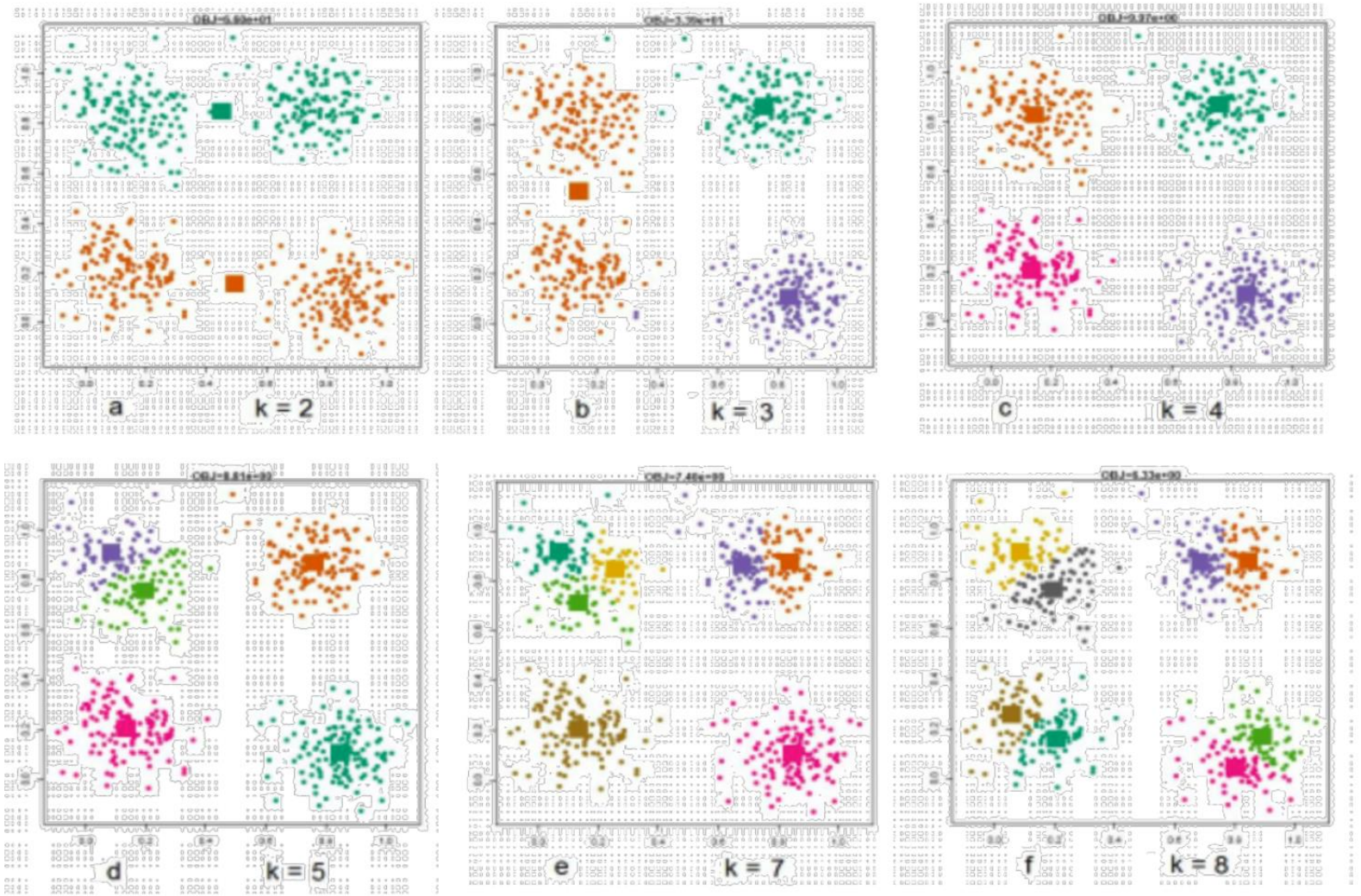
- The clustering result is *sensitive* to centroid initialization.
- A bad initialization could lead to
  - **exponential running time** in worst case
  - and **low quality final assignments** which are far away from global optimum
- In this example, the result of k-means clustering (the right figure) contradicts the obvious cluster structure of the data set.
  - Empty cluster
  - Two centroids were placed in close proximity to each other

The illustration was prepared with the Mirkes Java applet.

Initialization

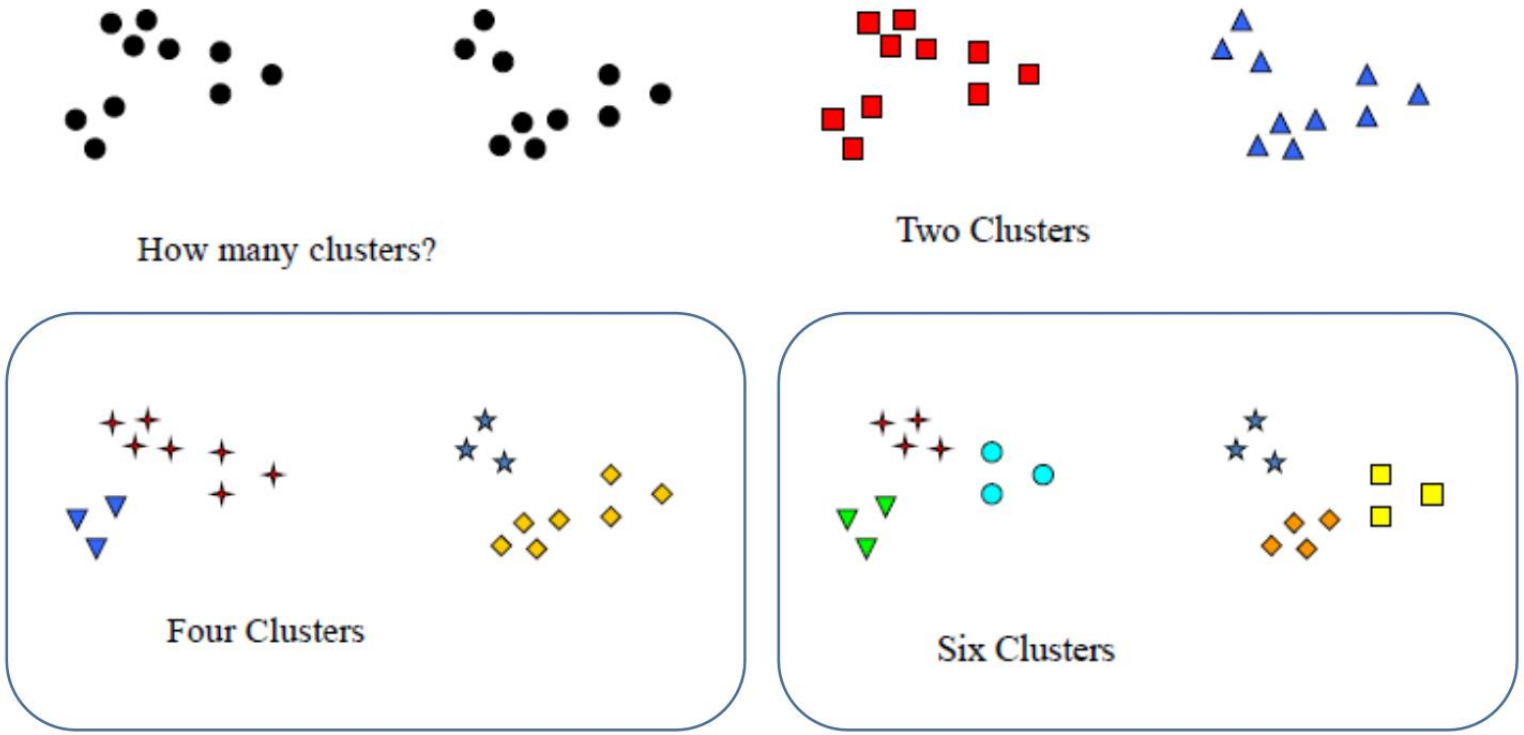
- **How to Initialize?**
  - **Forgy method**: randomly pick k training examples and use them as initial centroids.
  - **Random Partition**: first randomly assigns a cluster to each sample and then compute the cluster mean as initial centroids.
  - **k-means++** (Arthur & Vassilvitskii'07):
    - The first centroid is selected at random
    - The next centroid selected is the one that is farthest from the currently selected (selection is based on a weighted probability score)
    - The selection continues until *K* centroids are obtained
- **Solve bad local minima?**
  - Initialize many times and pick the solution with smallest training loss

Number of Clusters



- The selection of k greatly impacts the quality of clustering solution.

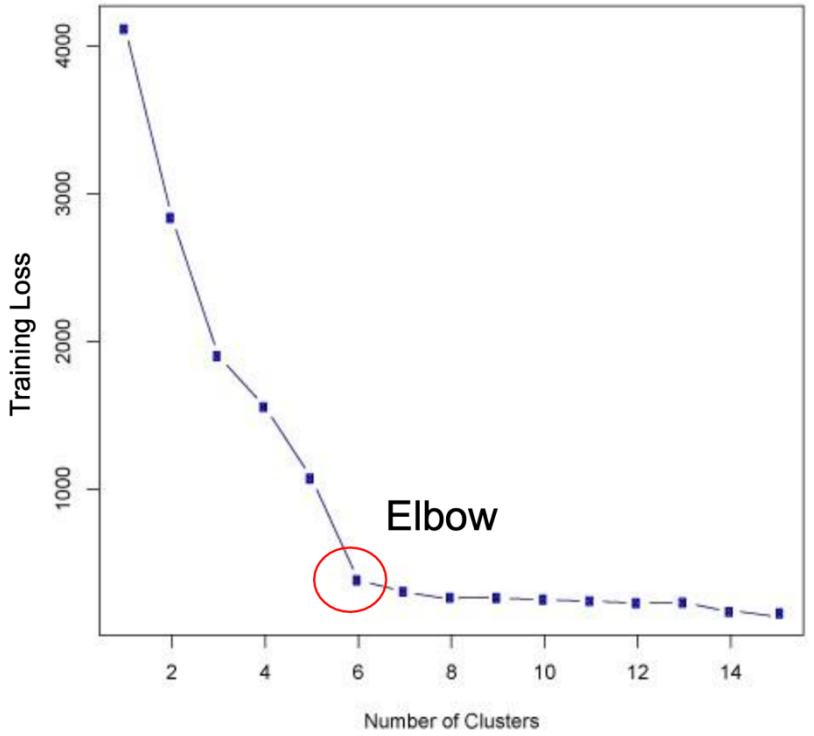
Number of Clusters



- The correct choice of k is often *ambiguous*, depending on
  - the shape and scale of the distribution of points in a data set
  - the desired clustering resolution of the user.

How to choose the optimal k?

- **Not an easy problem**: There is no definitive way to find the optimal number
- But we can perform clustering on different values of k and use several methods to estimate the clustering result:
  - **Elbow method**
  - Silhouette method
  - Information Criteria
  - Cross-validation
  - and etc..



- Elbow method is probably the most popular method to determine the optimal number of clusters.
- Though finding elbow points can be a challenge, because in practice there may not be a sharp elbow.

Summary

1. What is k-medoids algorithm and how is it different from k-means?

   k-medoids is also a *partition*-based clustering method, sharing a similar objective as k-means. However, it use actual data points as representatives and can use other distance metrics than Euclidean distance.

2. Why does initialization of the k-means algorithm matter and what can you do?

   It may result in long running time and bad clustering result.

   We can use Forgy method or k-means++ to choose initial centroids, and we can also initialize many times and pick the best solution.

3. How to determine the k used in the k-means algorithm?

   We can perform clustering on different values of k and use several techniques such as Elbow method to determine the optimal k.

Acknowledgements

- Some slides and content of this lecture are adopted from:
  - MIT 6.036 Introduction to Machine Learning
  - SUTD 50.007 Machine Learning, Spring 2023 (Asst Prof. Malika Meghjani)
  - Bishop, C. M. (2006). Pattern recognition and machine learning. Springer. (Chapter 9)