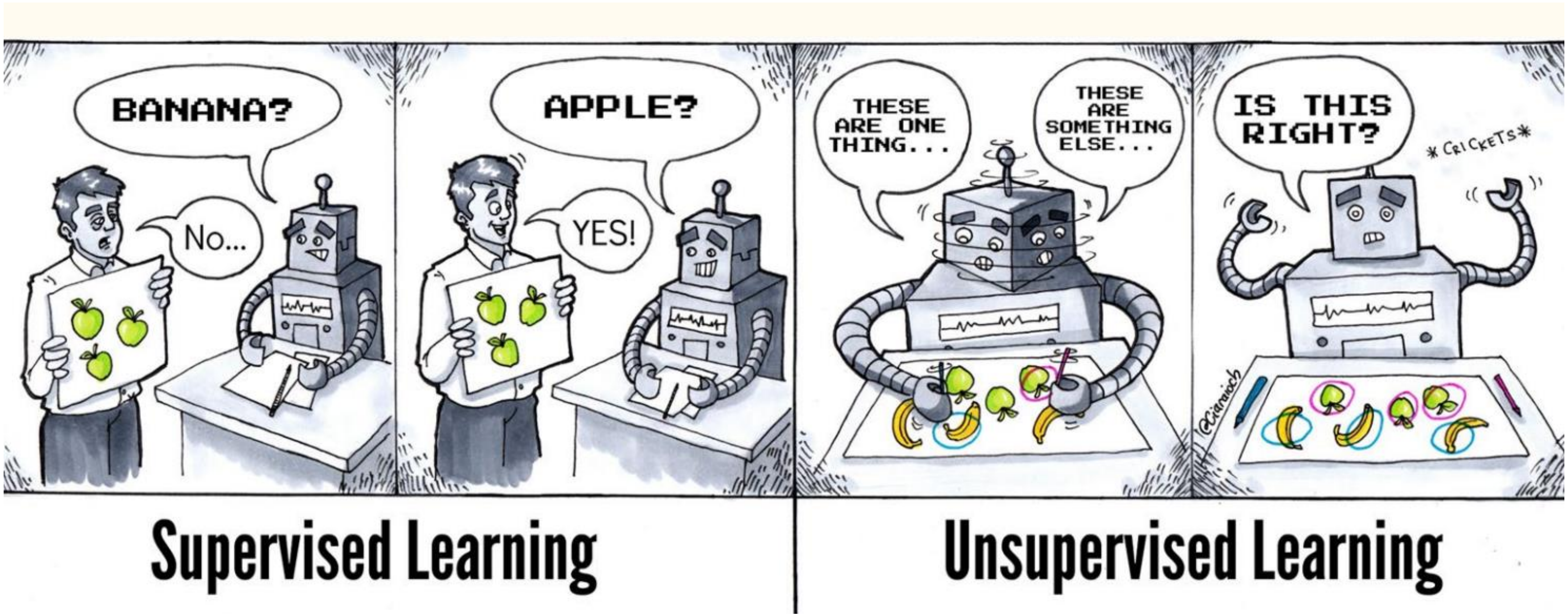
Cluster Analysis

Unsupervised Learning



Task 1 : Group These Set of Document into 3 Groups based on meaning

Doc1 : Health , Medicine, Doctor

Doc 2 : Machine Learning, Computer

Doc 3 : Environment, Planet

Doc 4 : Pollution, Climate Crisis

Doc 5 : Covid, Health , Doctor

Task 1 : Group These Set of Document into 3 Groups based on meaning

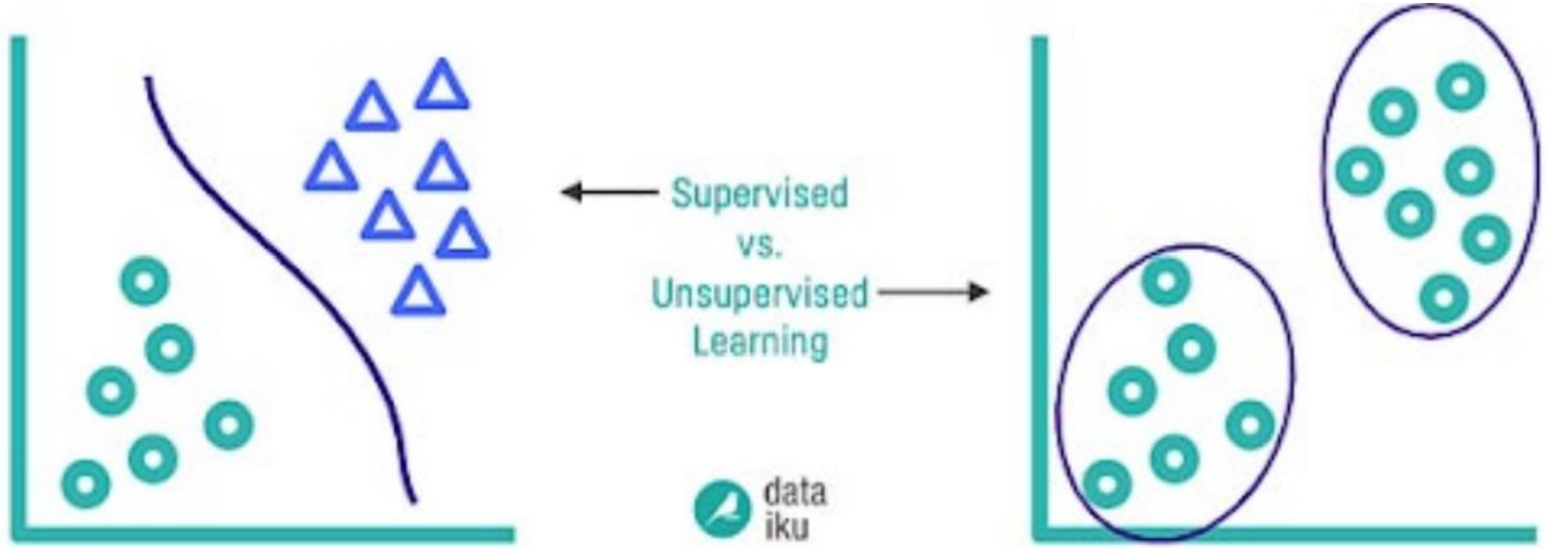
Doc1 : Health , Medicine, Doctor

Doc 5 : Covid, Health , Doctor

Doc 3 : Environment, Planet
Doc 4 : Pollution, Climate Crisis

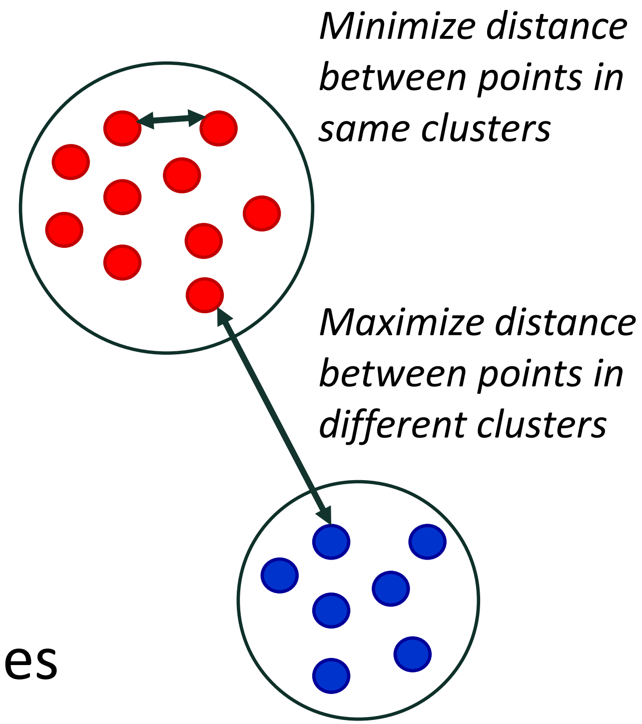
Doc 2 : Machine Learning, Computer

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering



Definition

- Cluster analysis: groups data objects based on information found in the data that describes object relationships
- Goal:
  - Objects within a group are similar/related
  - Objects in different groups are different/unrelated
- Applications:
  - Clustering for understanding
  - Clustering for utility: as a starting point for other purposes
  - Clustering for outlier detection



Applications

- Customer Relationships:
  - Divide customers into groups according to their business patterns
  - Develop campaigns to target each group specifically
- Credit Card Customers:
  - Evaluate features and profit contributions of different customers
- Information Retrieval:
  - Group similar search results together: More effective presentation for users than flat list (specifically when a term has more than one meaning):
  - Group similar documents in entire collection: Improve search results

Applications

- Summarization:
  - Techniques with high complexity such as PCA/Regression
  - Instead of applying technique to a large dataset, pick a prototype for each cluster
- Compression:
  - Vector quantization of images, audio and video data
- Nearest Neighbor:
  - Instead of computing all pairwise distances, only compute distance to prototypes
  - Rational: If an object is far from the prototype of a cluster, it is far from all points in the cluster

Clustering Examples

Original Points

Two Clusters

Four Clusters

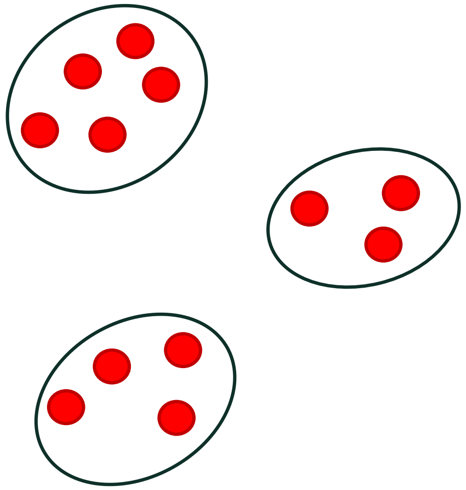
Six Clusters

Challenges

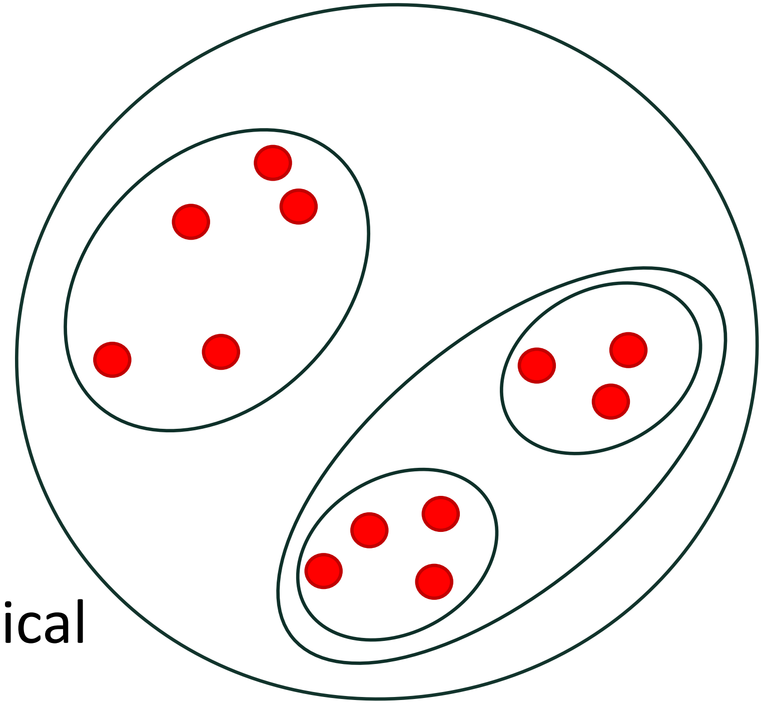
- Scalability
- Ability to deal with different types of attributes
- Ability to discover clusters with different shapes
- Ability to deal with noisy data
- Incremental/Insensitivity to input order
- Capability for dealing with high dimensions
- Ability to handle constraints
- Interpretability

Clustering Types

- Hierarchical vs Partitional:
  - **Partitional**: divides data points into non-overlapping subsets
  - **Hierarchical**: divides into nested clusters, organized as tree



Partitional



Hierarchical

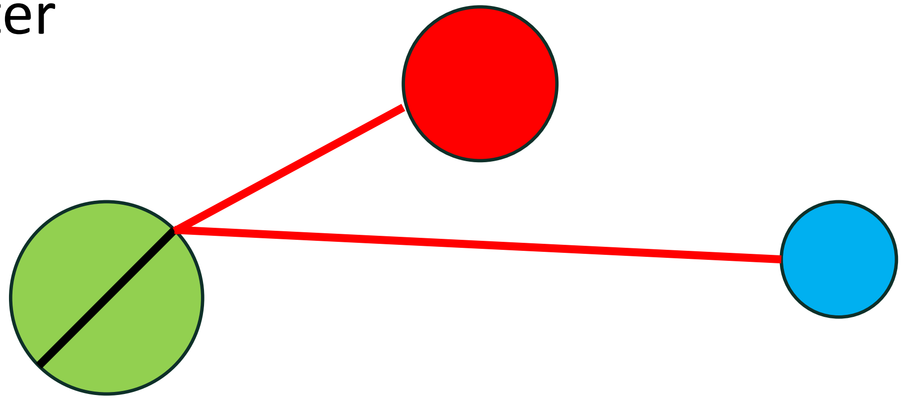
Clustering Types

- Exclusive vs Overlapping vs Fuzzy
  - **Exclusive**: each object belongs to one cluster
  - **Overlapping**: an object can simultaneously belong to multiple groups
  - **Fuzzy**: each object belongs to each cluster with a given weight between 0 (does not belong) and 1 (definitely belongs)
    - Weights must sum to 1

- Complete vs Partial
  - **Complete**: every object is assigned to a cluster
  - **Partial**: some objects (noise for example) may not be assigned to a cluster

Types of Clusters

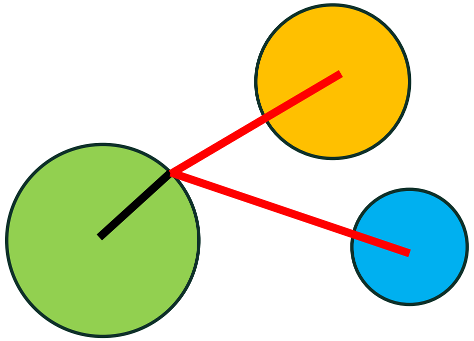
- **Well-Separated**: each object is closer to every other object in its cluster than any object in another cluster



- Sometimes a threshold is used to specify that all the objects in a cluster must sufficiently close to one another.
- Definition of a cluster is satisfied only when the data contains natural clusters. These clusters can have any shape.

Types of Clusters

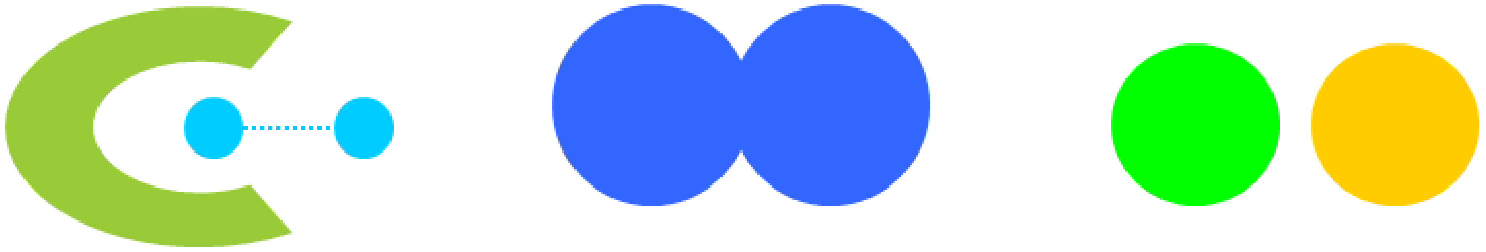
- **Prototype-Based**: each object is closer to the prototype (center) that defines the cluster than to the prototype of any other cluster



- If the data is numerical, the prototype of the cluster is often a **centroid** i.e., the average of all the points in the cluster.
- If the data has categorical attributes, the prototype of the cluster is often a **medoid** i.e., the most representative point of the cluster.
- These clusters tend to be globular (spherical shape)

Types of Clusters

- **Contiguity based**: each object is closer to some point in its cluster than any other point outside its cluster

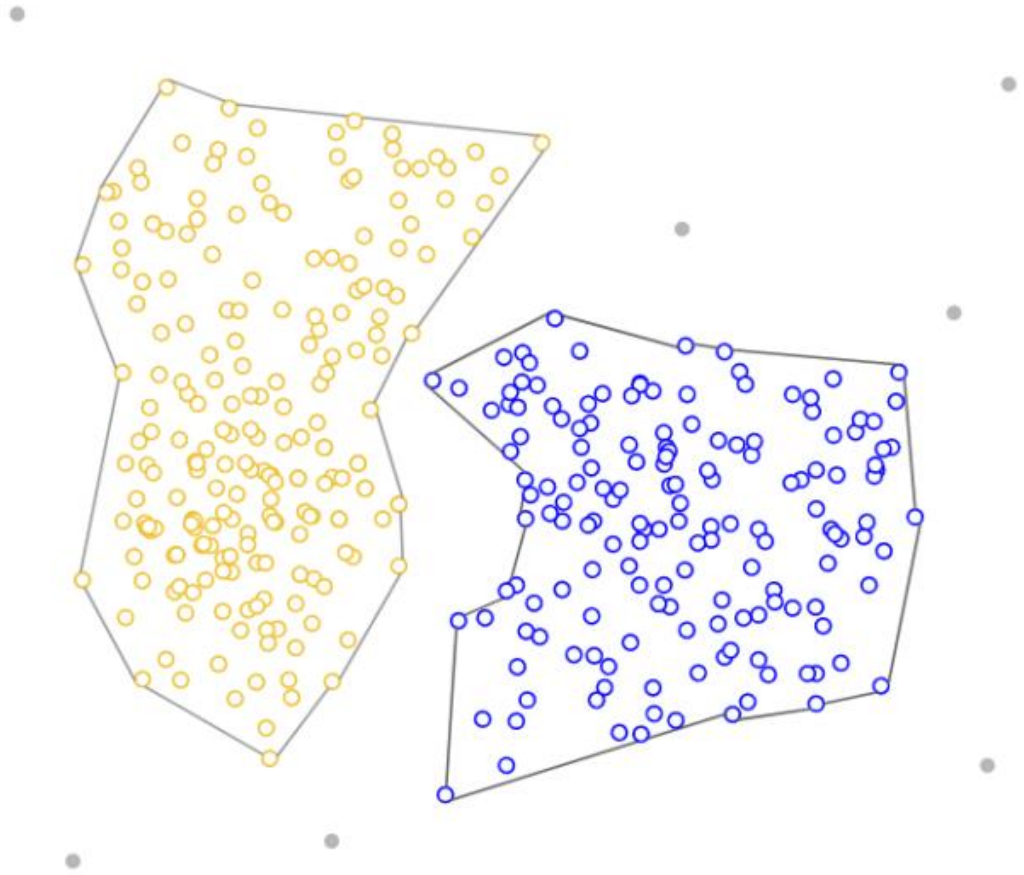


- nearest neighbor or transitive clusters
- when data is represented as a graph, a cluster is defined as a connected component which is a group of points that are connected to each other but has no connections to points outside of the group.
- 2 points are connected only if they are within a specified distance of each other

- Useful when clusters are irregular and intertwined.
- This does not work efficiently when there is noise in the data. For example, a small bridge of points can merge two distinct clusters into one.

Types of Clusters

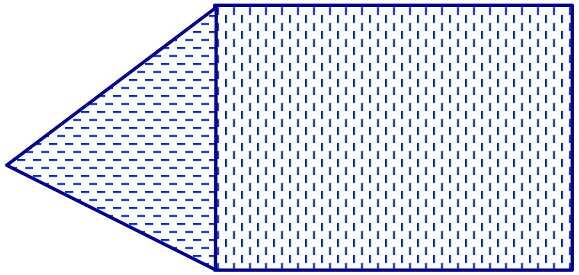
- **Density-Based**: a cluster is a dense region surrounded by a region of low density

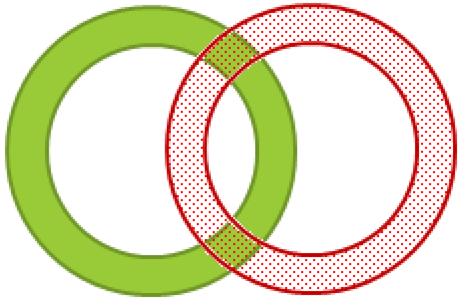


- Density based clusters are employed when the clusters are irregular, intertwined and when noise and outliers are present.
- Points in low density region are classified as noise and omitted.

Types of Clusters

- **Conceptual-Based**: points in the cluster share some general property





In all the previous clustering techniques:

- provide a number of clusters
- clusters are relatively arbitrary
- if you want to understand them better you need to go in and figure out what the clusters really "mean".

In conceptual clustering,

- provide it with a list of concepts and any info and requirements needed for an item to fit in that concept.
- The algorithm creates a structure (usually heiarchal) that defines how those concepts interact and what points belong to which concepts.

Techniques

Method	Algorithms
Partitioning	K-Means
Hierarchical	Agglomerative Hierarchical Clustering
Density	DBSCAN

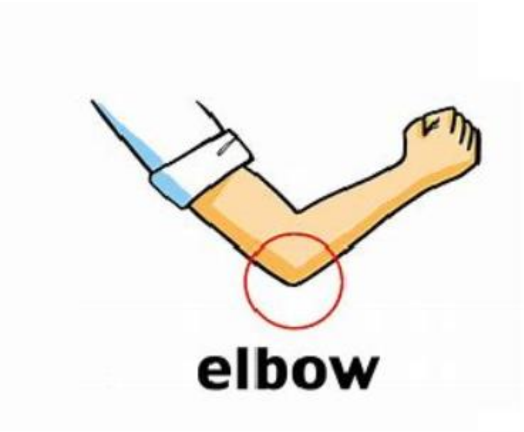
K-Means

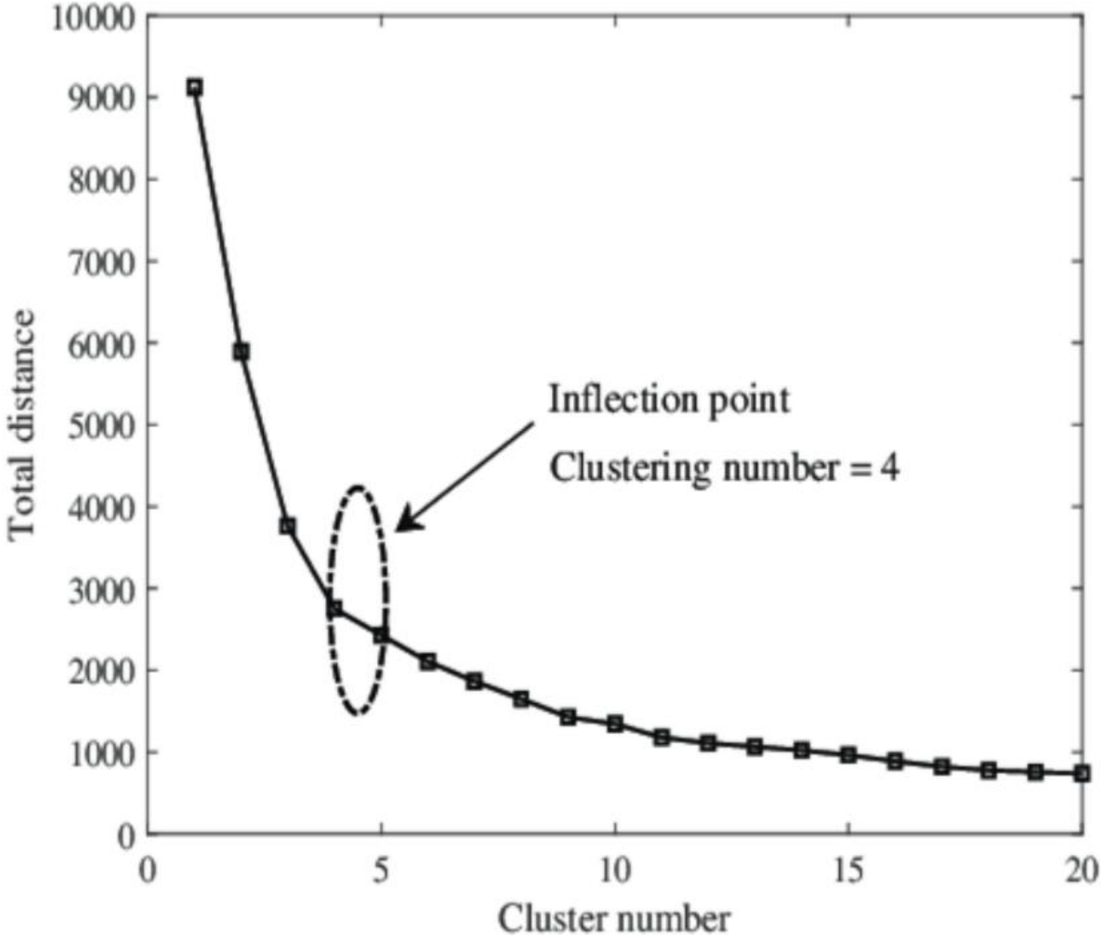
- ***k*-means clustering** aims to partition *n* observations into *k* clusters
  - each observation belongs to the cluster with the nearest mean
  - cluster centers or cluster centroid serves as a prototype of the cluster
  - *k*-means clustering minimizes within-cluster variances

- K: user-specified parameter

How to choose Number of K?

The Elbow Method





K-Means

- K: user-specified parameter

Select k points as initial centroids

Repeat

  Form k-clusters by assigning each point to the closest centroid

  Recompute the centroid of each cluster

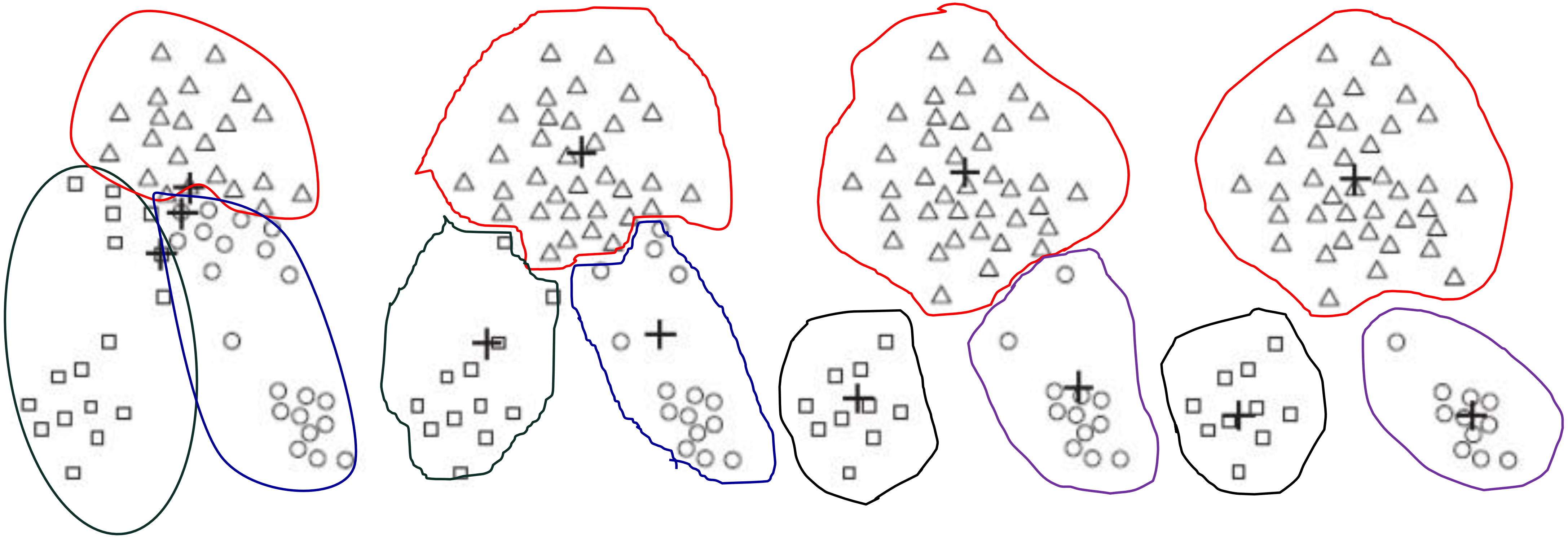
Until centroids do not change

How?

Need a proximity measure

Typically as the mean of each cluster

Example



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-Means

- K: user-specified parameter

Select k points as initial centroids
Repeat
    Form k-clusters by assigning each point to the closest centroid
    Recompute the centroid of each cluster
Until small enough change

Weaker condition for stopping: for example: until only 1% of points change clusters

Finding the closest centroid

- Proximity measures to quantify the notion of "closest"
- Euclidean distance: $d(a,b)=\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\cdots+(a_n-b_n)^2}$

Suitable for points in Euclidean space

- Manhattan distance: $d(a,b)=\sqrt{|a_1-b_1|+|a_2-b_2|+\cdots+|a_n-b_n|}$
- Cosine similarity measure: $\cos(a,b)=\dfrac{\sum_i a_i b_i}{\|a\|\|b\|}$

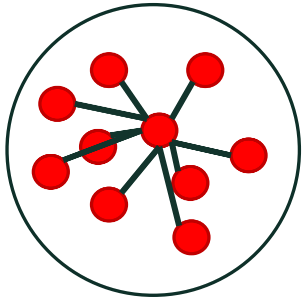
Suitable for documents and binary data

- Jaccard measure: (for binary data) $J=\dfrac{f_{11}}{f_{10}+f_{01}+f_{11}}$

Re-computing the centroid

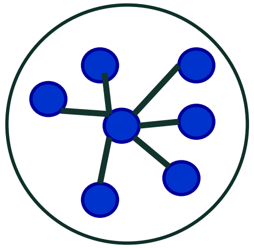
- Goal of clustering: expressed by an objective function
- When objective function is given: centroid can be computed mathematically
- Sum of Squared Error:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$



  - The mean of the cluster minimizes SSE
- Document Data:

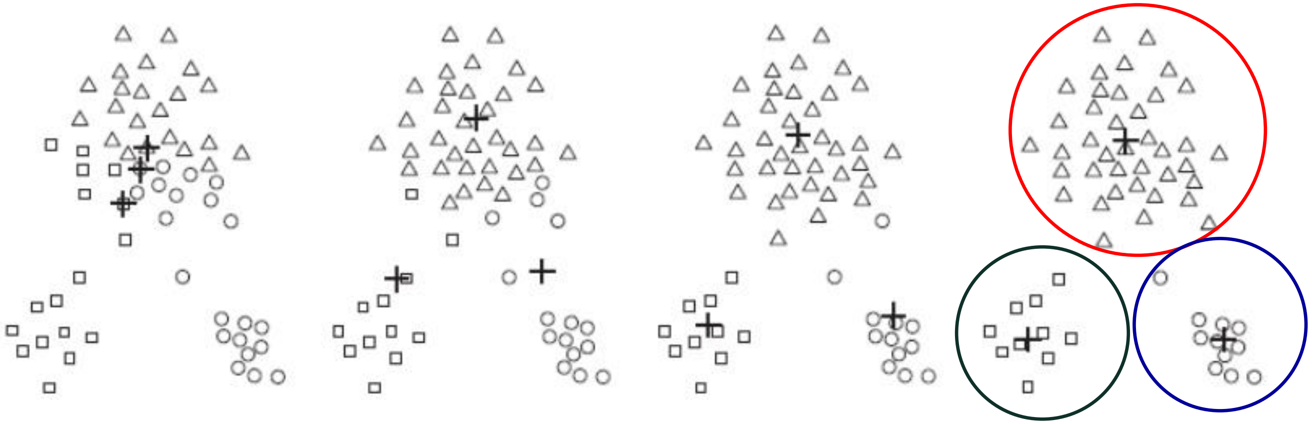
$$\text{Total Cohesion} = \sum_{i=1}^{k} \sum_{x \in C_i} \text{cosine}(x, c_i)$$



Choosing initial centroids

(1) Random initialization:
Different initial points result in different final clusters

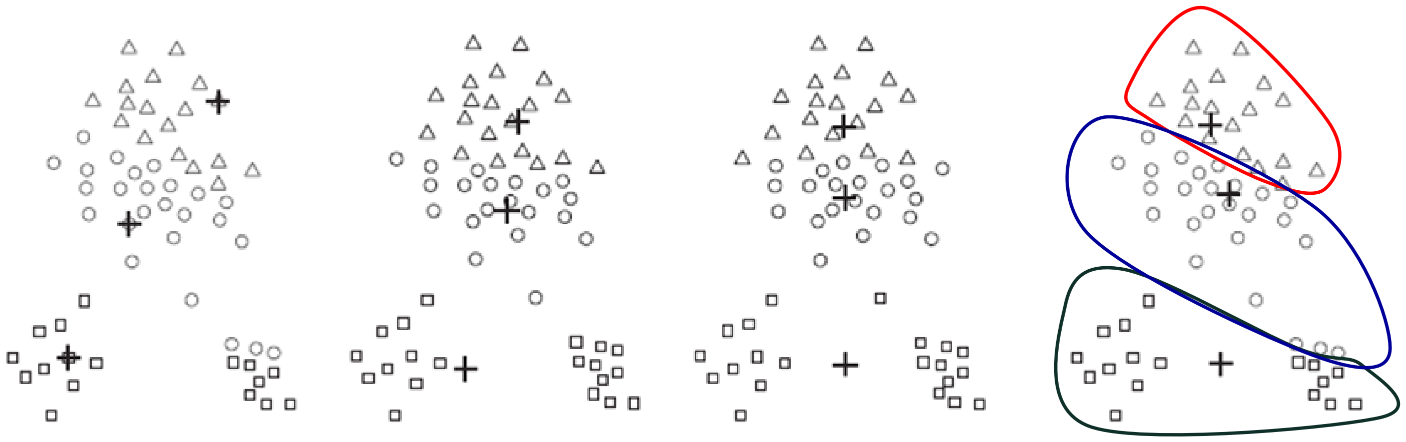
Try different random runs and select best one. Might not always generate a good choice



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

Choosing initial centroids

(2) Well separated initial centroids: Select initial centroid randomly. Then successively select **farthest point** from centroid as the next centroid. Might select outliers as centroids

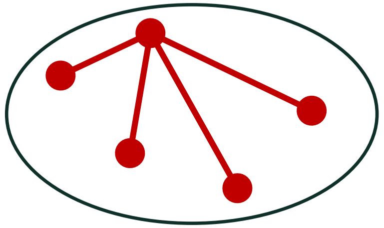


(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

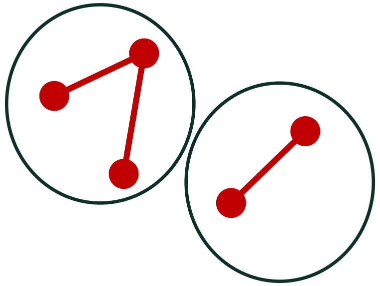
Empty Clusters

- Empty clusters may be obtained if no points are allocated to a given cluster
- Will increase the squared error unnecessarily
- Approaches:
  1. Choose a new centroid from the cluster that has the highest SSE

     ➡ This will split the cluster and reduce the SSE
  2. Choose a point that is farthest away from any cluster centroid



One cluster



Two clusters

Outliers

- Unnecessarily increase the error
- Not as representative as without outliers
- Useful to eliminate outliers beforehand

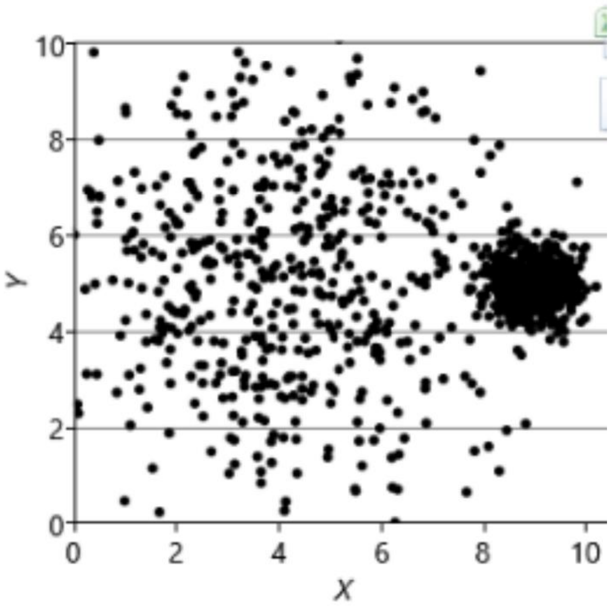
Post-Processing

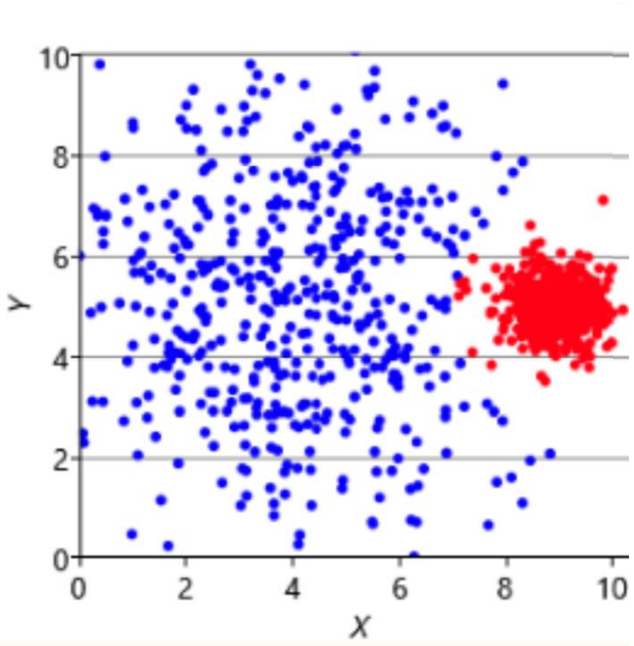
- Increase number of clusters to reduce SSE
  - Split a cluster: usually with the largest SSE
  - Introduce a new centroid: the point farthest from any centroid

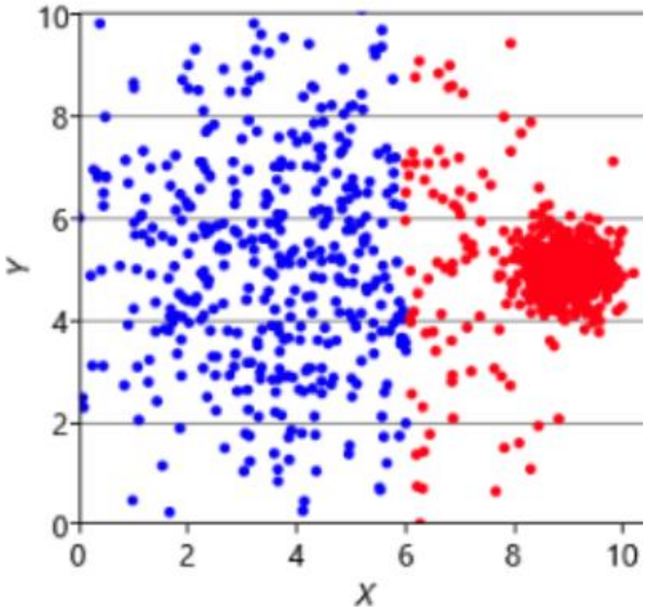
- Reduce number of clusters while trying to minimize SSE
  - Disperse a cluster: remove its centroid, reassign its points to closest clusters
  - Merge two clusters: merge clusters with closest centroids or that result in smallest SSE increase

Assumptions made by K-means

All clusters are the same size.(Area not Cardinality)



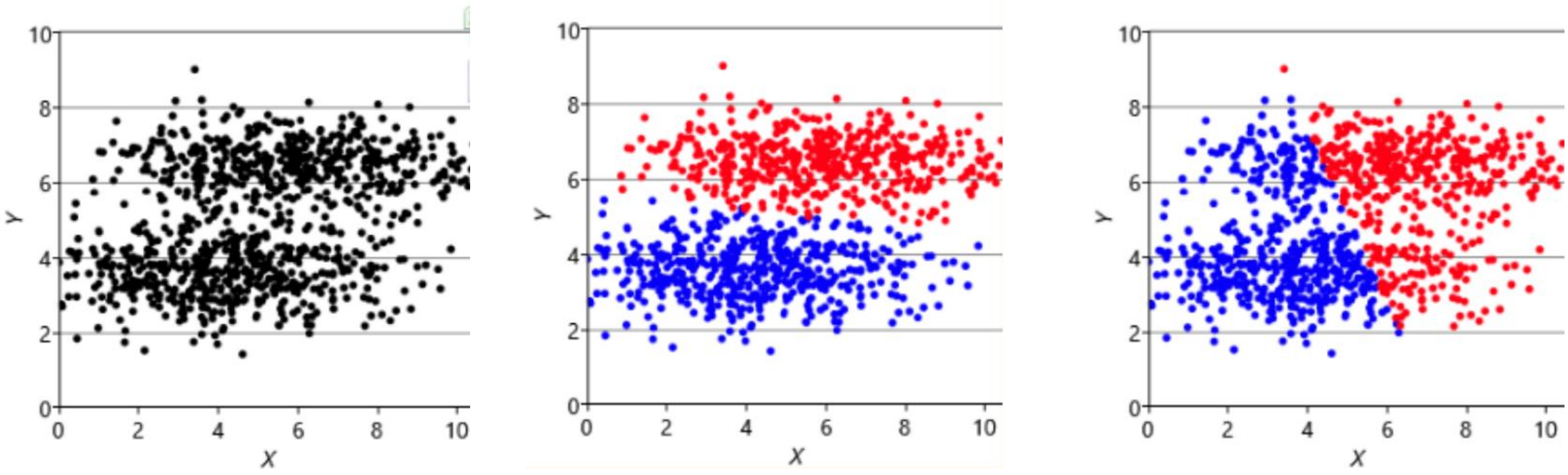




K-means

Assumptions made by K-means

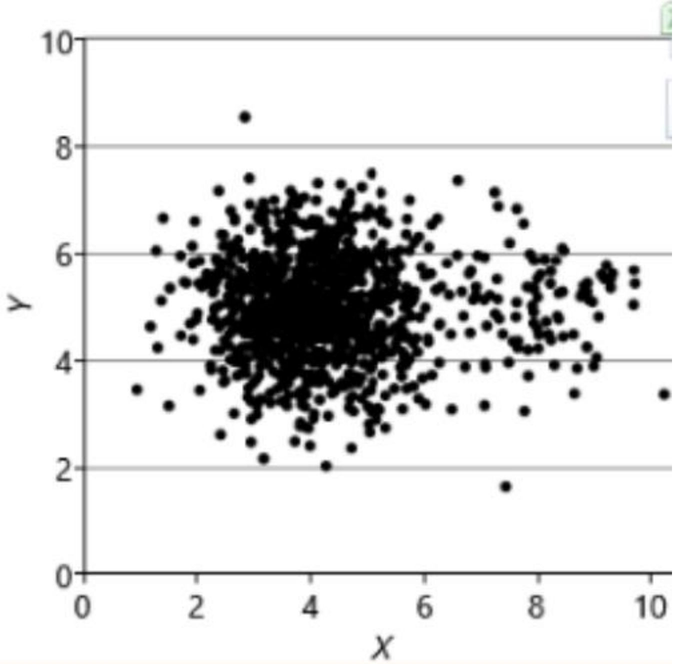
Clusters have the same extent in every direction.

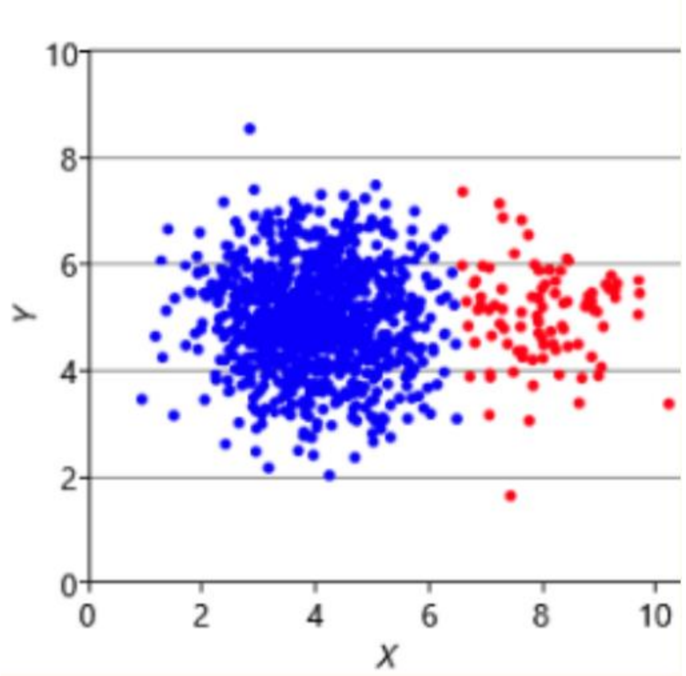


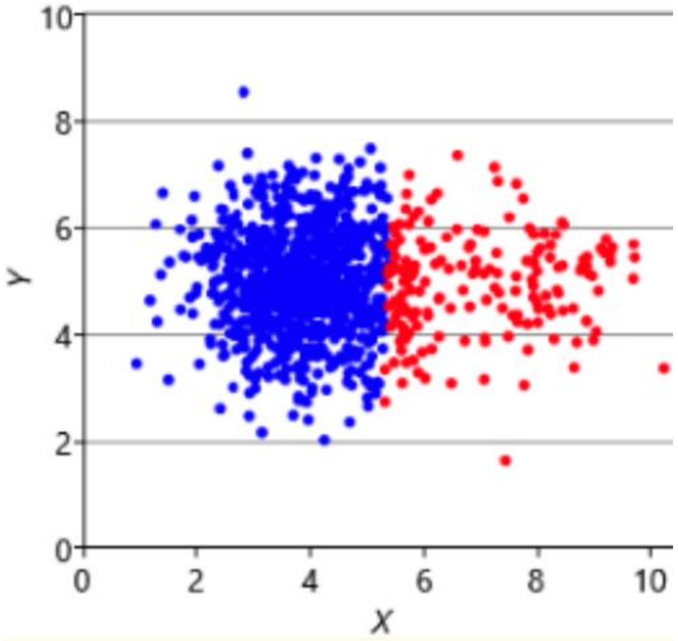
K-means

Assumptions made by K-means

Clusters have similar numbers of points assigned to them







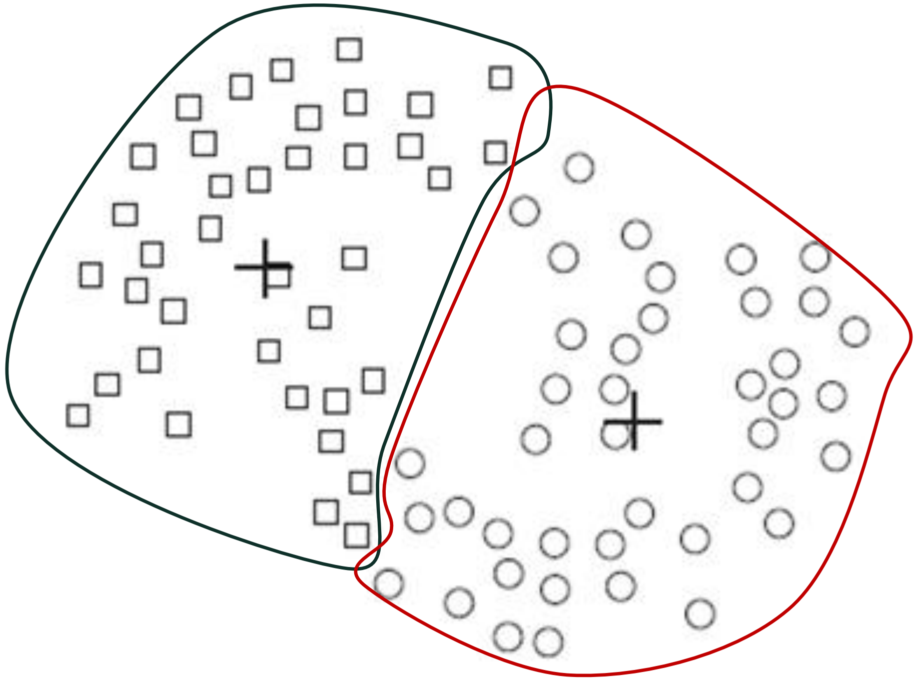
K-means

Weaknesses

- Can’t well detect natural clusters when clusters have:
  - Non-spherical shapes
  - Widely different sizes
  - Widely different densities

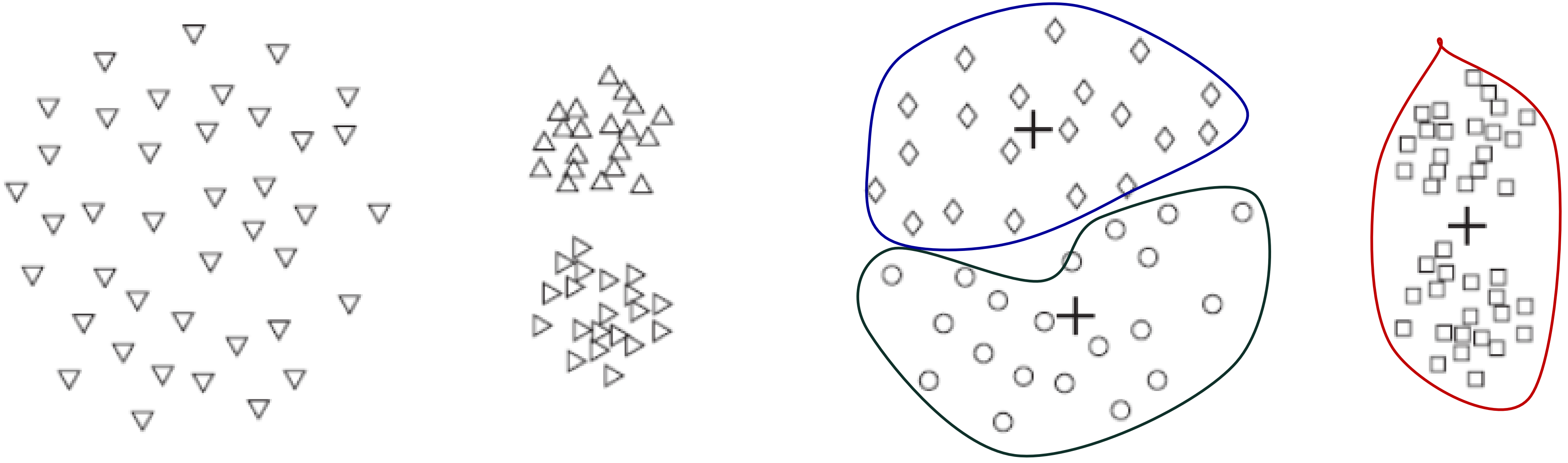
Clusters with different shapes

(a) Original points.



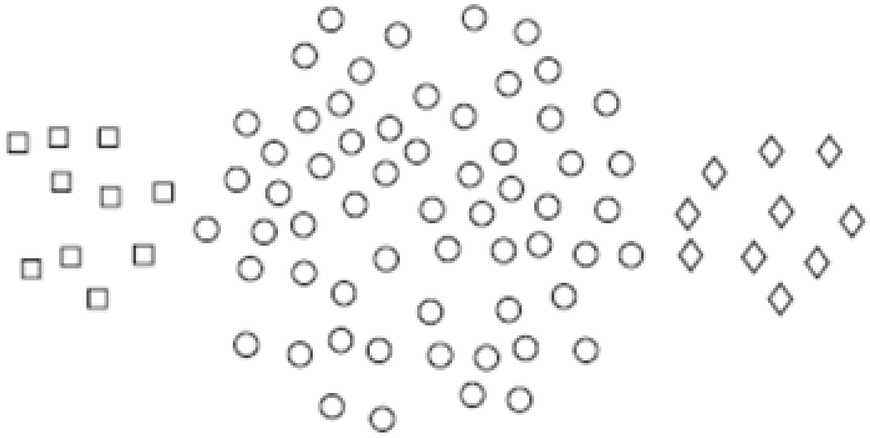
(b) Two K-means clusters.

Clusters with different densities

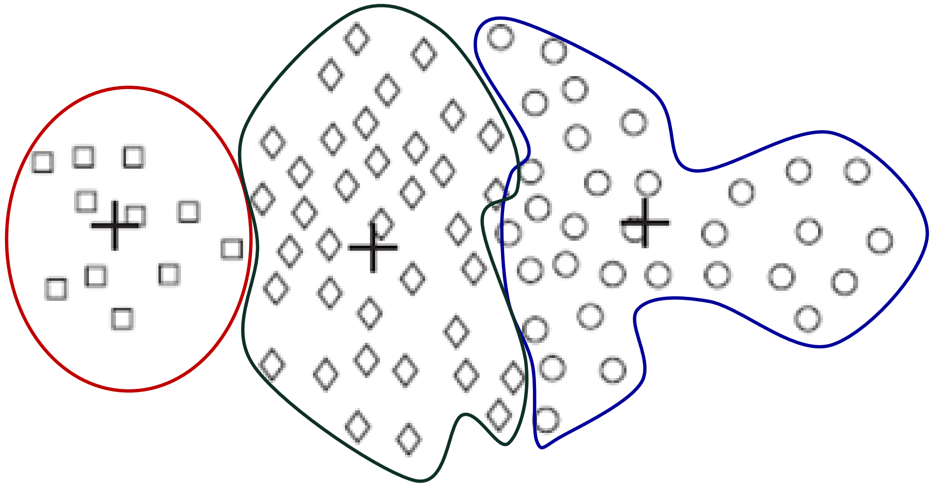


(a) Original points. (b) Three K-means clusters.

Clusters with different sizes

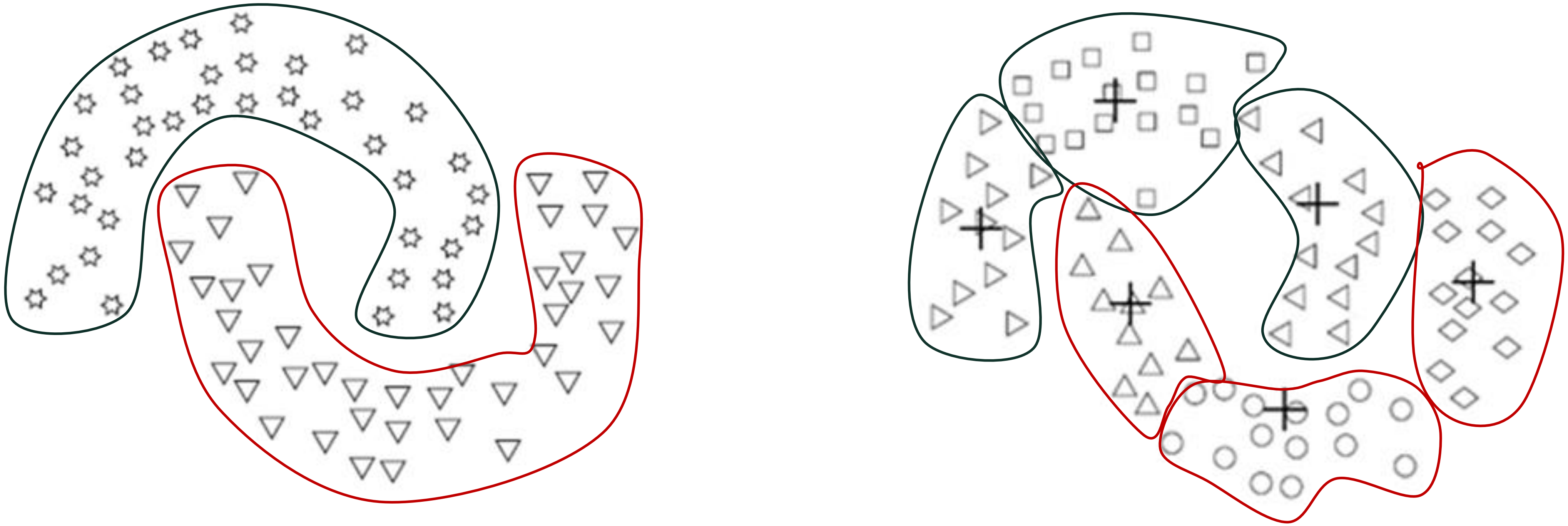


(a) Original points.

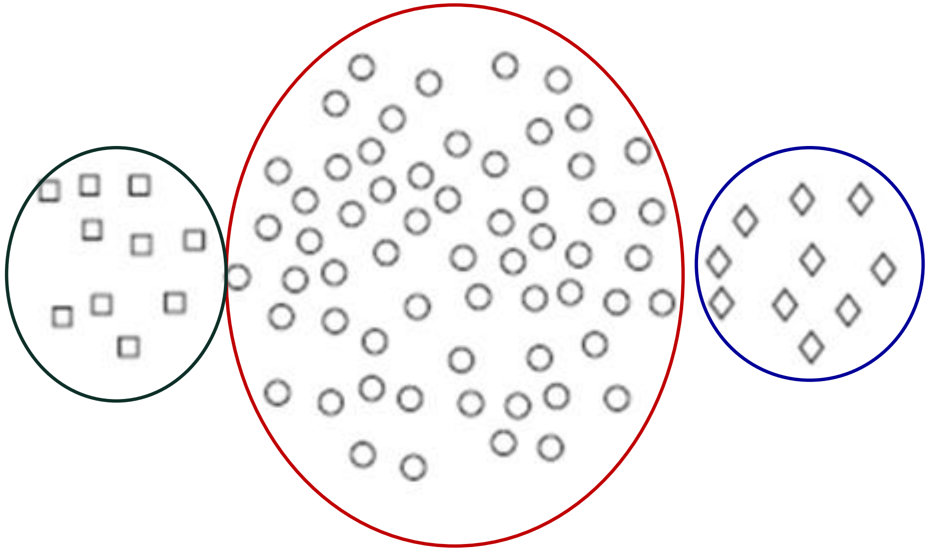


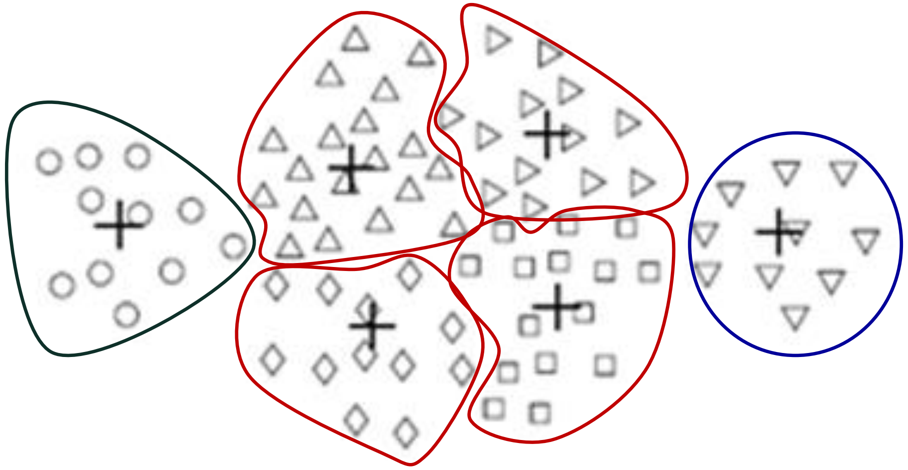
(b) Three K-means clusters.

Increasing number of clusters

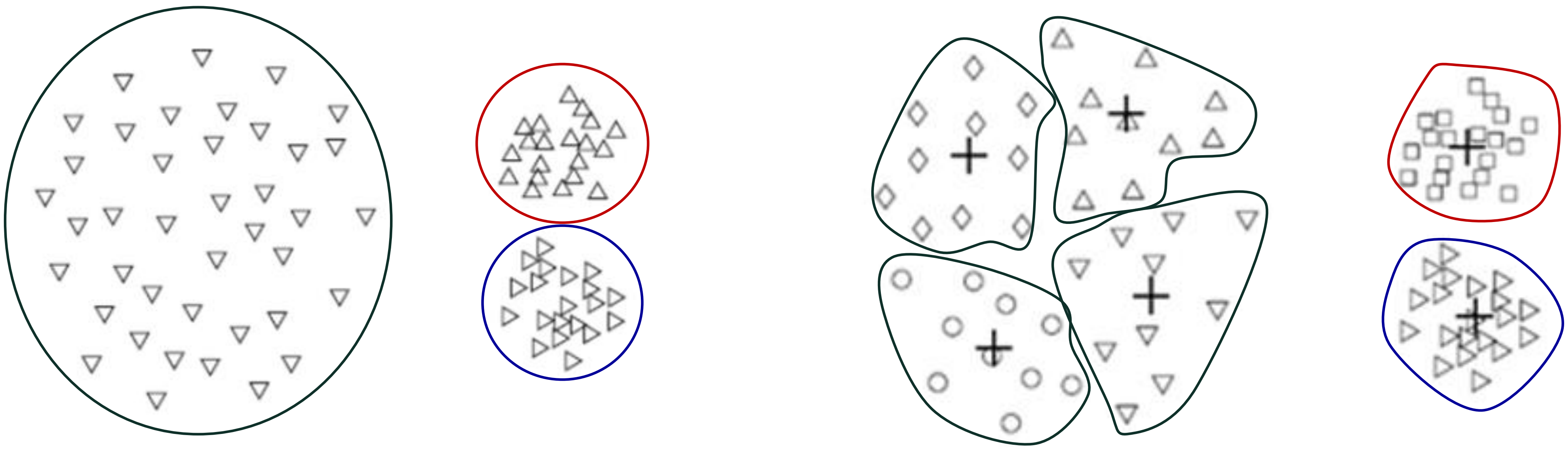


Increasing number of clusters





Increasing number of clusters



Strengths

- Efficient
- Converges relatively quickly
- Can be used with a variety of data

Variations

- Kmeans ++:
  - Improves K-Means by selecting initial cluster centers more strategically
- K-medoid:
  - Partitionning around medoid, PAM
  - More efficient than k-means in presence of outliers and noise
  - The complexity of each iteration is more costly than k-means

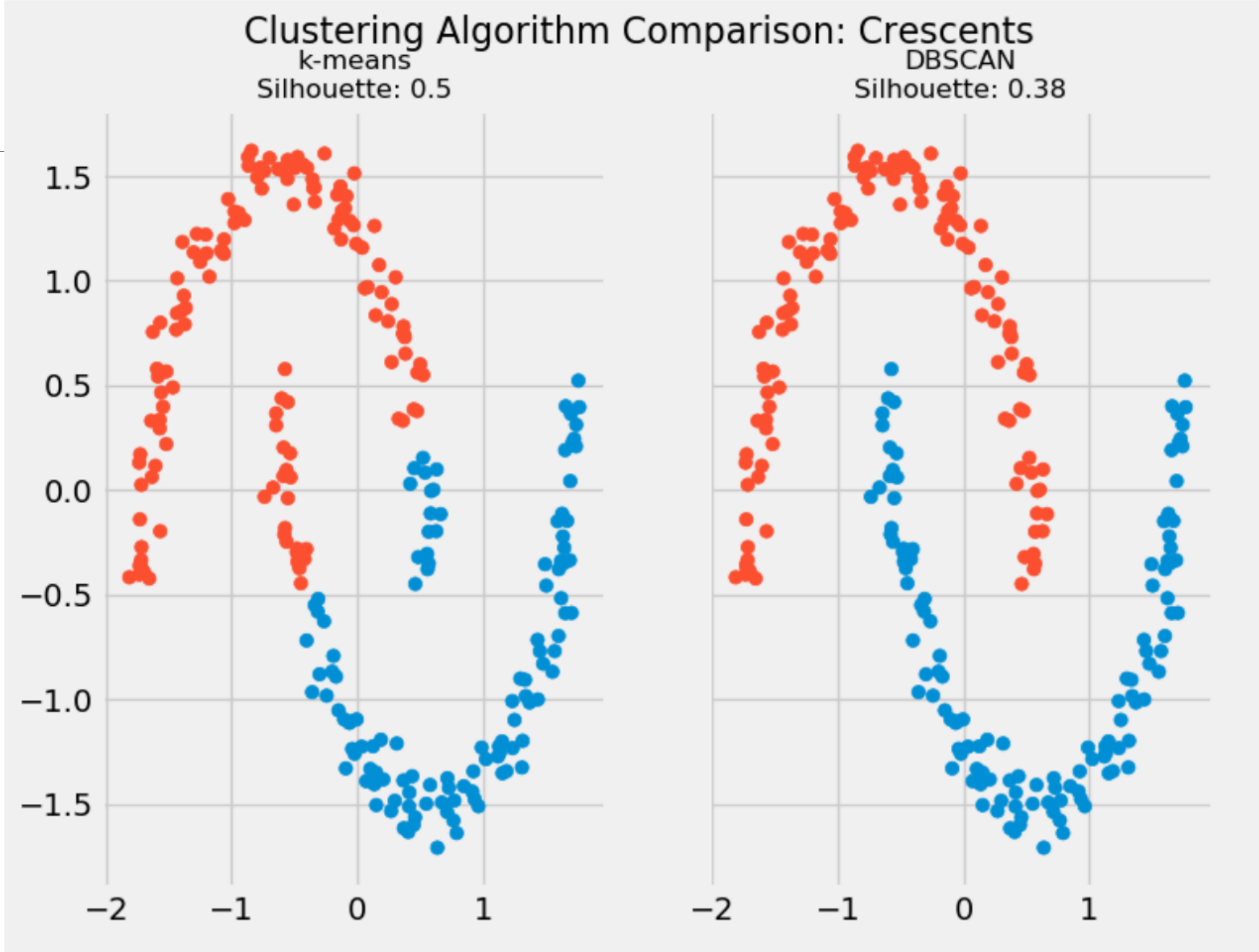
  - **Update centroids**
    - Unlike k-means, where centroids can be any point in space, a medoid is an actual data point within the dataset
    - Initially, the k medoids are randomly selected from the dataset. The algorithm then iteratively replaces non-medoid points with medoids that minimize the total dissimilarity within the cluster
    - This process continues until the medoids converge and the clusters stabilize.

Variations

Clustering Large Applications (CLARA)

- Apply PAM on a sample of the original set
- Performance depends on sampled medoids (how close to best medoids)

DBSCAN


Clustering Algorithm Comparison: Crescents
k-means
Silhouette: 0.5
DBSCAN
Silhouette: 0.38
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
−2
−1
0
1

Insurance Fraud Data

Case	Age	Gender	Claim	Tickets	Prior Claims	Attorney	Outcome
1	1	1	0.6	1	0.5	0	OK
2	0.9	1	0.64	1	1	1	OK
…							
10	0.3	0	0.48	0.6	1	1	Fraudulent

<u>Normalize data</u>

Tickets:
- 1: more than 2
- 0.6: 1 ticket
- 0: 0 ticket

Prior claims:
- 0: no claims
- 0.5: 1 claim
- 1: 2 or more claims

Gender: 1 for Male, 0 for Female

Claim amount: (claim -MIN)/(MAX-MIN)

Insurance Fraud Data

Select randomly 1 fraudulent and 1 ok claims as centroids

Cluster	Age	Gender	Claim	Tickets	Prior Claims	Attorney	Outcome
1	1	1	0.6	1	0.5	0	0.0
2	0.05	0	0.0	0.6	0	0	1.0

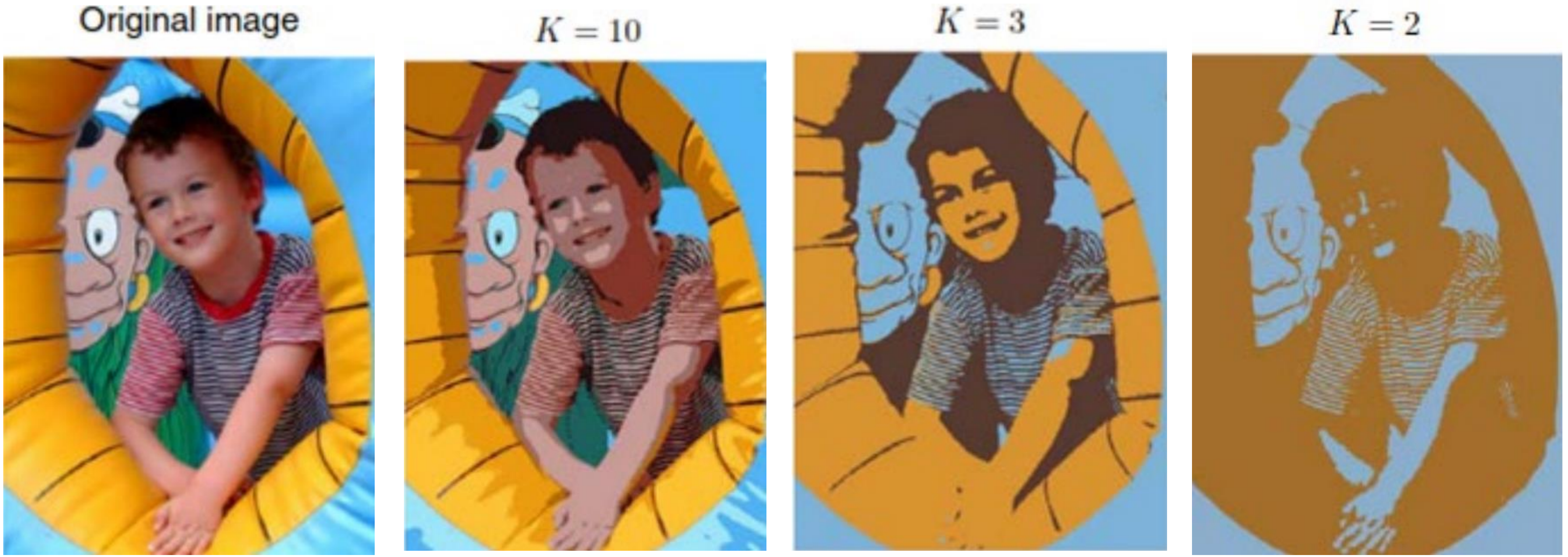
Find distances from each point to each centroid and assign point to cluster

Training Case	Cluster 1	Cluster 2	Outcome
1	0	2.673	Cluster 1
2	1.262	4.292	Cluster 1
3	2.673	0	Cluster 2
4	2.170	2.030	Cluster 2
5	2.328	2.137	Cluster 2
6	0.604	1.927	Cluster 1
7	1.280	4.094	Cluster 1
8	2.133	2.020	Cluster 2
9	3.270	2.710	Cluster 2
10	2.754	3.653	Cluster 1

Repeat for iterations 2, 3, … until convergence

Image Segmentation

Segmentation is to partition an image into regions each of which has a reasonably homogeneous visual appearance or which corresponds to objects or parts of objects



How to use

sklearn.cluster.KMeans

class sklearn.cluster.KMeans(*n_clusters=8*, ***, *init='k-means++'*, *n_init=10*, *max_iter=300*, *tol=0.0001*, *verbose=0*, *random_state=None*, *copy_x=True*, *algorithm='auto'*) [source]

```
>>> from sklearn.cluster import KMeans
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4], [1, 0],
...               [10, 2], [10, 4], [10, 0]])
>>> kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
>>> kmeans.labels_
array([1, 1, 1, 0, 0, 0], dtype=int32)
>>> kmeans.predict([[0, 0], [12, 3]])
array([1, 0], dtype=int32)
>>> kmeans.cluster_centers_
array([[10.,  2.],
       [ 1.,  2.]])
```

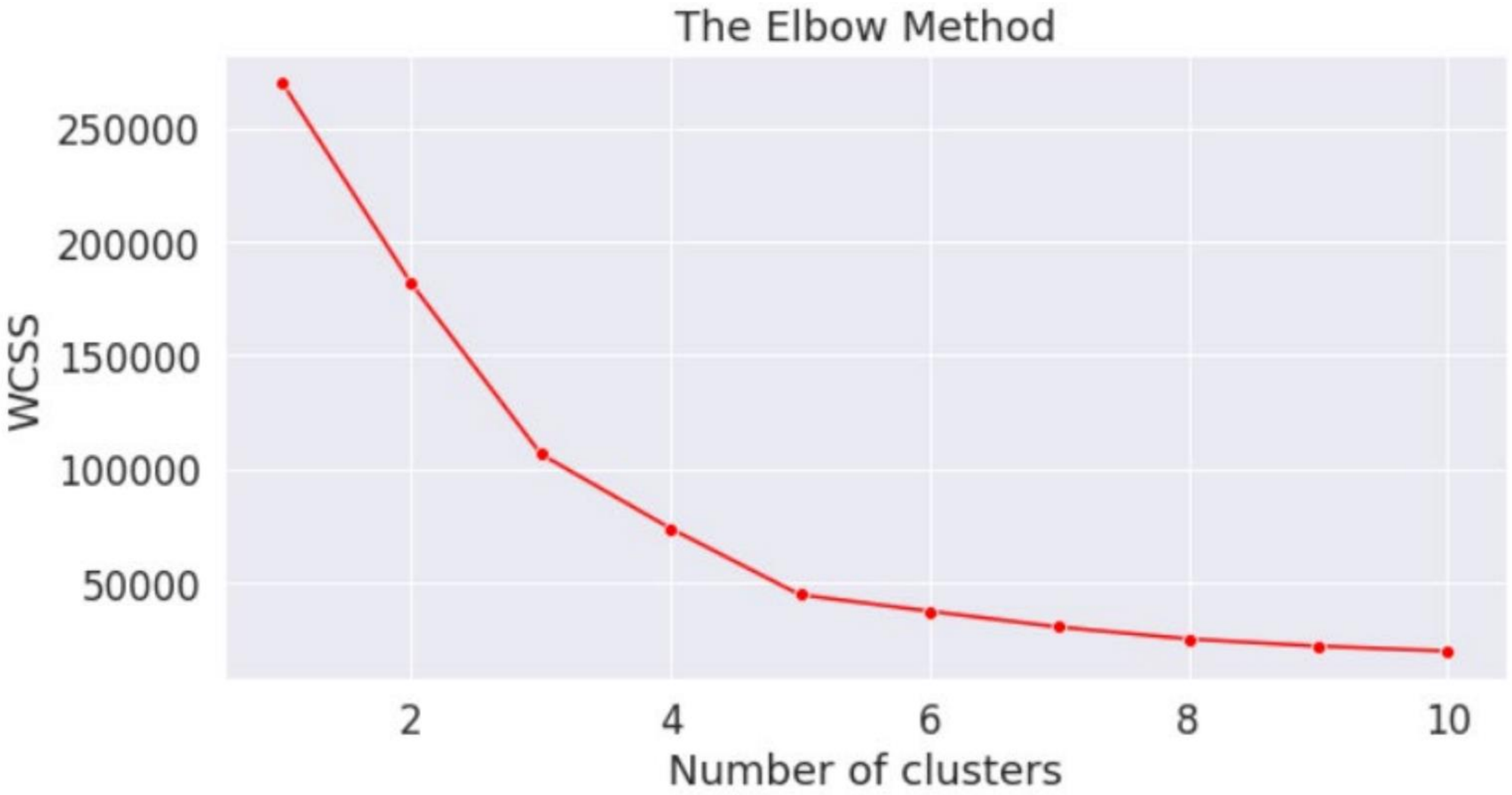
```python
# Importing the dataset
dataset = pd.read_csv('../input/Mall_Customers.csv',index_col='CustomerID')
```

```python
dataset.head()
```

	Genre	Age	Annual_Income_(k$)	Spending_Score
CustomerID				
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40

```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    # inertia method returns wcss for that model
    wcss.append(kmeans.inertia_)
```

```
plt.figure(figsize=(10,5))
sns.lineplot(range(1, 11), wcss,marker='o',color='red')
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```


The Elbow Method
250000
200000
150000
100000
50000
WCSS
2
4
6
8
10
Number of clusters

```python
# Fitting K-Means to the dataset
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(X)
```

```python
# Visualising the clusters
plt.figure(figsize=(15,7))
sns.scatterplot(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], color = 'yellow', label = 'Cluster 1',
s=50)
sns.scatterplot(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], color = 'blue', label = 'Cluster 2',s=
50)
sns.scatterplot(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], color = 'green', label = 'Cluster 3',s
=50)
sns.scatterplot(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], color = 'grey', label = 'Cluster 4',s=
50)
sns.scatterplot(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], color = 'orange', label = 'Cluster 5',
s=50)
sns.scatterplot(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color = 'red',
                label = 'Centroids',s=300,marker=',')
plt.grid(False)
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```


Clusters of customers
100
80
60
40
20
0
Spending Score (1-100)
20
40
60
80
100
120
140
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids