

# **CS5228: Knowledge Discovery and Data Mining**

## Lecture 2 — Clustering I

# Course Logistics — Update

- Project — Team Formation

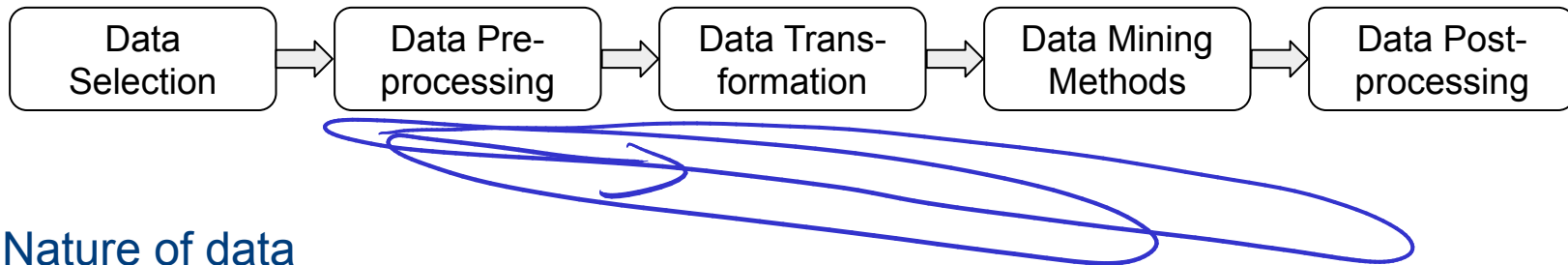
- Default team size: 4 (exception maybe later if needed)
- Final allocation at the teaching team's discretion
- Canvas self-signup group

- Project — Deadline

- Dataset release: Week ~3
  - Progress Report: Week ~7
  - Final Report: Week 12 (Fri, Nov 7)
- } Earlier submission are always welcome! :)

# Quick Recap

- Data Mining — from data to knowledge



- Nature of data

- Types of attributes: categorical (nominal / ordinal) vs. numerical (interval / ratio)
- Types of data and data representations  
(e.g., data matrix, transactions, graph, ordered data)
- Data quality

# Quick Recap — Data Preparation

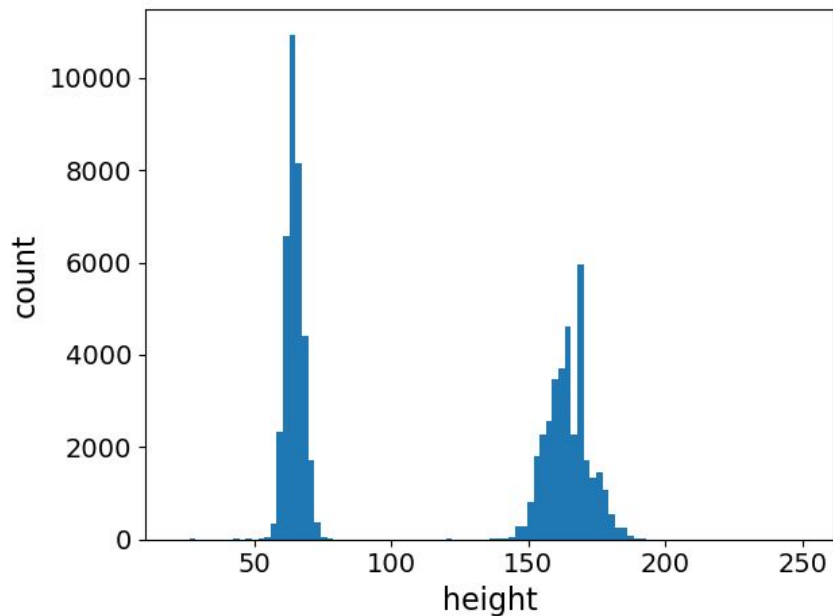
- Exploratory Data Analysis (EDA)
  - Assess data quality
  - Get to know your data
- Data Preprocessing
  - Improve data quality ("*Garbage in, garbage out!*")
  - Generate valid input for data mining algorithms
  - Remove complexity from data to ease analysis

Example of noise: missing values

Age	Edu- cation	Marital Status	Annual Income	Credit Default
23	Masters	Single	75k	Yes
N/A	Bachelor	Married	N/A	No
26	Masters	Single	70k	Yes
41	PhD	Single	95k	Yes
18	Bachelor	Single	40k	No
55	Master	Married	N/A	No
30	Bachelor	Single	N/A	No
35	PhD	Married	60k	Yes
N/A	PhD	Married	65k	Yes

# Quick Recap — Clarifications

- Suspicious EDA results  $\Rightarrow$  errors in the data



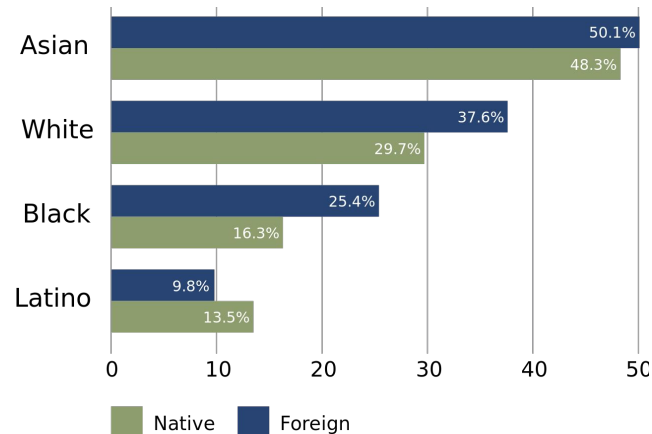
Data may be absolutely correct

Example: infant care dataset (infants + parents)

# Quick Recap — Clarifications

- Removing "questionable" attributes  $\Rightarrow$  better results
  - e.g., removing zodiac sign from credit default prediction might lower accuracy
- Removing "questionable" attributes  $\Rightarrow$  no biases (or perfect privacy)
  - e.g., removing ethnicity not foolproof if it correlates with other attributes

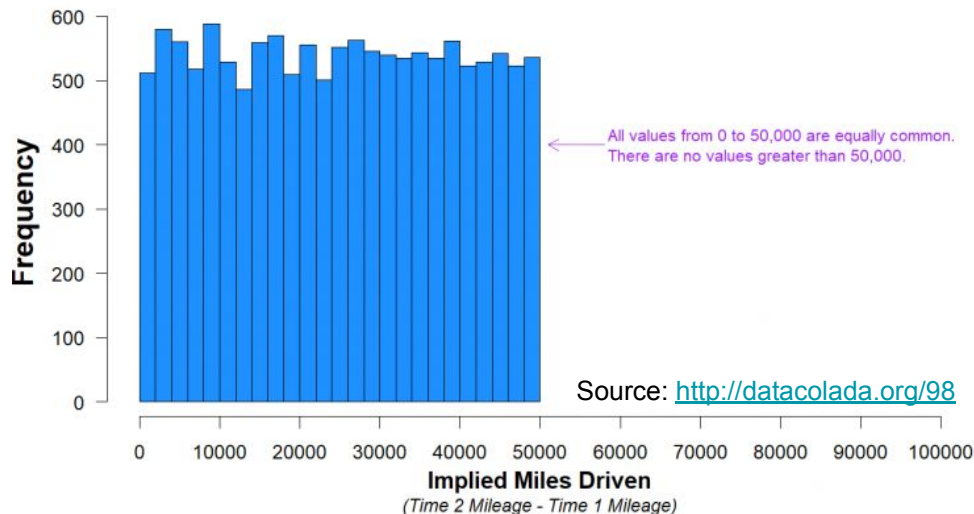
Age	Ethnicity	Education	Annual Income	Credit Approval
23	White	Masters	75k	Yes
35	Latino	Bachelor	50k	No
26	Asian	Masters	70k	Yes
41	Asian	PhD	95k	Yes
18	Black	Bachelor	40k	No
...	...	...	...	...



# Quick Recap — EDA: Additional Insights

- Manipulated / fudged data
  - Sometimes data just does not "look right"
  - Example: unexpected/unintuitive data distributions

Figure 1. Histogram of Miles Driven - Car #1 (N=13,488)



# Outline

- **Clustering**

- Overview
- Applications
- Concepts

- **Clustering algorithms**

- K-Means
- DBSCAN
- Hierarchical Clustering

- **Cluster Evaluation**

L2

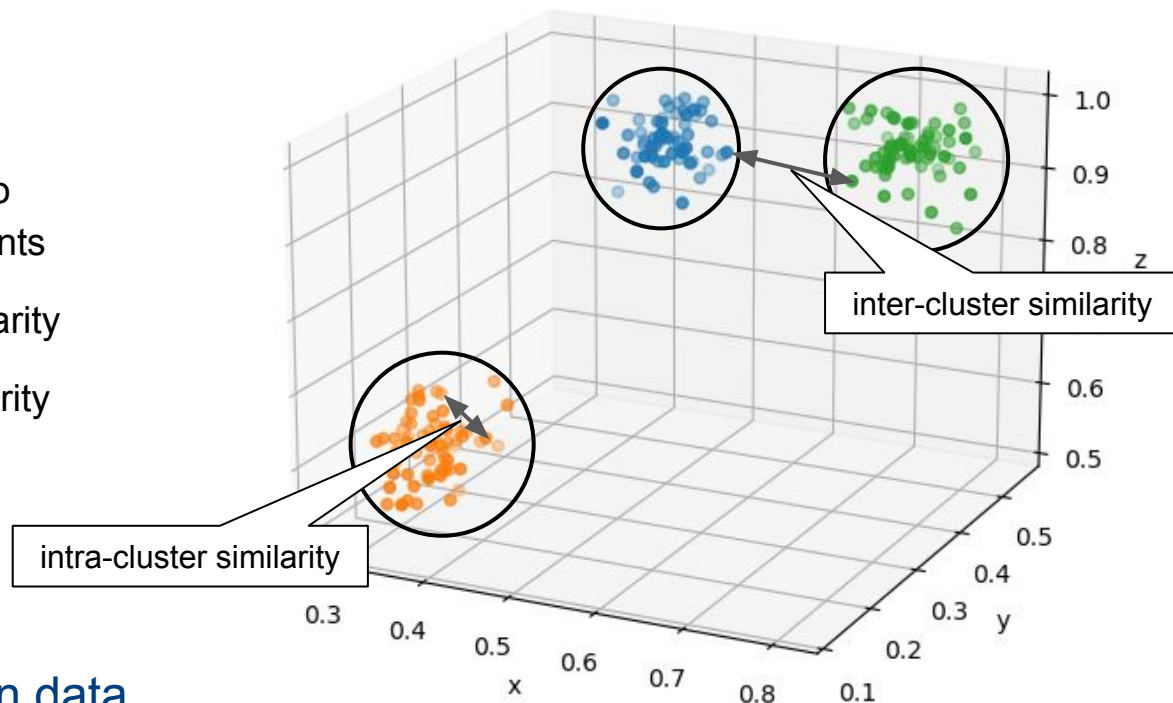
L3



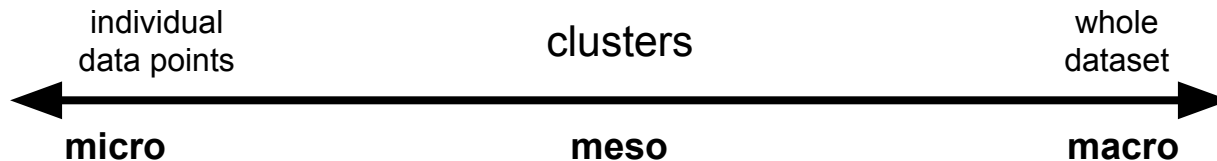
# What is Clustering?

- Goal of Clustering

- Separate unlabeled data into groups of **similar** objects/points
- Maximize **intra-cluster** similarity
- Minimize **inter-cluster** similarity



- Meso-level perspective on data



# Applications

- Market segmentation

- Group customers based on behavior and/or preferences
- Push tailored promotions to all customers in cluster

- Recommender systems

- Group items (e.g., movies) based on their attributes (e.g., genre, length, budget)
- Recommend movies from a cluster with movies a user liked

- Web Search Diversification

- Group Web pages (e.g., news articles) based on content, source (type), etc.
- Return search results from different clusters to ensure diversity

- ...and many more applications

⇒ speeding up LLMs

# Ingredients for Clustering

- Representation of objects, e.g.:

- (Multidimensional) point coordinates  $x, y$
- Sets  $A, B$  (e.g., items in a transaction)
- Vectors  $u, v$  (e.g., TF-IDF)

- Similarity Measure, e.g.:

- Euclidean Distance
- Jaccard Similarity
- Cosine Similarity

- Clustering Algorithm

- Process that determines if an object belongs to a cluster

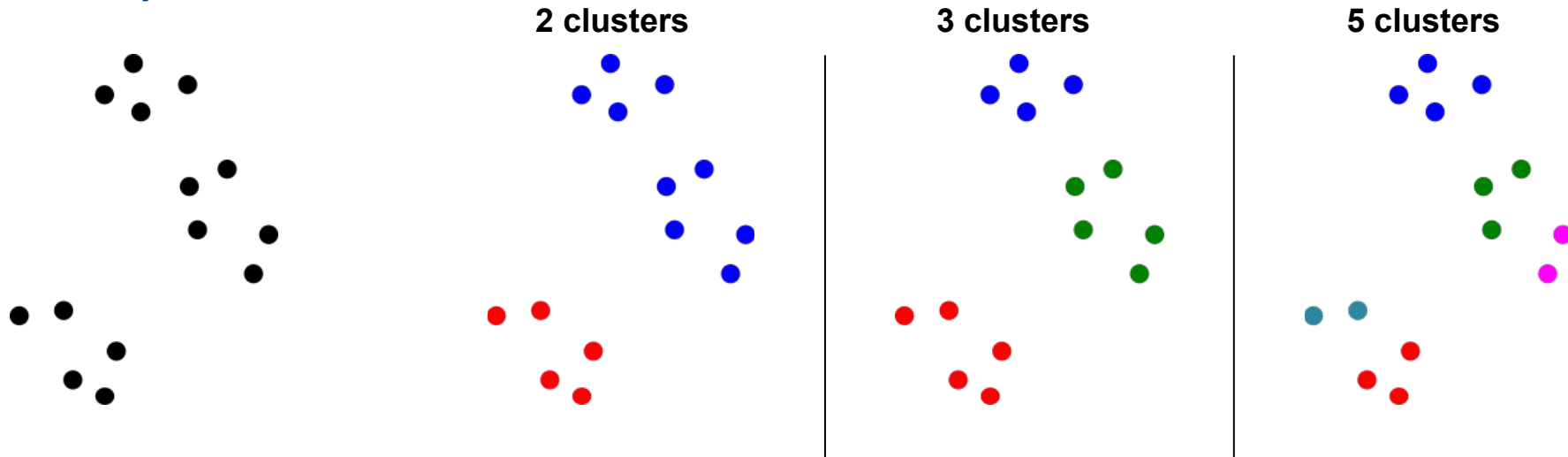
$$dist_{euclidean}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

$$sim_{jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$sim_{cosine}(u, v) = \frac{u \cdot v}{\|u\| \|v\|}$$

# What makes a clustering "good"?

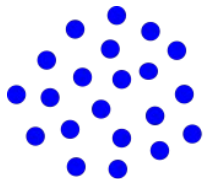
How many clusters?



→ Deciding on a good / meaningful / useful set of clusters not obvious

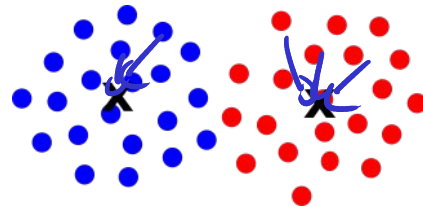
# Types of Clusters

- Well-separated



- Any object in a cluster is closer to every other object in the cluster than to any point outside the cluster

- Center-based

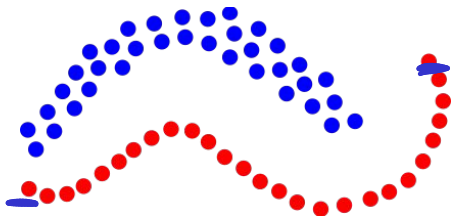


- Any object in a cluster is closer to the "center" of a cluster than to the center of any other cluster
- Example: mean of all data points (in Euclidean space)
- Cluster center commonly called centroid

*K-means*

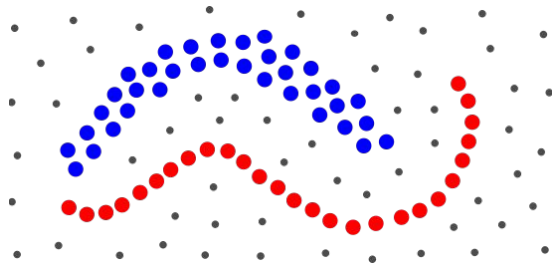
# Types of Clusters

- Contiguity-based



- 2 objects are in a cluster if they are more similar than a specified threshold
- Each object is more similar to some object in that cluster than to any point in a different cluster

- Density-based



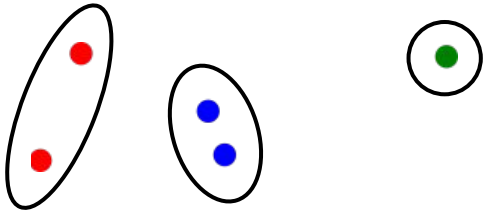
- Cluster = dense(r) region of objects surrounded by region of low(er) density
- Can typically address noise better than contiguity-based clusters

DBSCAN

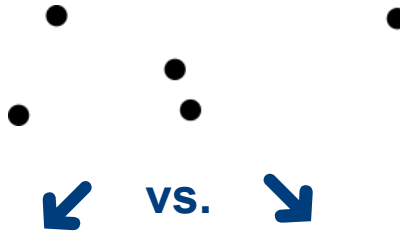
# Types of Clusterings (i.e., sets of clusters)

*K-Means*  
*DBSCAN*

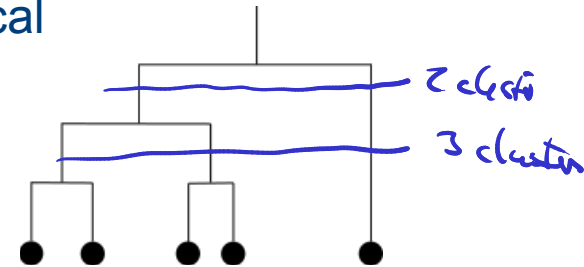
- Partitional



- Division of the set of data objects into non-overlapping subsets (i.e., clusters)
- Each object is in exactly 1 cluster (or in no cluster at all)

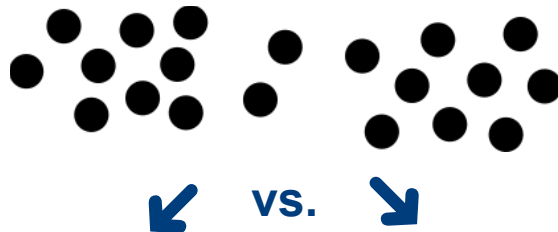


- Hierarchical



- Clusters can be nested
- A point can belong to different clusters depending on the hierarchy level

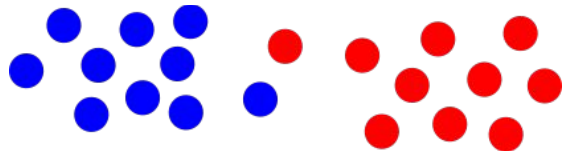
# Types of Clusterings (i.e., sets of clusters)



*k-Means*

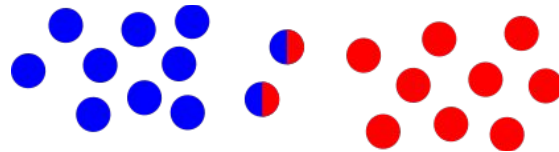
*DBSCAN*

- Exclusive



- Each object belongs to 1 cluster

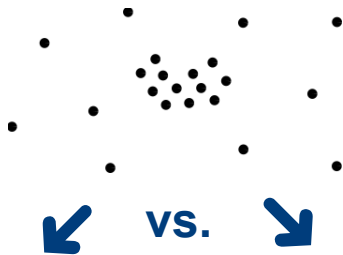
- Non-exclusive / overlapping



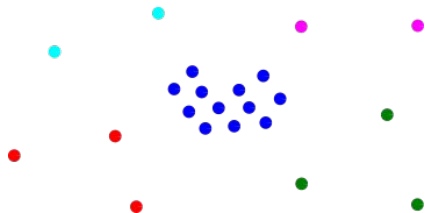
- An object can belong to more than 1 cluster at a time
- Fuzzy clustering: each object belongs to all clusters with a certain probability



# Types of Clusterings (i.e., sets of clusters)



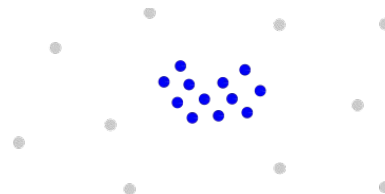
- Complete



- Every object is assigned to (at least) 1 cluster

*K-Means*

- Partial



- An object might not belong to a cluster
- Examples: noise, outliers

*DBSCAN*

# Quick Quiz

In what situation can I NOT apply clustering on a dataset?

**A**

All attributes of my dataset are nominal attributes

**B**

My dataset is 1-dimensional, i.e., there is only one attribute

**C**

The values of the attributes are not normally distributed

**D**



There is no similarity or distance between the data points defined

# Outline

- Clustering
  - Overview
  - Applications
  - Concepts
- Clustering Algorithms
  - **K-Means**
  - DBSCAN
  - Hierarchical Clustering
- Cluster Evaluation

# K-Means

- Basic characteristics

- Clusters: centroid-based
- Clustering: partitional, exclusive, complete

- Inputs (for d-dimensional Euclidean space)

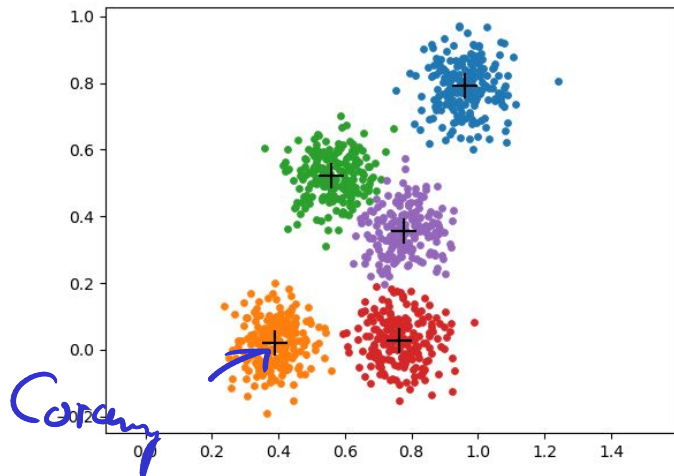
- $(x_1, x_2, \dots, x_N), x_i \in R^d$
- Number of clusters  $K \rightarrow c_1, c_2, \dots, c_K$  cluster centers

- Optimization objective

- Minimize Sum of Squared Error
- Finding optimal solution is NP-hard

$$O(N^{Kd+1})$$

→ Greedy solutions



$$\underline{SSE} = \sum_{i=1}^K \underbrace{\sum_{x \in C_i}}_{\text{set of points in } i\text{-th cluster}} \overbrace{\|x - \underline{c_i}\|^2}^{\text{squared distance between data point and center}}$$

centroid of cluster  $c_i$

# K-Means — How to Define the Centroid of a Cluster?

- Simple case in Euclidean space  $SSE = \sum_{i=1}^K \sum_{x \in C_i} ||x - c_i||^2$

$$\frac{\delta}{\delta c_k} SSE = \frac{\delta}{\delta c_k} \sum_{i=1}^K \sum_{x \in C_i} (x - c_i)^2$$

$$= \sum_{i=1}^K \sum_{x \in C_i} \frac{\delta}{\delta c_k} (x - c_i)^2$$

$$\Rightarrow \sum_{x \in C_k} 2 \cdot (x - c_k) \stackrel{!}{=} 0$$

# K-Means — How to Define the Centroid of a Cluster?

- Simple case in Euclidean space  $SSE = \sum_{i=1}^K \sum_{x \in C_i} \|x - c_i\|^2$

$m_k = \# \text{ samples in } C_k$

$$\frac{\partial}{\partial c_k} SSE = \frac{\partial}{\partial c_k} \sum_{i=1}^K \sum_{x \in C_i} (x - c_i)^2$$

$$= \sum_{i=1}^K \sum_{x \in C_i} \frac{\partial}{\partial c_k} (x - c_i)^2$$

$$\Rightarrow \sum_{x \in C_k} 2 \cdot (x - c_k) \stackrel{!}{=} 0$$

$$\sum_{x \in C_k} (x - c_k) \stackrel{!}{=} 0$$

$$\Rightarrow \sum_{x \in C_k} x - \sum_{x \in C_k} c_k = 0$$

$$\Rightarrow m_k c_k = \sum_{x \in C_k} x$$

$$\Rightarrow c_k = \frac{1}{m_k} \sum_{x \in C_k} x$$

→ Centroid of cluster = Mean of all points in that cluster

# K-Means — Basic Algorithm (Lloyd's Algorithm)

1. Initialization: Select  $K$  points as initial centroids  $c_1, c_2, \dots, c_K$

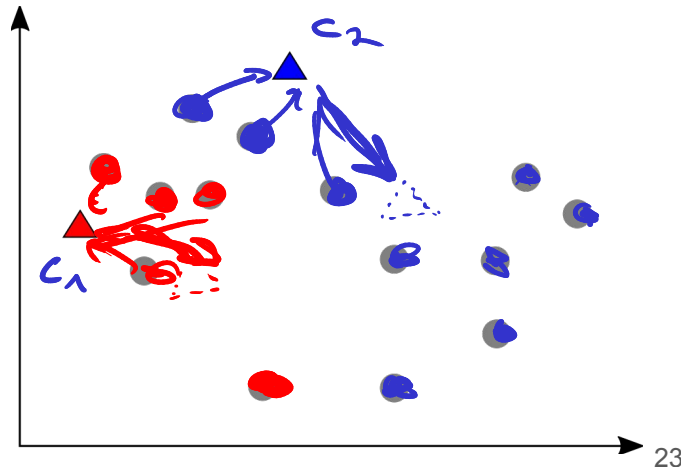
2. Repeat

2a) **Assignment**: assign each point to nearest cluster (i.e., centroid)

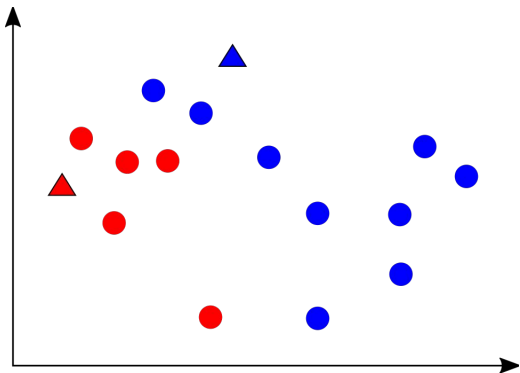
2b) **Update**: move each centroid to the average of its assigned points

**Until** no change in assignments

Example:  $K=2$ , after initialization

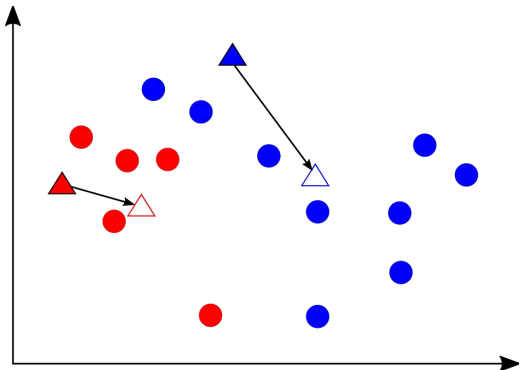


# K-Means — Repeated Steps



- Assignment

- For each data point  $x$ , find nearest centroid  $c_i$
- Assign  $x$  to cluster  $i$

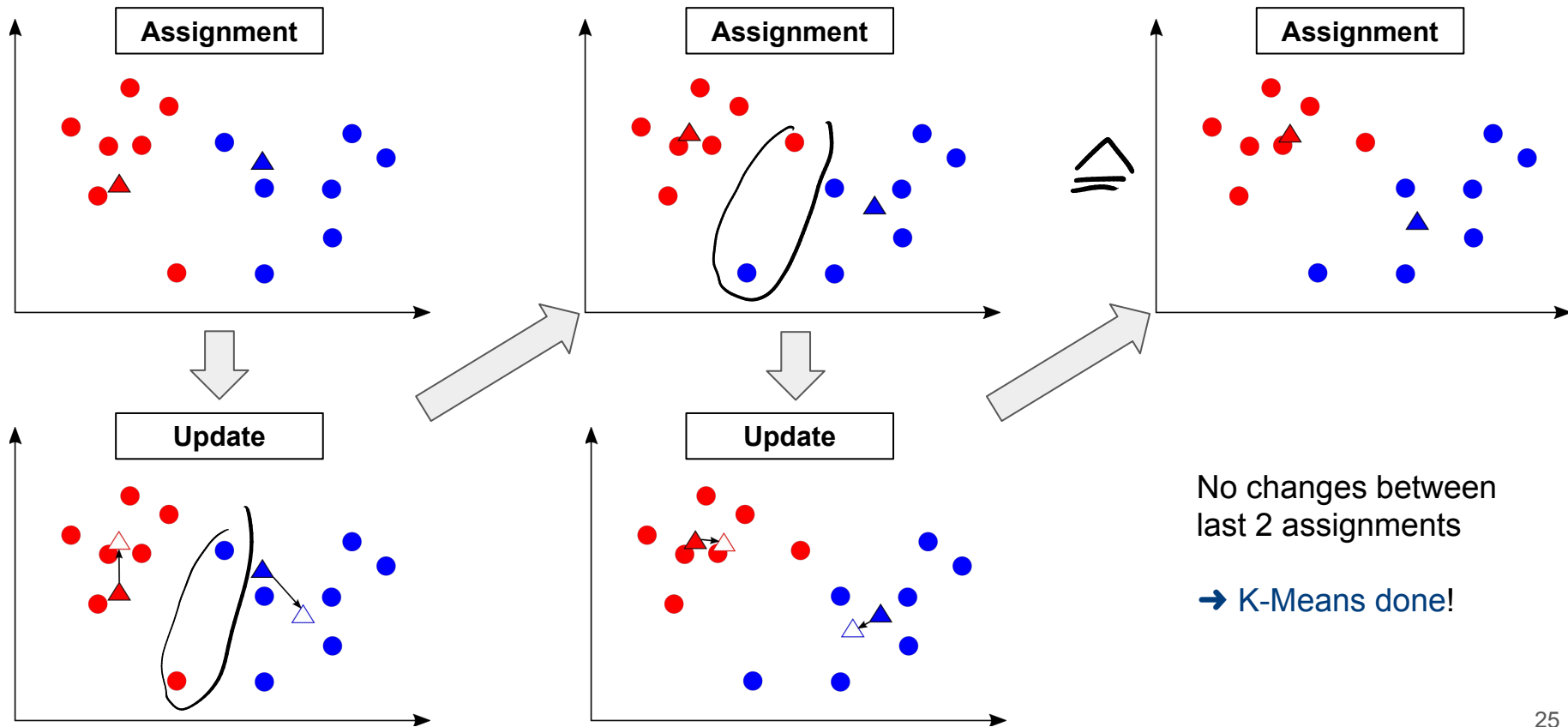


- Update

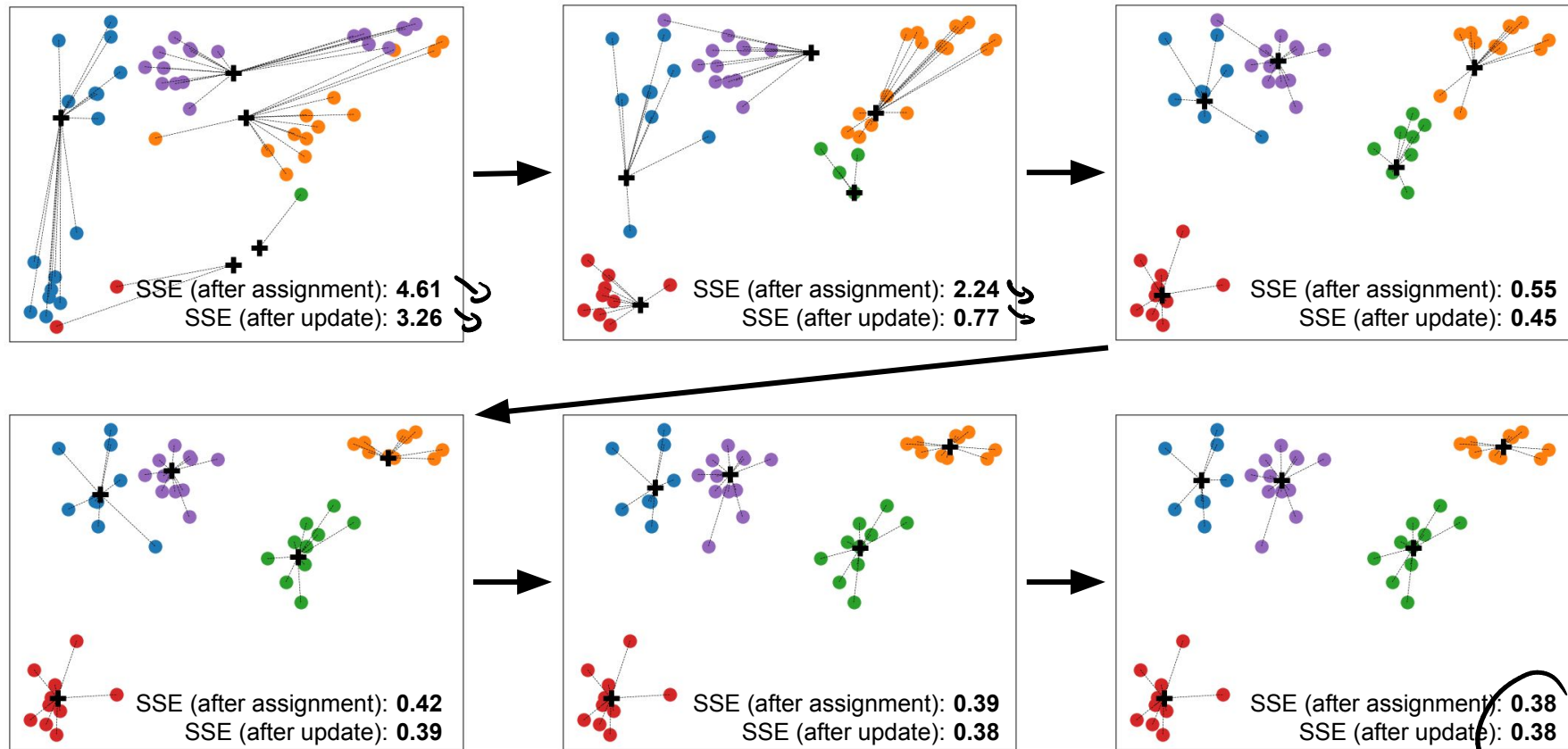
- Calculate mean of all data points of cluster  $i$
- Set centroid  $c_i$  to mean of cluster  $i$



# K-Means — Iteration until Convergence



# K-Means — Convergence



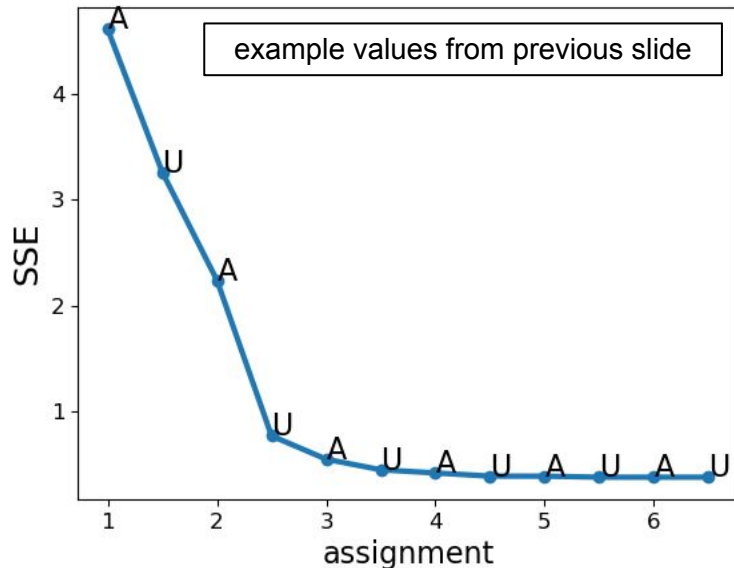
# K-Means — Convergence

- Good news: K-Means always converges!

- Both assignment (A) and update (U) reduce SSE (or no changes)
- Most improvement during the first iterations

- Bad news

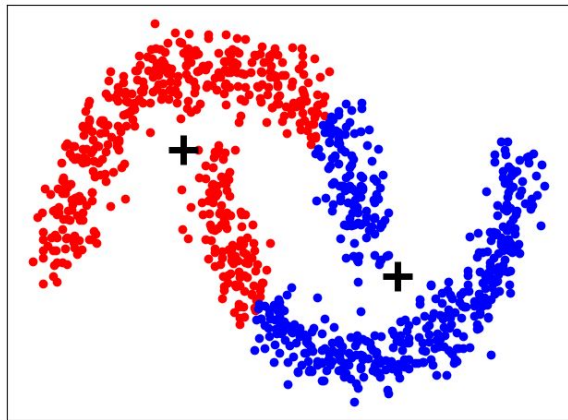
- Lloyd's algorithm returns a **local optimum**, not necessarily a global optimum
- Important: initialization of centroids  
(discussed in more detail later)



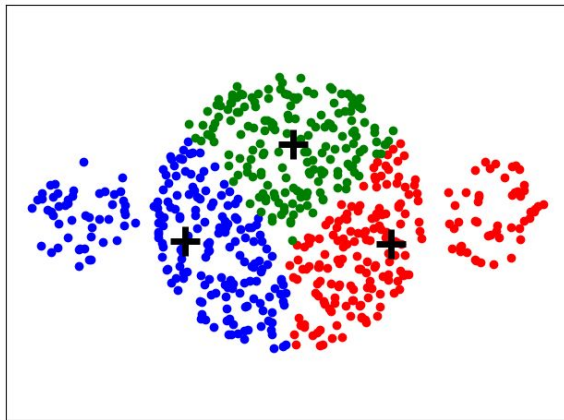
# K-Means — Limitations (Data Distribution)

- K-Means is susceptible to "natural" clusters

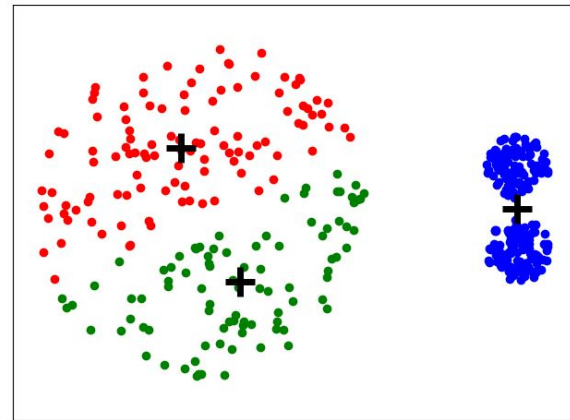
Non-Spherical Clusters



Clusters of different sizes

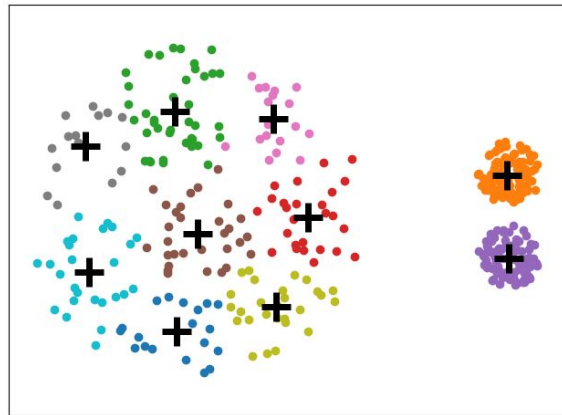
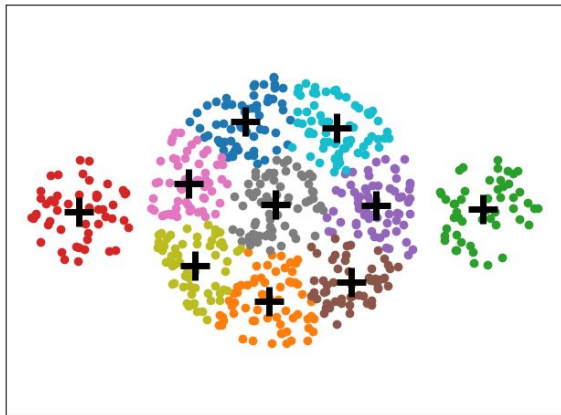
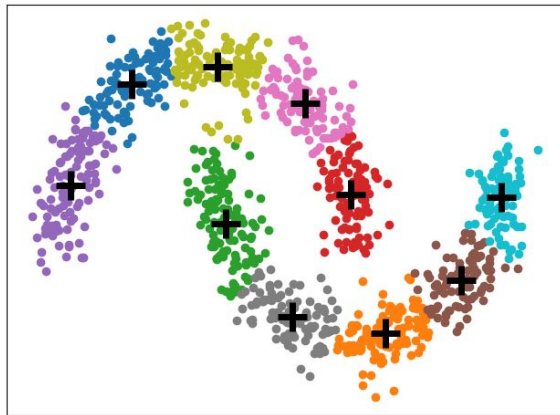


Clusters of different densities



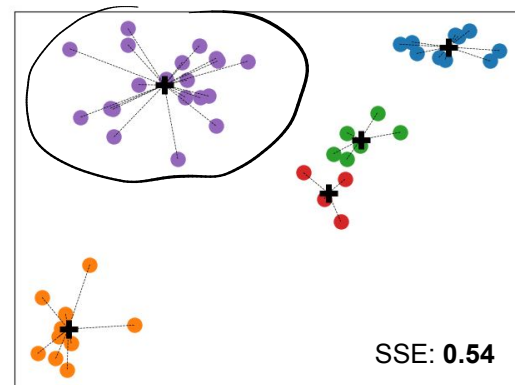
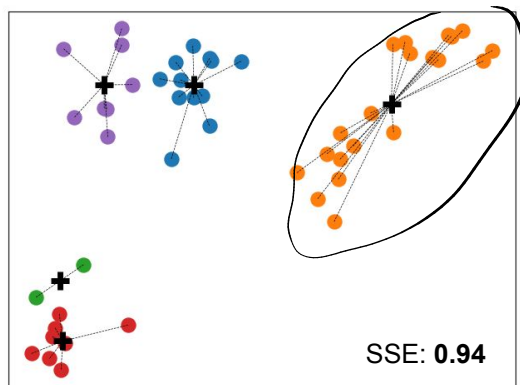
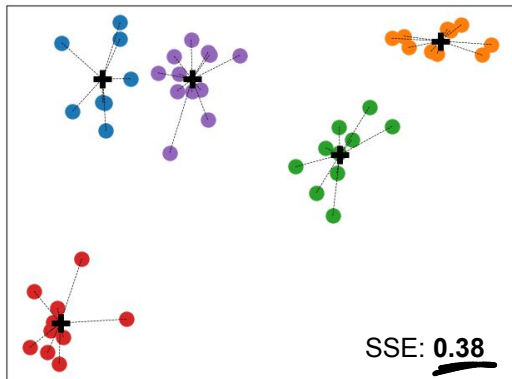
# K-Means — Limitations (Data Distribution)

- Potential workaround: Choose large( $r$ ) value for  $K$ 
  - Intuition: split natural clusters into multiple "well-behaved" (blob-like) subclusters
  - Apply suitable postprocessing steps to merge subclusters



# K-Means — Limitations (Initial Centroids Issue)

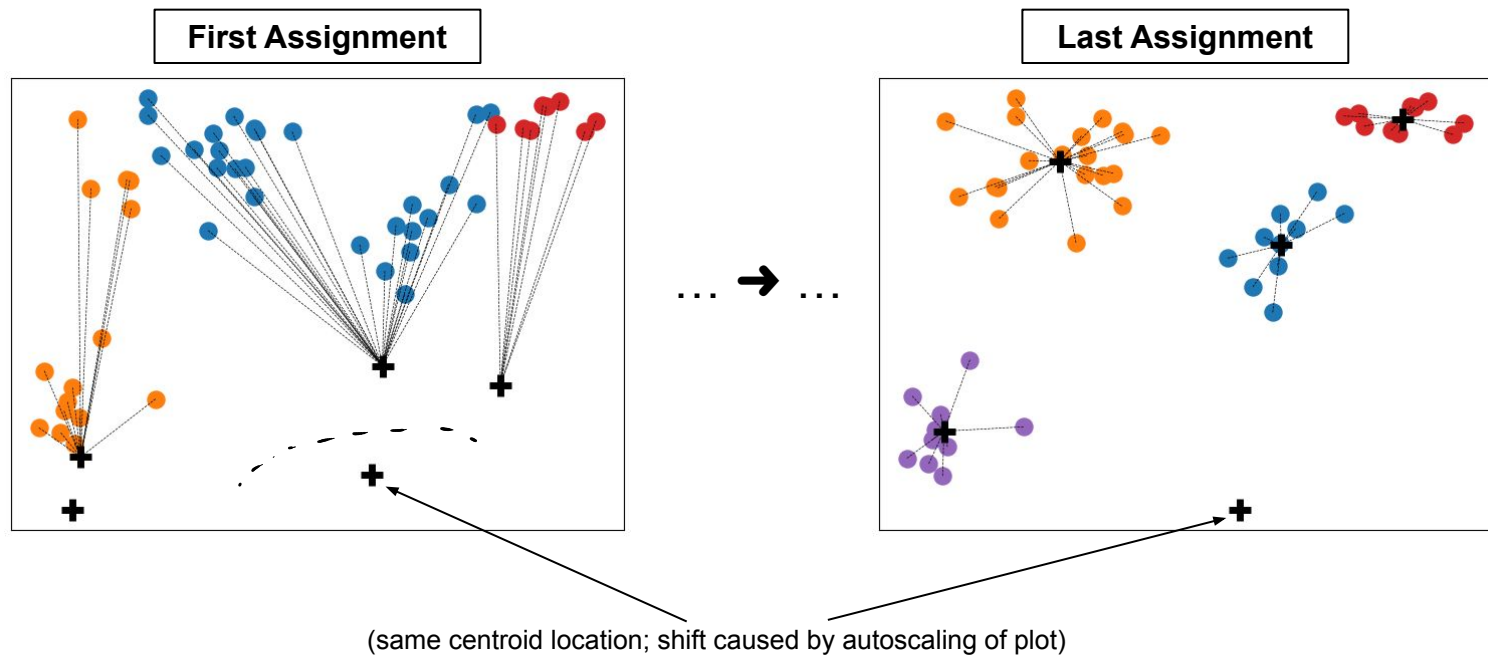
- Different initializations of centroids may yield different clusterings
  - Different clusterings typically have different SSEs → global minimum!



# K-Means — Limitations (Initial Centroids Issue)

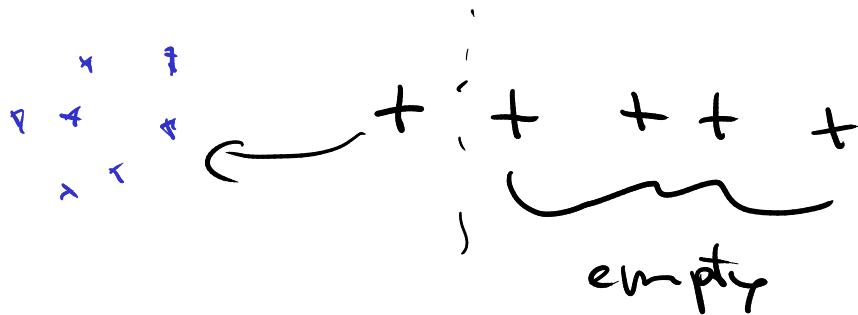
- Some initialization of centroids can yield empty clusters
  - Occurs when a centroid is "blocked off" data points by other centroids

$K=5$



# Quick Quiz

What is the **maximum number of empty clusters** with K-Means with a really bad initialization of the centroids and  $K \geq 2$ ?



**A**

0

**B** ✓

$K - 1$

**C**

$K$

**D**

$K + 1$



# K-Means — Handling Empty Clusters

- Artificially fill empty clusters after assignment step (and continue iterations)
  - Replace empty cluster with point that contributes most to SSE
  - Replace empty cluster with a point from the cluster with the highest SSE
- Post processing
  - Split "loose" clusters = clusters with very high SSE
- Modification of Lloyd's algorithm → K-Means variants
  - Typically aim to address the initial centroids issue

# K-Means Variants — K-Means++

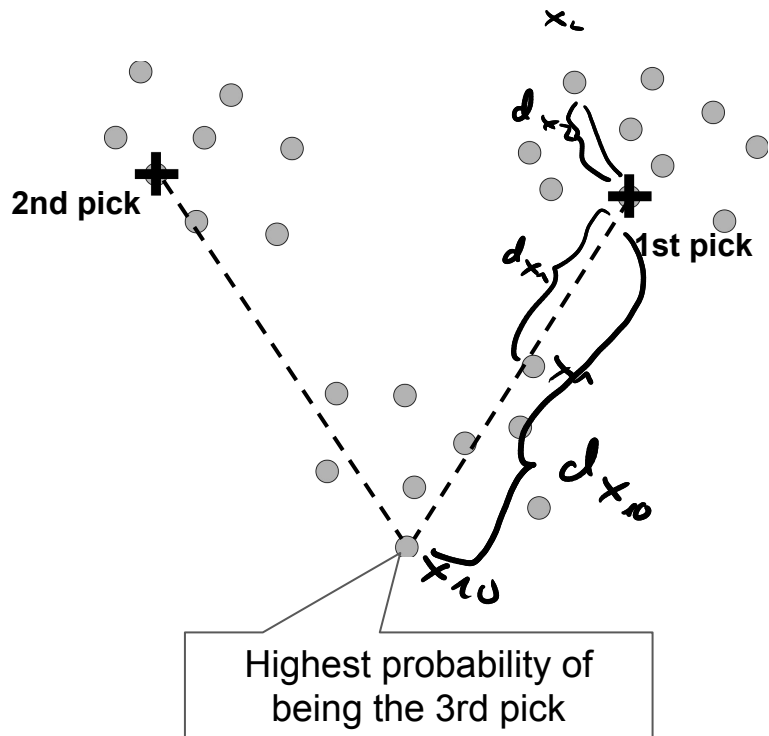
- Only changes initialization of centroids

(assignment/update steps remain the same)

- Goal: spread out centroids
- Better performance in practice
- Theoretical guarantees

- Initialization process

1. Pick random point as first centroid  $c_1$
  2. Repeat
    - 2a) For each point  $x$ , calculate distance  $d_x$  to nearest existing centroid
    - 2b) Pick random point for next centroid with probability proportional to  $d_x^2$
- Until K centroids have been picked



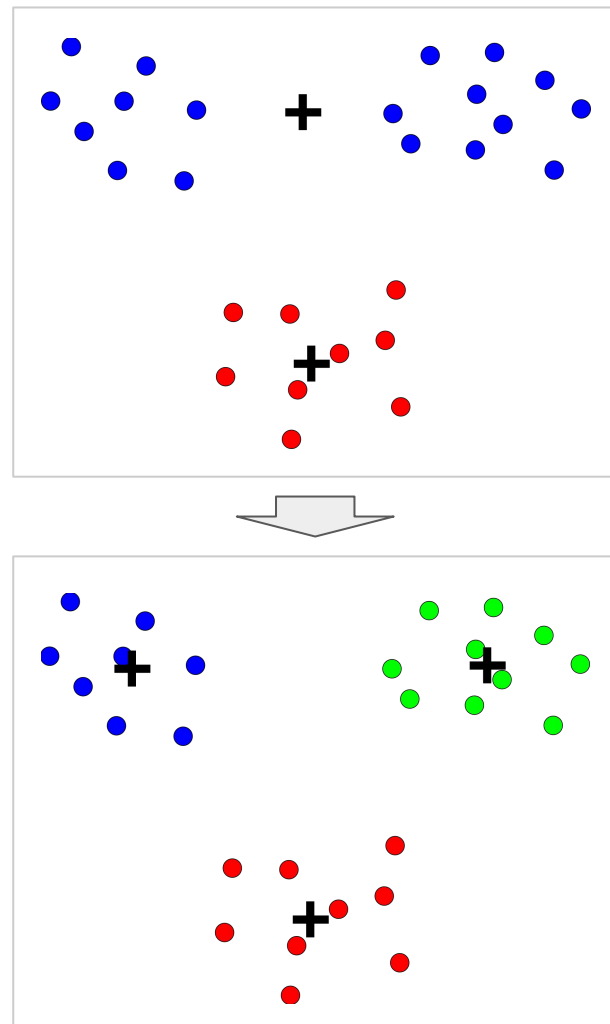
# K-Means Variants — X-Means

- Automatic method to choose K

- Run K-Means with K=2
- Iteratively, run K-Means with K=2 over each subcluster
- Split subcluster only if meaningful w.r.t. a scoring function

- Example scoring functions

- Bayesian Information criterion (BIC)
- Akaike information criterion (AIC)
- Minimum Description Length (MDL)



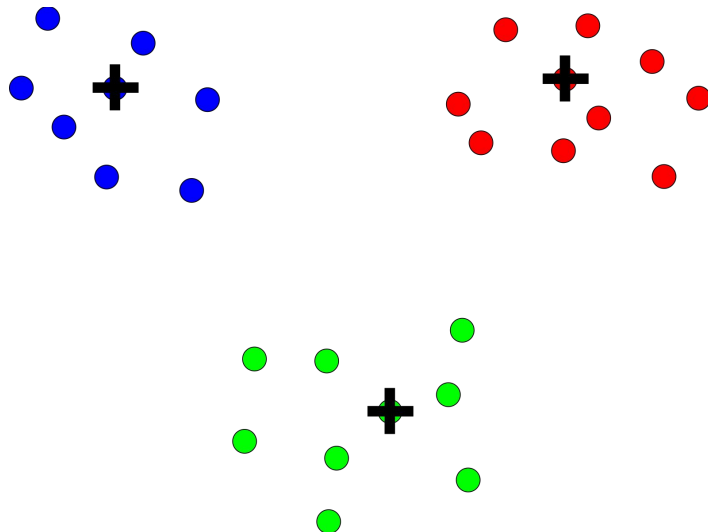
# K-Means Variants — K-Medoids

- Restriction: centroids are chosen from the data points

- Does not require the calculation of the averages  
(no average of sets or more complex objects)
- Only notion of distance or similarity still needed  
(e.g., Jaccard similarity for sets or custom metrics)
- More robust to noise and outliers

- Main issue: performance

- More expensive Update step
- Swap medoid with each point in cluster and calculate change in cost (e.g., SSE)
- Choose the point as new medoid that minimizes the cost after swapping



# Quick Quiz

Can a theoretically optimal initialization of centroids **guarantee** no empty clusters in any case of running K-Means?

$k > \# \text{ unique data points}$

**A**



No

**B**

Yes

**C**

Impossible  
to say

**D**

Unlikely

# Outline

- Clustering
  - Overview
  - Applications
  - Concepts
- Clustering Algorithms
  - K-Means
  - **DBSCAN**
  - Hierarchical Clustering
- Cluster Evaluation

# DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

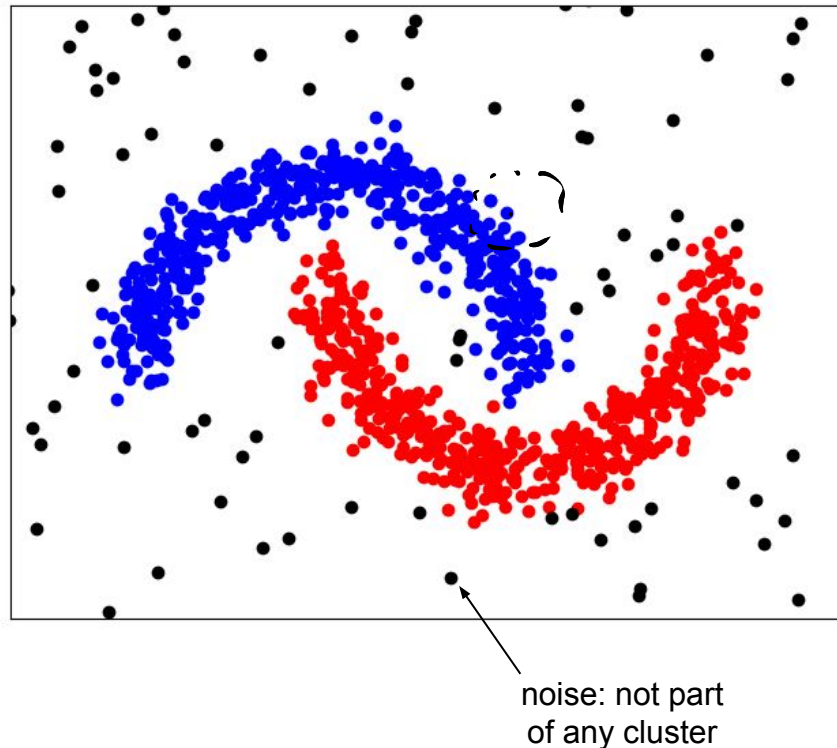
- Basic characteristics

- Clusters: density-based
- Clustering: partitional, exclusive, partial

- Inputs (for d-dimensional Euclidean space)

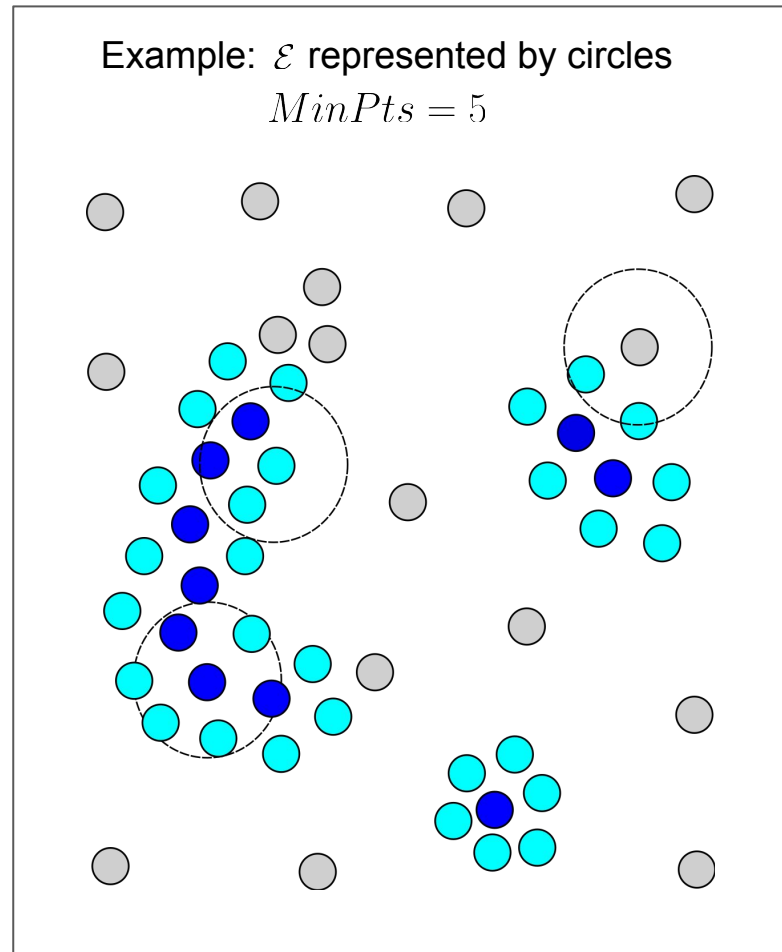
- $(x_1, x_2, \dots, x_N), x_i \in R^d$
- $\varepsilon$  — radius defining a points neighborhood
- *MinPts* — minimum number of points

$$density = \frac{mass}{volume} = \frac{MinPts}{\varepsilon}$$



# DBSCAN — Types of Points

- Core points ●
  - All points with at least  $MinPts$  neighbors with radius  $\varepsilon$  (this includes the point itself!)
  - Form the **interior** of a cluster
- Border Points ●
  - Non-core points with at least one core point in their neighborhood
  - Form the **border** of a cluster
- Outliers / noise ●
  - All other points
  - Default node type





# DBSCAN — Algorithm (2 Iterative Phases)

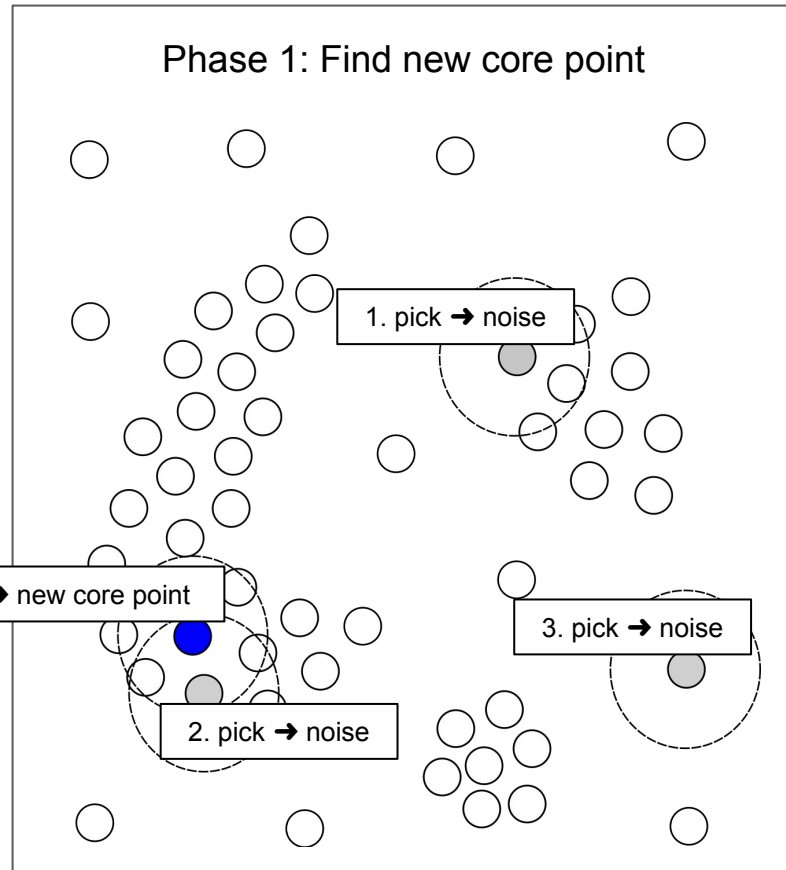
## 1. Find new cluster seed (core point)

Repeat

- 1a) Pick random unexplored point  $x$
- 1b) If  $x$  has less than  $MinPts$  neighbors within  $\epsilon$ , label  $x$  as noise (might change later)

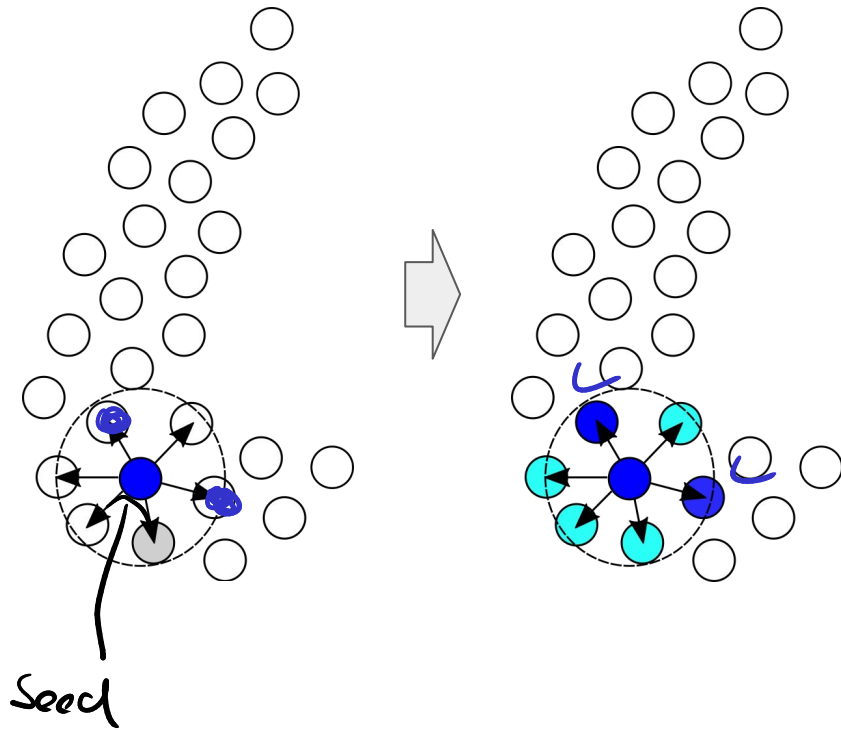
Until  $x$  is a new core point

## 2. Explore new cluster



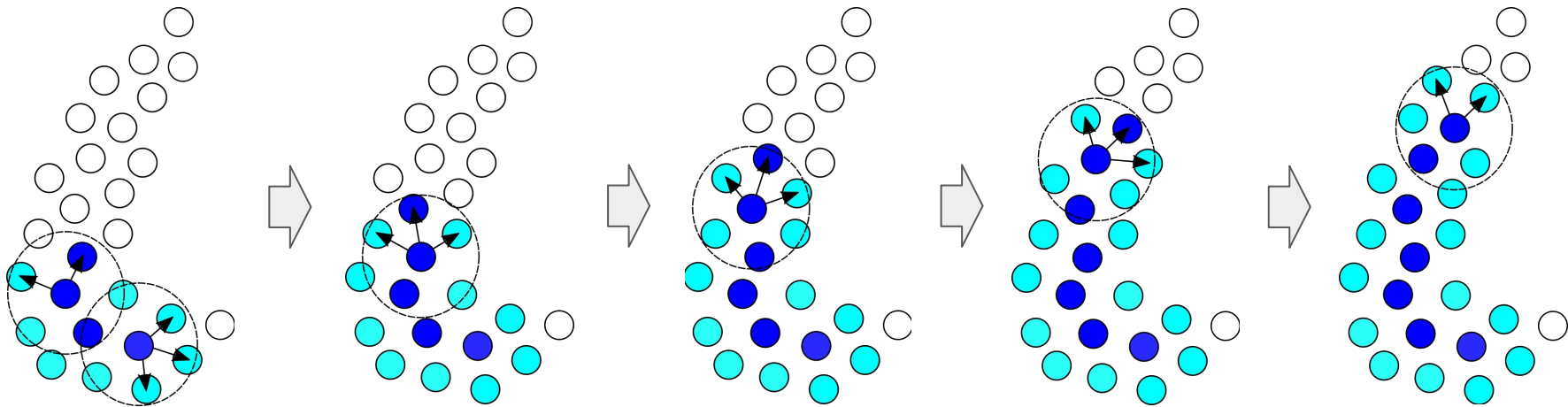
# DBSCAN — Algorithm (Cluster Exploration)

- Explore all neighbors of core point
  - Put all neighbors into the same cluster
  - A neighbor is either a core point or a border point (never noise)
  - New type of neighbor might overwrite previous noise label
- Recursively repeat for each newly found core point



# DBSCAN — Algorithm (Cluster Exploration)

- Example: complete exploration of a cluster
  - Further exploration stops at border points
  - Points beyond border points will become noise or part of a new cluster (after Phase 1)



# DBSCAN — Algorithm (Cluster Exploration)

- **Input:** newly found core  $x_c$  point signifying a new cluster

$S \leftarrow \text{get\_neighbors}(x_c, \mathcal{E})$

*seed*

WHILE  $|S| \neq 0$

$s \leftarrow S.\text{pop}()$

IF  $s.\text{label} = \text{"noise"}$  THEN

$s.\text{label} \leftarrow x_c.\text{cluster\_id}$

IF  $s.\text{label} \neq \text{"unknown"}$  THEN

CONTINUE

$s.\text{label} \leftarrow x_c.\text{cluster\_id}$

$\text{neighbors} \leftarrow \text{get\_neighbors}(s, \mathcal{E})$

IF  $|\text{neighbors}| \geq \text{MinPts}$  THEN

$S \leftarrow S \cup \text{neighbors}$

Initialize  $S$  with the the neighbors of  $x_c$

Pick next point  $s$  from  $S$  until  $S$  is empty

If point  $s$  is (currently) noise,  
add point to current cluster

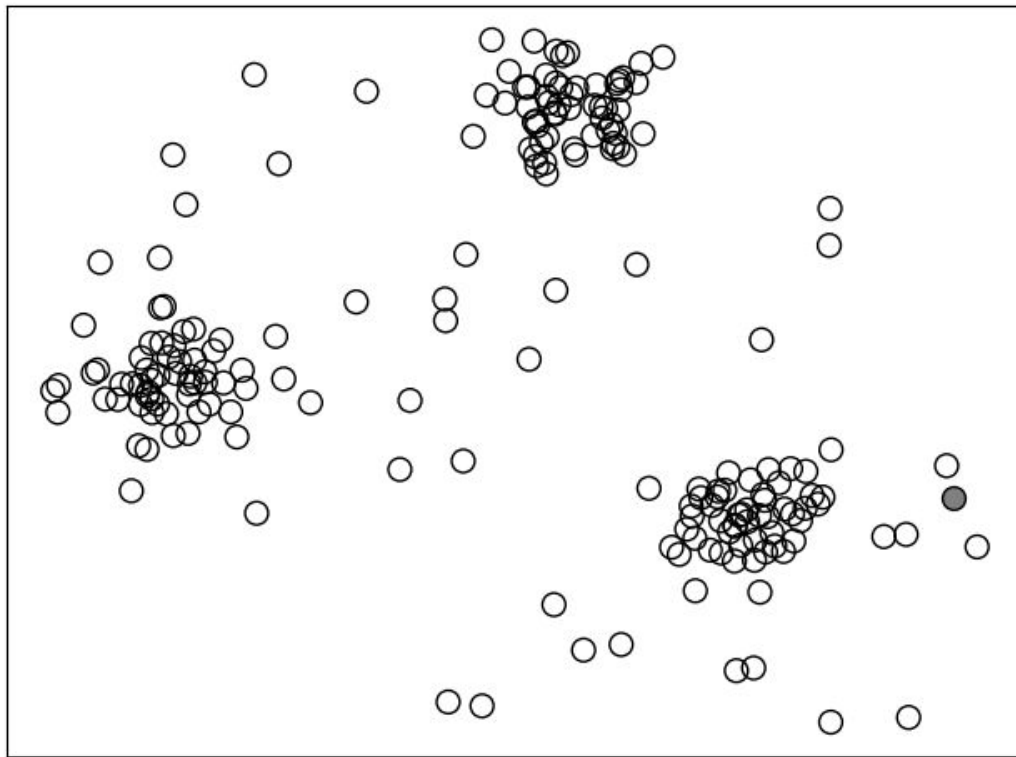
If  $s$  has already been explored,  
continue with next point

Add  $s$  current cluster ( $s$  so far unexplored)

Get all neighbors of  $s$

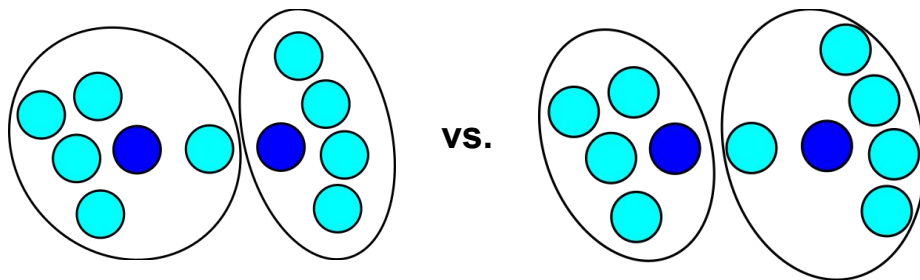
If  $s$  has more than  $\text{MinPts}$  neighbors,  $s$  is also a core point  
→ add neighbors to  $S$  (so they will be explored as well)

# DBSCAN — Example with Visualization



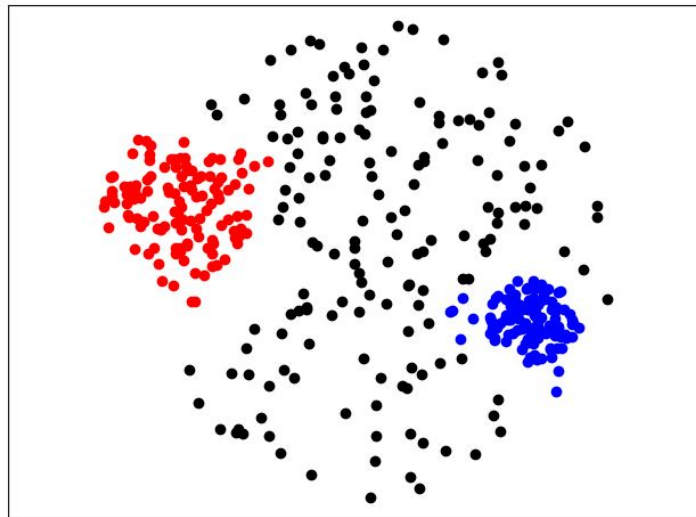
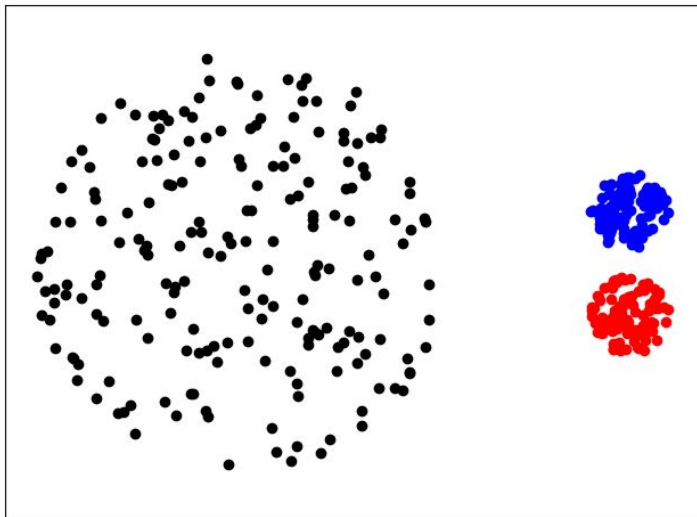
# DBSCAN — Characteristics

- DBSCAN always converges
  - Each data point gets explored (either in Phase 1 or 2)
  - A data point does not change its type (only exception: noise → border)
- DBSCAN is not completely deterministic
  - Phase 1 introduces randomness
  - Border points may be reachable from core points of different clusters
  - Noise and core points deterministic



# DBSCAN — Limitations

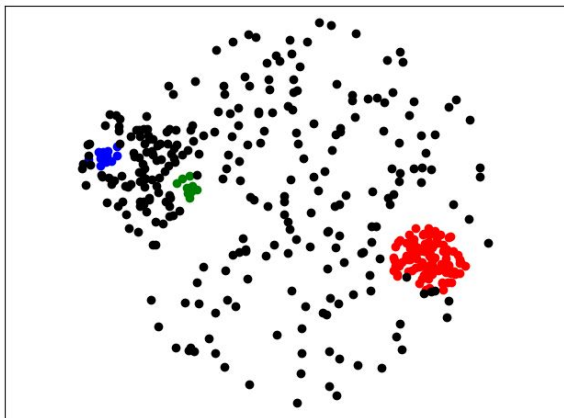
- DBSCAN cannot handle different densities



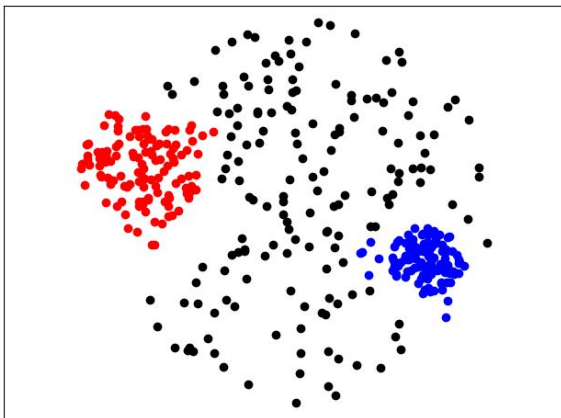
# DBSCAN — Limitations

- DBSCAN is generally very sensitive to parameters
  - Choosing  $\varepsilon$  and  $MinPts$  requires good understanding of data and context

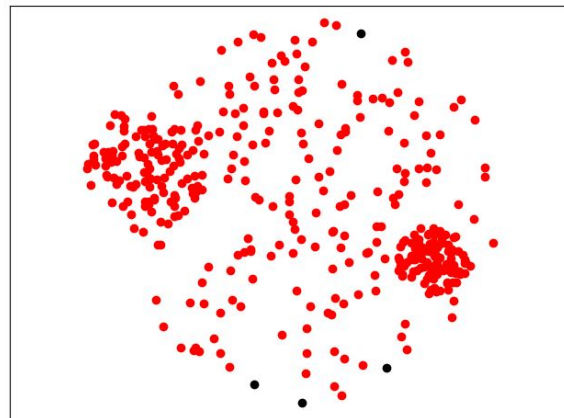
$\varepsilon = 0.05$



$\varepsilon = 0.1$



$\varepsilon = 0.2$

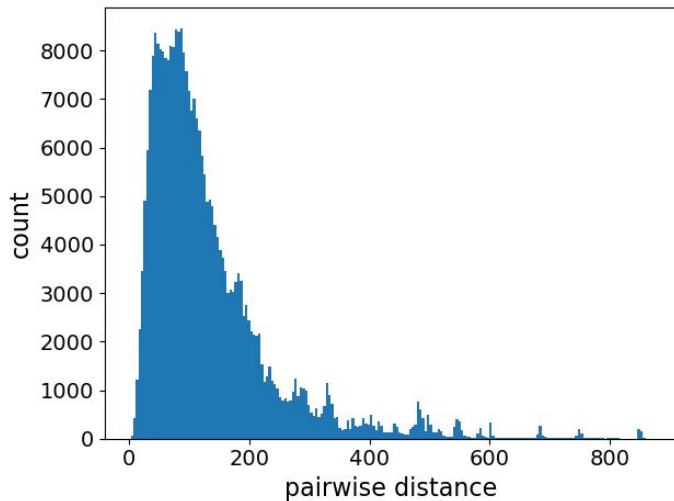


( $MinPts = \underline{10}$  for all three examples)



# DBSCAN — How to Choose Parameter Values?

- Informed by results of EDA, e.g.:



Distribution of all pairwise distances

First insights into suitable values for  $\varepsilon$

- Density of data points has intuitive semantic meaning, e.g.:
  - Geographic distance between bars in a city
  - Task: Find areas (clusters) with more than 10 bars within 500m

# DBSCAN — How to Choose Parameter Values?

- Intuitive interpretations of meaningful parameter values

- Example

- 141 McDonald's restaurants across Singapore

- Find areas with more than 5 restaurants within a 500m radius

*Min Pts*



*$\epsilon$*



# Quick Quiz

What is the **smallest cluster size**  
when using DBSCAN and a given  
value for *MinPts*? > 5

~~DBSCAN~~

A

1

B

*MinPts* - 1

C ✓

*MinPts*

D (✓)

*MinPts* + 1

# Quick Quiz

*no duplicate*

Given  $N=1,000$  data points and using DBSCAN with  $MinPts=10$ , what is the **minimum** possible number of clusters?

**A** ✓

0

**B**

1

**C**

MinPts

**D**

MinPts + 1

# Quick Quiz — Side Note

- Slight inconsistencies across different sources

- Relevant step in algorithm

$neighbors \leftarrow get\_neighbors(s, \mathcal{E})$

Does *neighbors* contain *s* itself?

Yes

- Original paper: Yes, *s* is part of neighborhood

- Smallest cluster size:  $MinPts$

- Maximum number of clusters:  $\left\lfloor \frac{N}{MinPts} \right\rfloor$   $N = \text{number of data points}$

# Clustering Algorithms

- K-Means
- DBSCAN
- **Hierarchical Clustering** (next lecture)



# Outline

- Clustering
  - Overview
  - Applications
  - Concepts
- Clustering Algorithms
  - K-Means
  - DBSCAN
  - **Hierarchical Clustering** (next lecture)
- Cluster Evaluation

# Summary — Clustering I

- Clustering as fundamental data mining algorithm
  - Cluster provide a "meso-view" on data
  - Required: well-defined notion of similarity between data points
  - No single definition what a good cluster / clustering is

→ Wide range of different clustering algorithms

- 
- In this lecture: K-Means & DBSCAN
    - K-Means: split all data points into k clusters based in their **relative similarities**
    - DBSCAN: find clusters based on **absolute similarities** between data points
- 



# Solutions to Quick Quizzes

- Slide 18: D
- Slide 32: B
- Slide 37: A
- Slide 51: C
- Slide 52: A