Machine learning

Clustering: Basic Concepts and Algorithms

Mohammad-Reza A. Dehaqani

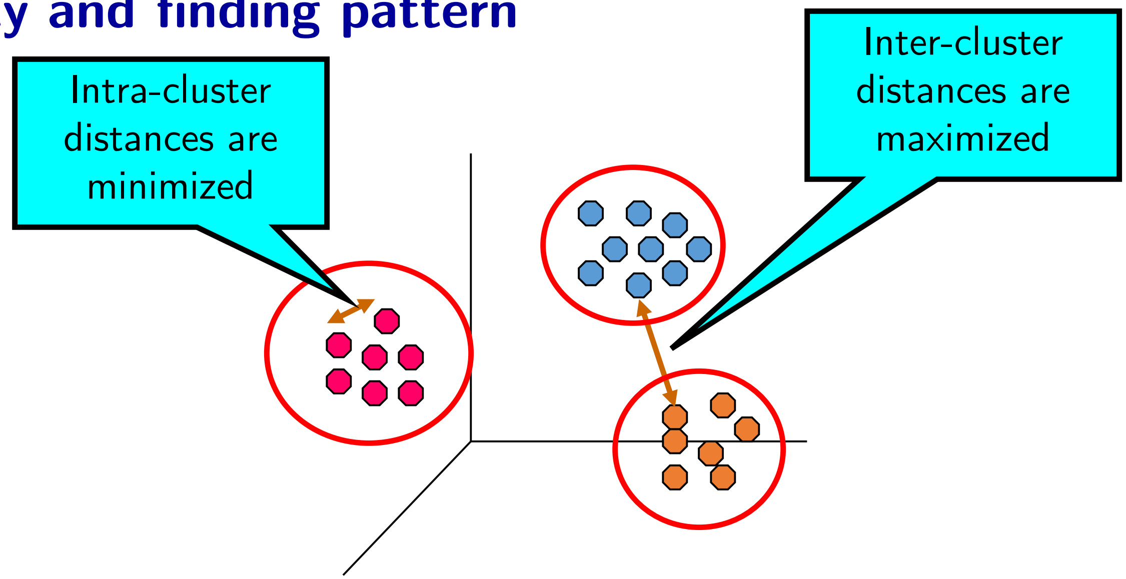
dehaqani@ut.ac.ir

Why cluster?

- **Unsupervised** method
- Labeling is **expensive**
- Gain insight into the **structure** of the data
- Find **prototypes** in the data

What is Cluster Analysis?

- Finding groups of objects such that the objects **in a group** will be **similar** (or related) to one another and **different from** (or unrelated to) the objects in **other groups**
- **Similarity and finding pattern**



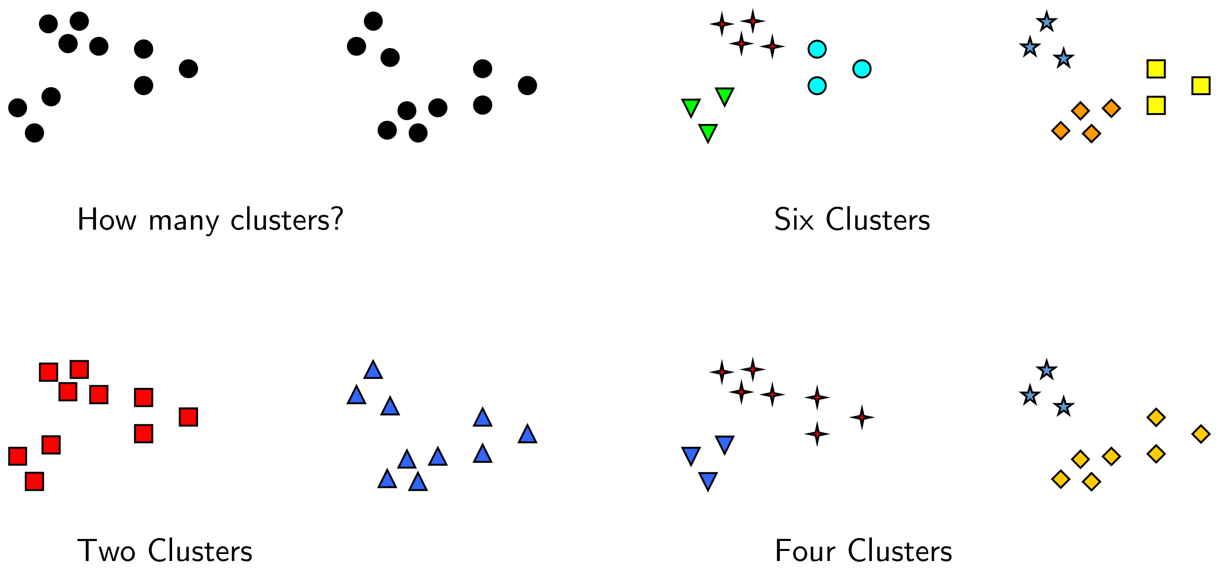
Quality: What Is Good Clustering?

- A **good clustering** method will produce high quality clusters with
  - **high intra-class** similarity
  - **low inter-class** similarity
- The **quality** of a clustering result depends on both the **similarity measure** used by the method and its **implementation**
- The **quality** of a clustering method is also measured by its ability to discover some or all of the **hidden patterns**

Dissimilarity/Similarity metric

- **Dissimilarity/Similarity metric**:
  - Similarity is expressed in terms of a **distance function**, typically metric: $d(i, j)$
- The definitions of **distance functions** are usually very different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables.
- **Weights** should be associated with **different variables** based on applications and data semantics.
- It is hard to define "similar enough" or "good enough"
  - the answer is typically **highly subjective**.

Notion of a Cluster can be Ambiguous



Data Structures

- **Data** matrix

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{id} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{nd} \end{bmatrix}$$

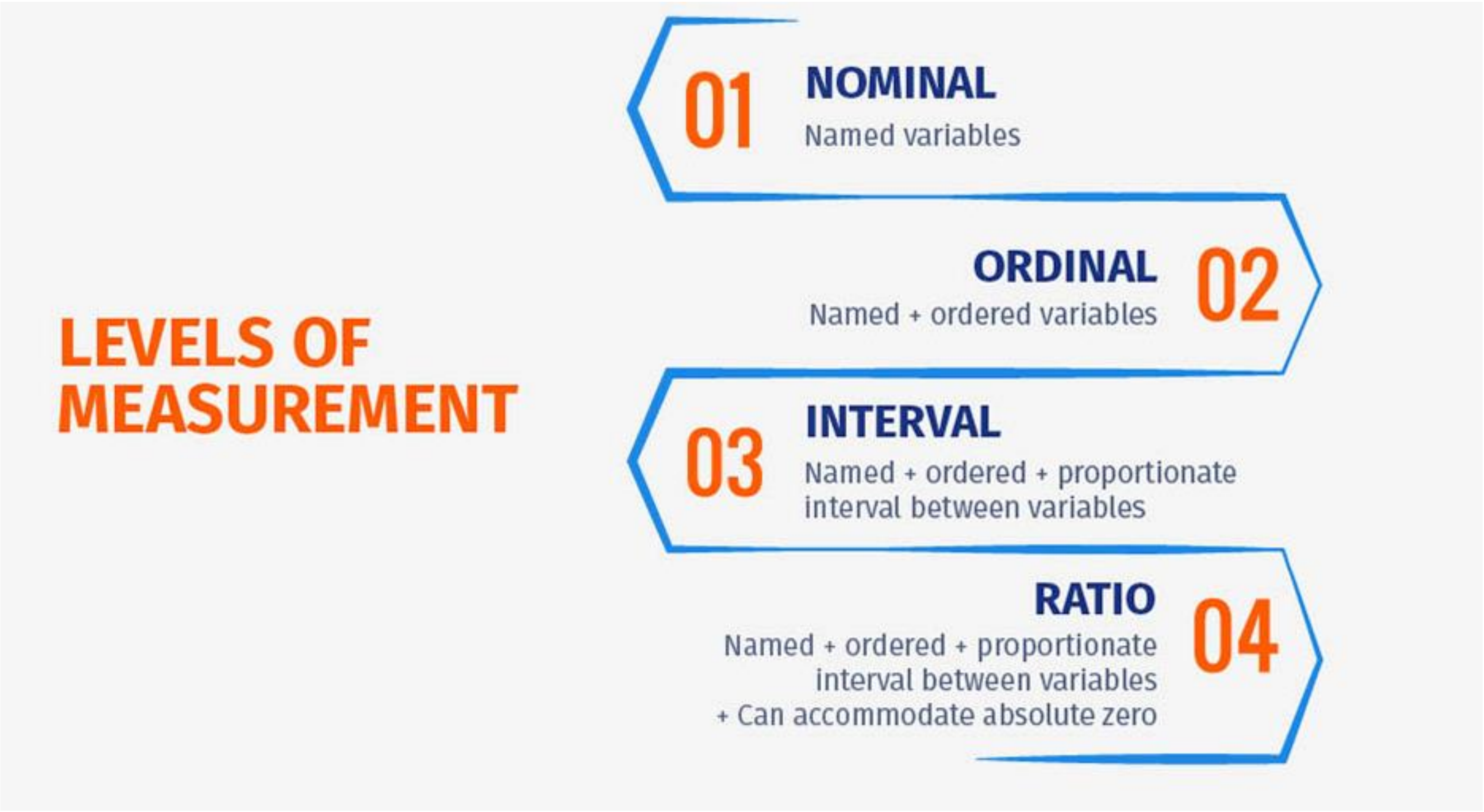
- **Dissimilarity** matrix

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \ldots & \ldots & 0 \end{bmatrix}$$

Type of data in clustering analysis

- Interval-scaled variables
  - You cannot calculate a ratio between them (arbitrary zero-point)
  - e.g. temperature in Celsius
- Binary variables
- Nominal and ordinal
- Ratio variables
  - It has **all the characteristics** of an interval scale, in addition, to be able to calculate ratios (absolute zero or character of origin)
  - e.g. age, weight, height,
- Variables of mixed types

Type of data	Nominal	Ordinal	Interval	Ratio
The sequence of variables is established	–	Yes	Yes	Yes
Mode	Yes	Yes	Yes	Yes
Median	–	Yes	Yes	Yes
Mean	–	–	Yes	Yes
Difference between variables can be evaluated	–	–	Yes	Yes
Addition and Subtraction of variables	–	–	Yes	Yes
Multiplication and Division of variables	–	–	–	Yes
Absolute zero	–	–	–	Yes



Interval-valued variables

- Standardize data
  - Calculate the **mean absolute deviation**:

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|\right)$$

where $m_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \dots + x_{nf}\right)$.

  - Calculate the standardized measurement (*z-score*)

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- Using mean absolute deviation **is more robust** than using **standard deviation**

Similarity and Dissimilarity Between Objects

- **Distances** are normally used to measure the **similarity** or **dissimilarity** between two data objects
- Some popular ones include: ***Minkowski distance***

$$d(i,j)=\sqrt[q]{\left(|x_{i_1}-x_{j_1}|^q+|x_{i_2}-x_{j_2}|^q+\ldots+|x_{i_d}-x_{j_d}|^q\right)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{id})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jd})$ are two p-dimensional data objects, and q is a positive integer

- If $q = 1$, d is **Manhattan distance**

$$d(i,j)=|x_{i_1}-x_{j_1}|+|x_{i_2}-x_{j_2}|+\ldots+|x_{i_d}-x_{j_d}|$$

Similarity and Dissimilarity Between Objects

- *If $q = 2$, d* is Euclidean distance

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{id} - x_{jd}|^2)}$$

  - Properties
    - $d(i,j) \geq 0$
    - $d(i,i) = 0$
    - $d(i,j) = d(j,i)$
    - $d(i,j) \leq d(i,k) + d(k,j)$
- Also, one can use **weighted distance**, **Pearson correlation coefficient**, or other dissimilarity measures

Binary Variables

- A contingency table for binary data

		Object j		
		1	0	sum
Object i	1	a	b	$a+b$
	0	c	d	$c+d$
	sum	$a+c$	$b+d$	p

- Distance measure for symmetric binary variables:

$$d(i,j) = \frac{b+c}{a+b+c+d}$$

- Distance measure for asymmetric binary variables:

$$d(i,j) = \frac{b+c}{a+b+c}$$

- Jaccard coefficient (***similarity*** measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i,j) = \frac{a}{a+b+c}$$

Nominal Variables

- A **generalization of the binary** variable in that it can take more than 2 states, e.g., red, yellow, blue, green
- Method 1: **Simple matching**
  - m: # of matches, p: total # of variables

$$d(i,j) = \frac{p-m}{p}$$

- Method 2: use a **large number** of binary variables
  - creating a **new binary variable** for each of the M nominal states

Ordinal Variables

- An ordinal variable can be **discrete** or **continuous**
  - **Order** is important, e.g., rank
- Can be treated like interval-scaled
  - replace x_{if} by **their rank** $r_{if} \in \{1, \ldots, M_f\}$
  - map the range of each variable onto [0, 1] by replacing i-th object in the f-th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

  - compute the **dissimilarity** using methods for interval-scaled variables

Variables of Mixed Types

- A database may contain all types of variables
  - symmetric binary, asymmetric binary, nominal, ordinal, interval and ratio
- One may use a **weighted formula** to combine their effects

$$d(i,j) = \frac{\sum_{f=1}^{d} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{d} \delta_{ij}^{(f)}}$$

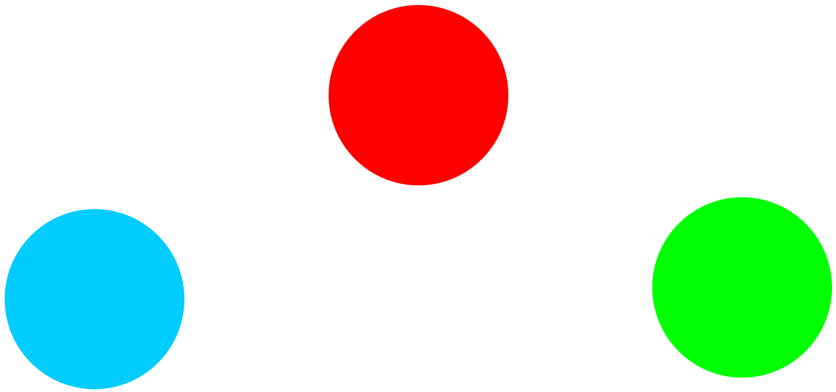
  - f is binary or nominal:

    $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ otherwise
  - f is interval-based: use the normalized distance
  - f is ordinal or ratio-scaled
    - compute ranks r_{if} and
    - and treat z_{if} as interval-scaled

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

Types of Clusters: Well-Separated

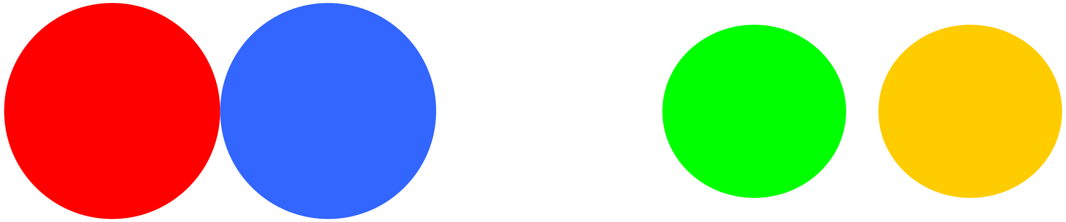
A cluster is a set of points such that any point in a cluster is closer (or more similar) **to every other point** in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Center-Based

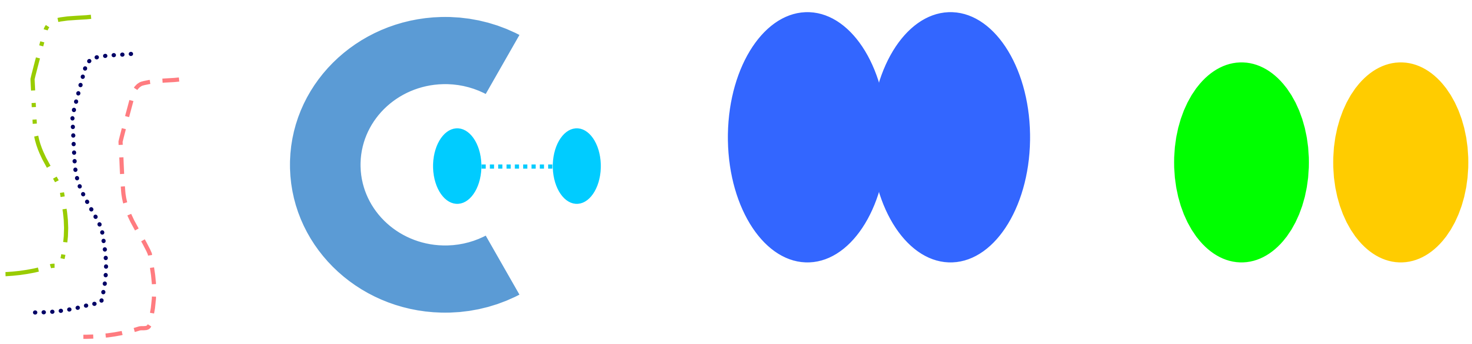
- A cluster is a set of objects such that an object in a cluster is closer (more similar) **to the "center"** of a cluster, than to the center of any other cluster
- The center of a cluster is often a **centroid**, the **average** of all the points in the cluster, or a **medoid**, the most "**representative**" point of a cluster



4 center-based clusters

Types of Clusters: Contiguity-Based

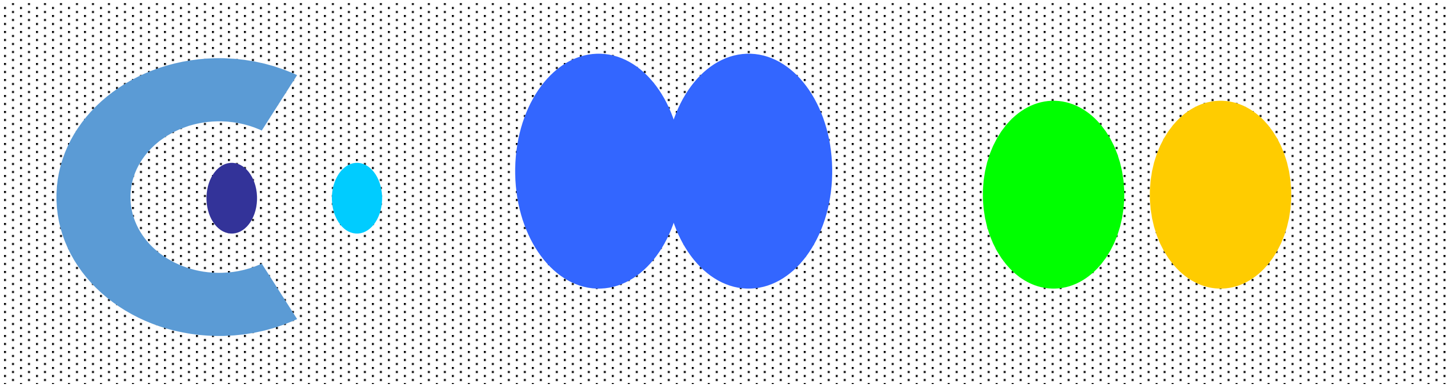
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) **to one or more other points** in the cluster than to any point not in the cluster.



8 contiguous clusters

Types of Clusters: Density-Based

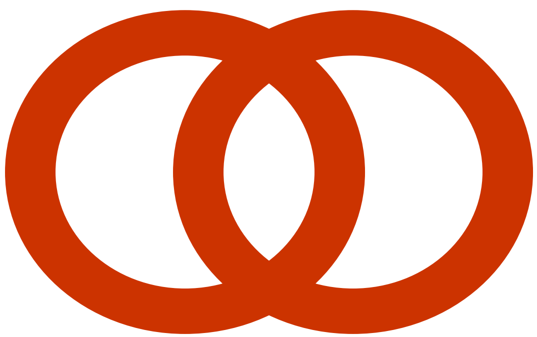
- A cluster is a **dense region of points**, which is separated by low-density regions, from other regions of high density.
- Used when the clusters are **irregular or intertwined**, and when **noise** and outliers are **present**.



6 density-based clusters

Types of Clusters: Conceptual Clusters

- Shared Property or Conceptual Clusters
  - Finds clusters that **share some common property** or represent a particular concept.



2 Overlapping Circles

Types of Clusters: Objective Function

- Clusters Defined by an **Objective Function**
  - Finds clusters **that minimize or maximize** an objective function.
  - Enumerate **all possible ways** of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (**NP Hard**)
  - Can **have global or local objectives**.
    - **Hierarchical** clustering algorithms typically have local objectives
    - **Partitional algorithms** typically have global objectives

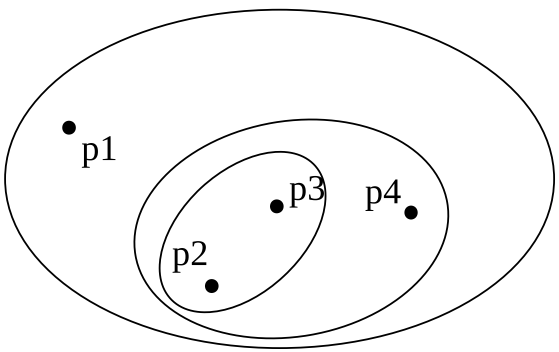
Types of Clusters: Objective Function
Graph approach

- **Map** the clustering problem to **a different domain** and solve a related problem in that domain
  - **Proximity matrix** defines a weighted graph, where the **nodes are the points being clustered**, and the **weighted** edges represent the **proximities** between points
  - Clustering is equivalent to **breaking the graph into connected** components, one for each cluster
  - Want to **minimize the edge weight** between clusters and **maximize the edge weight** within clusters

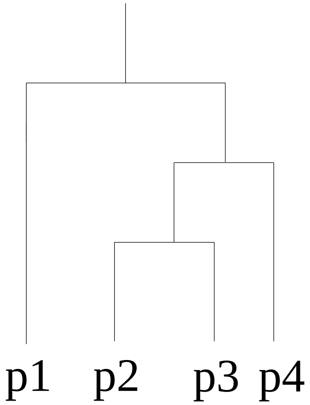
Types of Clustering

- A clustering is a process to find a set of clusters
- Important distinction between **hierarchical** and **partitional** sets of clusters
- **Partitional** Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- **Hierarchical** clustering
  - A set of nested clusters organized as a hierarchical tree
- **Density based** clustering
  - Discover clusters of **arbitrary** shape.
  - Clusters **dense regions** of objects **separated by regions of low density**

Hierarchical Clustering

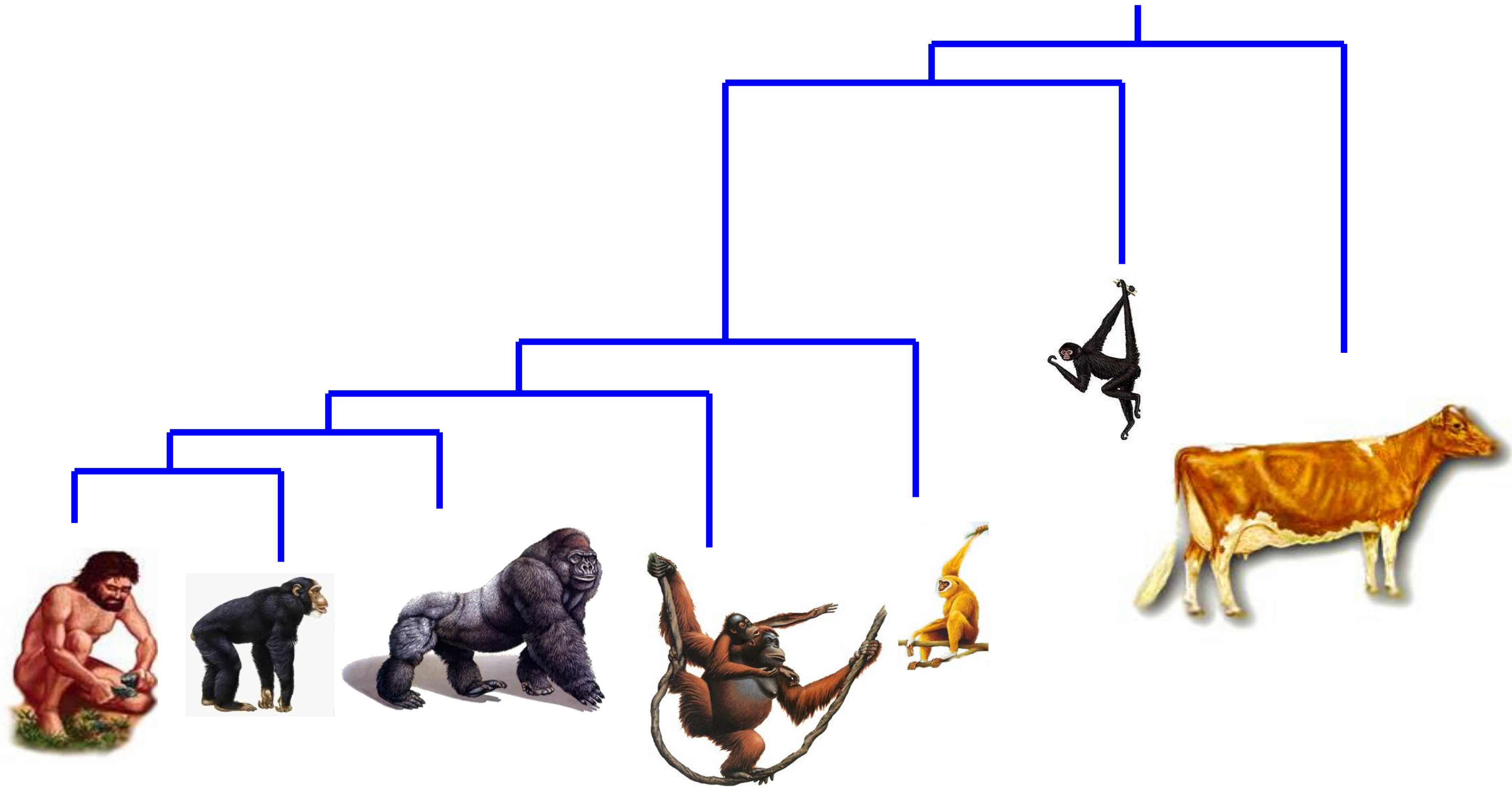


Hierarchical Clustering



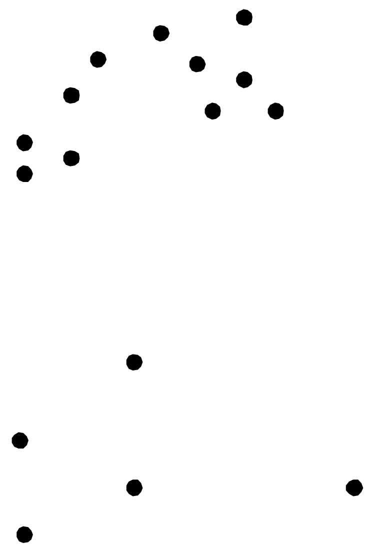
Dendrogram

One dataset that can be perfectly clustered using a hierarchy

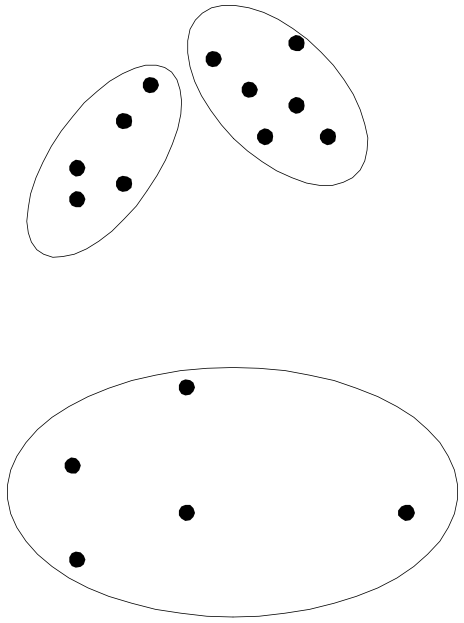


(Bovine:0.69395, (Spider Monkey 0.390, (Gibbon:0.36079,(Orang:0.33636,(Gorilla:0.17147,(Chimp:0.19268, Human:0.11927):0.08386):0.06124):0.15057):0.54939);

Partitional Clustering



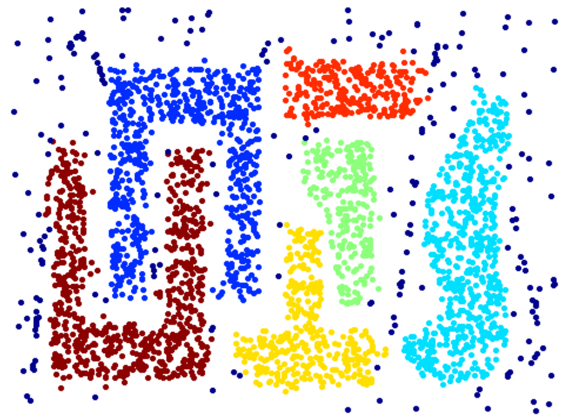
Original Points



A Partitional Clustering

Density based Clustering

Original Points



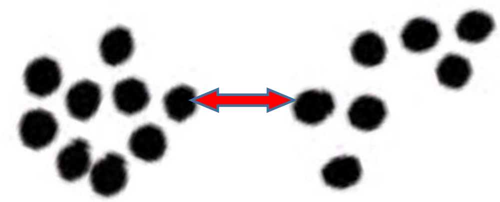
Clusters

Hierarchical Clustering: Bottom-Up Agglomerative

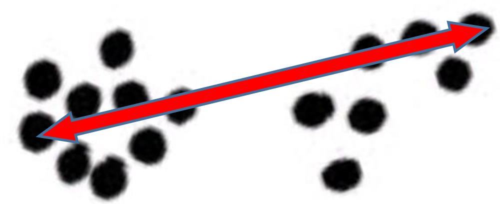
- Clustering starts with **each object in a separate cluster**, and repeat:
  - **Joins** the most similar pair of clusters,
  - **Update** the similarity of the new cluster to others until there is only one cluster.
- **Greedy** – less accurate but simple to implement (there is no way to revise clustering)

Bottom-up Agglomerative clustering

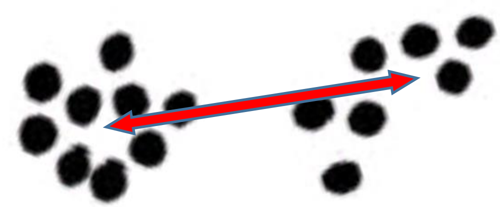
- Different algorithms differ in **how the similarities are defined** (and hence updated) **between two clusters**
- **Single-Linkage**
  - **Nearest** Neighbor: similarity between their closest members.



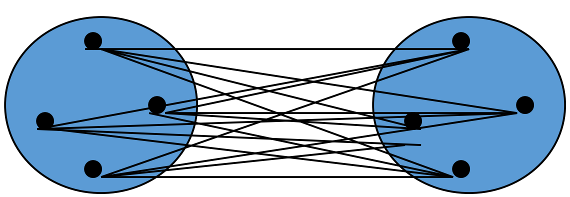
- **Complete-Linkage**
  - **Furthest** Neighbor: similarity between their furthest members.



- **Centroid**
  - Similarity between the **centers of gravity**

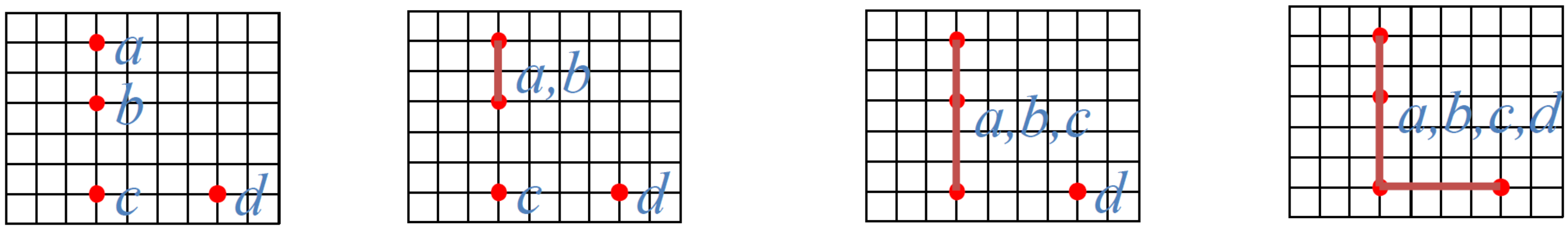


- **Average-Linkage**
  - Average similarity of **all cross-cluster pairs**



Single-Linkage Method

Euclidean Distance



(1) (2) (3)

	b	c	d
a	2	5	6
b		3	5
c			4

	b	c	d
a	2	5	6
b		3	5
c			4

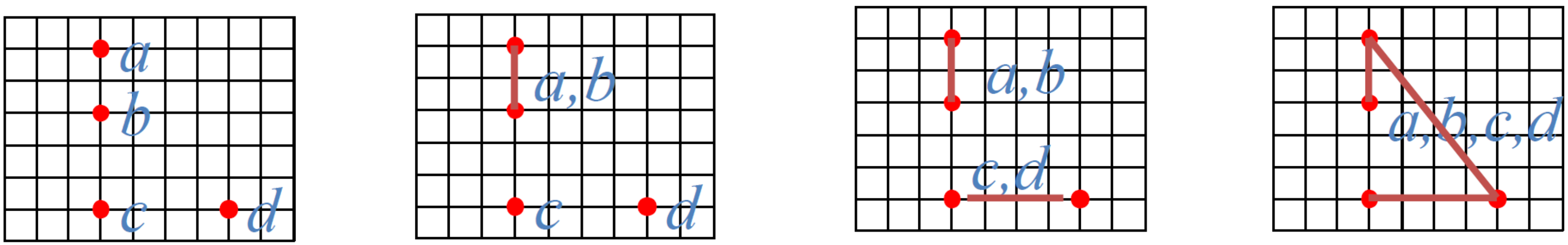
	c	d
a,b	3	5
c		4

	d
a,b,c	4

Distance Matrix

Complete-Linkage Method

Euclidean Distance



(1) (2) (3)

	b	c	d
a	2	5	6
b		3	5
c			4

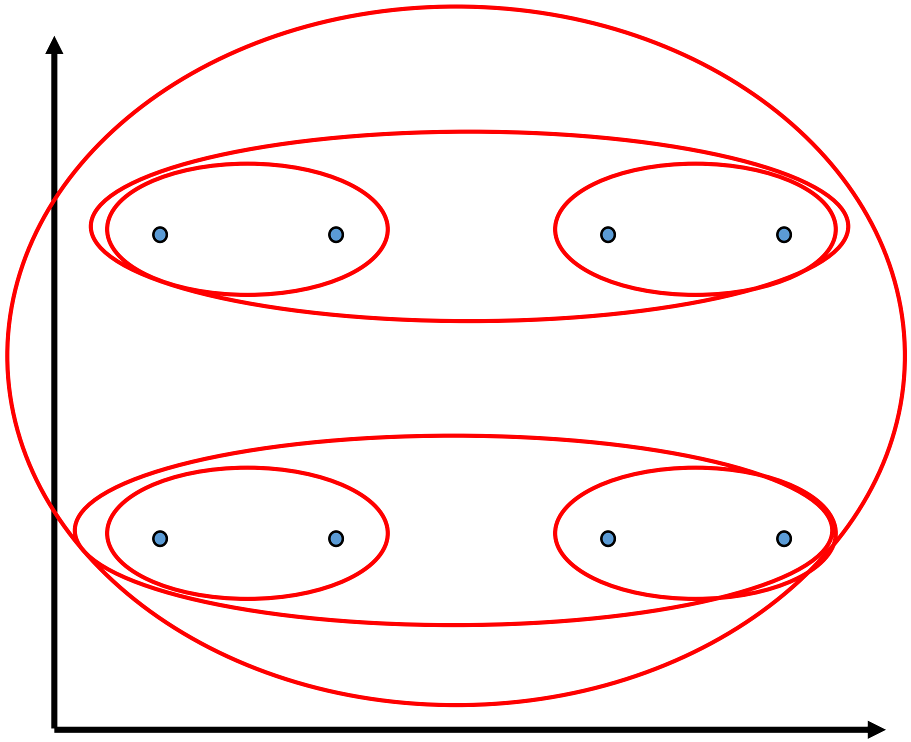
	b	c	d
a	2	5	6
b		3	5
c			4

	c	d
a,b	5	6
c		4

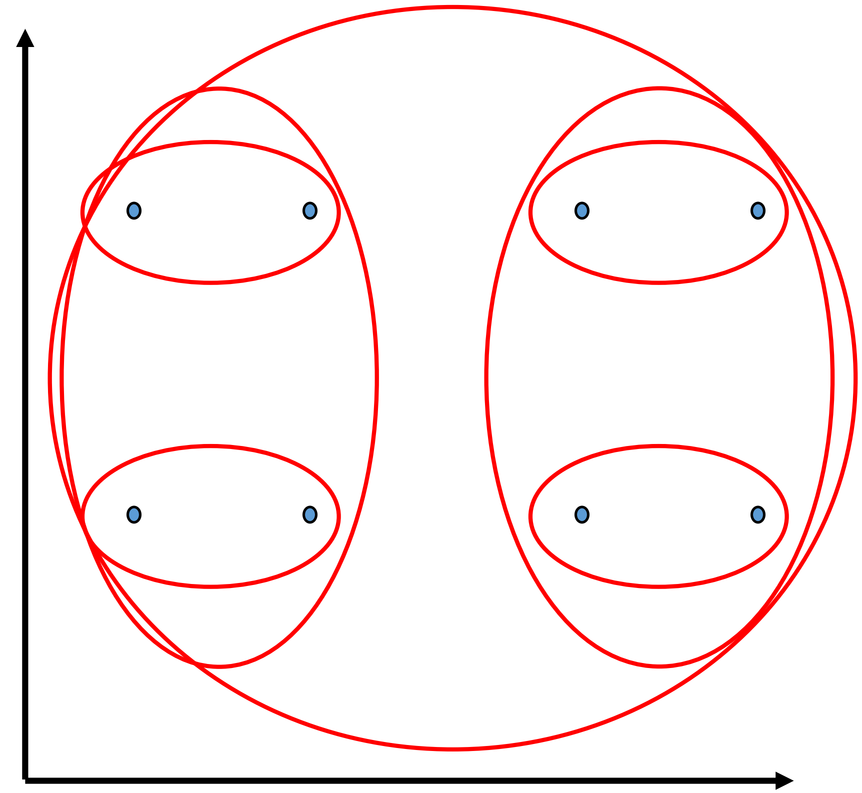
	c,d
a,b	6

Distance Matrix

Single Link

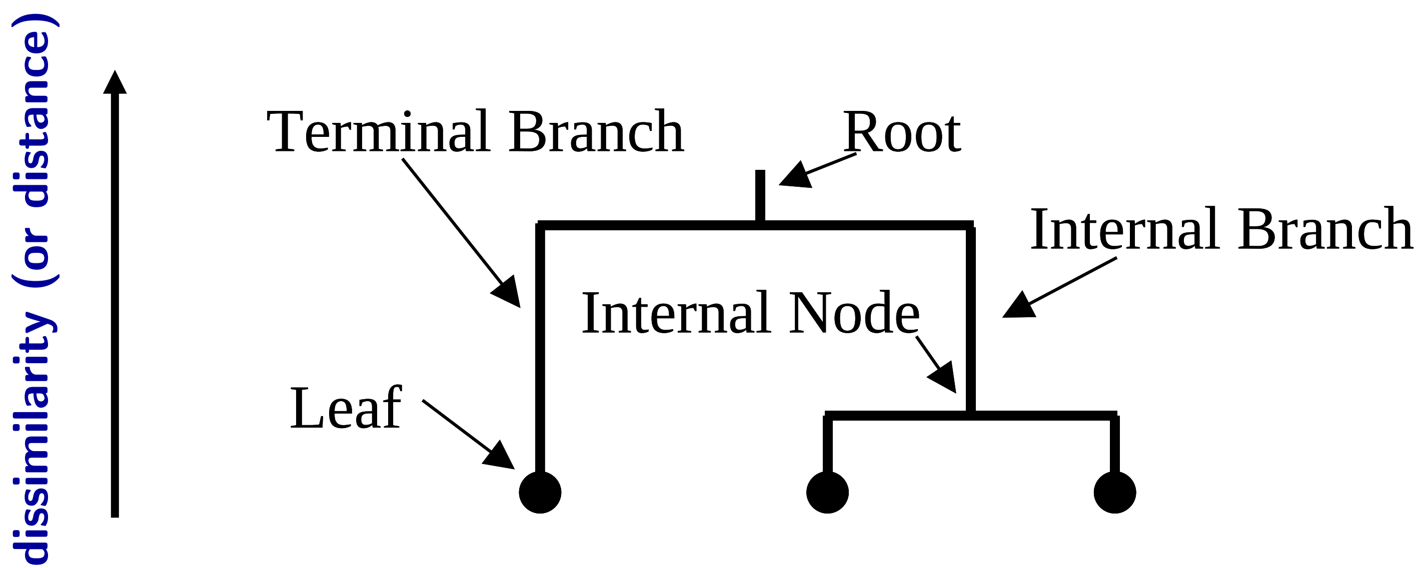


Complete Link

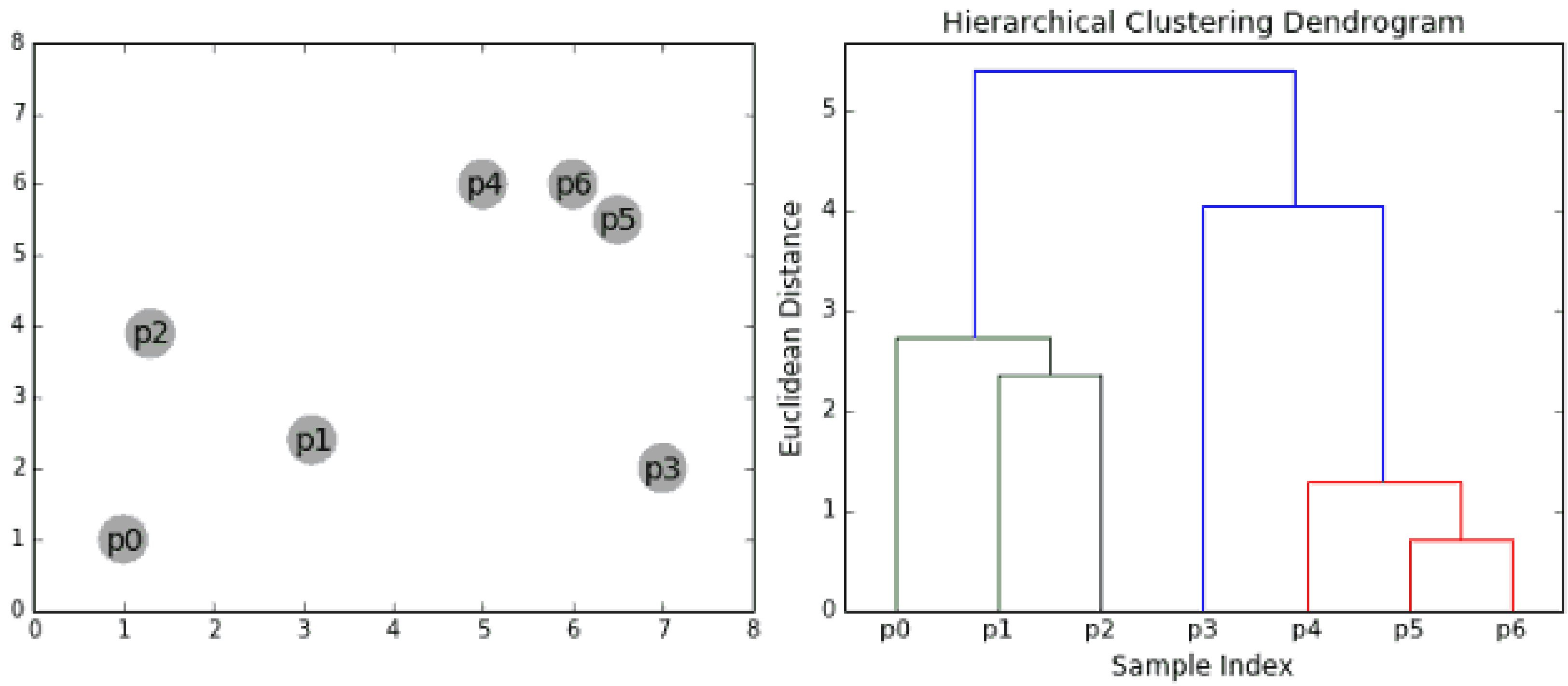


Dendogram

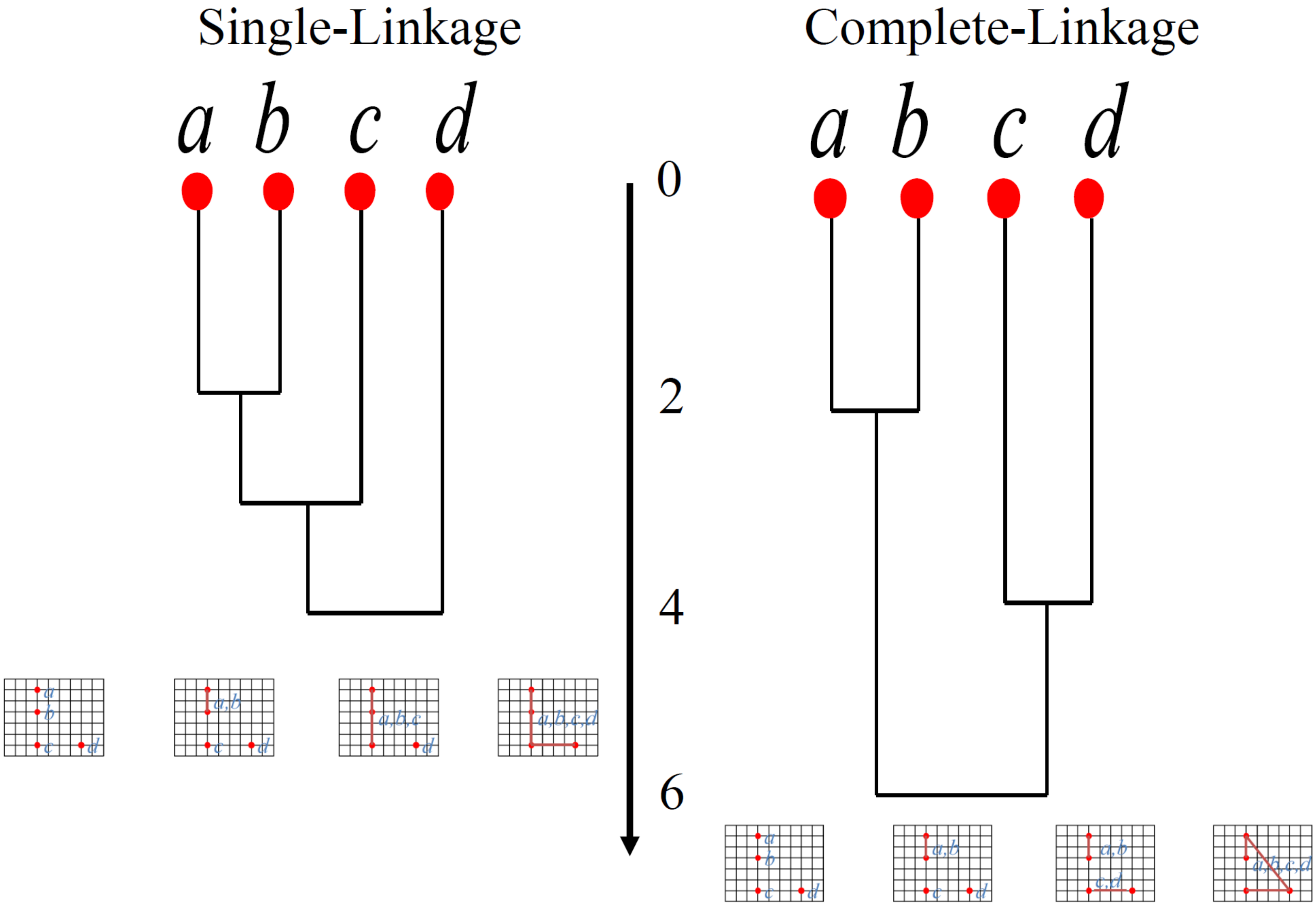
- A Useful Tool for **Summarizing Similarity Measurements**
- Clustering obtained by **cutting the dendrogram** at a desired level: each connected component forms a cluster.
- The dissimilarity (or distance) between two objects in a dendrogram is represented as the height of the lowest internal node they share.



Simple example



Dendrograms



Single vs. Complete Linkage

- Shape of clusters
- Single-linkage:
  - allows anisotropic and non-convex shapes
- Complete-linkage:
  - assumes isotopic, convex shapes

Hierarchical Clustering: Top-Down divisive

- Starts with **all the data** in a single cluster, and repeat:
  - Split each cluster into two using a **partition algorithm**
  - Until each object is a separate cluster.
- More accurate than bottom-up but **complex to implement**

Computational Complexity

- All hierarchical clustering methods need to compute similarity of all pairs of n individual instances which is $O(n^2)$.
- At each iteration,
  - **Sort** similarities to find largest one $O(n^2 \log n)$.
  - **Update** similarity between merged cluster and other clusters.
    - Computing similarity to each other cluster can be done in constant time.
- we get $O(n^2 \log n)$ or $O(n^3)$ (if naïvely implemented)

Partitioning Algorithms

- Partitioning method: **Construct a partition** of n objects into a set of K clusters
  - **Given**: a set of objects and the number K
  - **Find**: a partition of K clusters that optimizes the chosen **partitioning criterion**
- **Globally optimal**: **exhaustively** enumerate all partitions
- Effective **heuristic** method: **K-means algorithm**

K-Means Algorithm

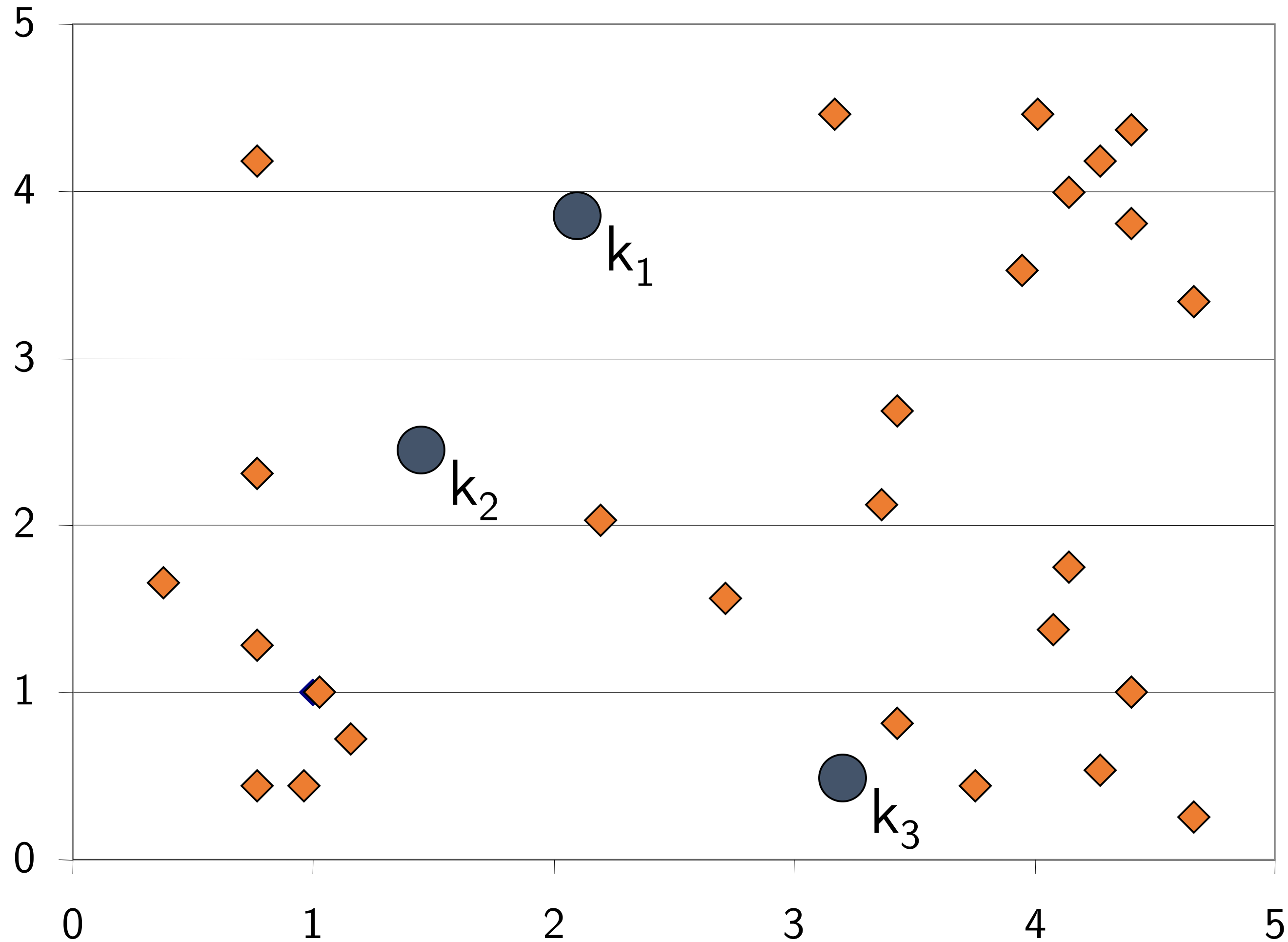
- **Input**:
  - Desired number of clusters, k
- **Initialize**:
  - the k cluster centers (randomly if necessary)
- **Iterate**:
  1. **Assign** points to the **nearest** cluster centers
  2. **Re-estimate** the k cluster centers (aka the **centroid** or mean), by assuming the memberships found above are correct.

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

- **Termination**
  - If **none of the objects changed membership** in the last iteration, exit. Otherwise go to 1.

K-means Clustering: Step 1

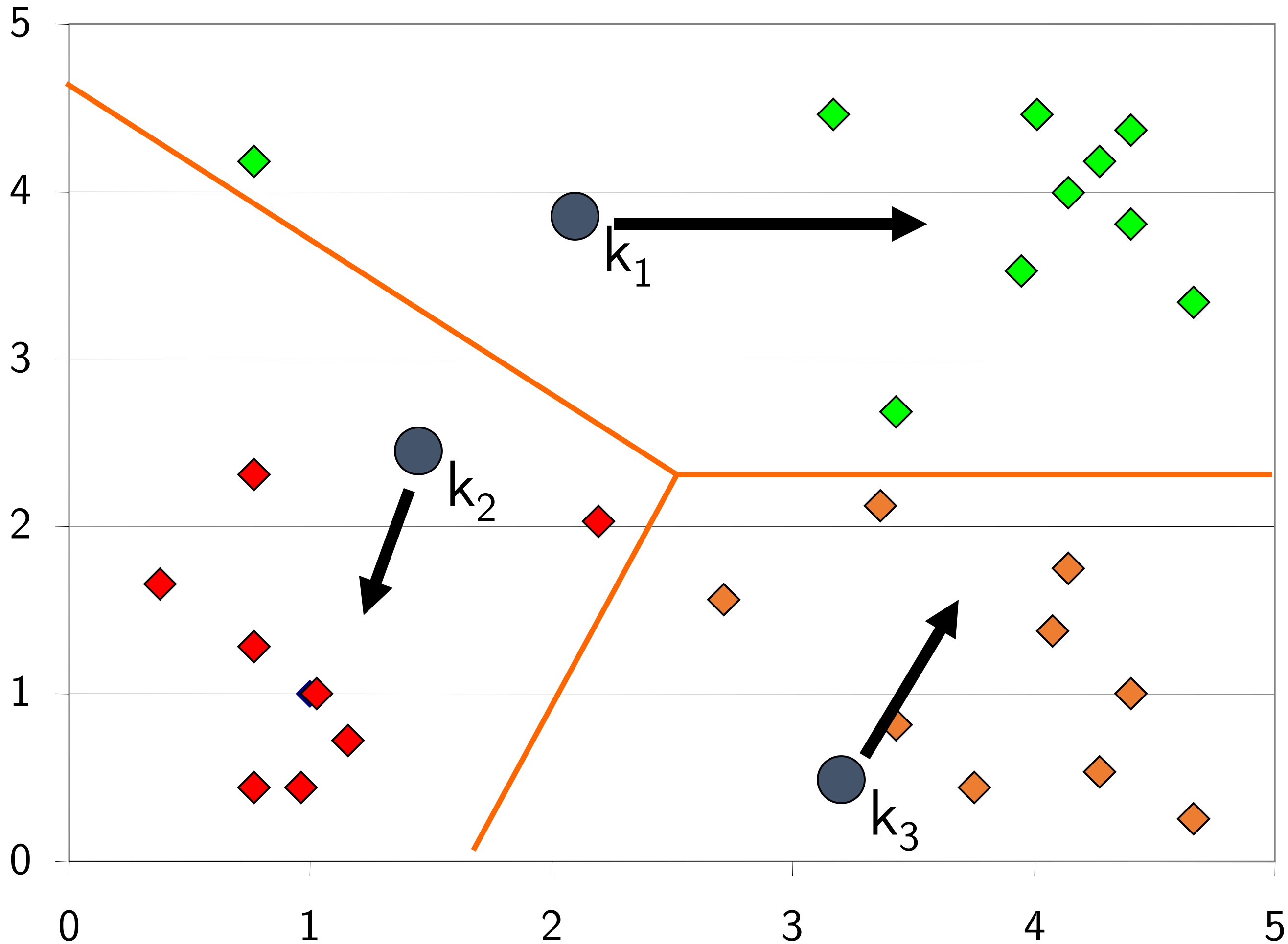
Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 2

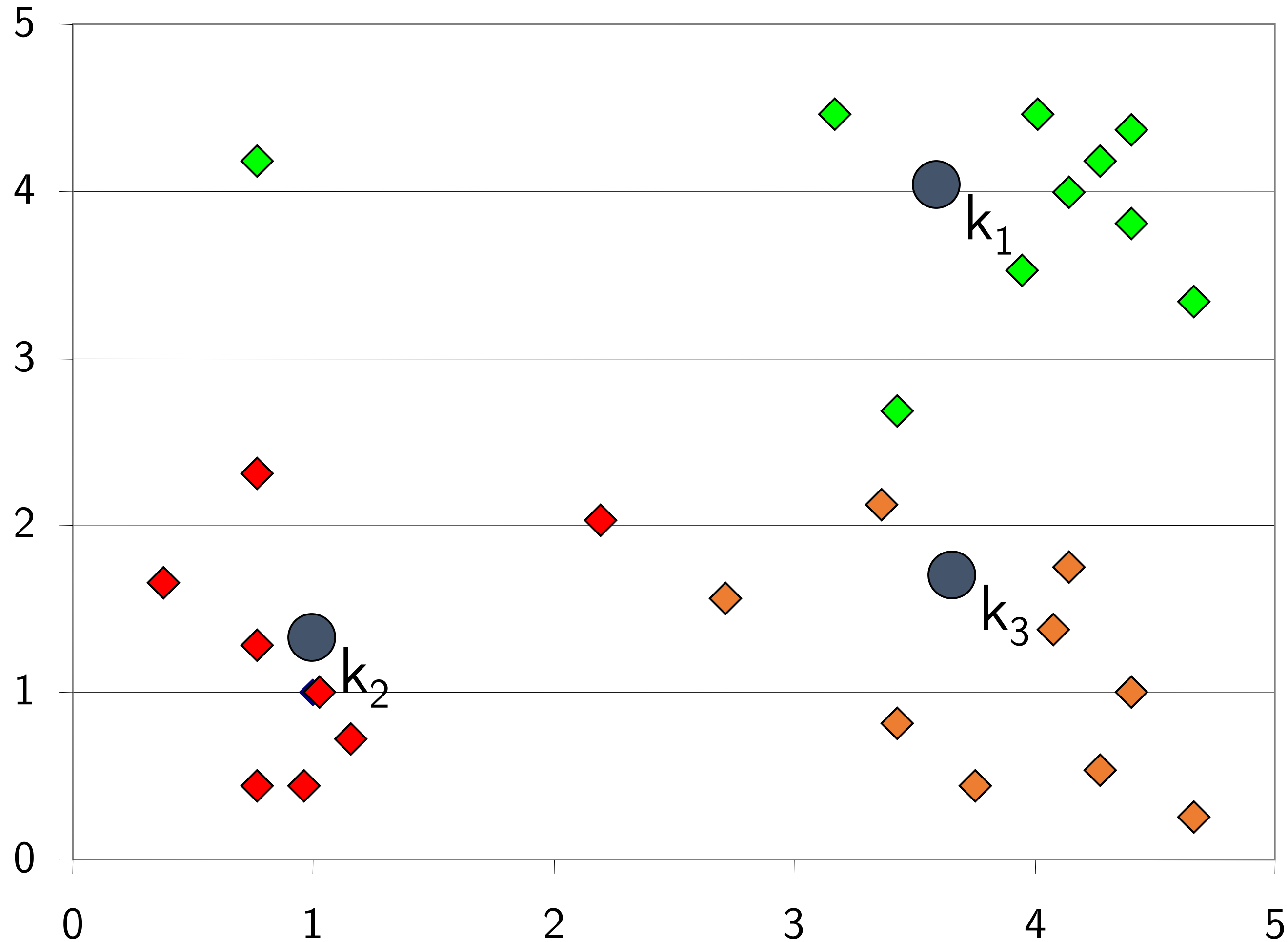
Algorithm: k-means, Distance Metric: Euclidean Distance

Voronoi diagram



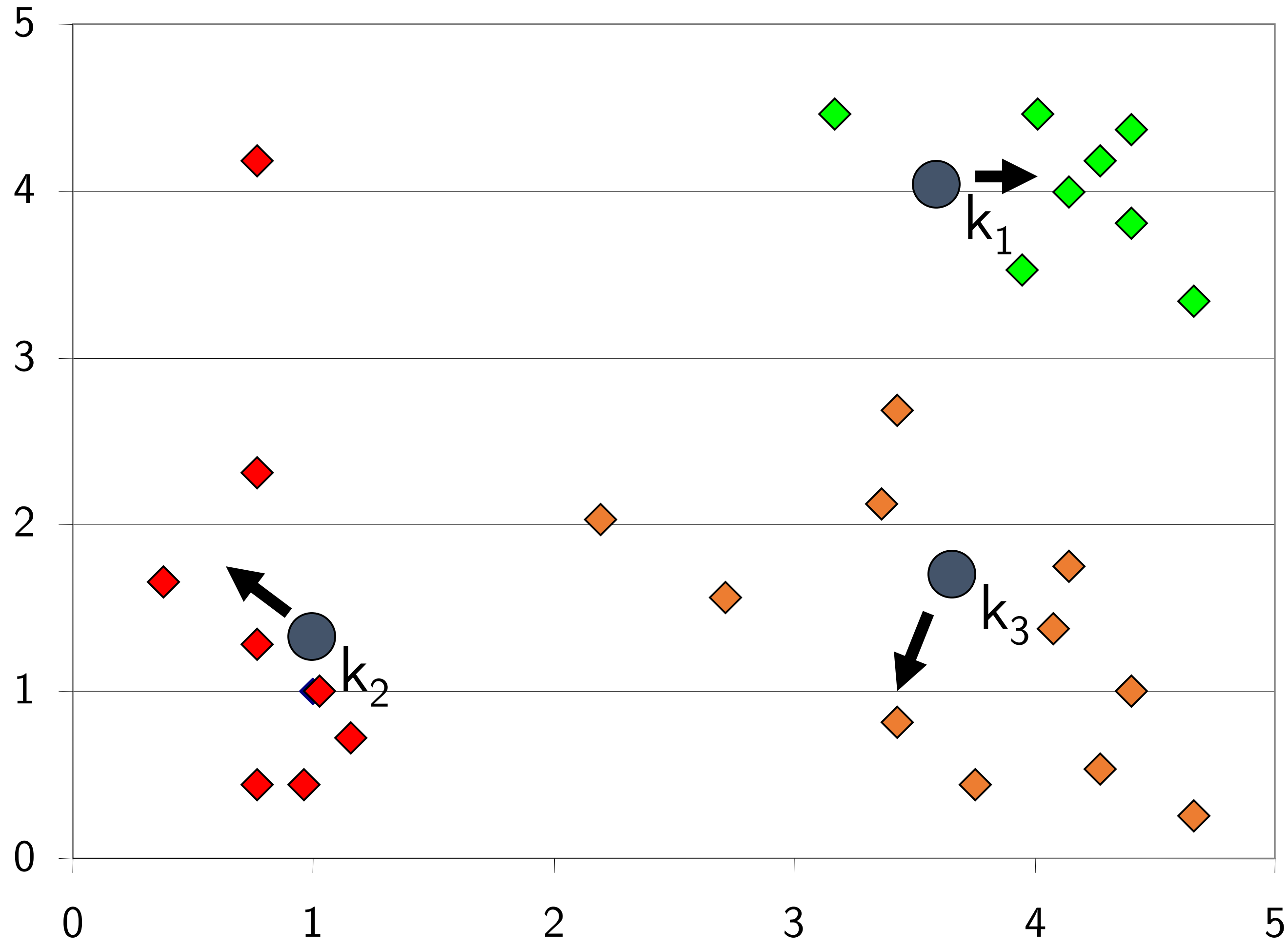
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



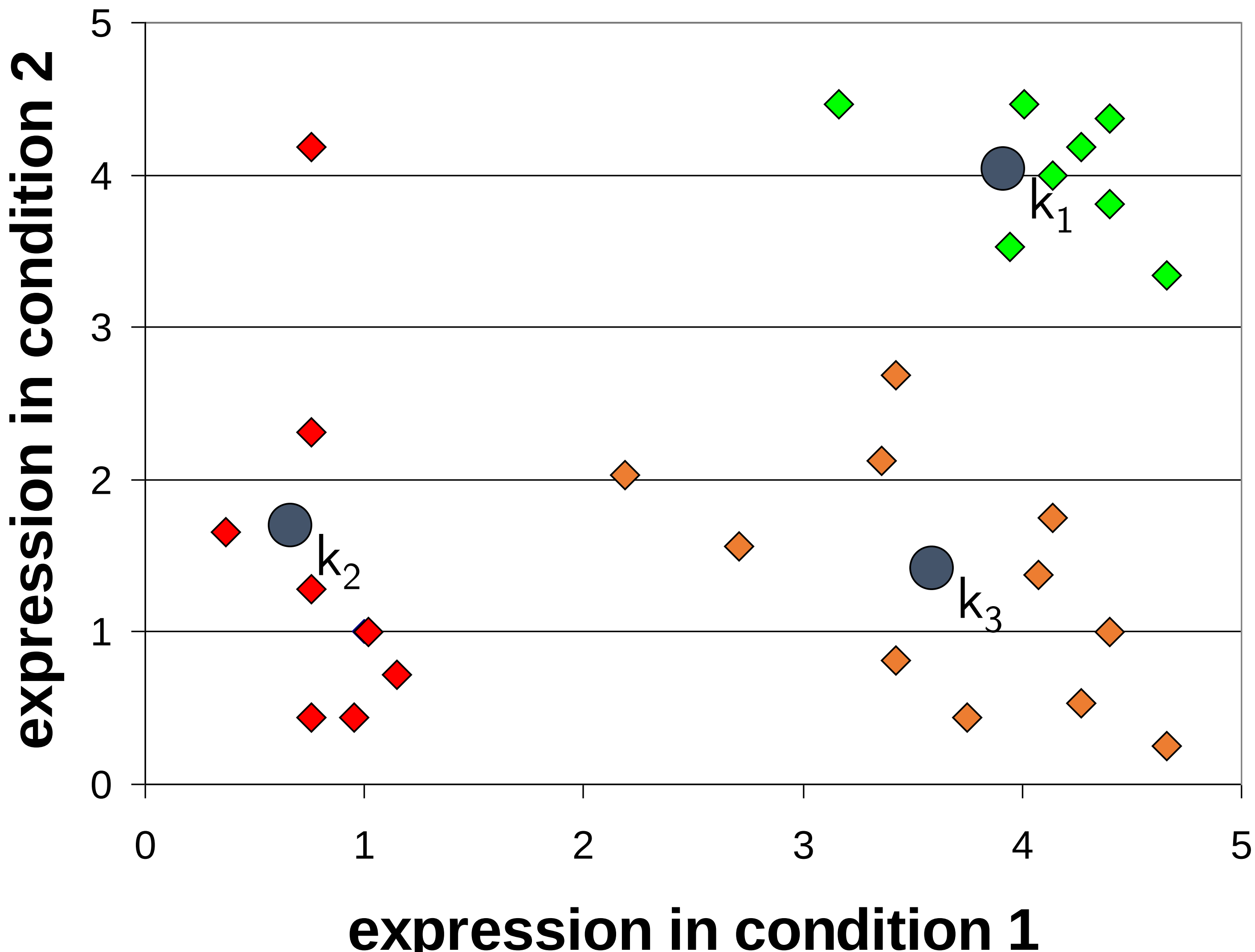
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Recap ...

- Randomly initialize k centers
  - $\mu^{(0)} = \mu_1^{(0)}, \mu_2^{(0)}, \mu_3^{(0)}, \ldots, \mu_k^{(0)}$
- Interate t=0, 1, 2, . . .
  - **Classify**: Assign each point j∈{1,2,…,m} to nearest center

$$C^{(t)}(j) \leftarrow \arg \min_{i=1,\ldots,k} \|\mu_i^{(t)} - x_j\|^2$$

  - **Recenter**: μ_i becomes centroid if its points:

$$\mu_i^{(t+1)} \leftarrow \arg\min_{\mu} \sum_{j:C^{(t)}(j)=i} \|\mu - x_j\|^2 \qquad i \in \{1, \ldots, k\}$$

  - Equivalent to $\mu_i \longleftarrow$ average of its points!

What is K-means optimizing? (Objective Function)

- **Potential function** (objective function) F(μ,C) of centers μ and point allocation C:

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

$$= \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- Optimal K-means:
  - $min_{\mu} min_{C} F(\mu, C)$

K-means algorithm

- Optimize **potential function**:

$$\min_{\mu} \min_{C} F(\mu, C) = \min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{j:C(j)=i} ||\mu_i - x_j||^2$$

- **K-means algorithm:** (coordinate descent on F)

  (1) Fix μ and optimize C; **Expected** cluster assignment

  (2) Fix C, optimize μ; **Maximum** likelihood for center

- A sample of **EM algorithm**

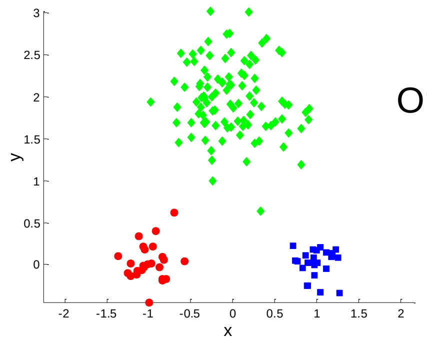
Computational Complexity

- At each **iteration**,
  - **Computing distance** between each of the n objects and the K cluster centers is O(Kn).
  - **Computing cluster centers**: Each object gets added once to some cluster: O(n).
- Assume these two steps are each done once for **I iterations**: O(IKn).

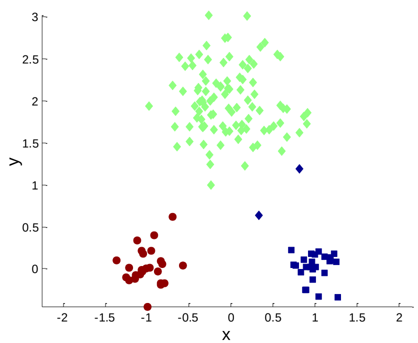
Results are quite sensitive to seed selection

- Results **can vary** based on random seed selection.
- Some seeds can result in **poor convergence** rate, or convergence to sub-optimal clustering.
  - Try out **multiple starting** points (very important!!!)
- **k-means++** algorithm of Arthur and Vassilvitskii
  - key idea: choose **centers that are far apart**
  - Choose one new data point at random as a new center, using a **weighted probability distribution** where a point x is chosen with probability proportional to **squared distance from nearest** center picked so far.

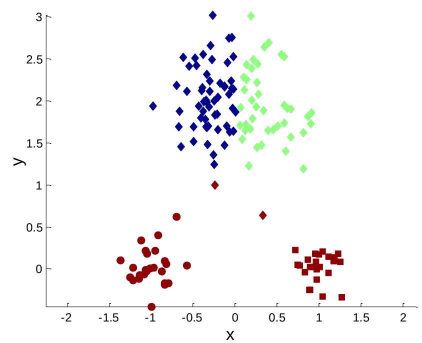
Two different K-means Clusterings



Original Points

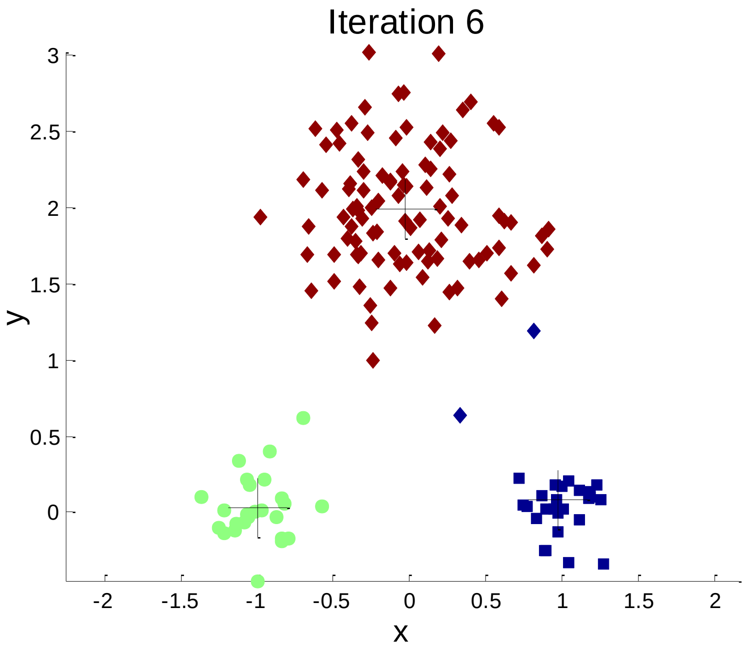


Optimal Clustering

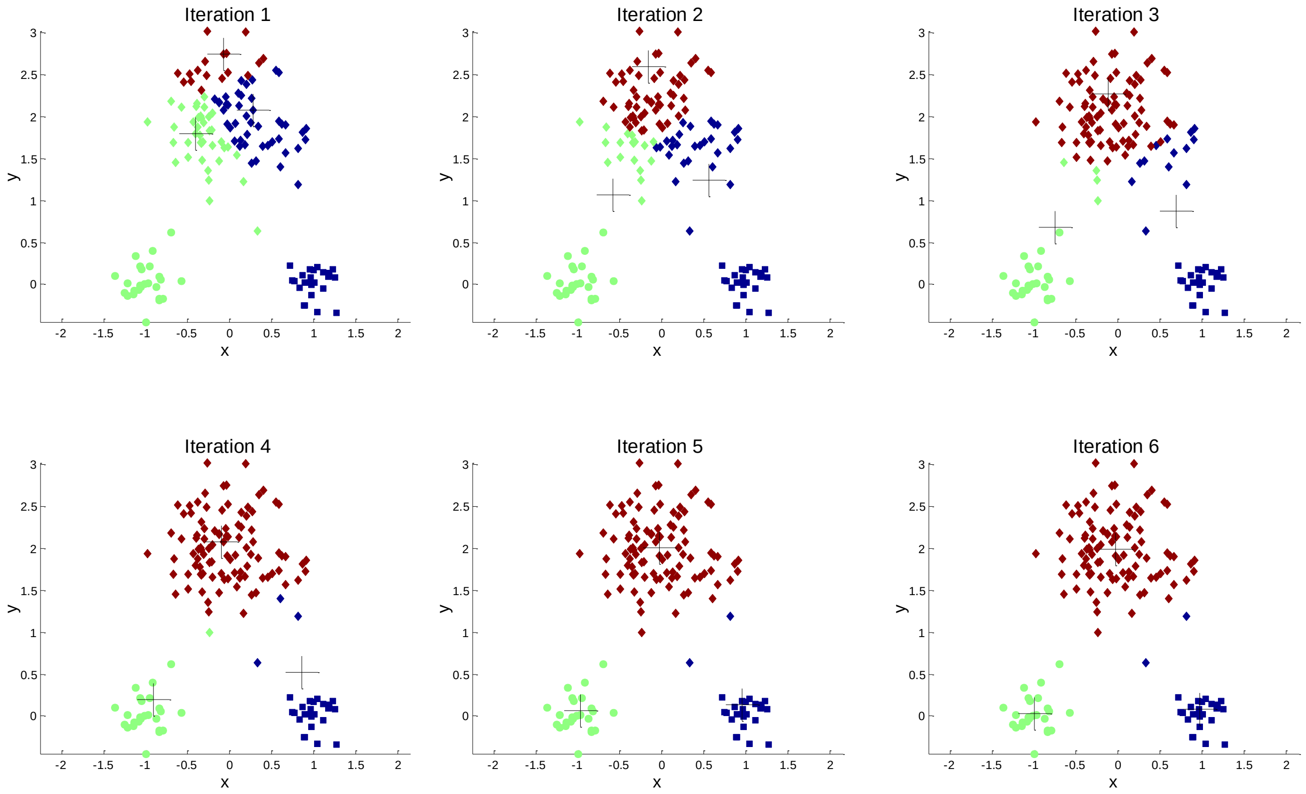


Sub-optimal Clustering

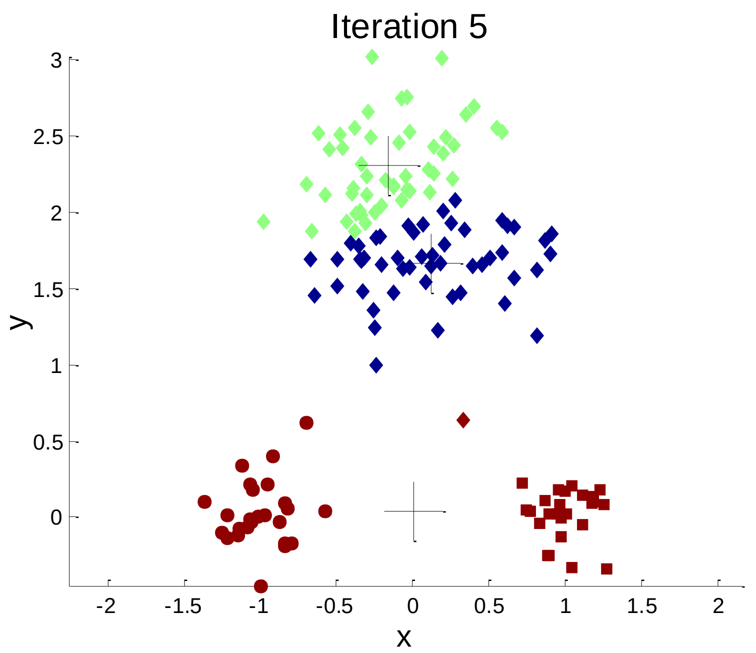
Importance of Choosing Initial Centroids (Case i)



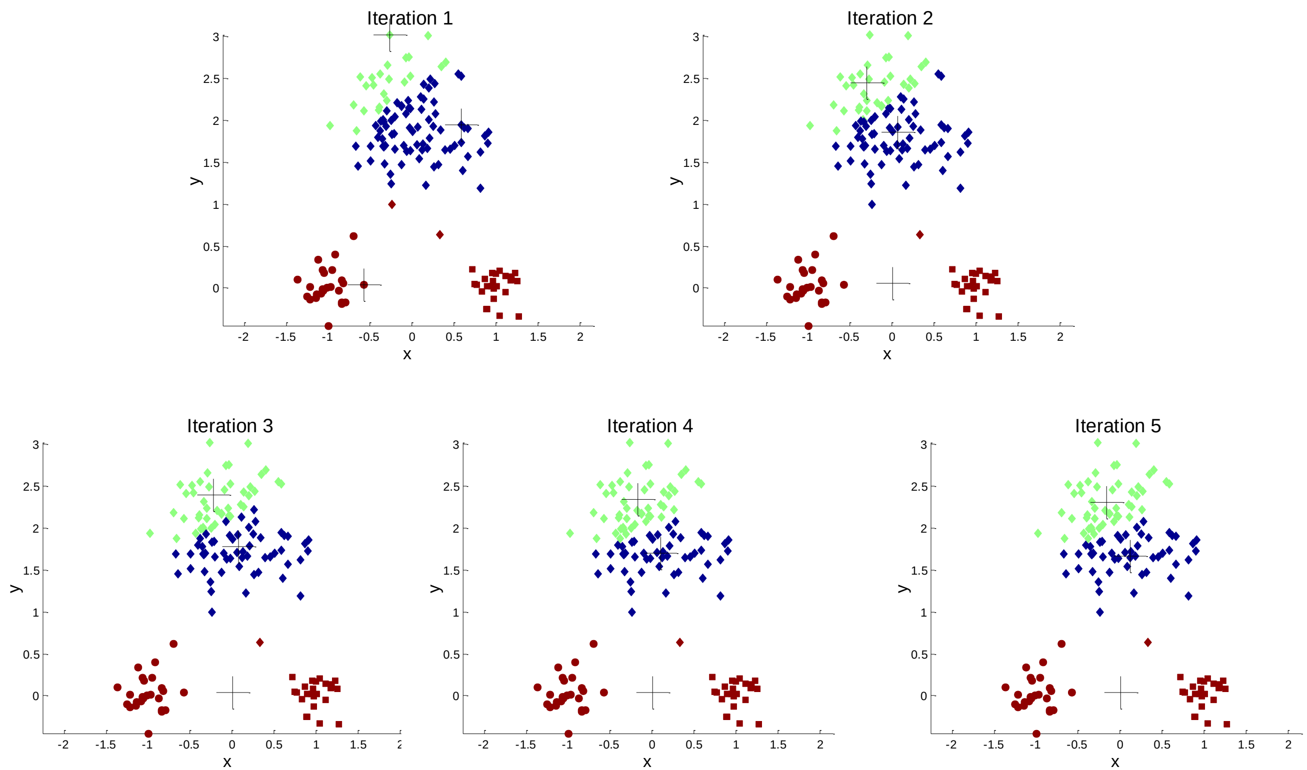
Importance of Choosing Initial Centroids (Case i)



Importance of Choosing Initial Centroids (Case ii)

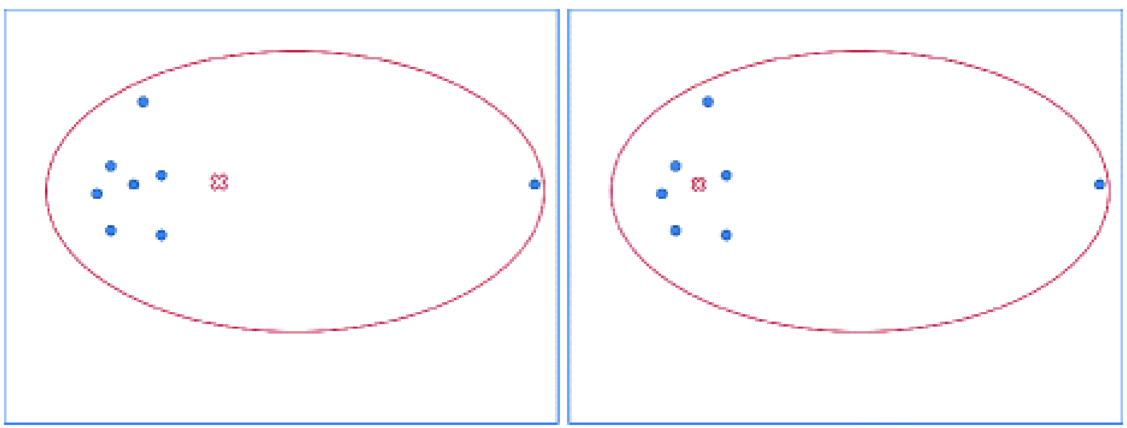


Importance of Choosing Initial Centroids (Case ii)



Other Issues

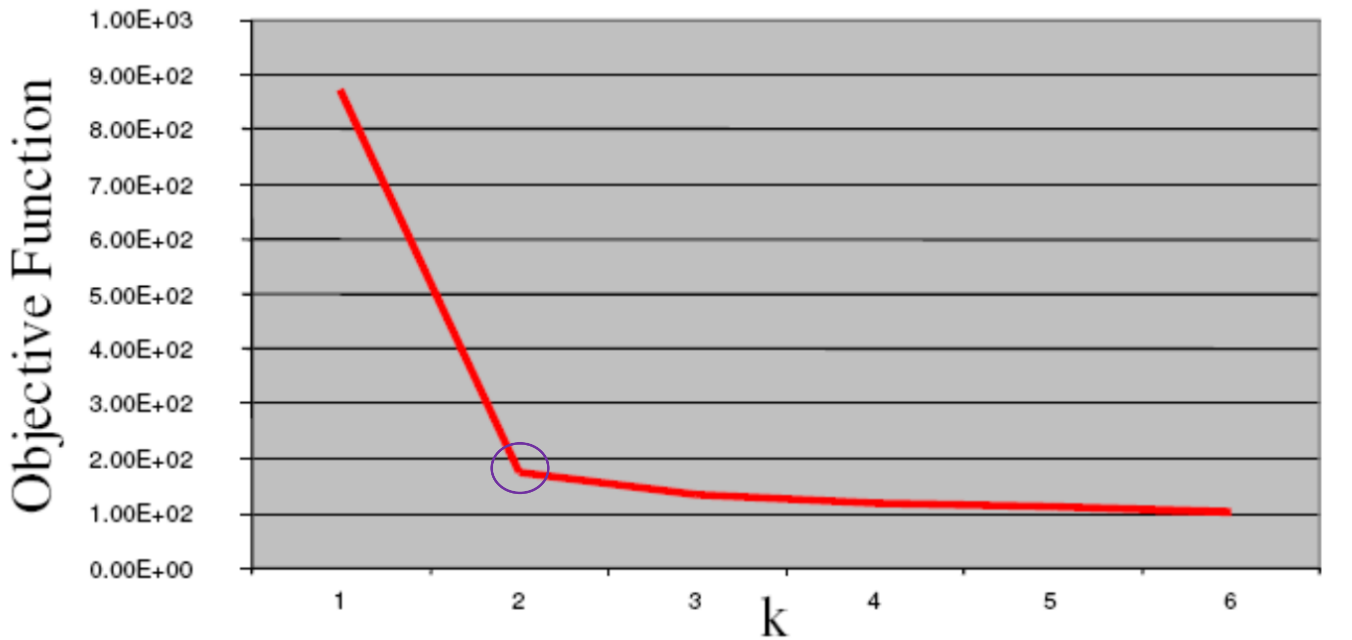
- Shape of clusters
    - Assumes **isotropic**, **equal variance**, **convex clusters**
- Sensitive to **Outliers**
    - use **K-medoids (***representative* objects)



(a) Mean (b) Medoid

Number of clusters K

- Objective function: $$\sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$
- Look for "Knee" in objective function



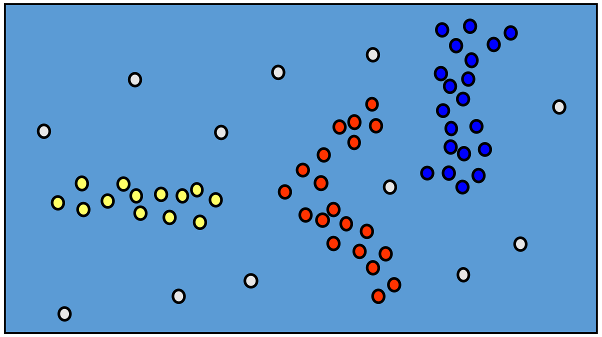
Density-based Approaches

- Why Density-Based Clustering methods?
  - Discover clusters of arbitrary shape.
  - Clusters – Dense regions of objects separated by regions of low density
- Proposed by Ester, Kriegel, Sander, and Xu (KDD96)
- Relies on a density-based notion of cluster: A cluster is defined as a maximal set of density-connected points.
- Discovers clusters of arbitrary shape in spatial databases with noise

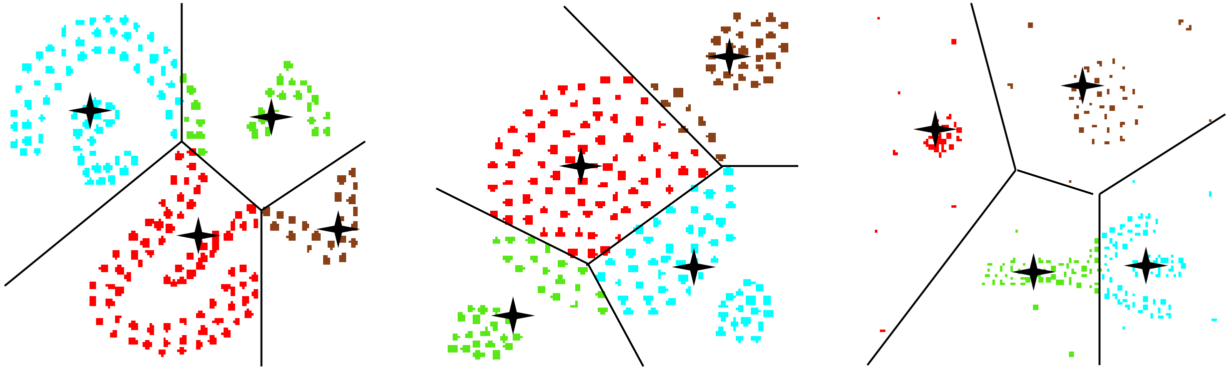
Density-Based Clustering

- **Basic Idea:**

  Clusters are dense regions in the data space, separated by regions of lower object density



- Why Density-Based Clustering?



Results of a *k*-medoid algorithm for *k*=4

Density Based Clustering: Basic Concept

- Intuition for the formalization of the basic idea
  - For any point in a cluster, the **local point density** around that point has to **exceed some threshold**
  - The set of points from one cluster is **spatially connected**
- Local point density at a point p defined by **two parameters**
  - ε – **radius** for the neighborhood of point p:

$$N_{\varepsilon}(p) := \{q \text{ in data set } D \mid dist(p, q) \leq \varepsilon\}$$

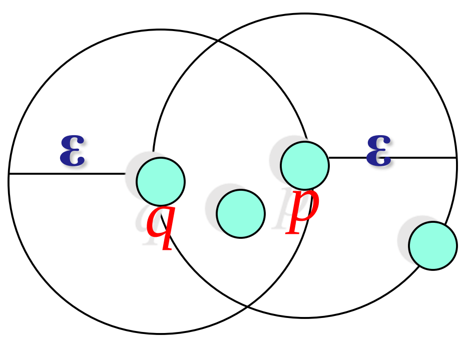
  - *MinPts* – **minimum number** of points in the given neighborhood $N(p)$

ε-Neighborhood

- **ε-Neighborhood** – Objects within a radius of ε from an object.

$$N_{\varepsilon}(p):\{q \mid d(p,q) \leq \varepsilon\}$$

- "**High density**" - ε-Neighborhood of an object contains at least *MinPts* of objects.

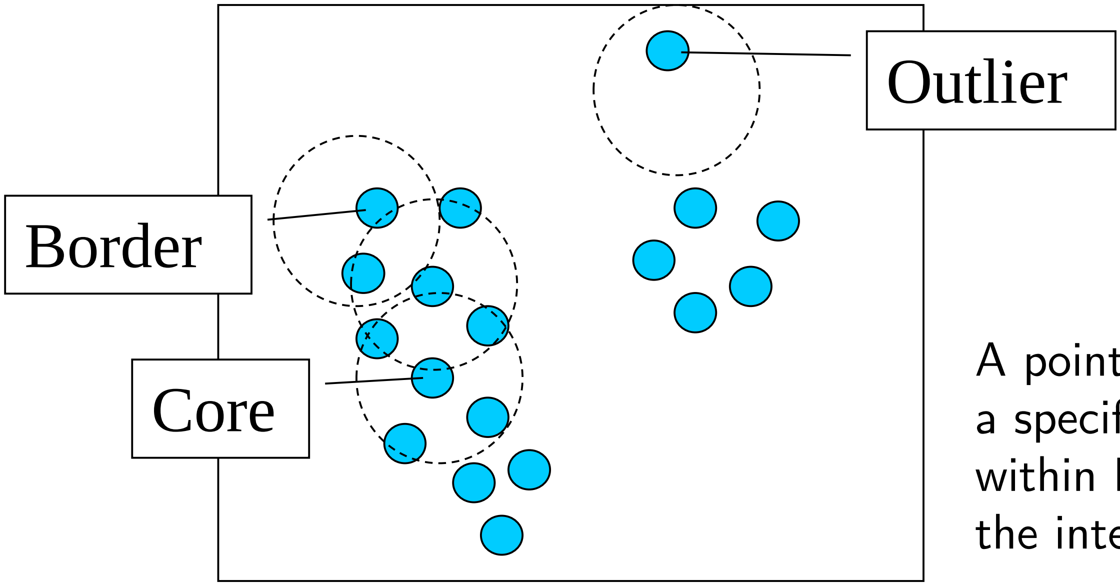


ε-Neighborhood of *p*
ε-Neighborhood of *q*

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

Core, Border & Outlier



ε = 1unit, MinPts = 5

Given ε and *MinPts*, categorize the objects into three exclusive groups.

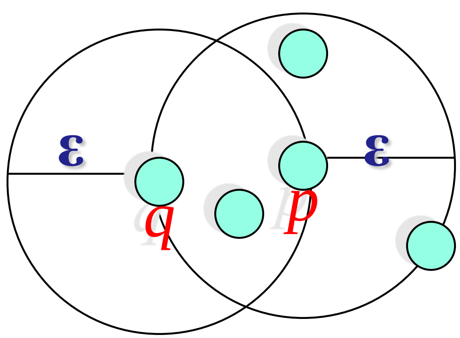
A point is a core point if it has more than a specified number of points (MinPts) within Eps These are points that are at the interior of a cluster.

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Density-Reachability

- **Directly density-reachable**
  - **An object q is directly density-reachable from object p if p is a core object and q is in p's ε-neighborhood.**

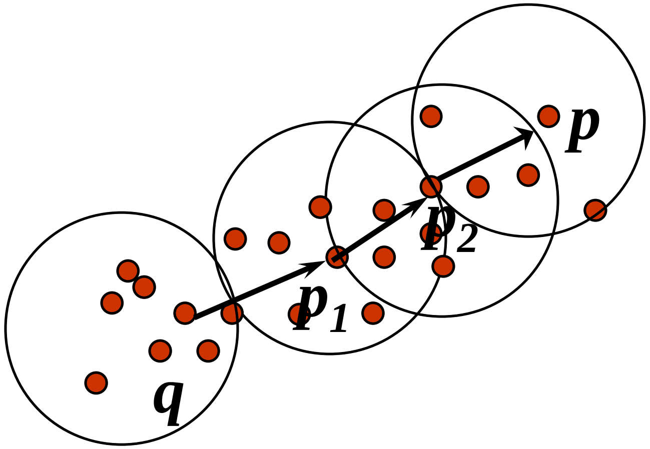


MinPts = 4

- q is directly density-reachable from p
- p is not directly density- reachable from q?
- Density-reachability is asymmetric.

Density-reachability

- Density-Reachable (directly and indirectly):
  - A point p is directly density-reachable from p_2;
  - p_2 is directly density-reachable from p_1;
  - p_1 is directly density-reachable from q;
  - $p \leftarrow p_2 \leftarrow p_1 \leftarrow q$ form a **chain**.

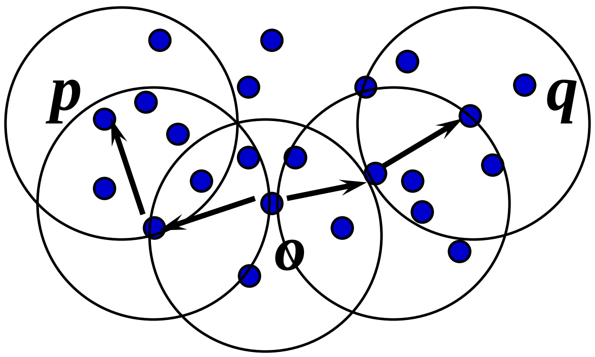


MinPts = 7

- p is (indirectly) density-reachable from q
- q is not density- reachable from p?

Density-Connectivity

- Density-reachable is **not symmetric**
  - not good enough to describe clusters
- Density-Connected
  - A pair of points p and q are density-connected if they are commonly density-reachable from a point o.
  - Density-connectivity **is symmetric**

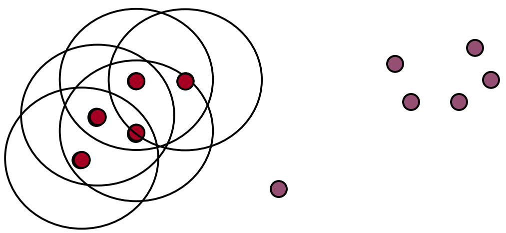


DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points **density-reachable from *p*** wrt *Eps* and *MinPts*.
- If *p* is a core point, a cluster is formed.
- If *p* is a border point, **no points are density-reachable from *p*** and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

DBSCAN Algorithm: Example

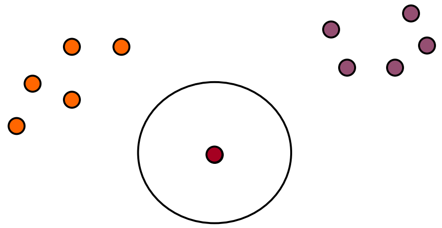
- Parameter
  - $\varepsilon = 2$ cm
  - $MinPts = 3$



for each $o \in D$ **do**
    if o is not yet classified **then**
        if o is a core-object **then**
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE

DBSCAN Algorithm: Example

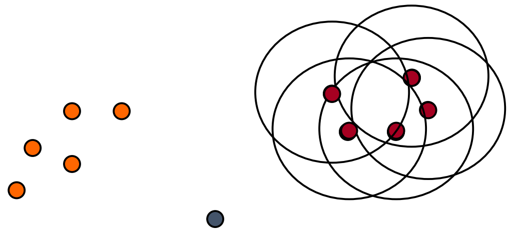
- Parameter
  - $\varepsilon = 2$ cm
  - $MinPts = 3$



for each $o \in D$ **do**
  if o is not yet classified **then**
    if o is a core-object **then**
      collect all objects density-reachable from o
      and assign them to a new cluster.
    else
      assign o to NOISE

DBSCAN Algorithm: Example

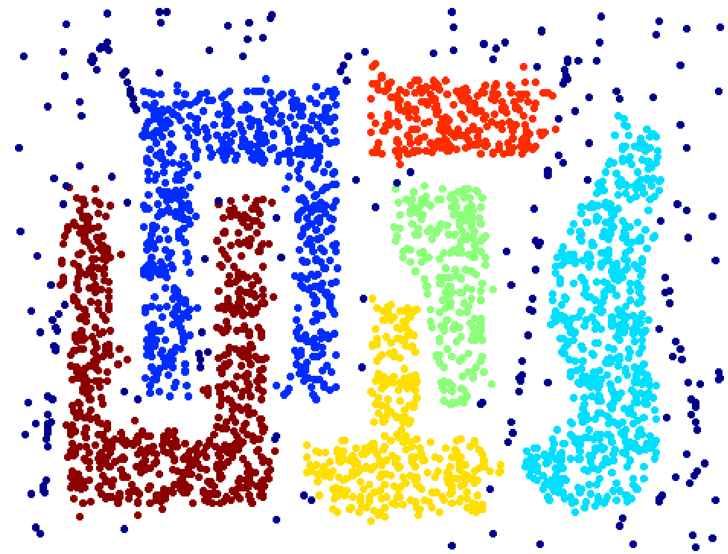
- Parameter
  - $\varepsilon = 2$ cm
  - $MinPts = 3$



for each $o \in D$ **do**
    if o is not yet classified **then**
        if o is a core-object **then**
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE

When DBSCAN Works Well

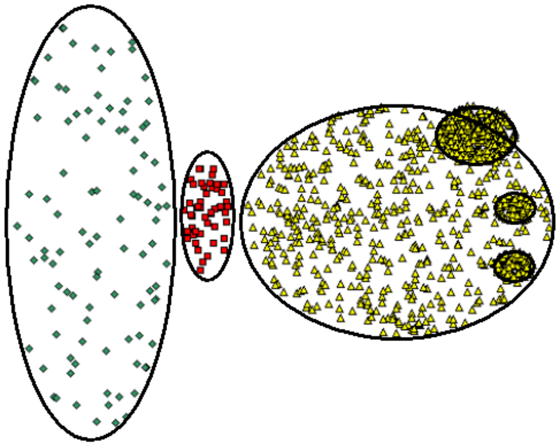
Original Points



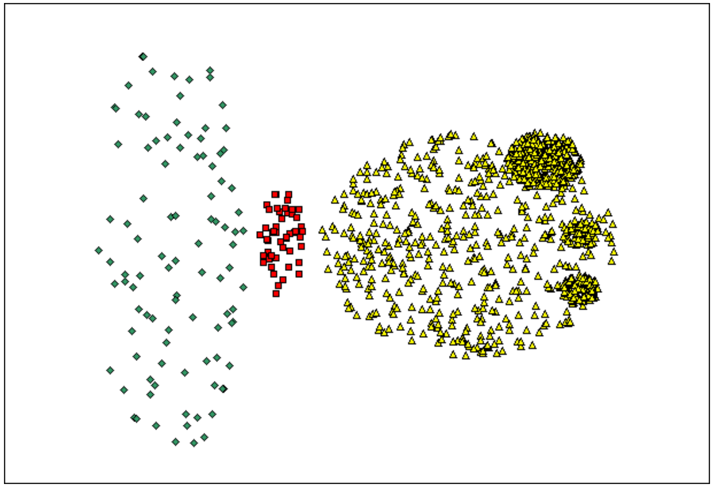
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

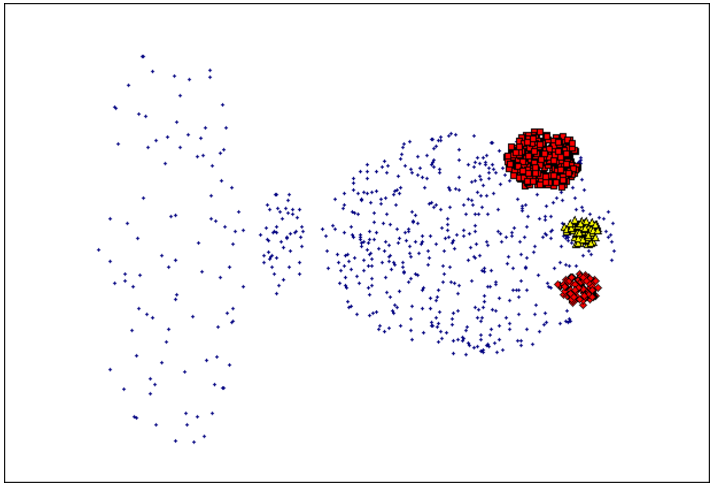
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- Cannot handle Varying densities
- sensitive to parameters

•Input parameters may be difficult to determine

What Is A Good Clustering?

- **Internal criterion:** A good clustering will produce high quality clusters in which:
  - the **intra-class** (that is, intra-cluster) similarity is high (**low within distance**)
  - the **inter-class** similarity is low high (**high between distance**)
  - The measured quality of a clustering depends on both the document **representation** and the **similarity measure** used

External criteria for clustering quality

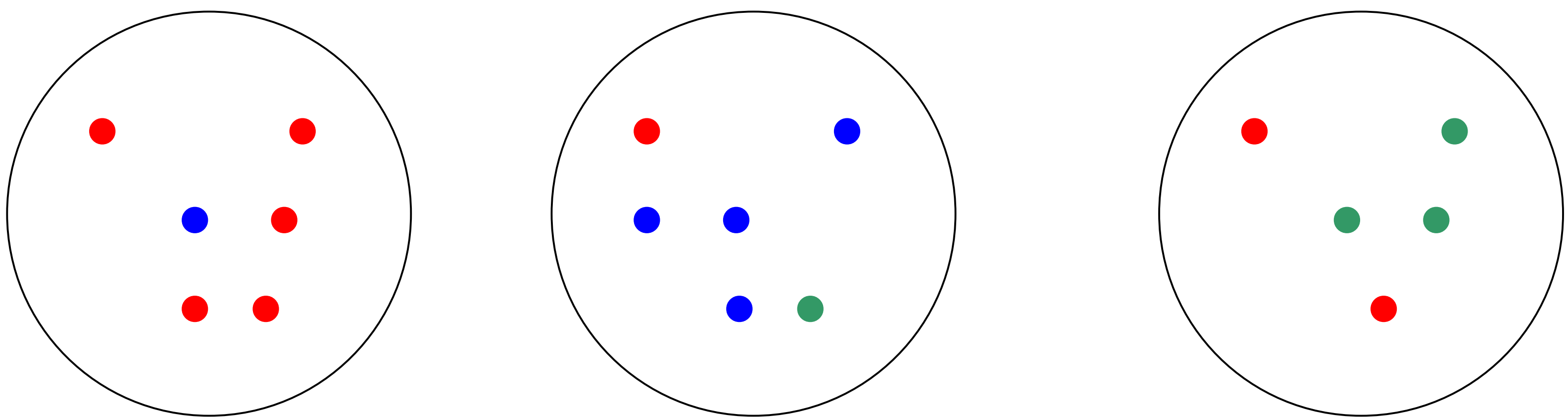
- Quality measured by **its ability** to discover some or all of the **hidden patterns** or latent classes in **gold standard** data
- Assesses a clustering with respect to ground truth requires ***labeled data***
- Assume documents with C gold standard classes, while our clustering algorithms produce K clusters, $\omega_1, \omega_2, \ldots, \omega_K$ with n_i members.

External Evaluation of Cluster Quality

- Simple measure: purity, the ratio between the dominant class in the cluster π_i and the size of cluster ω_i

$$Purity(\omega_i) = \frac{1}{n_i} \max_j (n_{ij}) \quad j \in C$$

= 5/6 = 4/6 = 3/5



Cluster I Cluster II Cluster III