CS60050
Machine Learning
Clustering

Somak Aditya, Sudeshna Sarkar
Department of CSE, IIT Kharagpur

Supervised learning vs. unsupervised learning

- Supervised learning: discover *discriminative* patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - discover *data generative* (all) patterns.

Clustering

Unsupervised learning: The data have no target attribute.

– Requires data, but no labels

– Detect patterns e.g. in

- Group emails or search results
- Customer shopping patterns
- Regions of images

– Useful when don’t know what you’re looking for

– But: can get gibberish

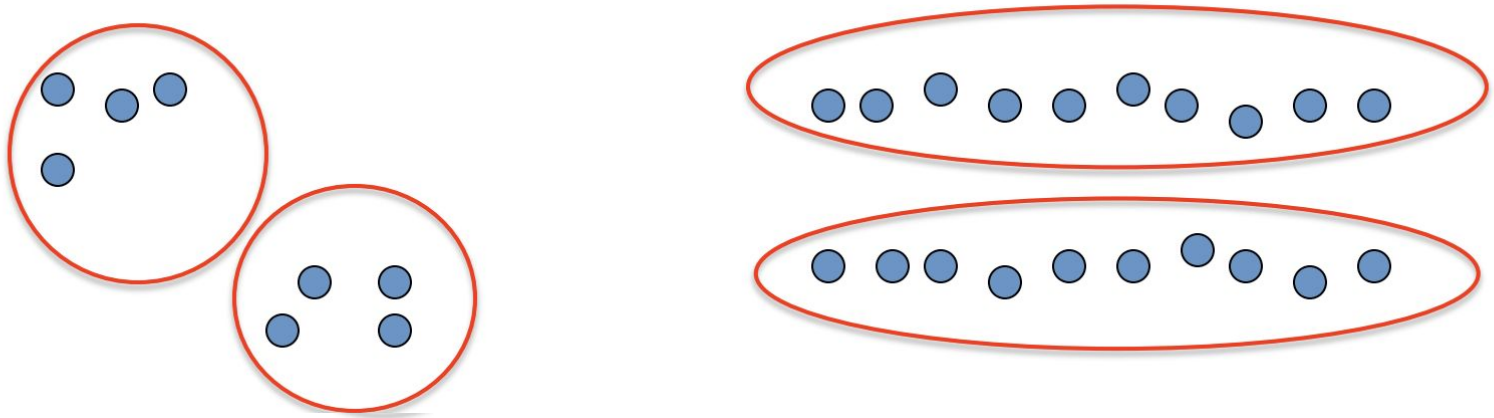
Applications

- Segmenting of customers with similar market characteristics — pricing , loyalty, spending behaviors etc.
- Grouping of products based on their properties
- Identify similar energy use customer profiles

  <x> = time series of energy usage
- Clustering weblog data to discover groups of similar access patterns.
- Recognize communities in social networks.
- Top 20 topics in Twitter

Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



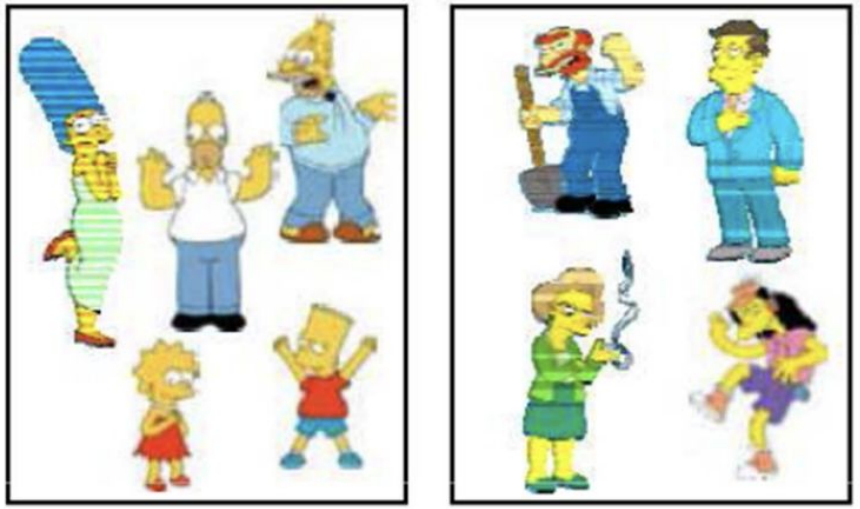
- What could "similar" mean?
  - One option: small Euclidean distance (squared)

$$\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

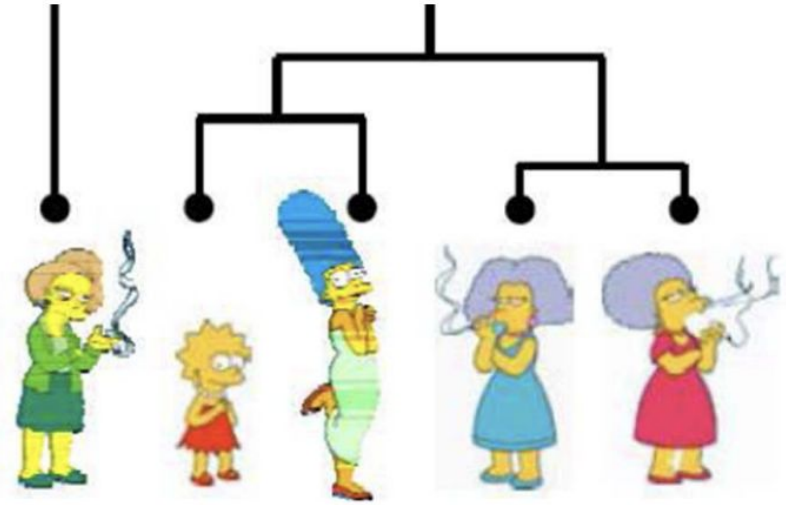
  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

Clustering Algorithms

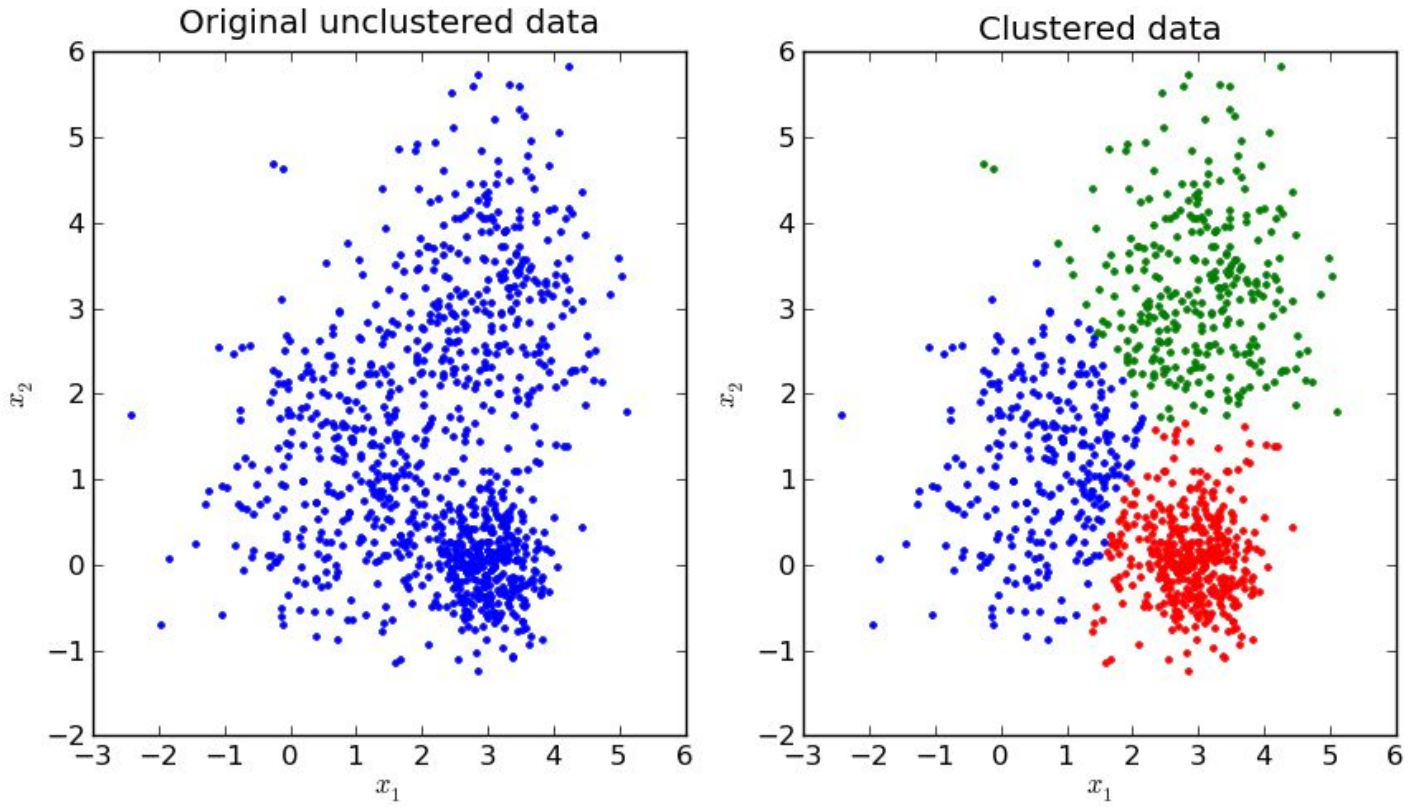
- Partition algorithms (Flat)
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering

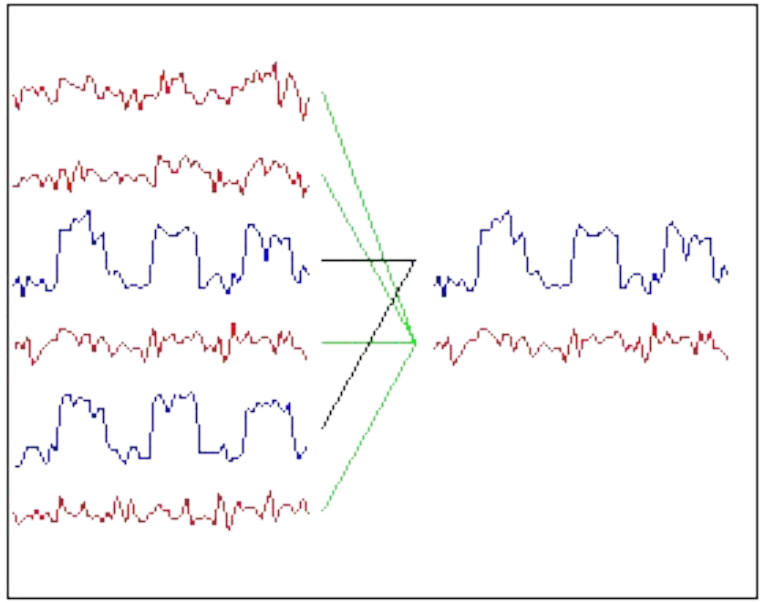


- Hierarchical algorithms
  - Bottom up – agglomerative
  - Top down – divisive



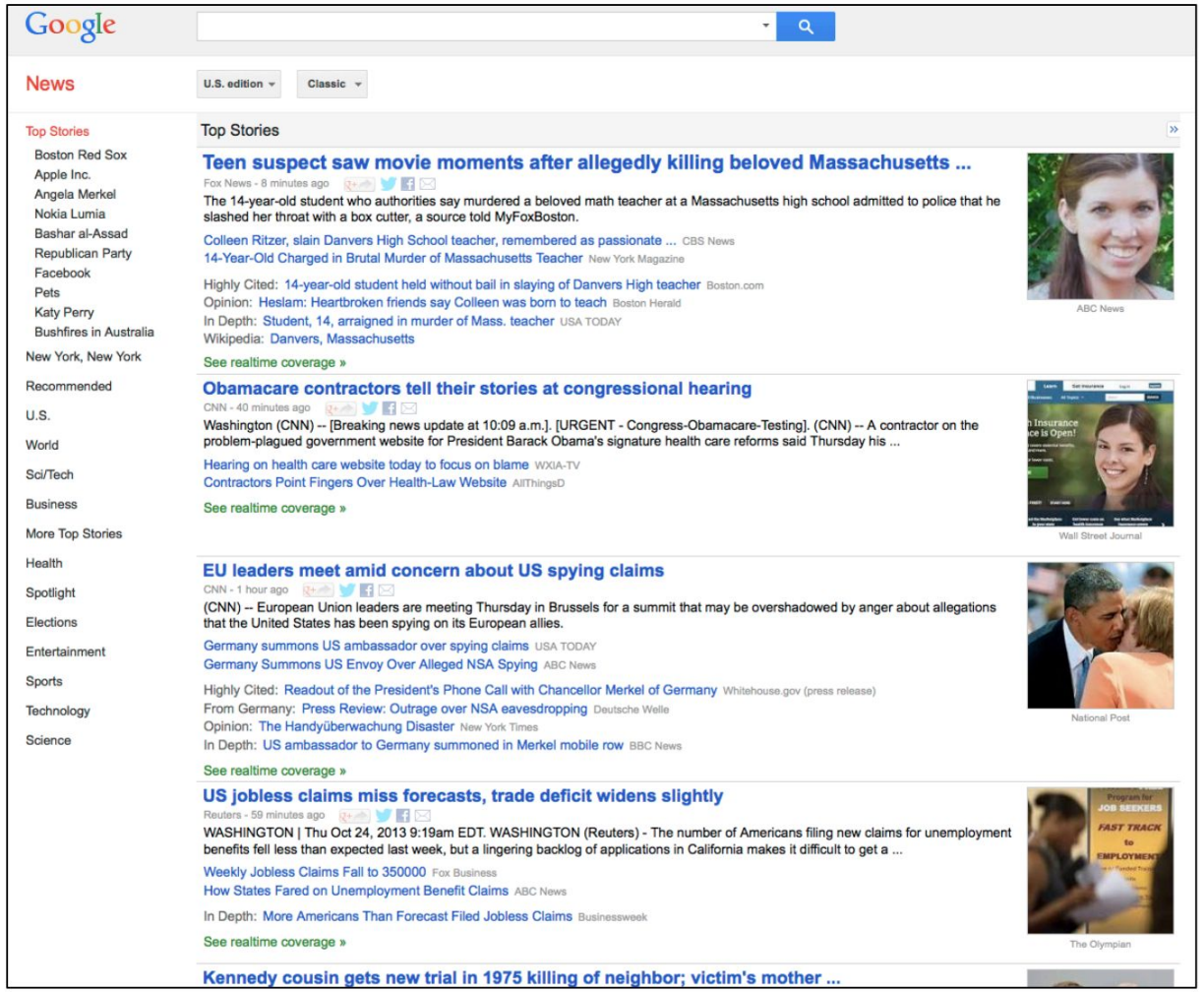
Clustering





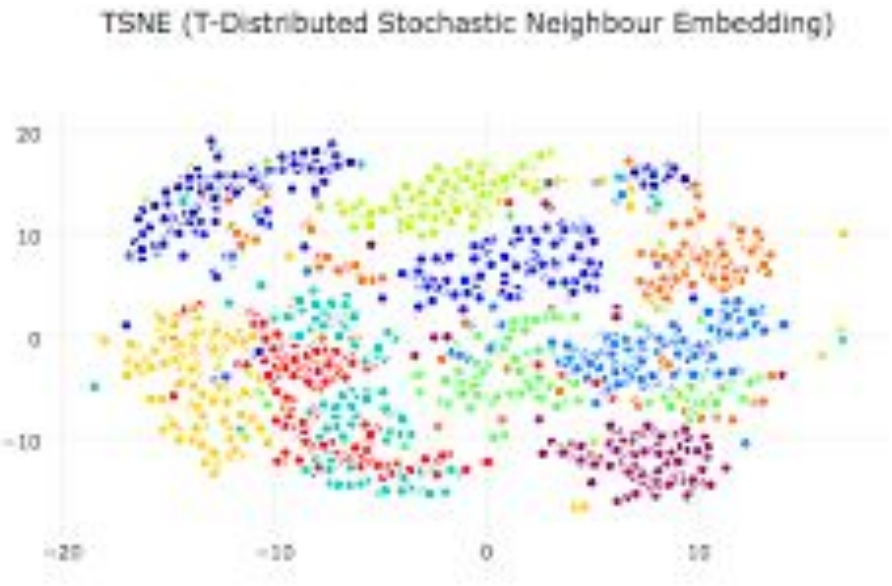
Clustering Examples

Cluster news articles



Customer Segmentation

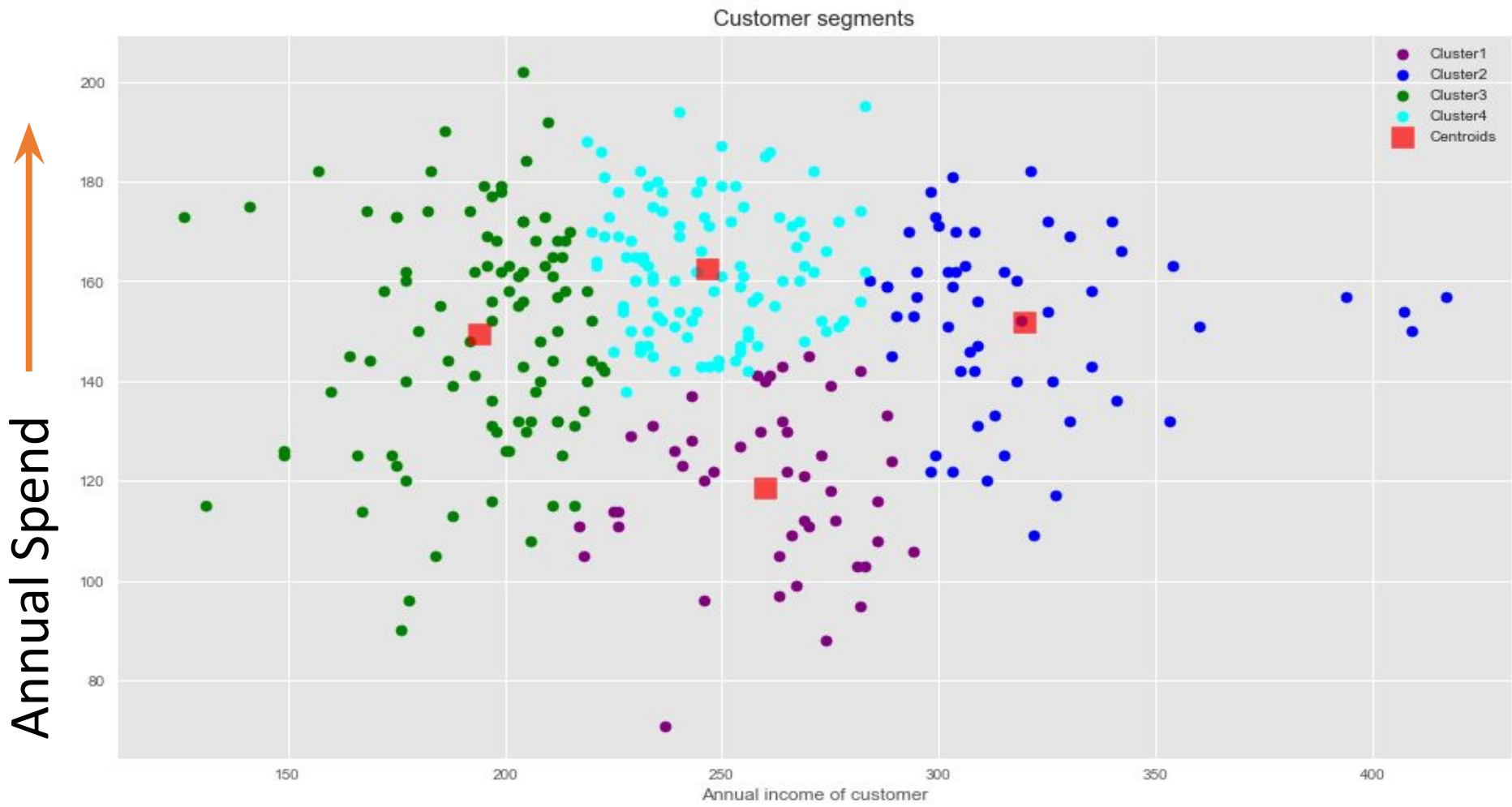
- Group customers based on their demography and activity
  - Purchase History
  - Demographic
  - Content engagement
  - Behavior
  - Customer Lifecycle Stage



Cater to customer groups for promotion, rec
development strategies

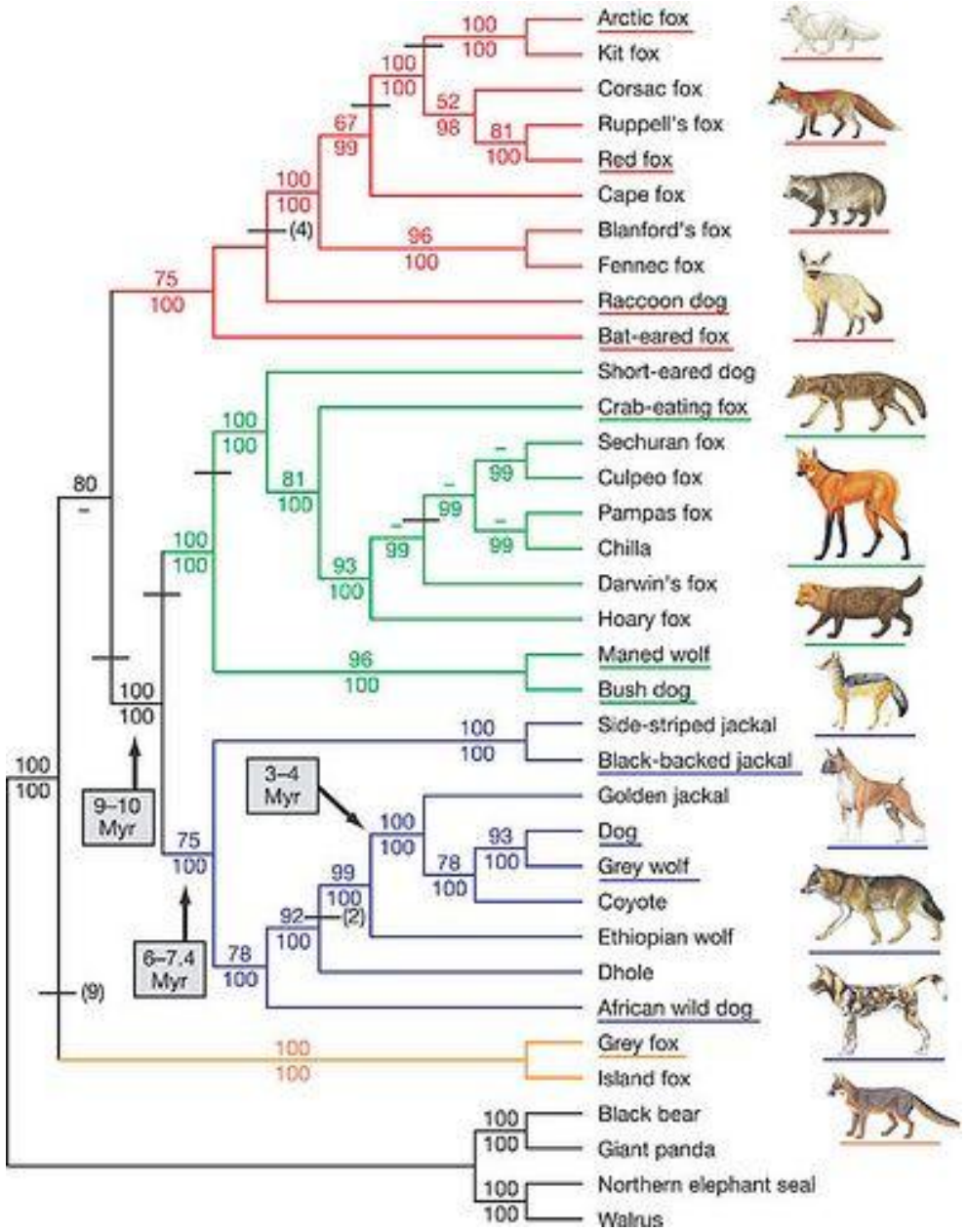
Customer Segments

INCOME	SPEND
233	150
250	187
204	172
236	178
354	163
192	148
294	153
263	173
199	162
168	174
239	160
275	139
266	171
211	144



Clustering

Clustering Species
(“phylogeny”)

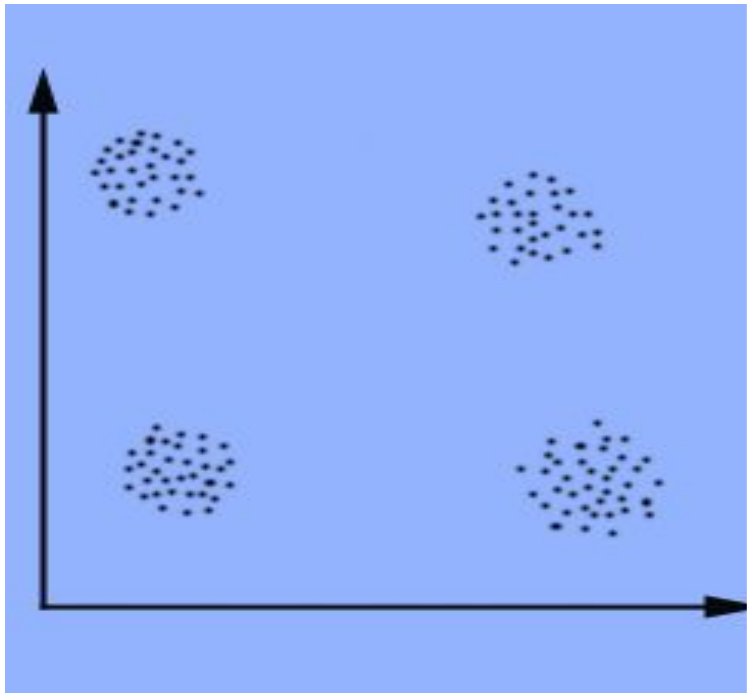


[Lindblad-Toh et al., Nature 2005]

Fundamental Aspects of clustering

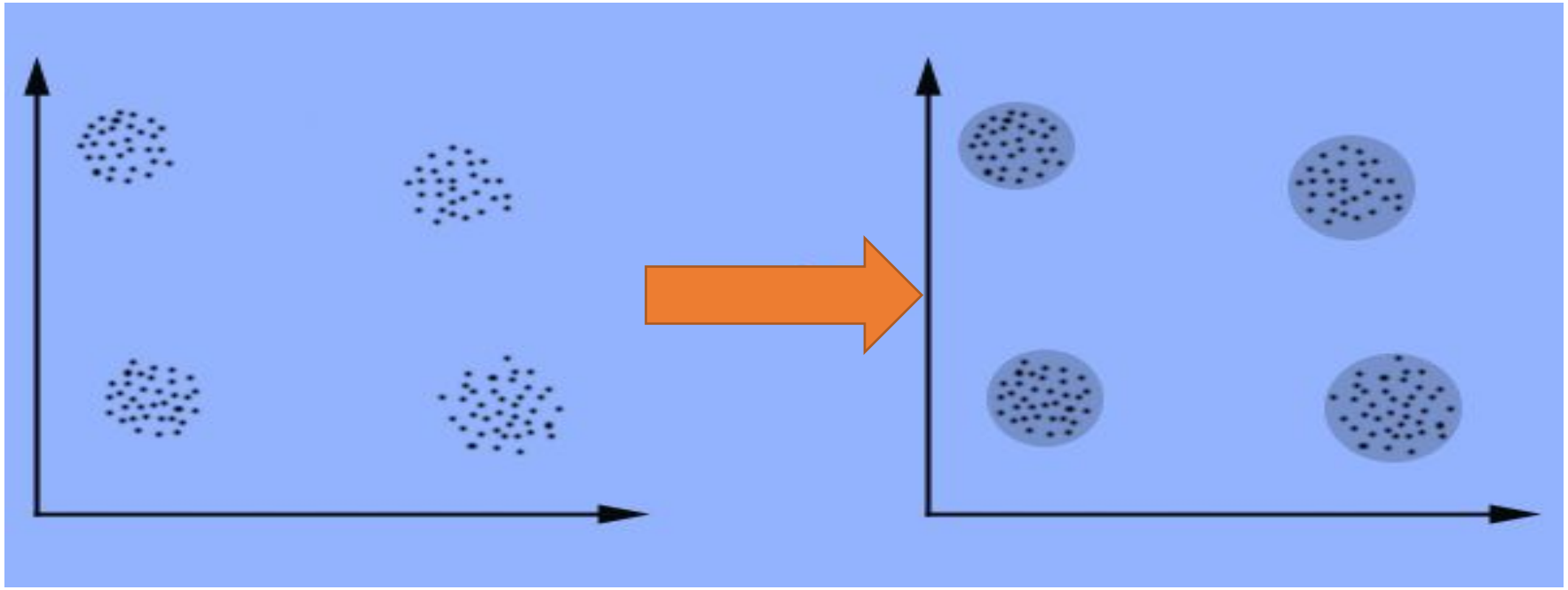
- A clustering algorithm
  - Partitional clustering
  - Hierarchical clustering
  - …
- A distance (similarity, or dissimilarity) function
  - Euclidean, cosine, Mahalanobis
- Clustering quality
  - Inter-clusters distance $\Rightarrow$ maximized
  - Intra-clusters distance $\Rightarrow$ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

How many clusters?



An illustration

This data set has four natural clusters.

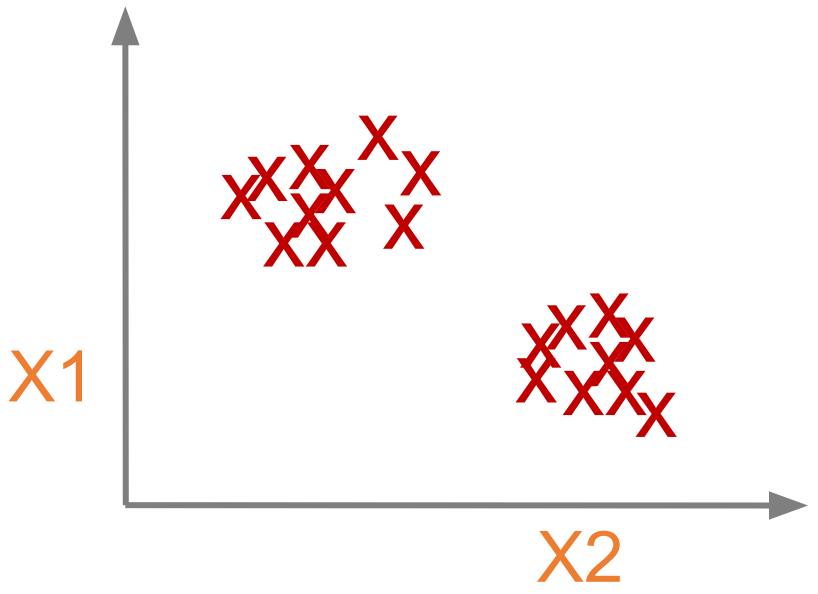


Clustering

- Given examples: $\{X_1, X_2, \ldots, X_m\},\ X_i = (x_{i1}, x_{i2}, \ldots x_{in})$

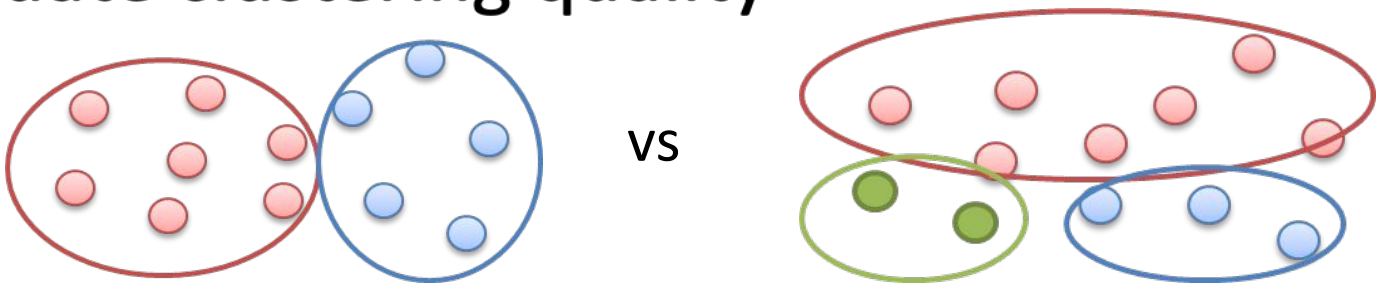
  Find a *natural* grouping of the data such that

  - Inter-cluster similarity is maximized
  - Intra-cluster similarity is minimized



Aspects of clustering

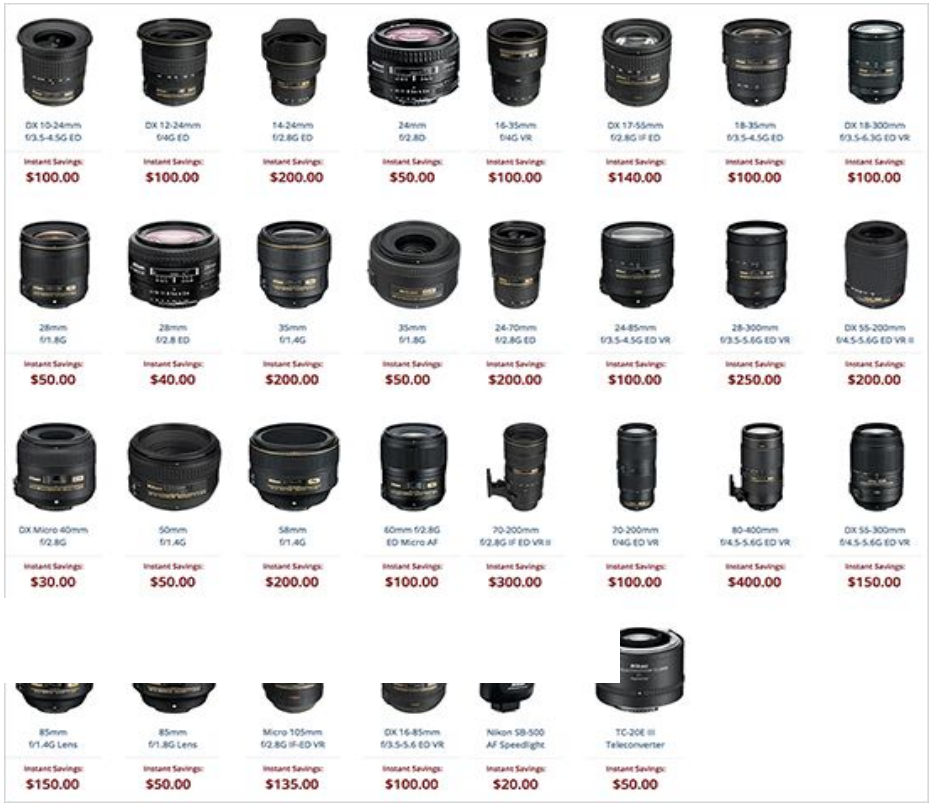
1. A proximity measure
   - Similarity measure $s(x_i, x_j)$: large if x_i and x_j are similar
   - Distance measure $\mathrm{d}(x_i, x_j)$: small if x_i and x_j are similar
2. A clustering algorithm
3. Criterion to evaluate clustering quality



Similarity / Distance Measures

Depends on the problem domai
data type

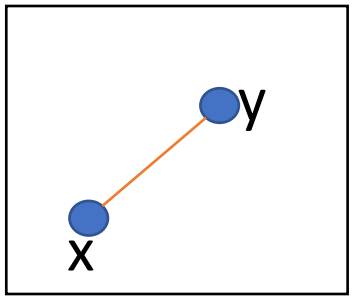
- Customers
- Time series
- Text
- Images



- Similarity or distance measures:
  - Euclidean
  - Manhattan distance
  - Cosine similarity
  - Pearson correlation
  - …

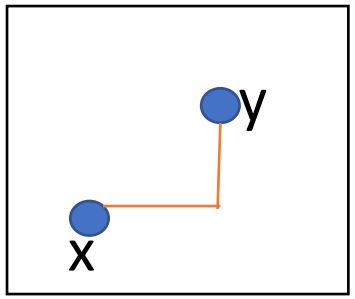
Distance/ Similarity measures

- Euclidean distance $d(\boldsymbol{X_i}, \boldsymbol{X_j}) = \sqrt{\sum_{s=1}^{d} |x_{is} - x_{js}|^2}$



Manhattan distance

$$d(\boldsymbol{X_i}, \boldsymbol{X_j}) = \sum_{s=1}^{d} |x_{is} - x_{js}|$$

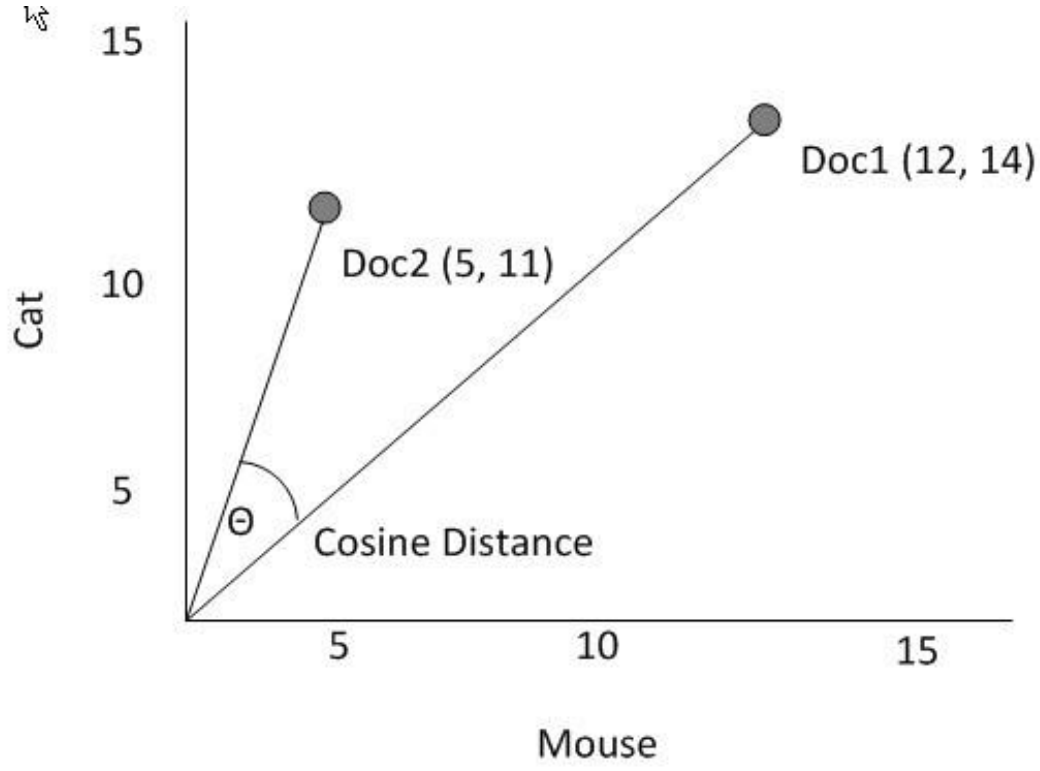


Minkowski family of distance measures

$$d(\boldsymbol{X_i}, \boldsymbol{X_j}) = \left(\sum_{s=1}^{d} |x_{is} - x_{js}|^p \right)^{1/p}$$

Similarity measures

•



(Dis)similarity measures

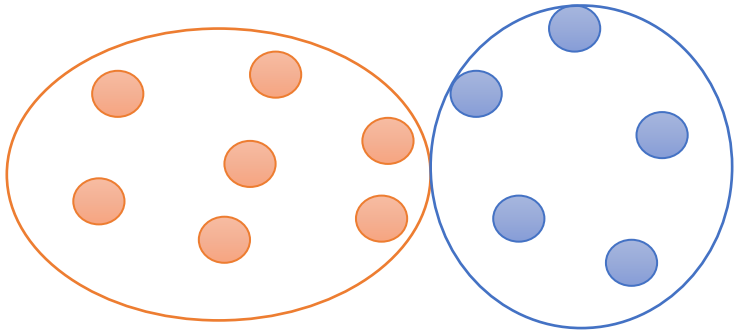
- Correlation coefficients (scale-invariant)
- Mahalanobis distance (wrt a Prob. Distribution P, Pos. semi-definite covariance matrix Σ)

$$d(x_i, x_i) = \sqrt{(x_i - x_j)\Sigma^{-1}(x_i - x_j)}$$

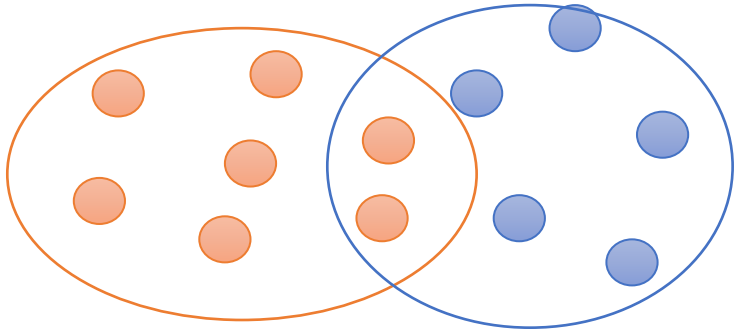
  - Euclidean distance weighted by the probability distribution spread.
- Pearson correlation

$$r(x_i, x_j) = \frac{Cov(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}}$$

Types of Clustering: Hard vs Soft



- Exclusive (Hard)
  - Non-overlapping subsets
  - Each item is a member of a single cluster



- Overlapping (Soft)
  - Potentially overlapping subsets
  - A item can simultaneously belong to multiple clusters

K-means Clustering

Clustering by Partitioning

Given K

- Construct a partition of *m* objects

$$X = \{\boldsymbol{X_1}, \boldsymbol{X_2}, \dots, \boldsymbol{X_m}\}$$

$\boldsymbol{X_i} = (x_{i1}, x_{i2}, \dots, x_{in})$ is a vector in a real-valued space $X \subseteq \mathbb{R}^n$

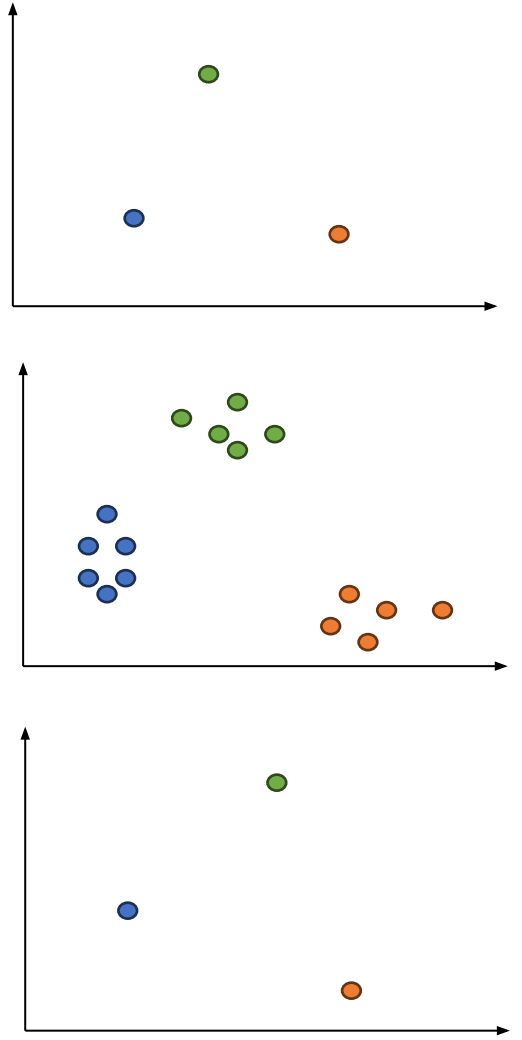
into a set of *K* clusters $C = \{C_1, C_2, \dots, C_K\}$

- The cluster mean μ_i serves as a prototype of the cluster C_i

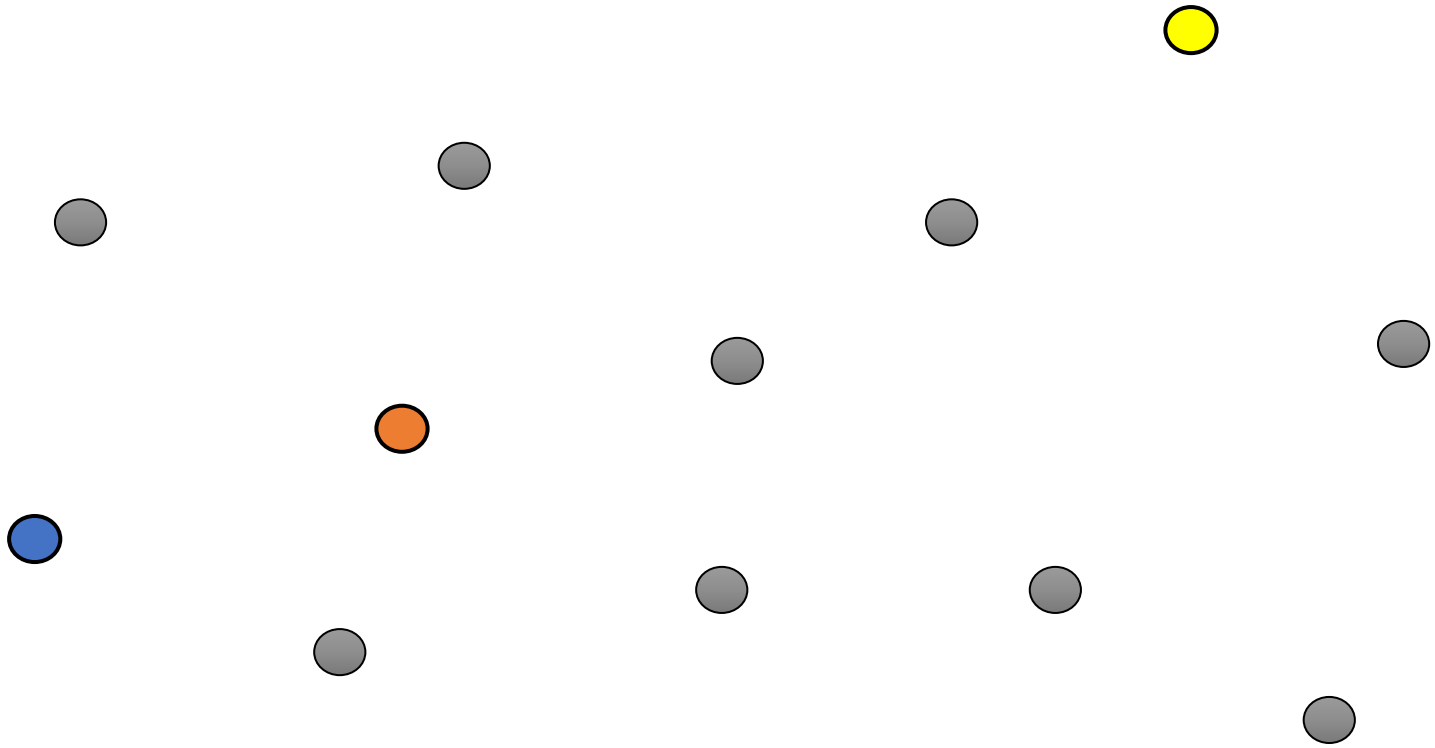
K-means algorithm (MacQueen, 1967)

Given K

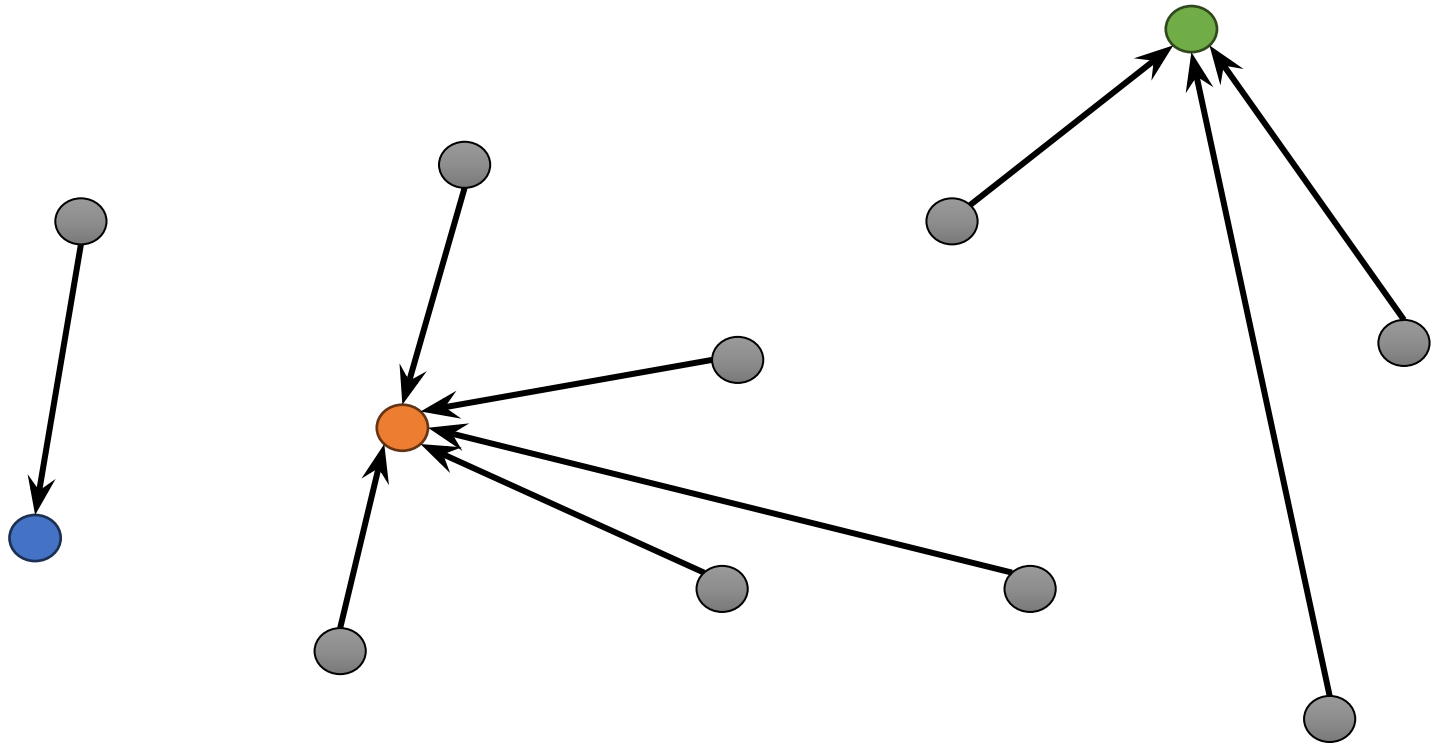
1. Initialization: Randomly choose *K* data points (seeds) to be the initial cluster centres
2. Cluster Assignment: Assign each data point to the closest cluster centre
3. Move Centroid: Re-compute the cluster centres using the current cluster memberships.
4. If a convergence criterion is not met, go to 2.



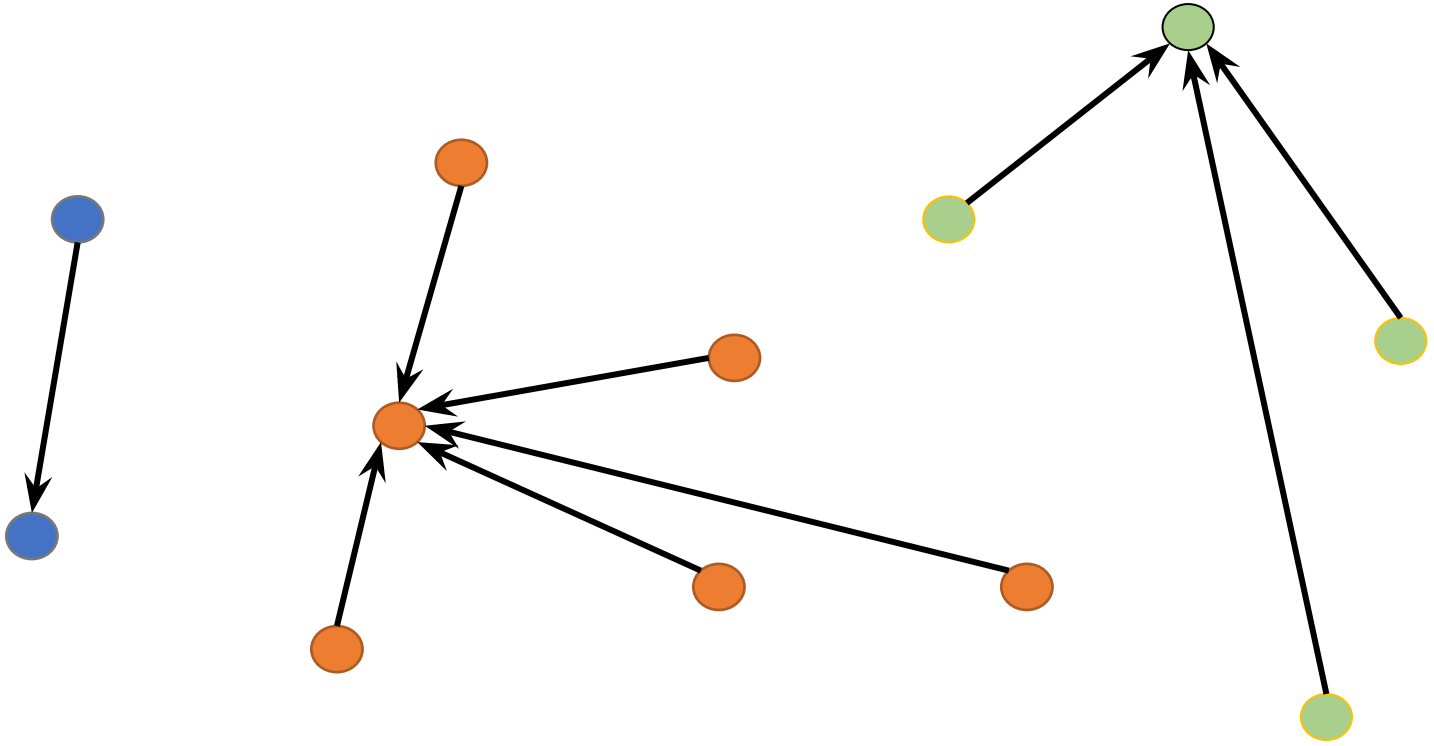
1. Random Initialization



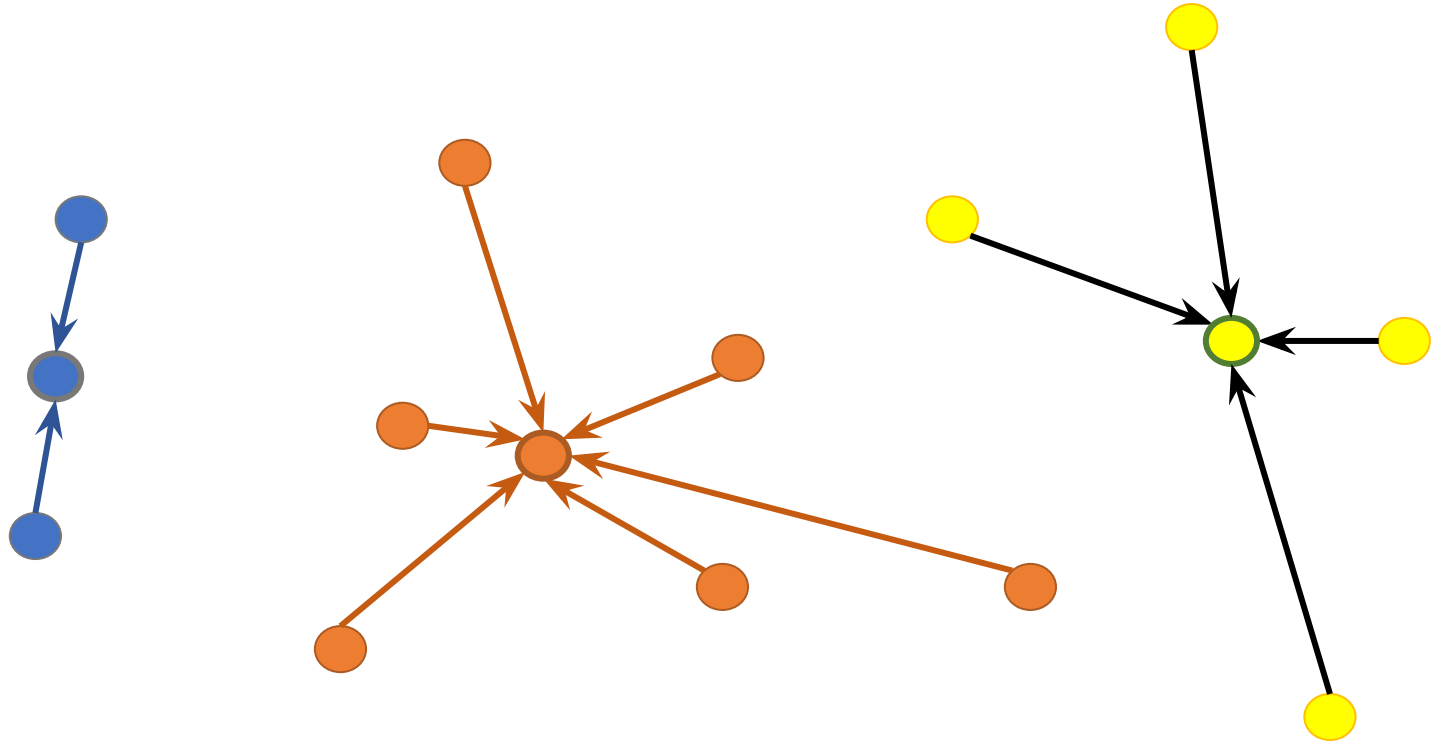
2. Cluster Assignment



2. Cluster Assignment



3. Move Centroid



K-Means & Its Stopping criterion

Given K

1. Initialization: Randomly choose *K* data points (seeds) to be the initial cluster centres
2. While not converged:
   I. Cluster Assignment: Assign each data point to the closest cluster centre
   II. Move Centroid: Re-compute the cluster centres using the current cluster memberships.
   III. If a convergence criterion is not met, go to 2.

- No re-assignments of data points to different clusters
- No (or minimum) change in centroids

OR

- Minimum decrease in the *sum of squared error*

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} \|x_i - \mu_i\|^2$$

K-Means Optimization Objective

- Good clustering: Within cluster variation is small

$$\underset{C}{\text{minimize}} \left\{ \sum_{i=1}^{K} WCV(C_i) \right\}$$

- E.g., the within-cluster sum of squares (WCSS) (sum of distance functions of each point in the cluster to the cluster mean)

$$\underset{C}{\text{argmin}} \sum_{i=1}^{K} \sum_{x \in C_k} \|X_i - \mu_i\|^2$$

K-Means Convergence Property

- $$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} \|X_i - \mu_i\|^2$$

- The algorithm is guaranteed to decrease the objective function SSE at every iteration.
- However it is not guaranteed to give the global optimum.

Convergence of *K*-Means

- Consider data points in Euclidean space
- Error of each data point = Euclidean distance of the point to its closest centroid

$$\min_{\mu} \min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

- SSE: total sum of the squared errors for each point
- Each step Re-computation monotonically decreases SSE (finds a local minima)

Convergence of *K*-Means

Given K

1. Initialization: Randomly choose *K* data points (seeds) to be the initial cluster centres
2. While not converged:
   - I. Cluster Assignment: Assign each data point to the closest cluster centre
   - II. Move Centroid: Re-compute the cluster centres using the current cluster memberships.
   - III. If a convergence criterion is not met, go to 2.

Recomputation monotonically decreases each square error

Proof:

Step 1: Fix means μ_i, then you minimize the expression values over C, $\sum_{i=1}^{k}\sum_{x_j \in C_i}\left|x_j - \mu_i\right|^2$

Step 2: Fix cluster assignment C. For fixed C, optimize over μ

(n_j = #members in cluster *j*)

$\sum_{x_j \in C_i}\left|x_j - a\right|^2$ reaches minimum for:

$\sum_{x_j \in C_i} -2\left(x_j - a\right) = 0$

i.e., $\sum x_j = \sum a = n_i a\,. \;\Rightarrow\; a = {}^1/_{n_i}\sum x_j = \mu_i$

K-means typically converges quickly

Convergence K-Means (summary)

Objective

$$\min_{\mu}\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

1. Fix μ, optimize *C:*

$$\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2 = \min_{c} \sum_{i}^{n} \left|x_i - \mu_{x_i}\right|^2$$

Step 1 of kmeans

2. Fix *C,* optimize μ:

$$\min_{\mu} \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|^2$$

   – Take partial derivative ~~of~~ with respect to μ_i and set to zero, we have

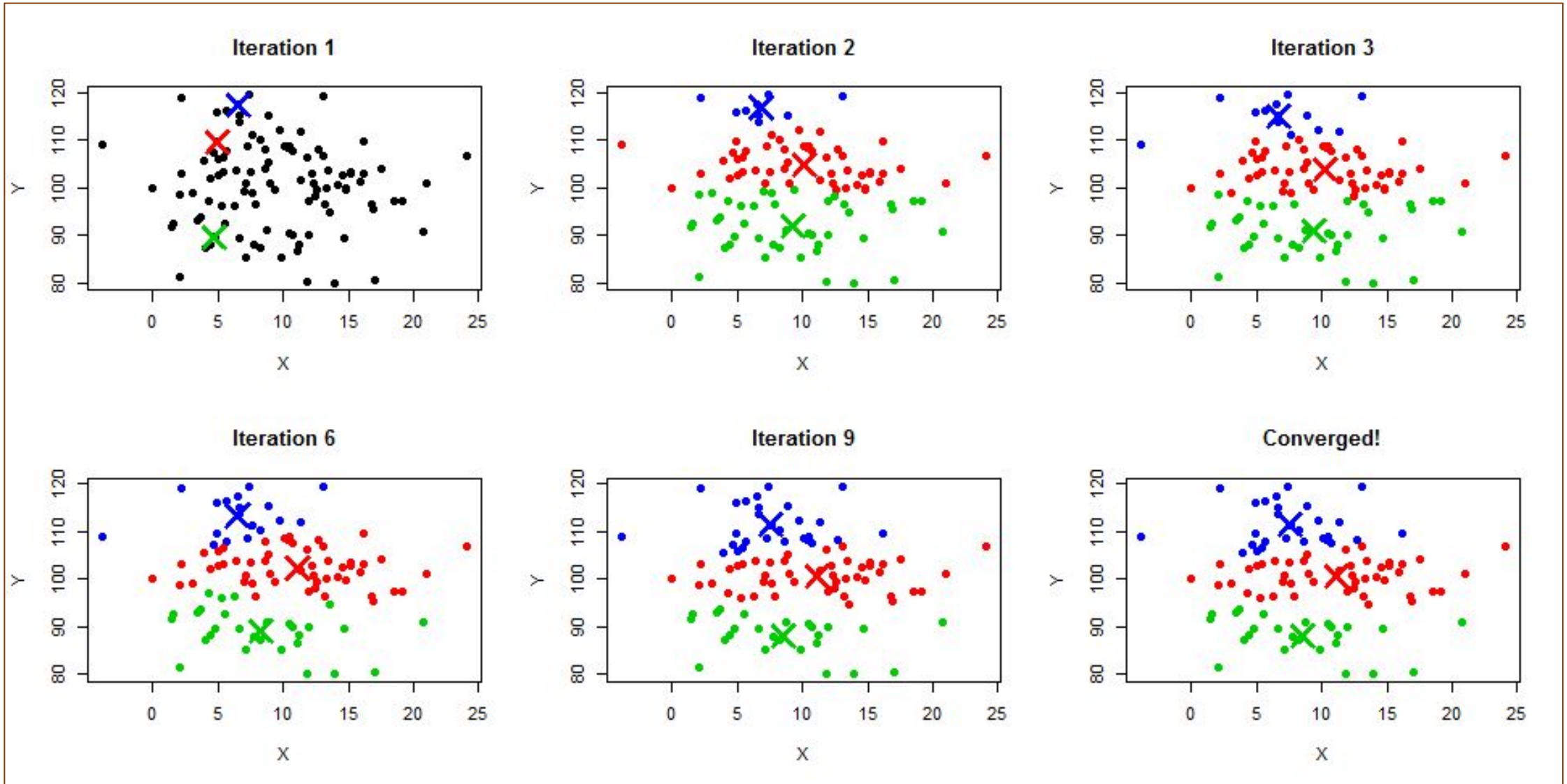
$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

Step 2 of kmeans

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

[Slide from Alan Fern]

K-means illustrated



Picking Cluster Seeds (Initial Values)

1. **Lloyd's Method**: Random Initialization

2. **K-Means++** : Iteratively construct a random sample with good spacing across the dataset.

Picking Cluster Seeds (Lloyd's Method)

Lloyd's Method: Random Initialization

- May converge at a local optimum

1. Perform multiple runs
   - Each run with a different set of randomly chosen seeds
2. Select that configuration that gives minimum SSE

Picking Cluster Seeds (K-means++)

- Choose centers at random from the data points
  - Weight the probability of choosing the centres according to their squared distance from the closest centre already chosen

In other words,

- Choose 1 center randomly.
- Choose second furthest from the first.
- Choose third furthest from first and second.
- … and so on.

- Let $D(X)$ be the distance between a point X and its nearest centre.
- Choose the next centre proportional to $D^2(X)$

- Initialization Algorithm
  - Choose c_1 at random.
  - For $j = 2$ to K
    - Pick c_j from the remaining data points $\{X_i\}$
      $\Pr(c_j = X_i) \propto D^2(X_i)$

How to select K?

1: Use cross validation to select K

- What should we optimize?

2: Let the domain expert look at the clustering and decide

3: The "knee" solution

- Plot the objective function values for different values of K
- "knee finding" or "elbow finding".

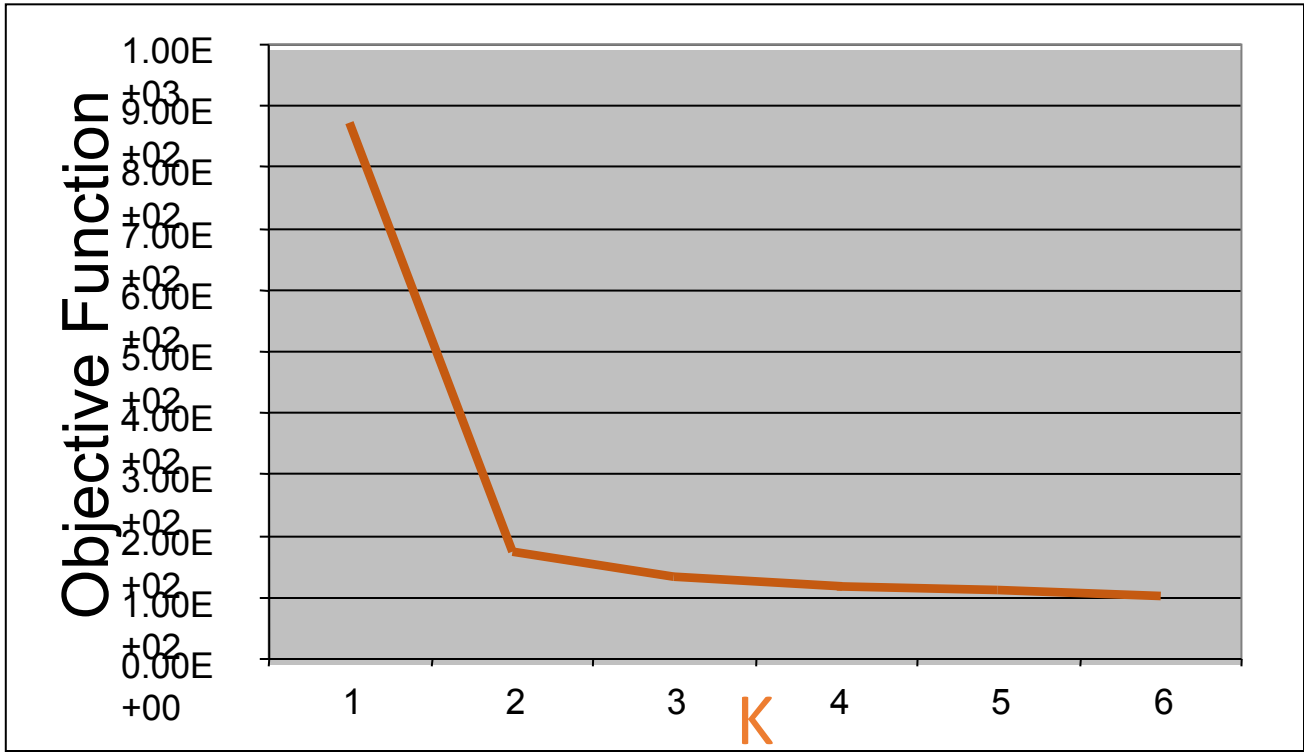


Figure from slide by Eamonn Keogh

K-Means Time Complexity

- m items, n dimensions, K clusters, I iterations
- Computing distance between two items is $O(n)$
- Reassigning clusters: $O(Km)$ distance computations
- Total for one iteration $O(Kmn)$
- Computing centroids: Each item gets added once to some centroid: $O(mn)$
- Assume these two steps are each done once for I iterations: $O(IKmn)$

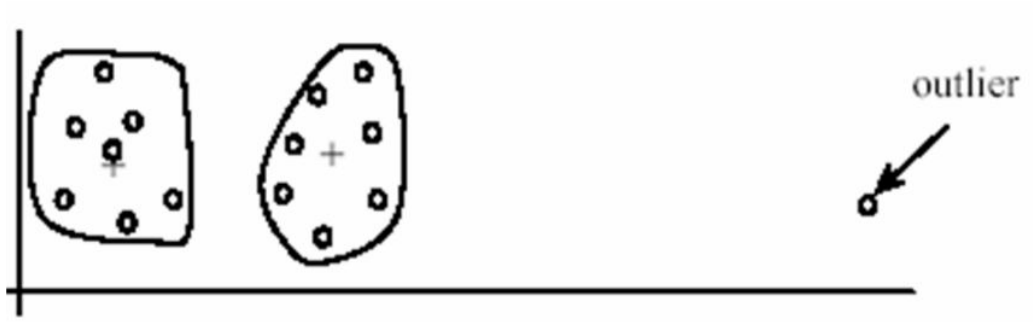
K-Means Pros and Cons

Pros

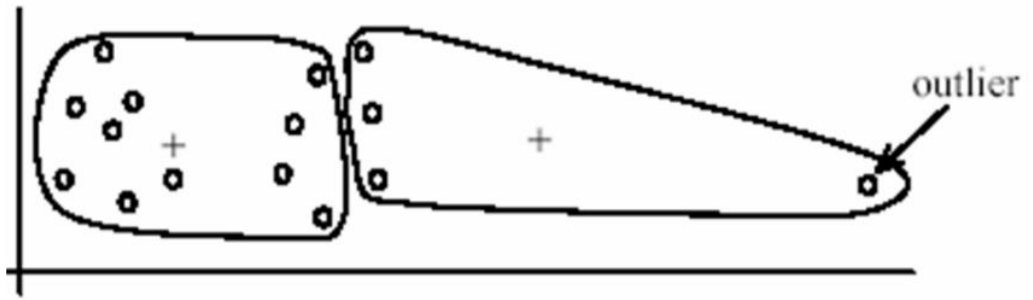
- Fast, robust easy to implement
- Relatively efficient: $O(IKmn)$
  - Normally, $K,\ I, m\ <<\ n$
- Gives best result when data set are distinct or well separated from each other.

Cons

- Requires K
- Sensitive to outliers
- Prone to local minima
- All clusters have the same parameters (e.g., distance measure is non-adaptive)

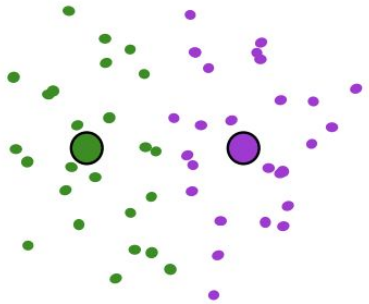


(B): Ideal clusters

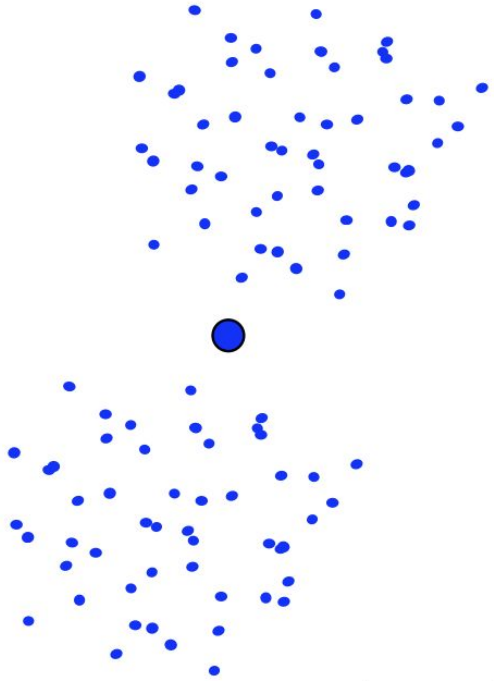


K-means Getting Stuck (Varying K)

A local optimum:

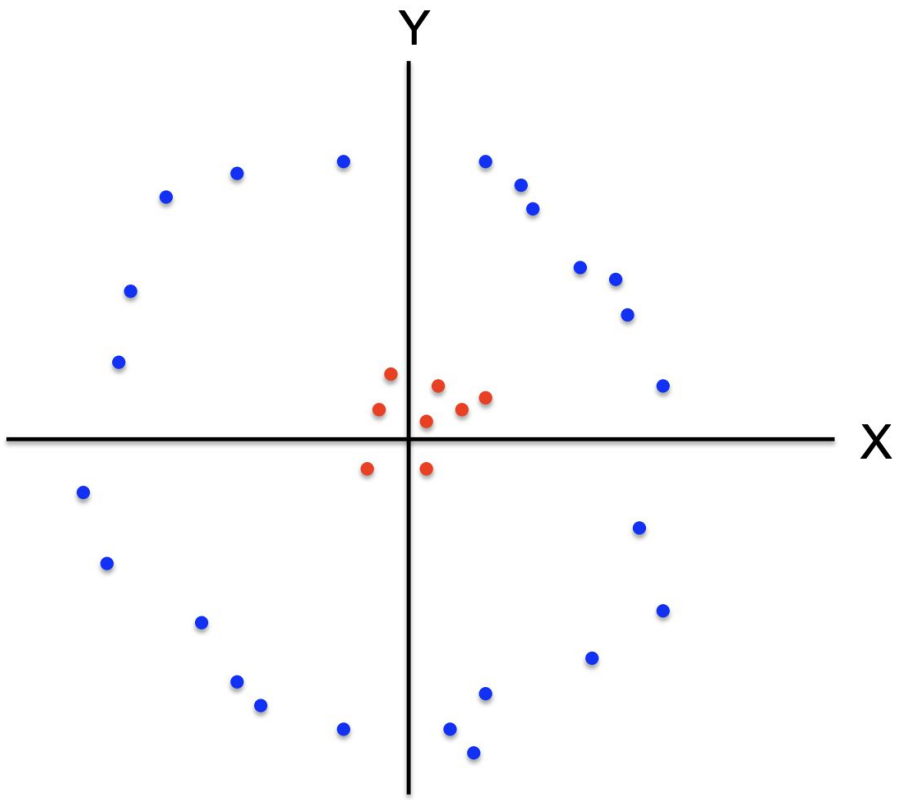


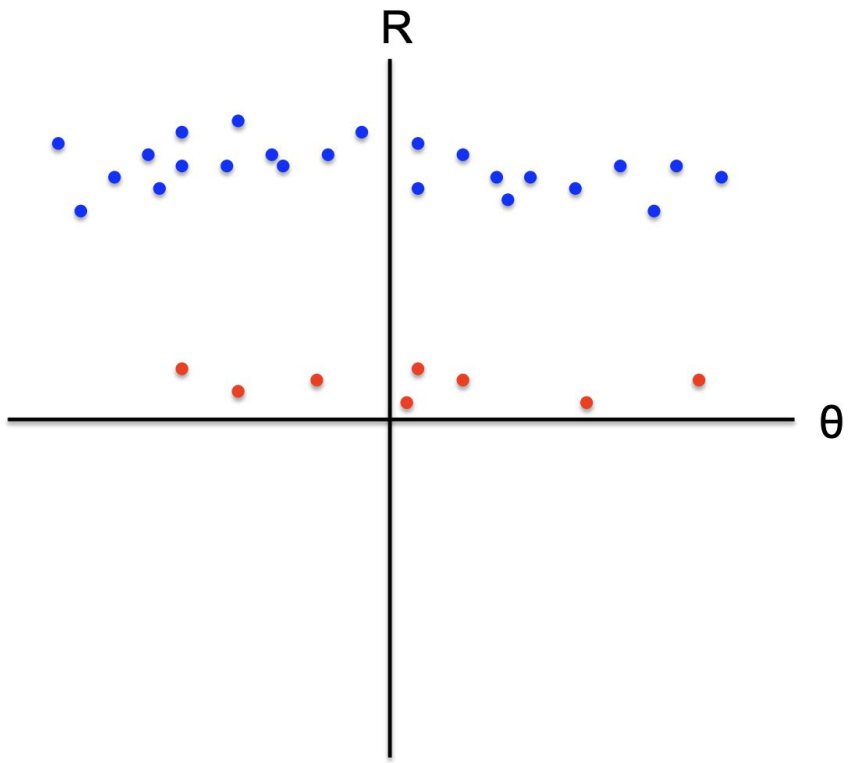
Would be better to have
one cluster here



… and two clusters here

K-means not able to properly cluster





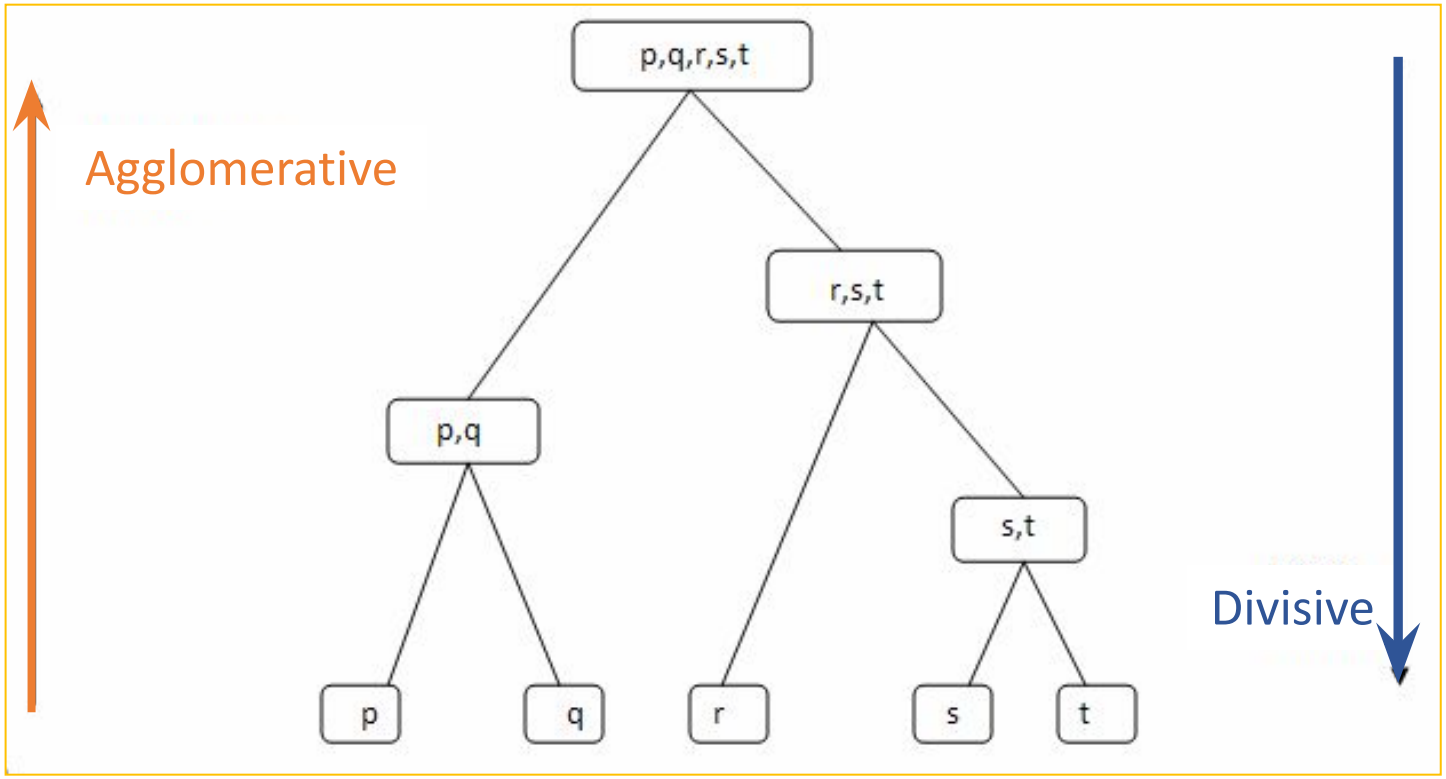
Changing the Features or distance function (kernel) May help

Hierarchical Clustering

Hierarchical Algorithms

Agglomerative (bottom-up): Start with each point as a cluster. Clusters are combined based on their "closeness".

Divisive (top-down): Start with one cluster including all points and recursively split each cluster.



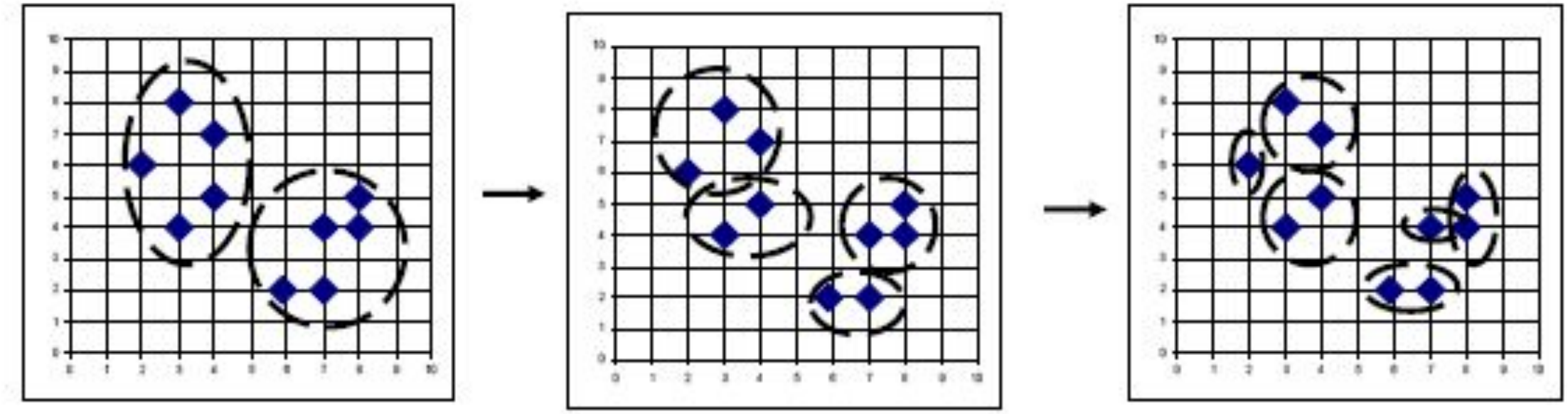
Types of hierarchical clustering

1. Divisive Hierarchical Clustering

- Hierarchical k-means
  - Start with all data points $\{x_1, x_2, \dots, x_m\}$ in one cluster
  - Run k-means on the data and split into k child clusters $\{c_1, c_2, \dots, c_k\}$
  - Recursively run k-means on each child cluster
  - Stop when only singleton clusters of individual data points remain.

Hierarchical clustering

Divisive (Top-down)



Slide credit: Min Zhang

Types of hierarchical clustering

2. Agglomerative (bottom up) clustering

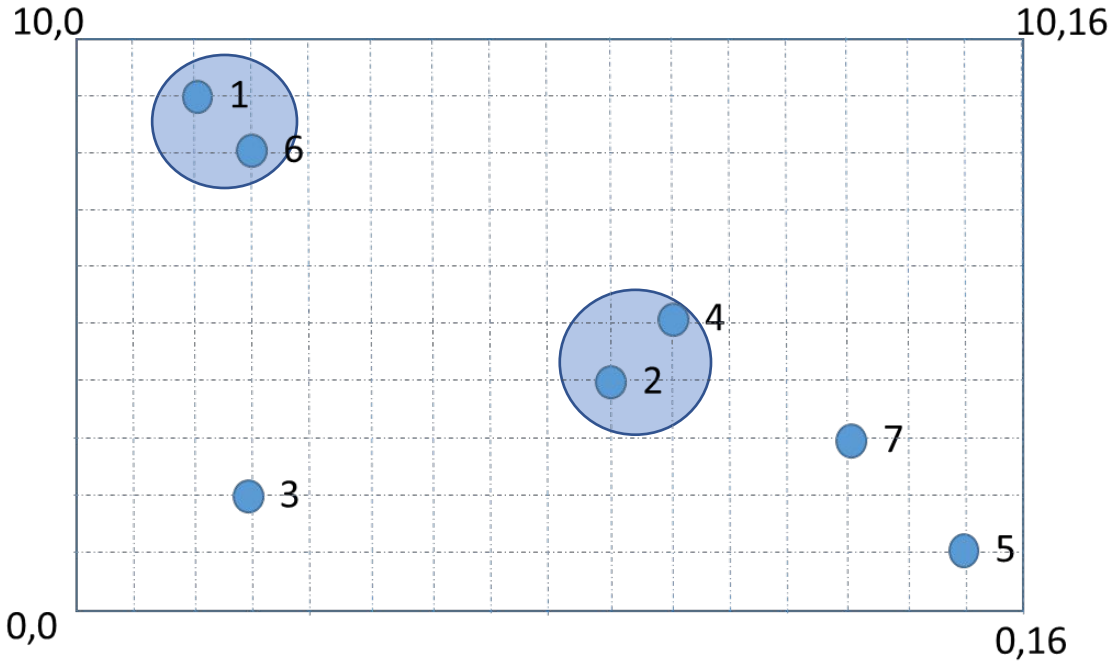
- **Agglomerative (bottom up) clustering**: Builds the hierarchical tree from the bottom level

1. Start with a collection of m singleton clusters $c_i = \{x_i\}$
2. Repeat
   1. Merge the most similar (or nearest) pair of clusters c_i, c_j into a new cluster c_{i+j}
   2. Remove c_i, c_j from the collection and add c_{i+j}
3. Until a single cluster is left

Produces a hierarchical tree of clusters or a dendrogram

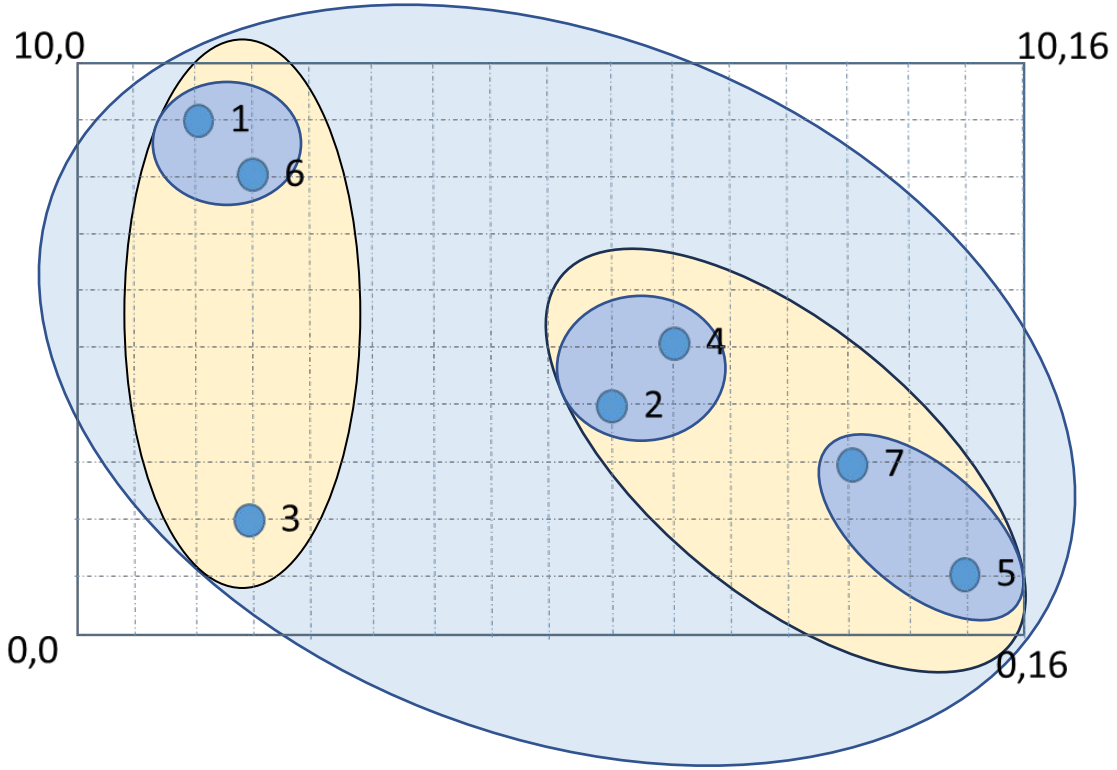
Hierarchical Clustering: Example

1. C= {1},{2},{3},{4},{5},{6},{7}
2. C = **{1,6}** ,{2},{3},{4},{5},{7}
3. {1,6}, **{2,4}** ,{3},{5},{7}

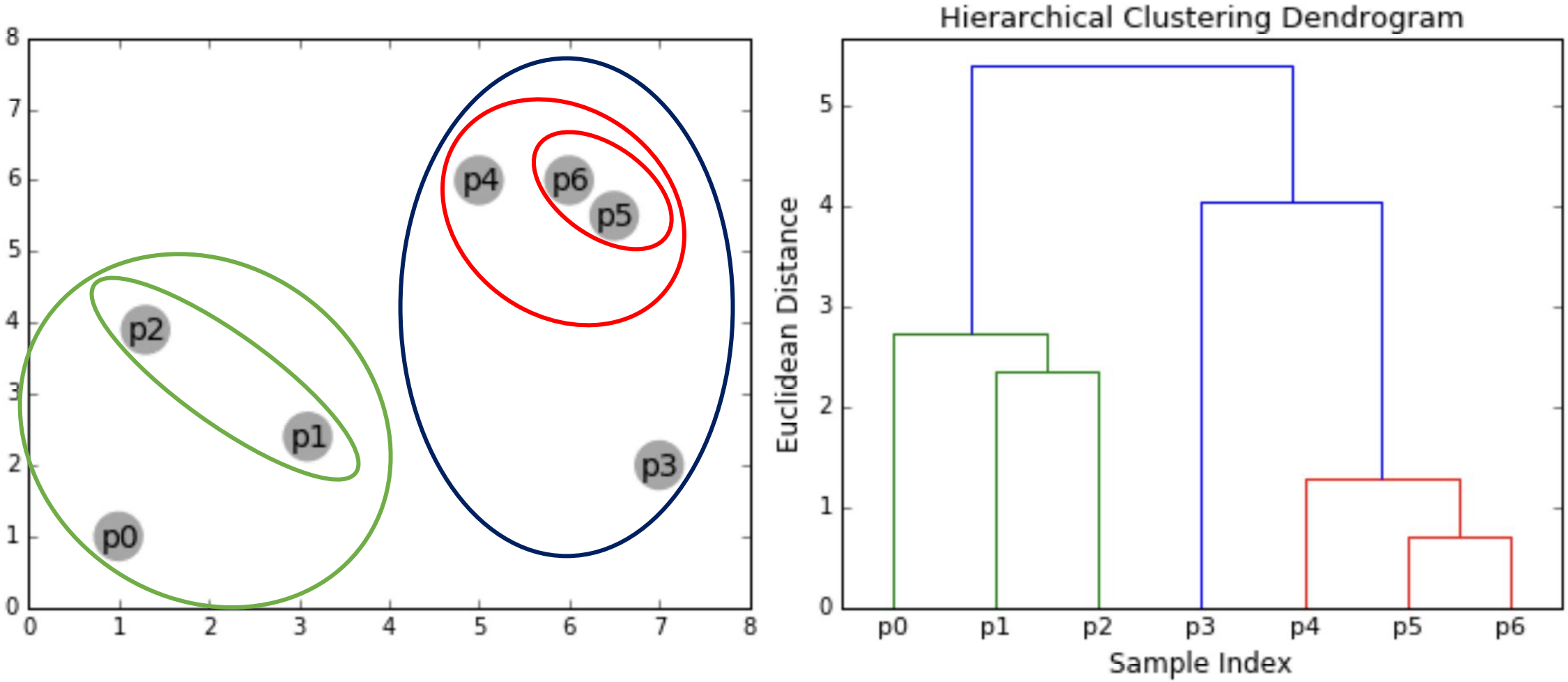


Hierarchical Clustering: Example

1. C= {1},{2},{3},{4},{5},{6},{7}
2. **{1,6}** ,{2},{3},{4},{5},{7}
3. {1,6}, **{2,4}** ,{3},{5},{7}
4. {1,6},{2,4},{3}, **{5,7}**
5. {1,6}, **{2,4,5,7}** ,{3}
6. **{1,6,3},** {2,4,5,7}
7. {1,6,3,2,4,5,7}



Dendrogram: Hierarchical Clustering



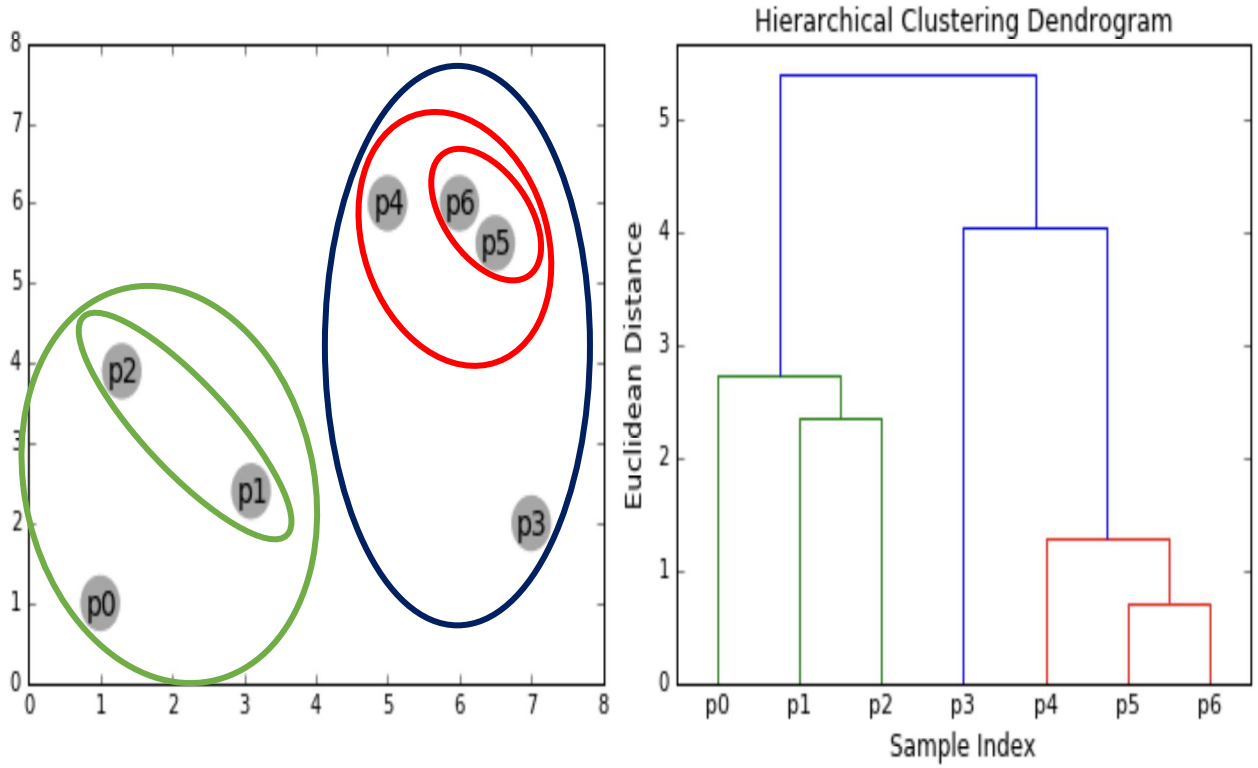
Dendrogram: Hierarchical Clustering

Dendrogram

- Input set S
- Nodes represent subsets of S

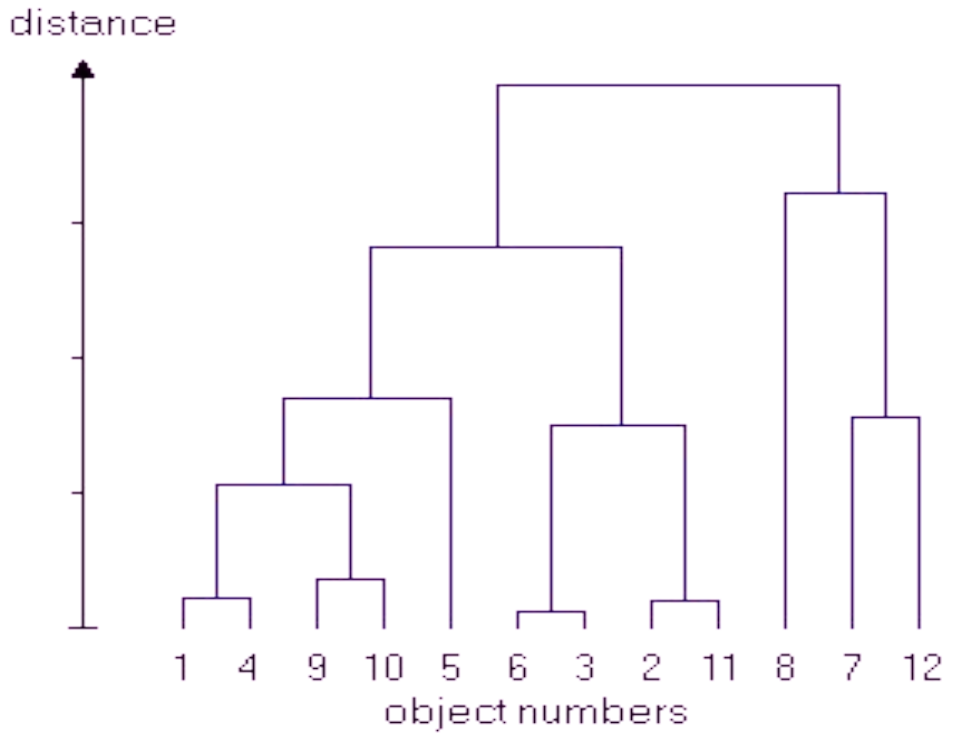
Features of the tree

- The root is S
- The leaves are the individual elements of S
- The internal nodes are defined as the union of their children.

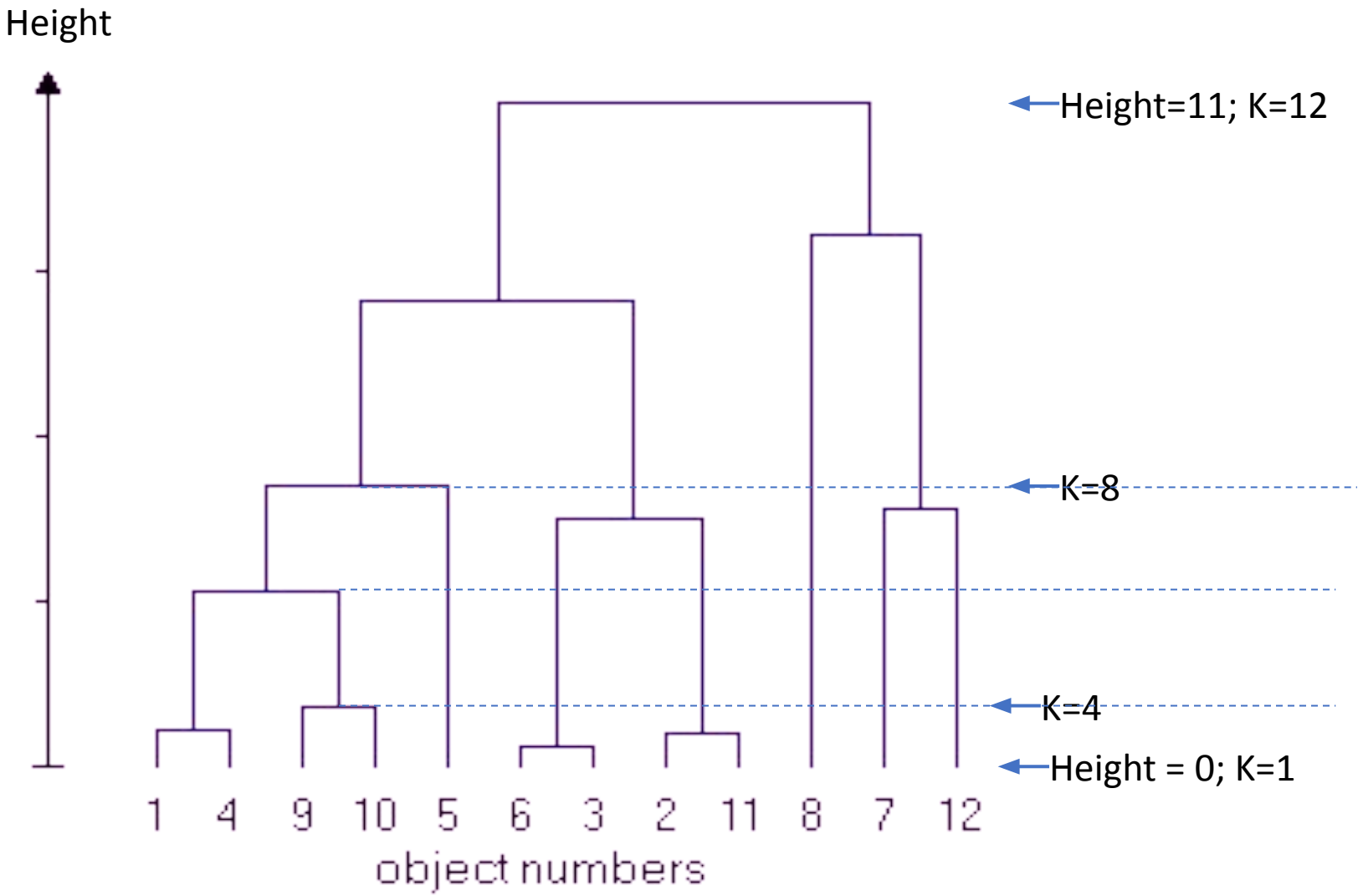


Dendrogram: Definition

- Height at leaf: 0
- May be cut at any level: Each connected component forms a cluster.
- Any height represents a horizontal cut and a partition of S into several (nested) clusters.



Hierarchical clustering



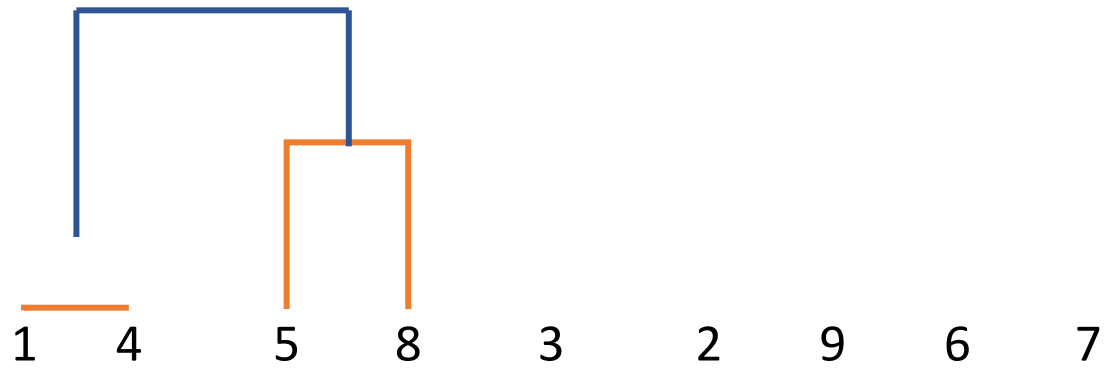
Height = 0
Clusters = 9

1 4 5 8 3 2 9 6 7

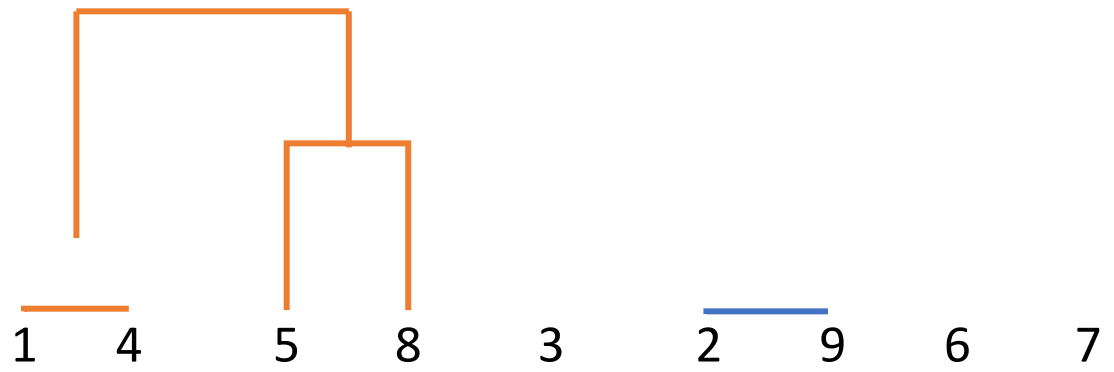
Height = 1
Clusters=8

Height = 2
Clusters=7

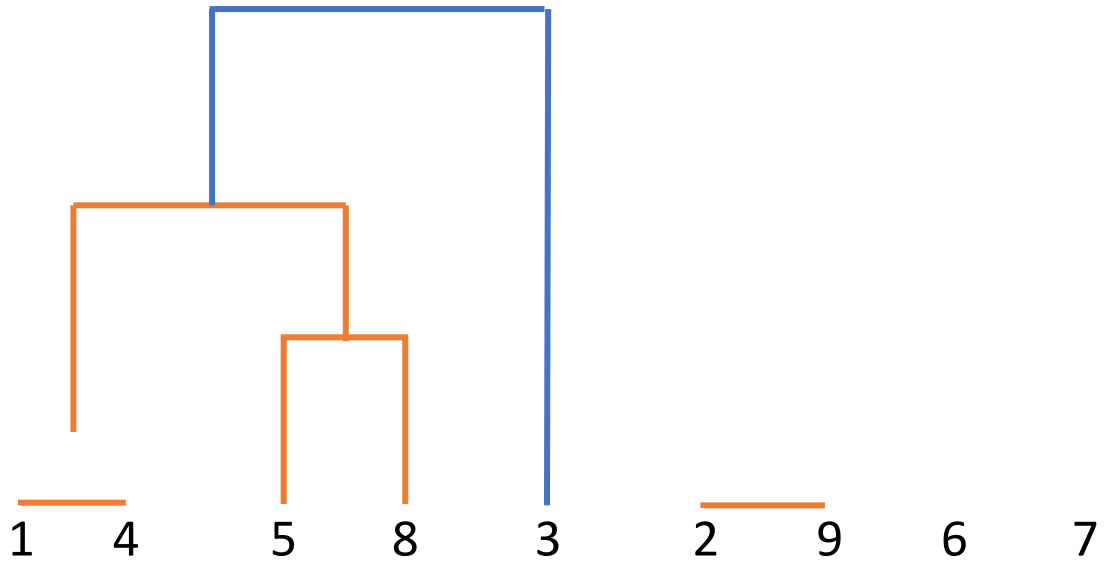
Height = 3
Clusters=6



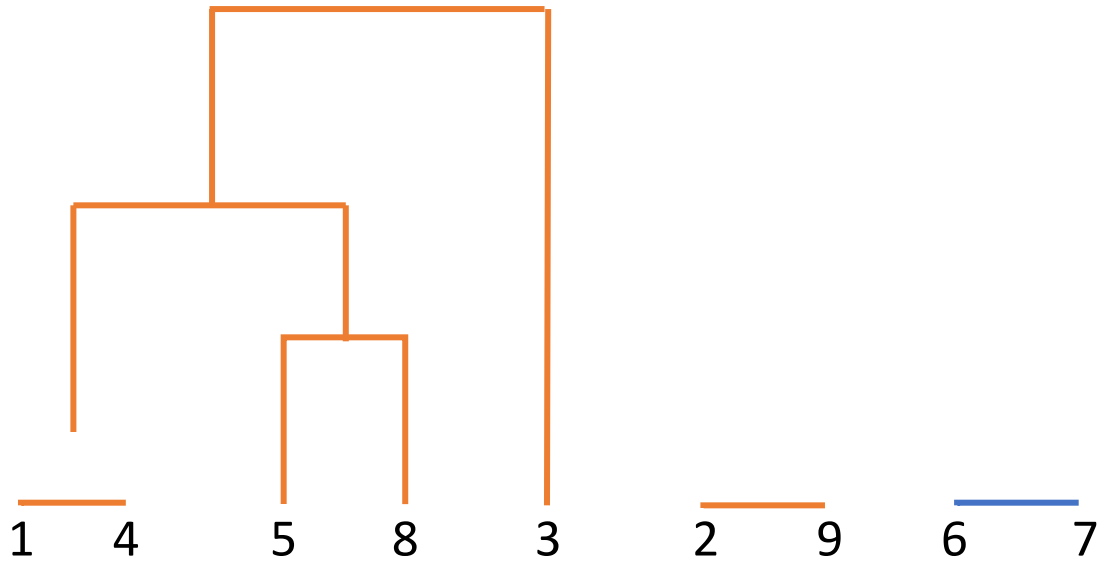
Height = 4
Clusters=5



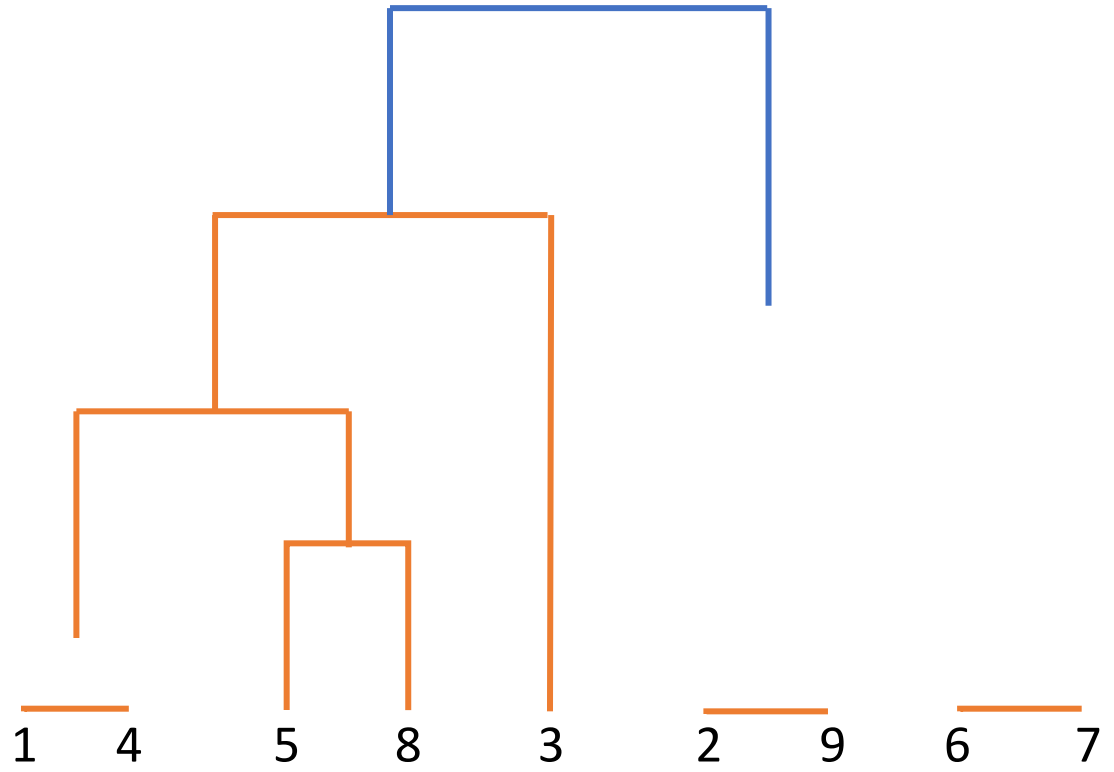
Height = 5
Clusters=4



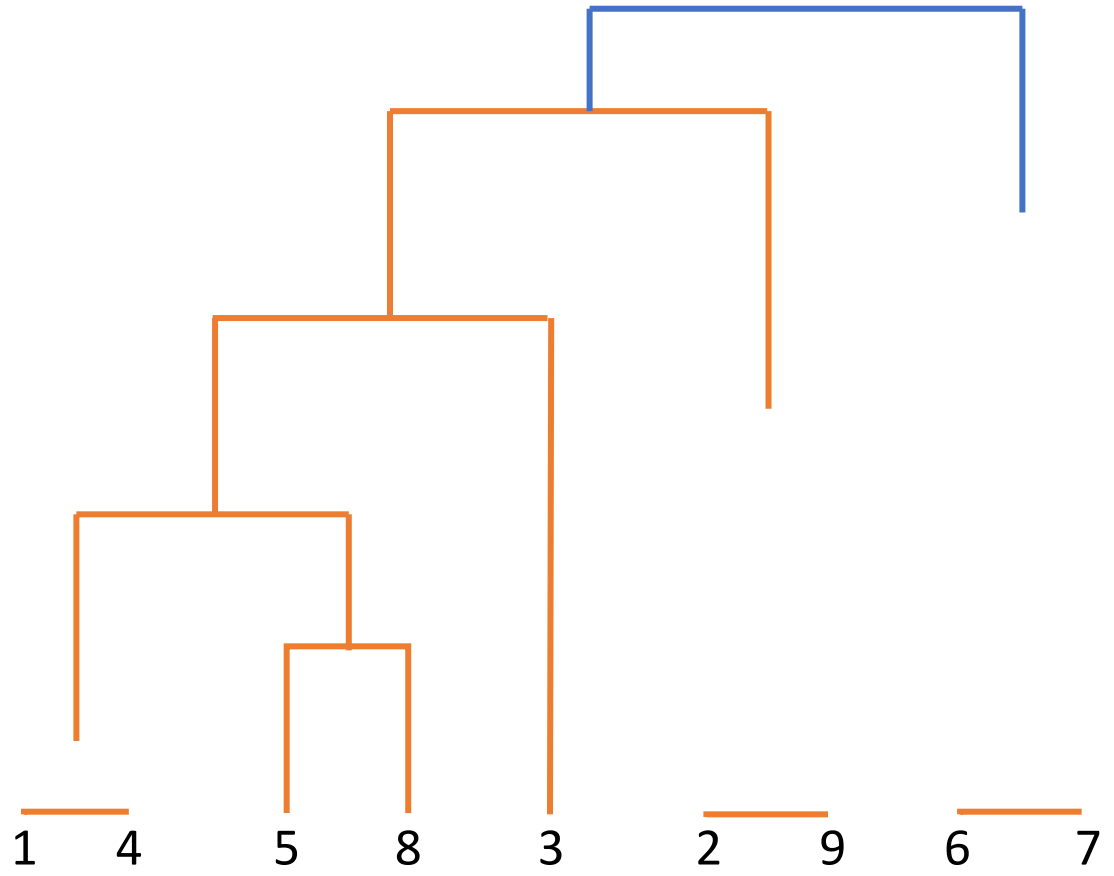
Height = 6
Clusters=3



Height = 7
Clusters=2



Height = 8
Clusters=1



Hierarchical Agglomerative clustering

- Start with a collection of m singleton clusters $c_i = \{x_i\}$
- Compute the *distance matrix* between the clusters.
- Repeat
  - Merge the two closest clusters c_i, c_j into a new cluster c_{i+j}
  - Update the distance matrix: Remove c_i, c_j from the collection and add c_{i+j}

  Until only a single cluster remains.

Different definitions of the distance leads to different algorithms.

Distance Measures

Real variables

- Euclidean
- Cosine
- Correlation
- Manhattan
- Minkowski
- Mahalanobis
- …

Discrete variables

- Hamming
- Jaccard
- …

Linkage: Definition

-

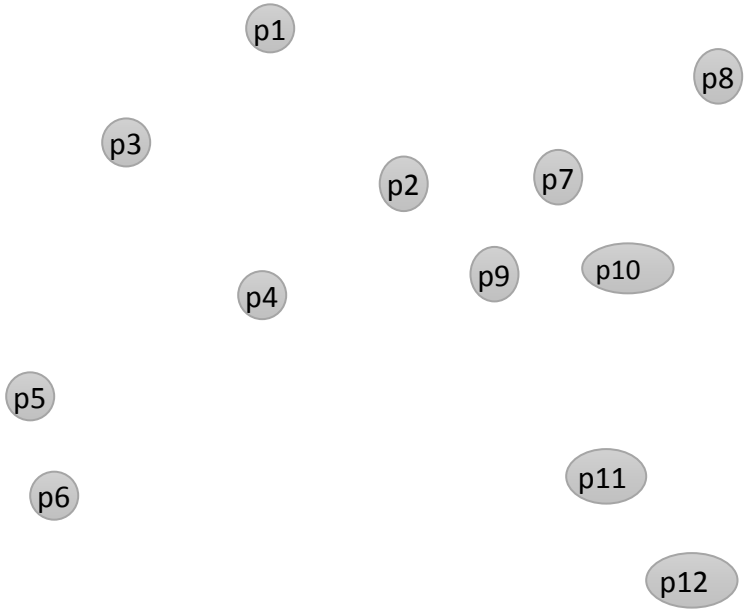
- Given distances d_{ij} between data points X_i and X_j

- Given two clusters $c_i = \{\dots\}$ and $c_j = \{\dots\}$
- Compute distance $d(c_i, c)$ between c_i and c_j using distances between their contained data points

- Required property
  - Monotonicity: combination similarities cannot go up in the sequence of merges

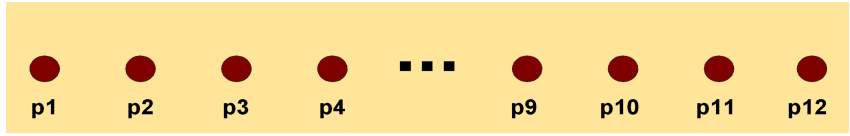
Initialization

- Each individual point is taken as a cluster
- Construct distance/proximity matrix

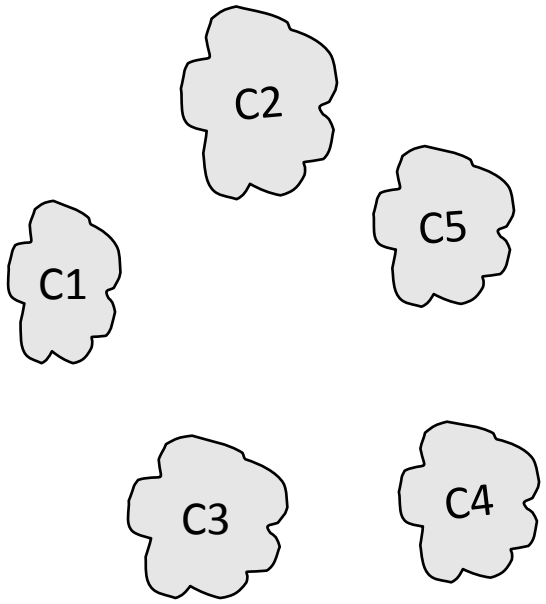


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Distance/Proximity Matrix



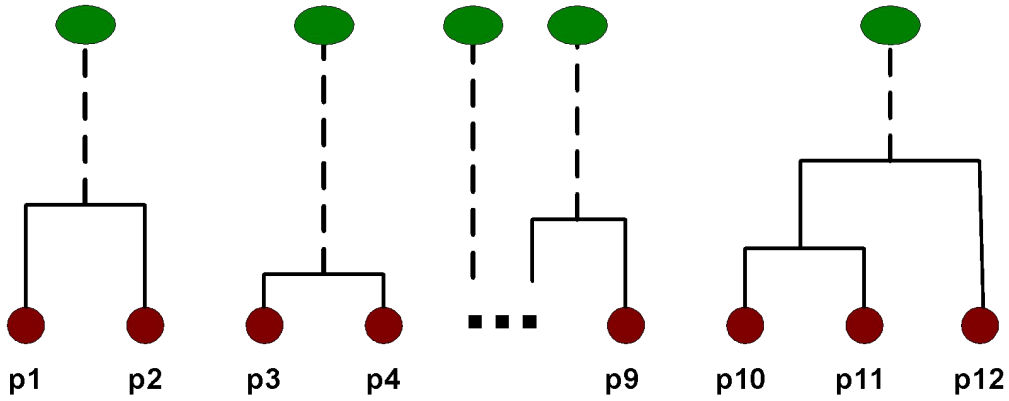
Intermediate State



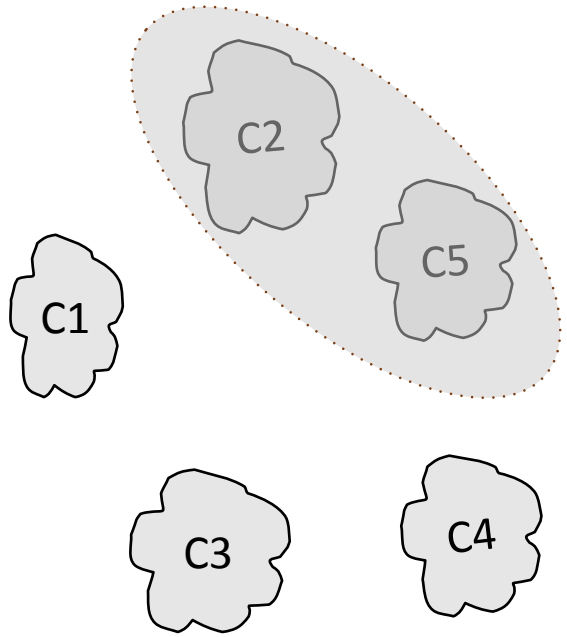
Distance/Proximity Matrix

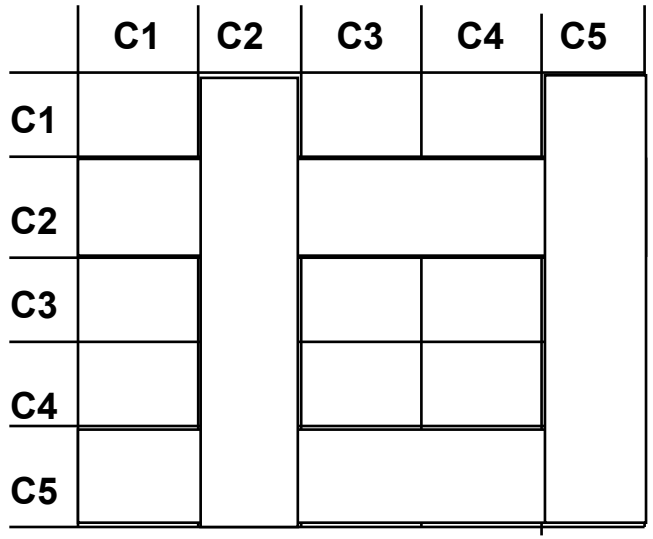
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

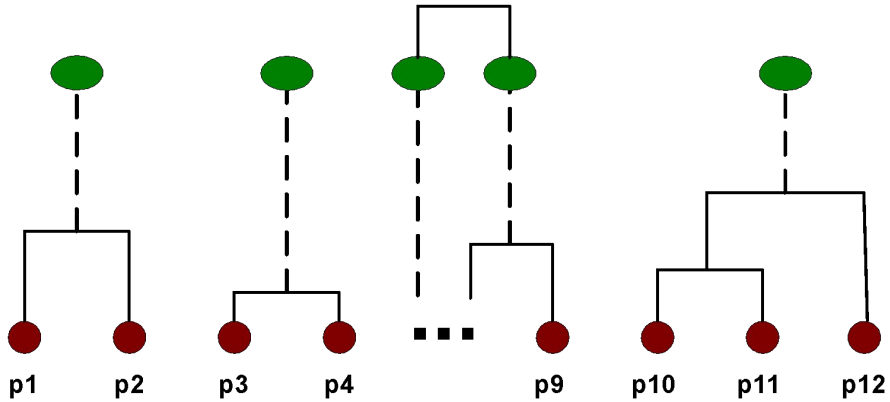
After some merging steps, we have some clusters



Intermediate State



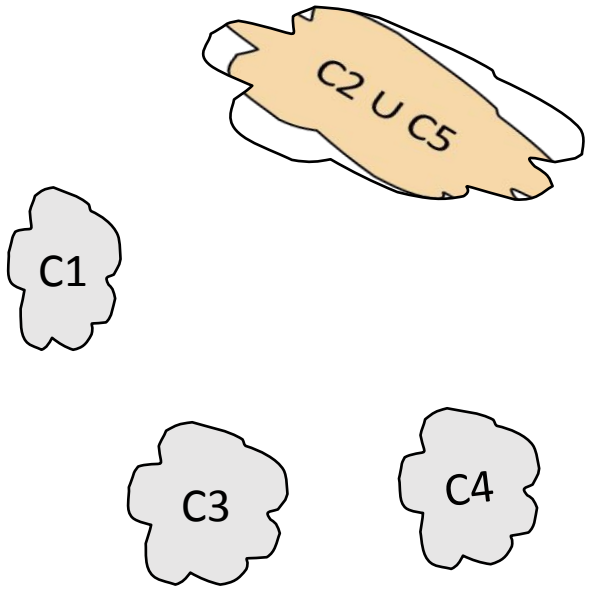




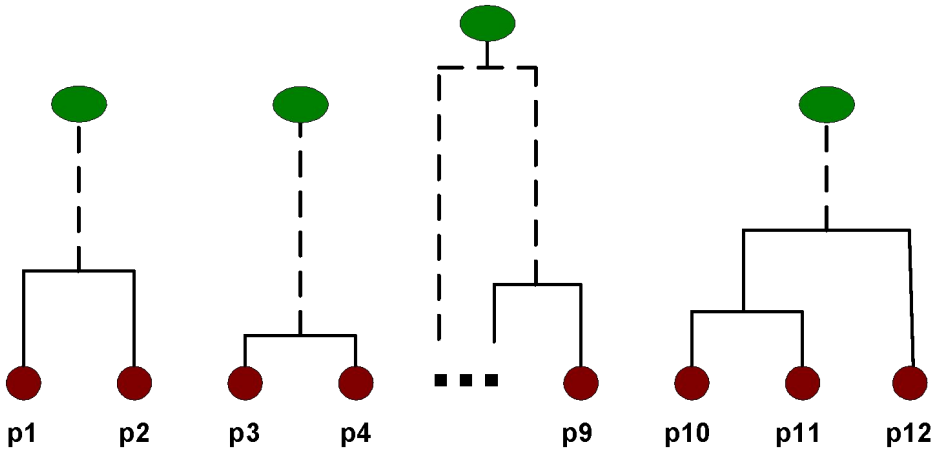
Merge the two closest clusters (C2 and C5) and update the distance matrix.

After Merging

Update the distance matrix



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		



Closest Pair

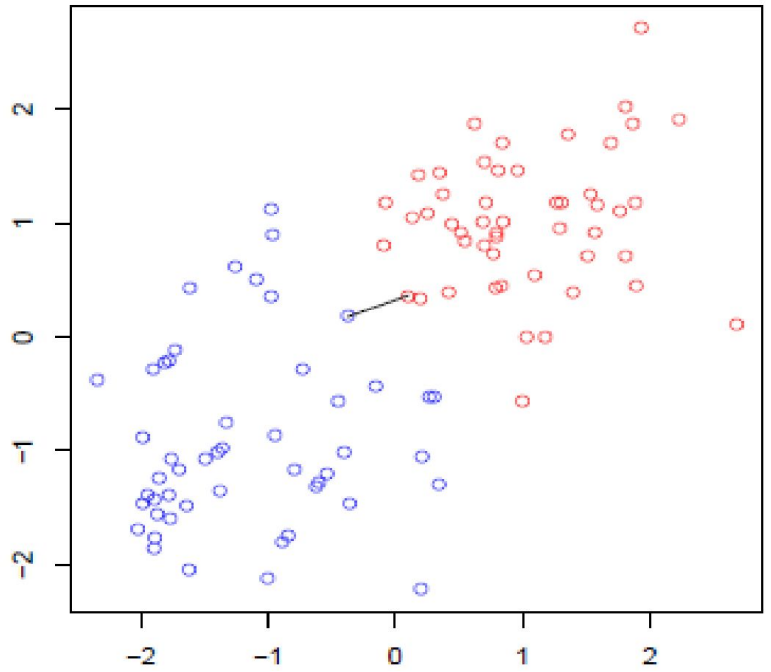
- A few ways to measure distances of two clusters.
- **Single-link**
  - Similarity of the *most* similar (single-link)
- **Complete-link**
  - Similarity of the *least* similar points
- **Centroid**
  - Clusters whose centroids (centers of gravity) are the most similar
- **Average-link**
  - Average cosine between pairs of elements

Distance between two clusters

- **Single-link** distance between clusters C_i and C_j is the *minimum distance* between any object in C_i and any object in C_j

$$\mathrm{sim}(C_i, C_j) = \max_{x \in C_i, y \in C_j} \mathrm{sim}(x, y)$$

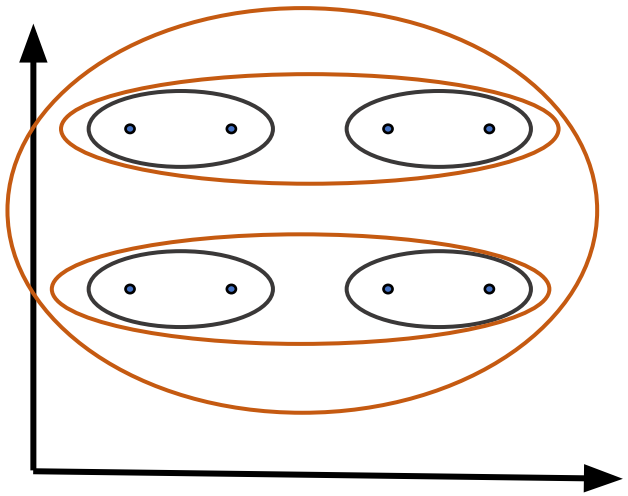
It can result in long and thin clusters.



Distance between two clusters

- **Single-link** distance between clusters C_i and C_j is the *minimum distance* between any object in C_i and any object in C_j

$$\mathrm{sim}(C_i, C_j) = \max_{x \in C_i, y \in C_j} \mathrm{sim}(x, y)$$

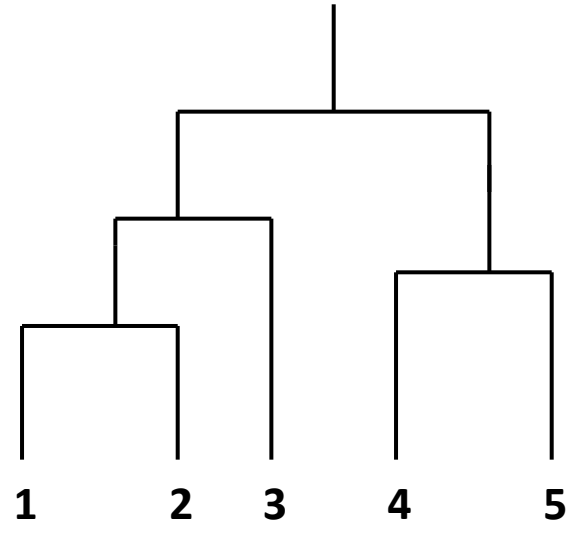


It can result in long and thin clusters.

Single-link clustering: example

- Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00

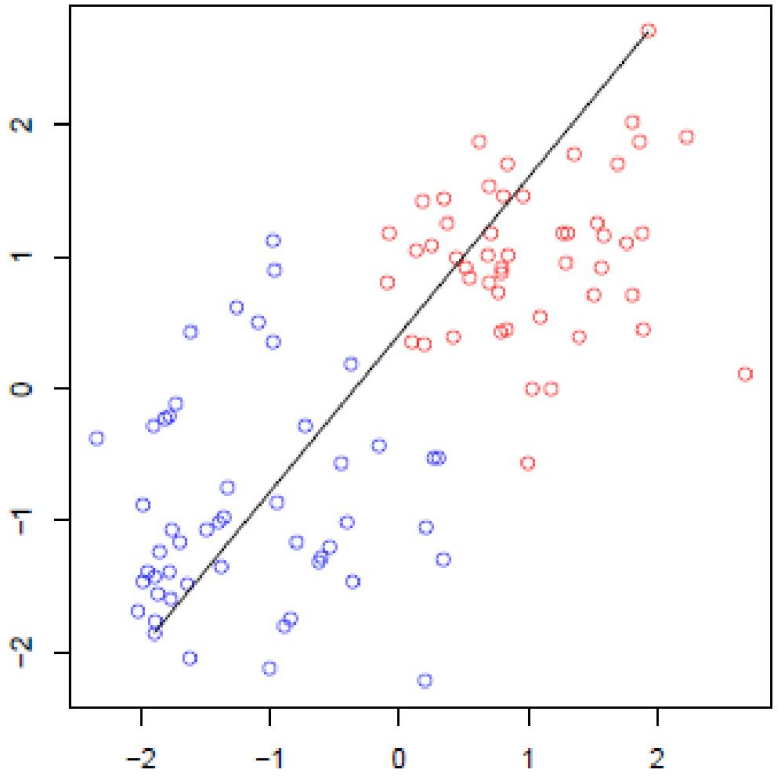


Complete link method

- The distance between two clusters is the distance of two furthest data points in the two clusters.

$$\text{sim}(C_i, C_j) = \min_{x \in C_i, y \in C_j} \text{sim}(x, y)$$

- Makes "tighter," spherical clusters that are typically preferable.

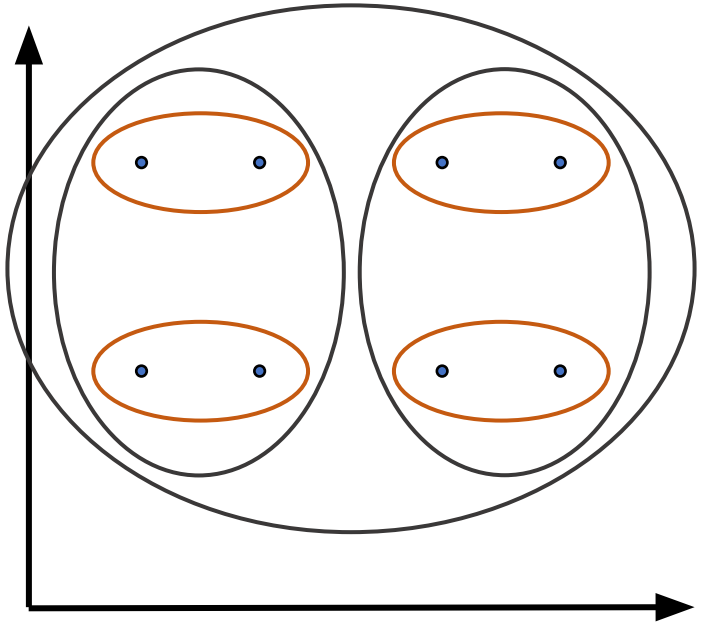


Complete link method

- The distance between two clusters is the distance of two furthest data points in the two clusters.

$$\text{sim}(C_i, C_j) = \min_{x \in C_i, y \in C_j} \text{sim}(x, y)$$

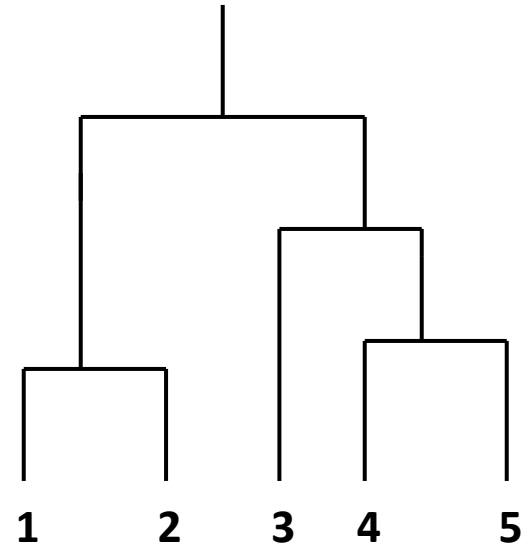
- Makes "tighter," spherical clusters that are typically preferable.



Complete-link clustering: example

- Distance between clusters is determined by the two most distant points in the different clusters

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Computational Complexity

•

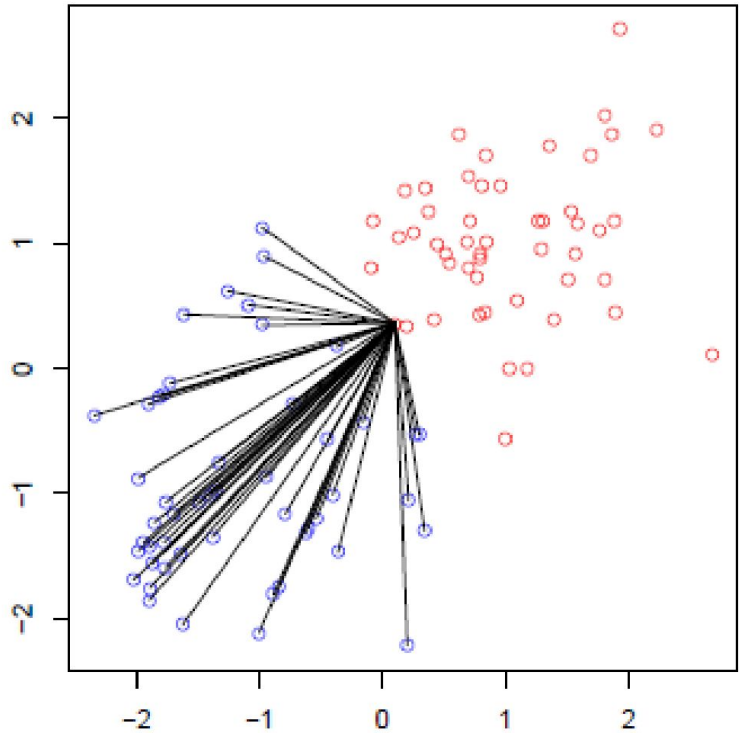
- In the first iteration, all HAC methods need to compute similarity of all pairs of N initial instances, which is $O(N^2)$.
- In each of the subsequent $N - 2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.
- In order to maintain an overall $O(N^2)$ performance, computing similarity to each other cluster must be done in constant time.
  - Often $O(N^3)$ if done naively or $O(N^2 \log N)$ if done more cleverly

Average Link Clustering

- Similarity of two clusters = average similarity between any object in Ci and any object in Cj

$$sim(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{x \in c_i} \sum_{y \in c_j} sim(x, y)$$

- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters



Compromise between single and complete link. Less susceptible to noise and outliers.

The complexity

- All the algorithms are at least $O(n^2)$. n is the number of data points.
- Single link can be done in $O(n^2)$.
- Complete and average links can be done in $O(n^2 \log n)$.
- Due the complexity, hard to use for large data sets.

Extra: Evaluation of Clustering

Evaluation

- Quality: “goodness” of clusters
- Aspects of validation
  - External Index: Measure the extent to which the clustering results match to ground truth labels
  - Internal Index: without reference to external information
  - Statistical framework: determine reliability
    - To what confidence level, the clusters are not formed by chance

External Evaluation

Evaluated based on data with known class labels e

- Rand Index

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

- Jaccard Index

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP + FP + FN}$$

- F-measure

External Evaluation

Entropy and Purity

- The number of objects in both the k-th cluster and j-th groundtruth: $|C_k \cap P_j|$

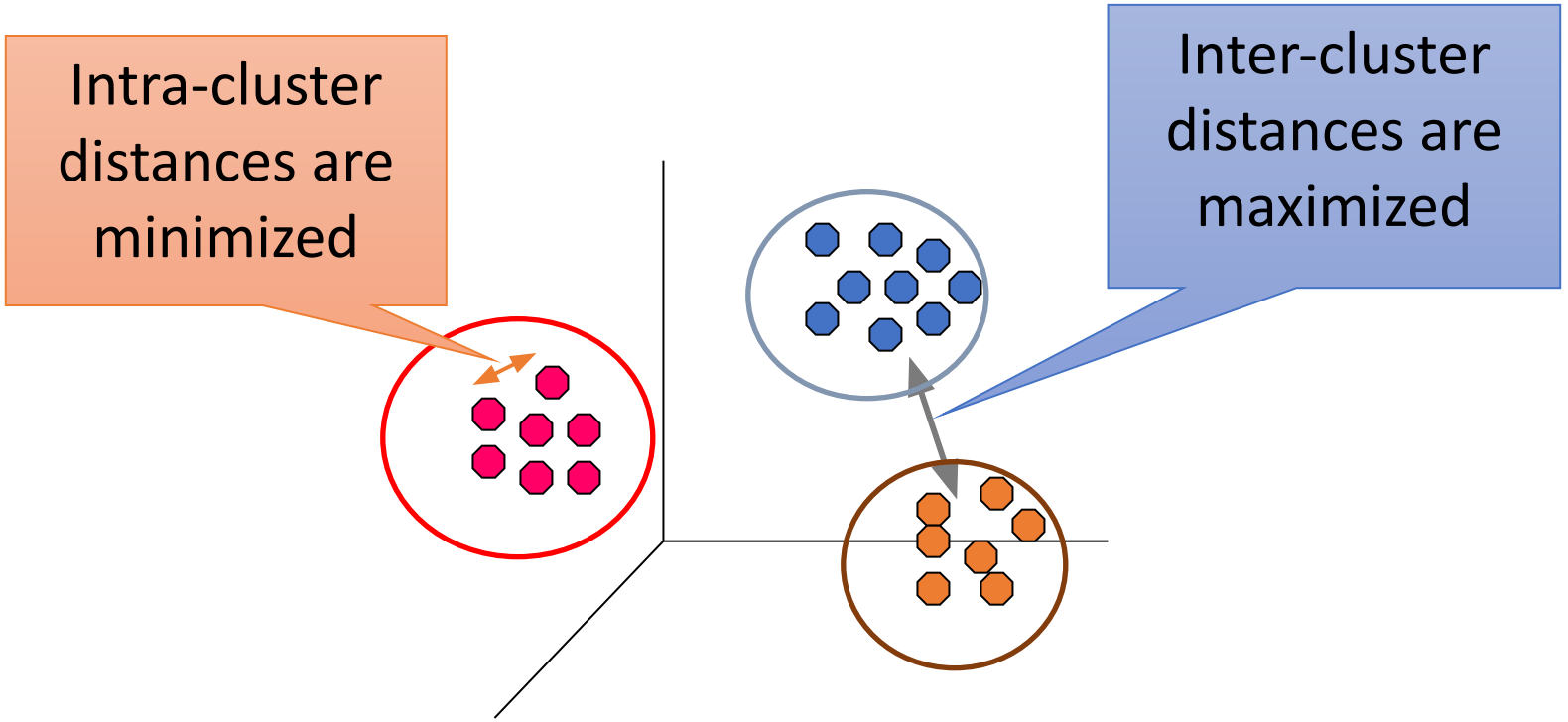
$$\text{Purity} = \frac{1}{n}\sum_{k} \max_{j} |C_k \cap P_j|$$

Homegeneity Score:

$$h = 1 - \frac{H(Y_{true}|Y_{predicted})}{H(Y_{true})}$$

Internal Evaluation

- Find groups of objects such that the objects in a group are similar and different from the objects in other groups



Cohesion and Separation

- WCSS/ SSE: Cohesion is measured by within cluster sum of squared distance

$$WCSS = \sum_{i} \sum_{x \in C_i} dist(x, \mu_i)^2$$

- Separation is measured by the between cluster sum of squares

$$BSS = \sum_{i} |C_i| \, dist(\mu, \mu_i)^2$$

$|C_i|$: size of cluster C_i μ: Centroid of whole data set

Davies-Bouldin Index

- Assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters

$$DB = \frac{1}{n}\sum_{i=1}^{k}\max_{j\neq i}\frac{\sigma_i + \sigma_j}{dist(\mu_i, \mu_j)}$$

Issues

- Representation & Similarity Measure
  - Representation of instances (features)
  - Proximity function (similarity / distance measure)
- Hard vs Soft clustering
  - Can an instance belong to more than one cluster?
- Clustering algorithm
  - Flat or Hierarchical
  - Density based
  - …
- Number of clusters
  - Fixed
  - Data driven