Foundations of Machine Learning

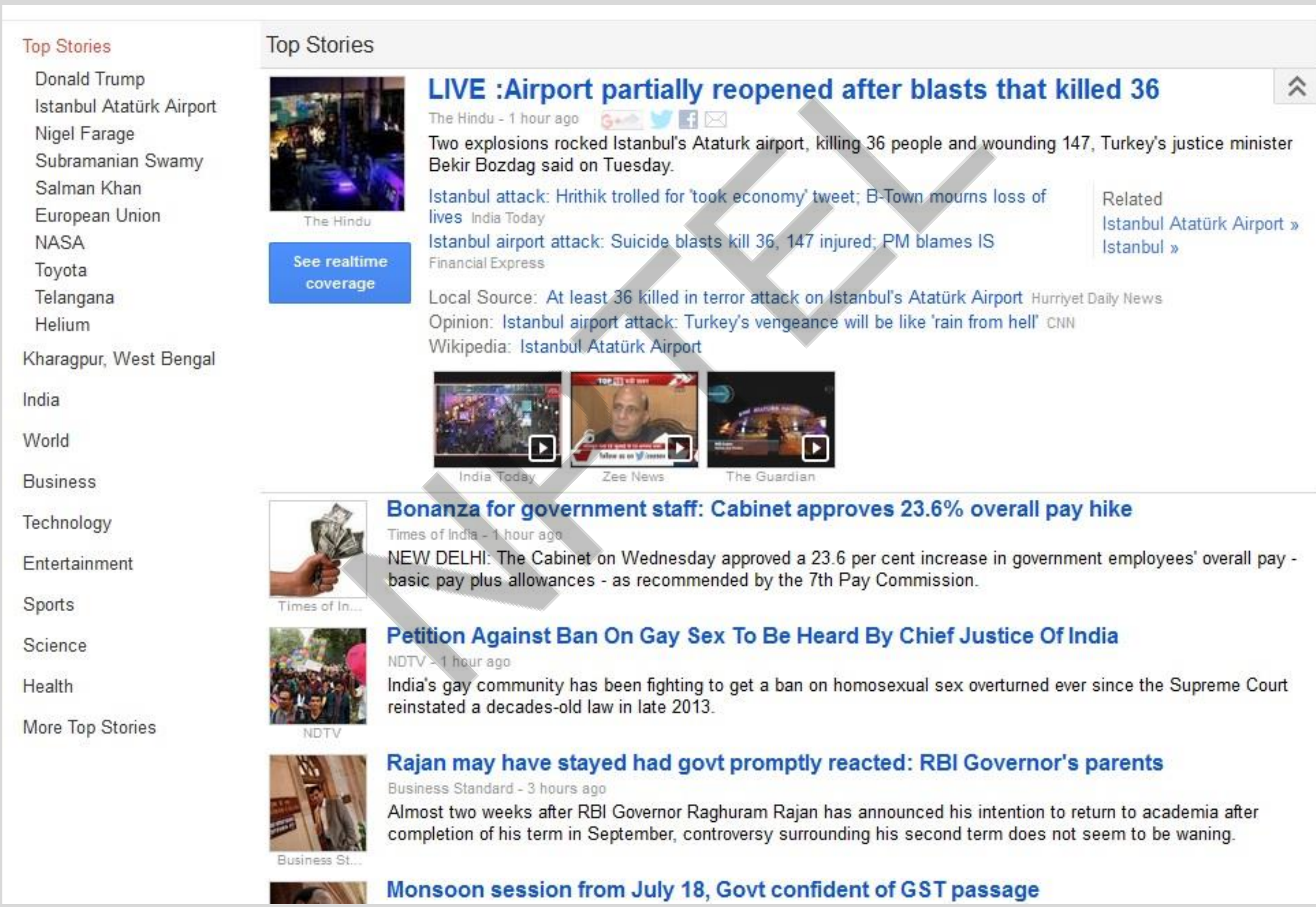
Module 9: Clustering
Part A: Introduction and kmeans

Sudeshna Sarkar
IIT Kharagpur

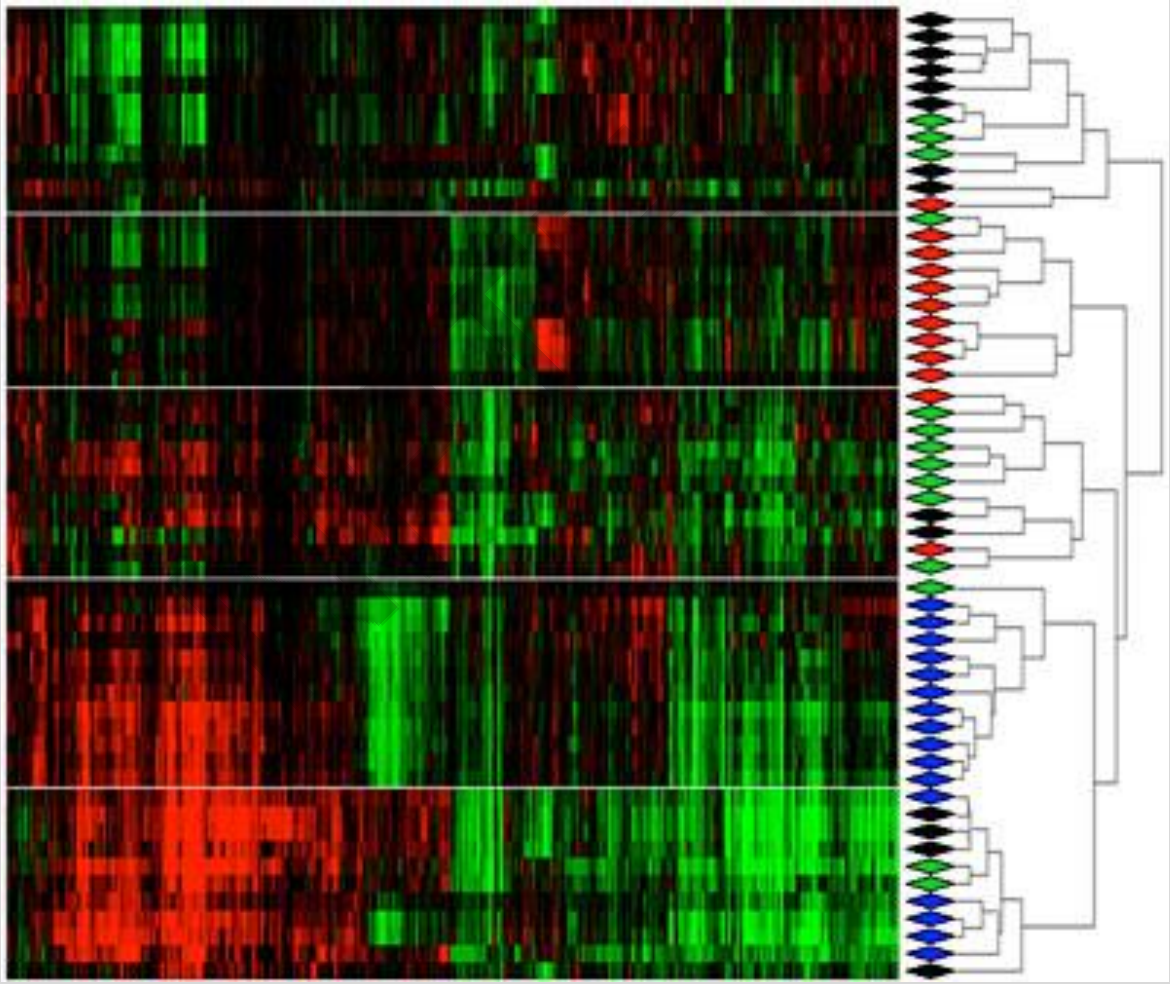
Unsupervised learning

- Unsupervised learning:
  - Data with no target attribute. Describe hidden structure from unlabeled data.
  - Explore the data to find some intrinsic structures in them.
- Clustering: the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other clusters.
- Useful for
  - Automatically organizing data.
  - Understanding hidden structure in data.
  - Preprocessing for further analysis.

Applications: News Clustering (Google)



Gene Expression Clustering

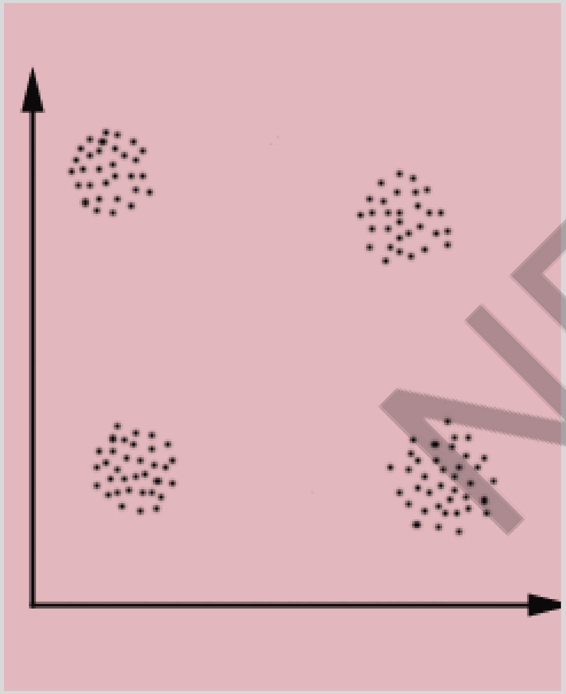


Other Applications

- Biology: classification of plants and animal kingdom given their features
- Marketing: Customer Segmentation based on a database of customer data containing their properties and past buying records
- Clustering weblog data to discover groups of similar access patterns.
- Recognize communities in social networks.

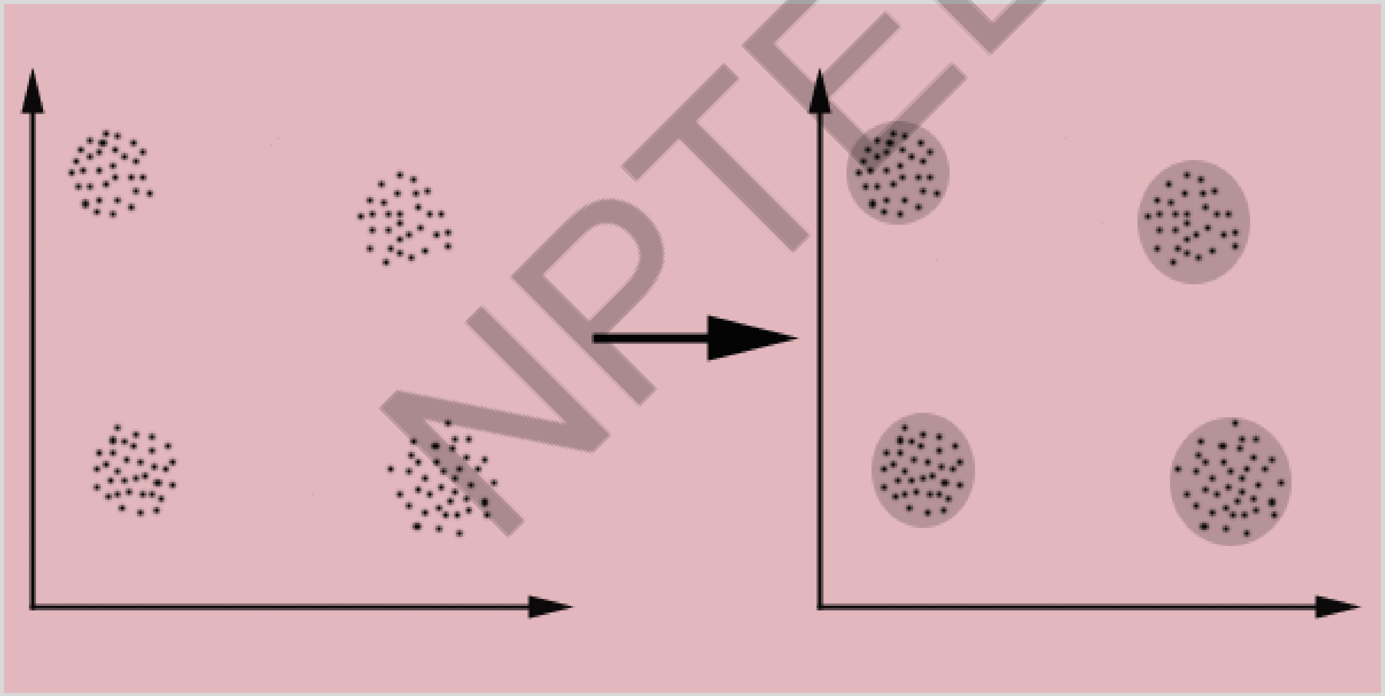
An illustration

- This data set has four natural clusters.



An illustration

- This data set has four natural clusters.



Aspects of clustering

- A clustering algorithm such as
  - Partitional clustering eg, kmeans
  - Hierarchical clustering eg, AHC
  - Mixture of Gaussians
- A distance or similarity function
  - such as Euclidean, Minkowski, cosine
- 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.

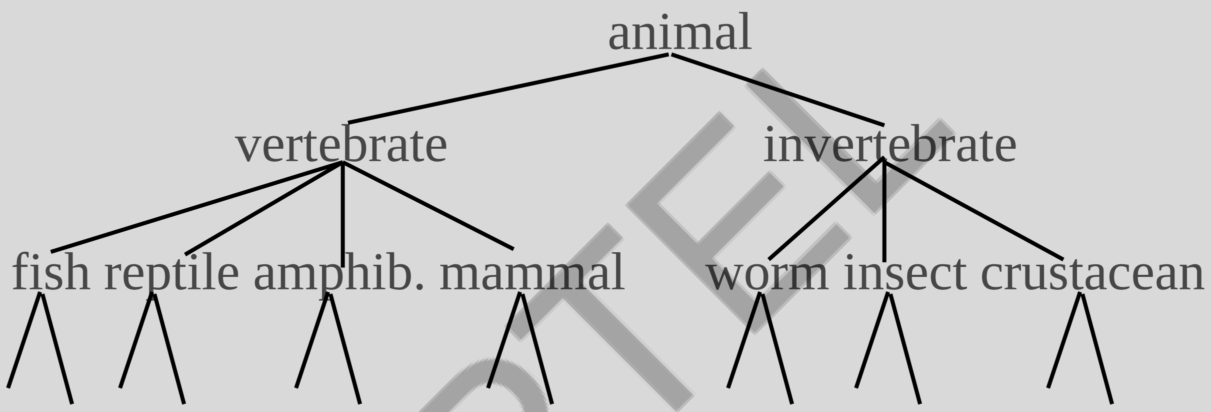
Major Clustering Approaches

- Partitioning: Construct various partitions and then evaluate them by some criterion
- Hierarchical: Create a hierarchical decomposition of the set of objects using some criterion
- Model-based: Hypothesize a model for each cluster and find best fit of models to data
- Density-based: Guided by connectivity and density functions
- Graph-Theoretic Clustering

Partitioning Algorithms

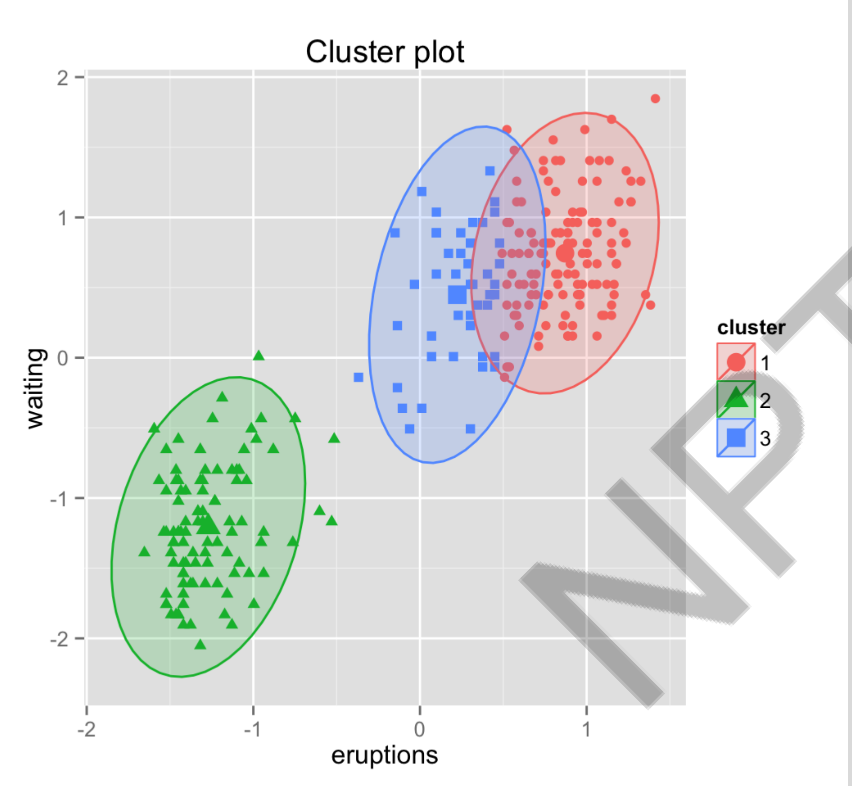
- Partitioning method: Construct a partition of a database *D* of *m* objects into a set of *k* clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic method: *k-means* (MacQueen, 1967)

Hierarchical Clustering



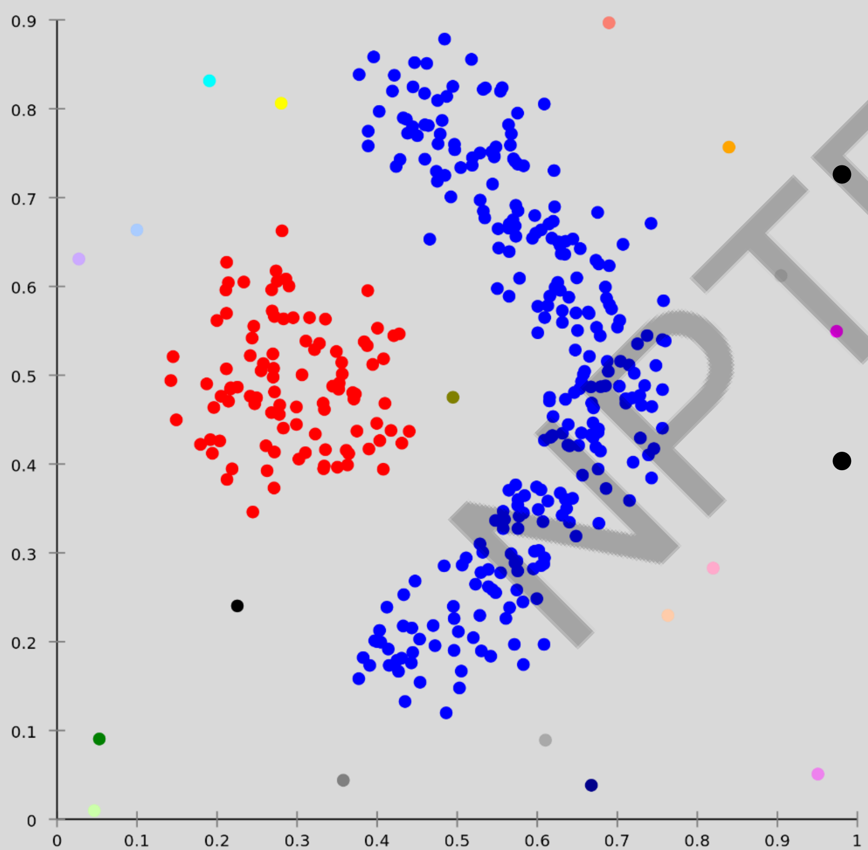
- Produce a nested sequence of clusters.
- One approach: recursive application of a partitional clustering algorithm.

Model Based Clustering



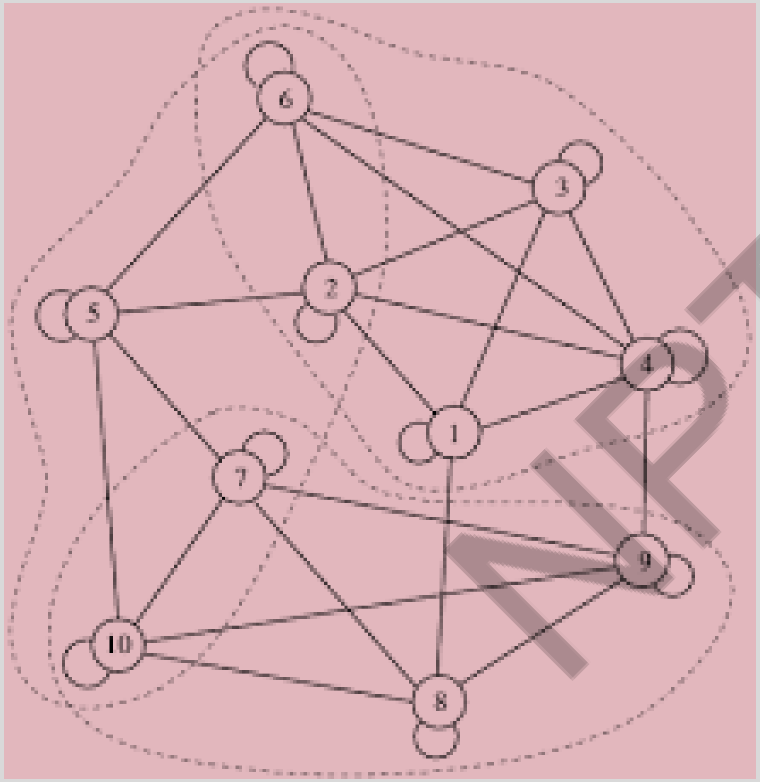
- A model is hypothesized
- e,g., Assume data is generated by a mixture of underlying probability distributions
- Fit the data to model

Density based Clustering



- Based on density connected points
- Locates regions of high density separated by regions of low density
- e.g., DBSCAN

Graph Theoretic Clustering



- Weights of edges between items (nodes) based on similarity
- E.g., look for minimum cut in a graph

(Dis)similarity measures

- Distance metric (scale-dependent)
  - Minkowski family of distance measures

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

Manhattan (p=1), Euclidean (p=2)

  - Cosine distance

$$\text{cosine}(x_i, x_j) = \frac{x_i . x_j}{\|x_i\| . \|x_i\|}$$

(Dis)similarity measures

- Correlation coefficients (scale-invariant)
- Mahalanobis distance

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

- Pearson correlation

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

Quality of Clustering

- Internal evaluation:
  - assign the best score to the algorithm that produces clusters with high similarity within a cluster and low similarity between clusters, e.g., Davies-Bouldin index

$$DB = \frac{1}{n} \sum_{i=1}^{k} \max_{j \neq i} \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

- External evaluation:
  - evaluated based on data such as known class labels and external benchmarks, eg, Rand Index, Jaccard Index, f-measure

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

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

Thank You

Foundations of Machine Learning

Module 9: Clustering

Part B: kmeans clustering

Sudeshna Sarkar

IIT Kharagpur

Partitioning Algorithms

- Given k
- Construct a partition of *m* objects $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_m}\}$

where $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \dots, x_{in})$ is a vector in a real-valued space $X \subseteq \mathbb{R}^n$, *n* is the number of attributes.

- into a set of *k* clusters $S = \{S_1, S_2, \dots, S_k\}$
- The cluster mean μ_i serves as a prototype of the cluster S_i.
- Find *k clusters* that optimizes a chosen criterion
  - E.g., the within-cluster sum of squares (WCSS) (sum of distance functions of each point in the cluster to the cluster mean)

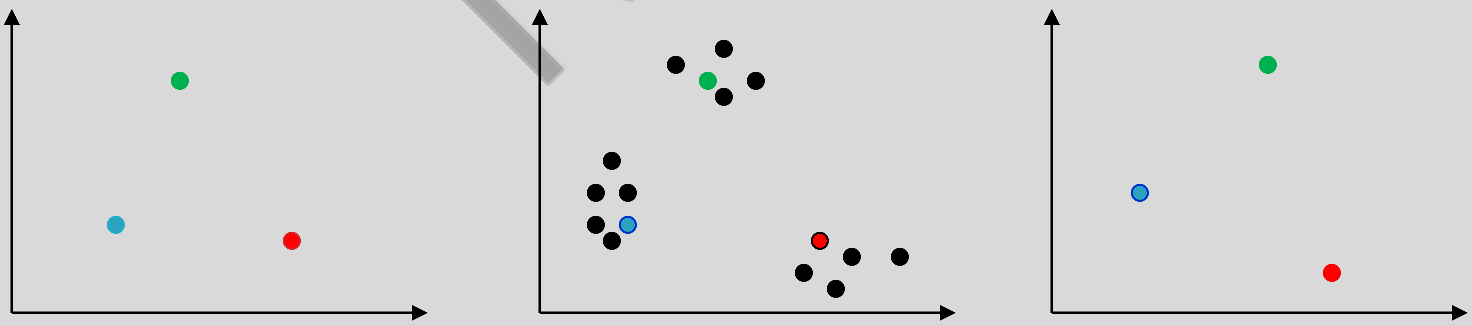
$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x_i - \mu_i\|^2$$

Heuristic method: k-means (MacQueen, 1967)

K-means algorithm

Given *k*

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



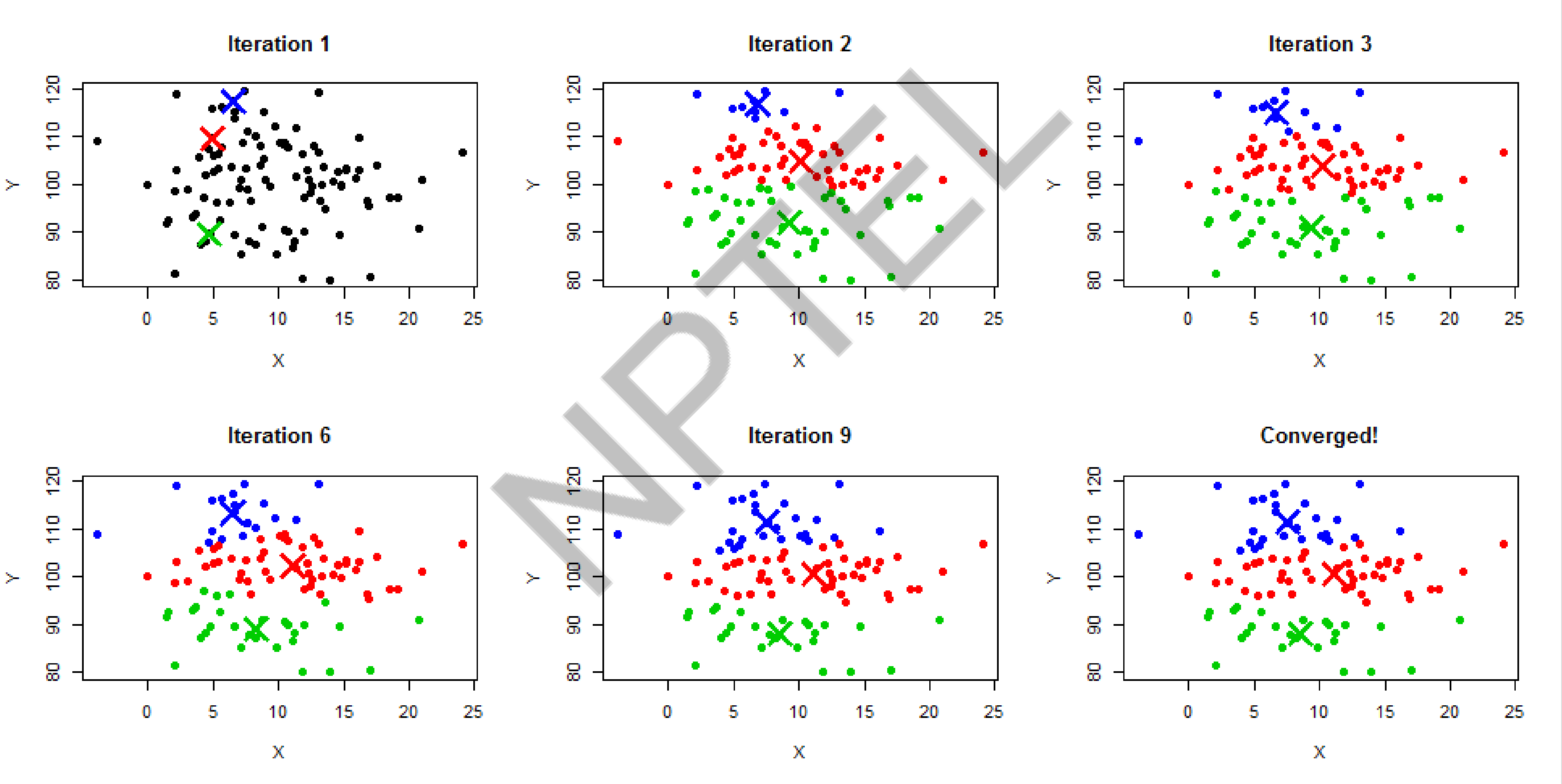
Stopping/convergence criterion

OR

1. no re-assignments of data points to different clusters
2. no (or minimum) change of centroids
3. minimum decrease in the *sum of squared error*

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

Kmeans illustrated



Similarity / Distance measures

- Distance metric (scale-dependent)
  - Minkowski family of distance measures

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

  Manhattan (p=1), Euclidean (p=2)

  - Cosine distance

Similarity / Distance measures

- Correlation coefficients (scale-invariant)
- Mahalanobis distance

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

- Pearson correlation

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

Convergence of K-Means

- Recomputation monotonically decreases each square error since
  (m_j is number of members in cluster j):
  $\sum (x_i - a)^2$ reaches minimum for:

$$\sum -2(x_i - a) = 0$$

$$\sum x_i = \sum a = m_j a$$

$$a = {}^{1}/_{m_j} \sum x_i = c_j$$

- K-means typically converges quickly

Time Complexity

- Computing distance between two items is O(n) where n is the dimensionality of the vectors.
- Reassigning clusters: O(km) distance computations, or O(kmn).
- Computing centroids: Each item gets added once to some centroid: O(mn).
- Assume these two steps are each done once for t iterations: O($tknm$).

Advantages

- Fast, robust easy to understand.
- Relatively efficient: O(tkmn)
- Gives best result when data set are distinct or well separated from each other.

Disadvantages

- Requires apriori specification of the number of cluster centers.
- Hard assignment of data points to clusters
- Euclidean distance measures can unequally weight underlying factors.
- Applicable only when mean is defined i.e. fails for categorical data.
- Only local optima

K-Means on RGB image

$x_1 = \{r_1, g_1, b_1\}$

$x_2 = \{r_2, g_2, b_2\}$

…

$x_i = \{r_i, g_i, b_i\}$

…

Classifier
(K-Means)

Classification Results

$x_1 \rightarrow C(x_1)$

$x_2 \rightarrow C(x_2)$

…

$x_i \rightarrow C(x_i)$

…

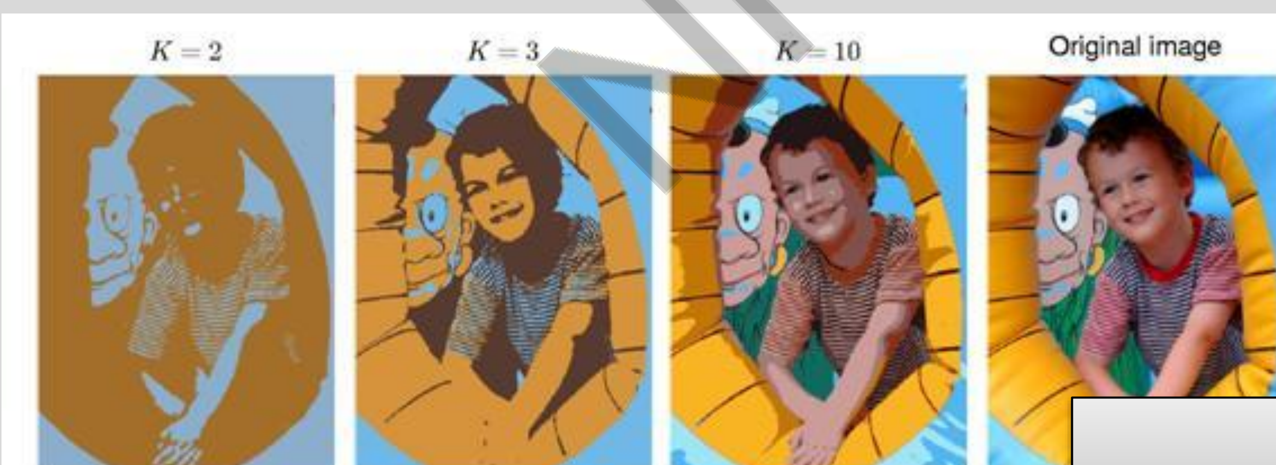
Cluster Parameters

θ_1 for C_1

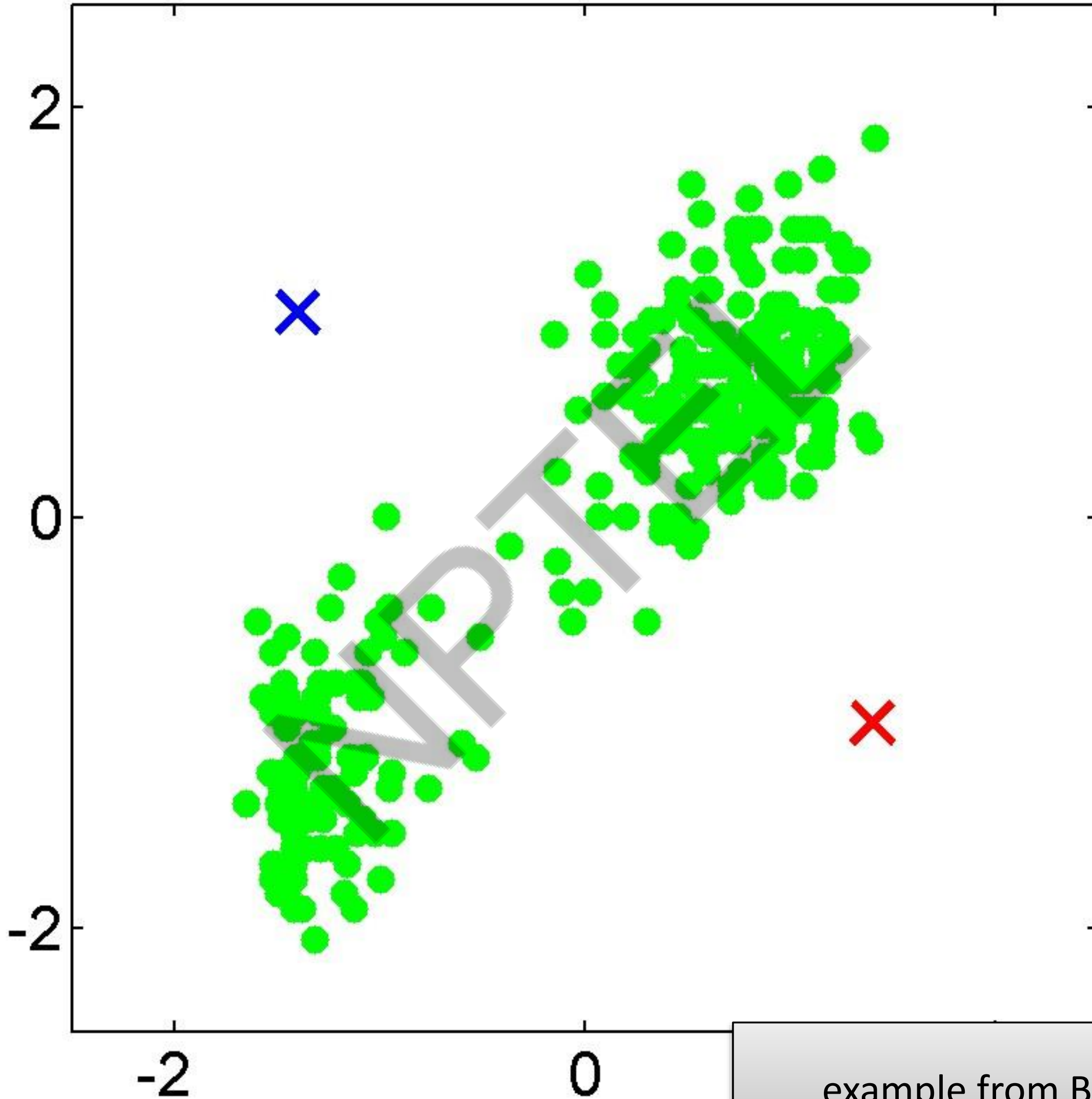
θ_2 for C_2

…

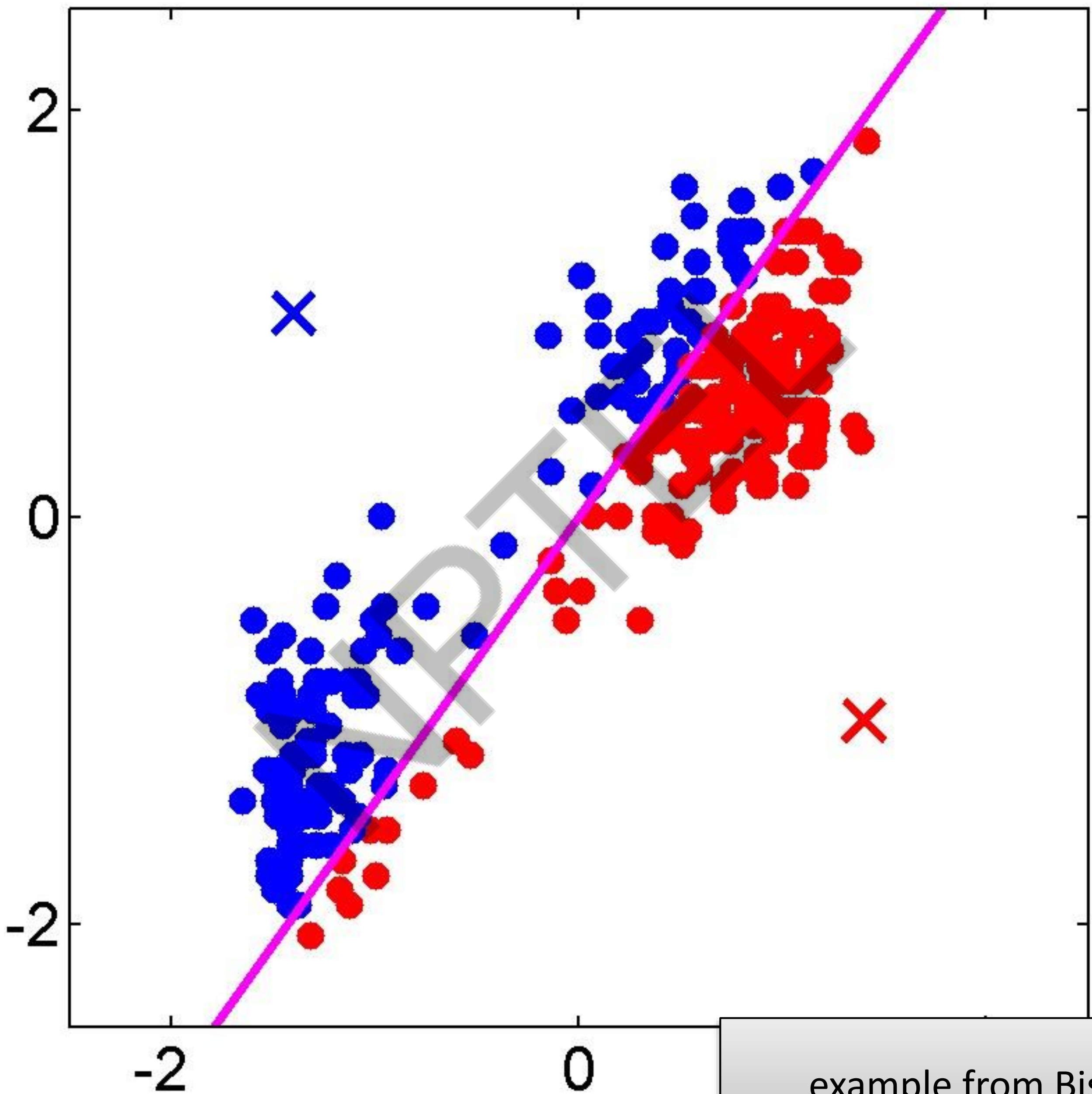
θ_k for C_k



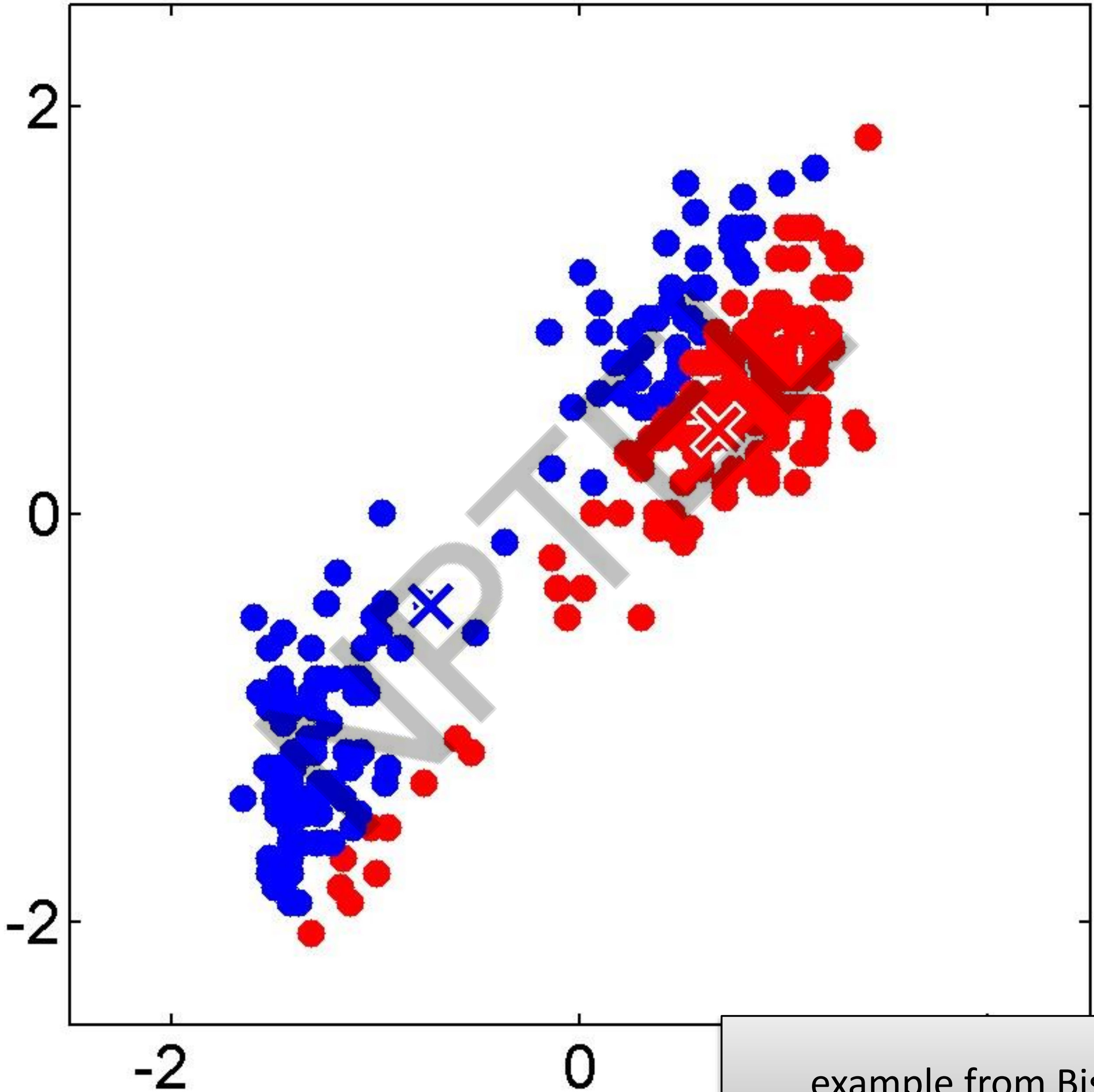
example from Bishop's Book



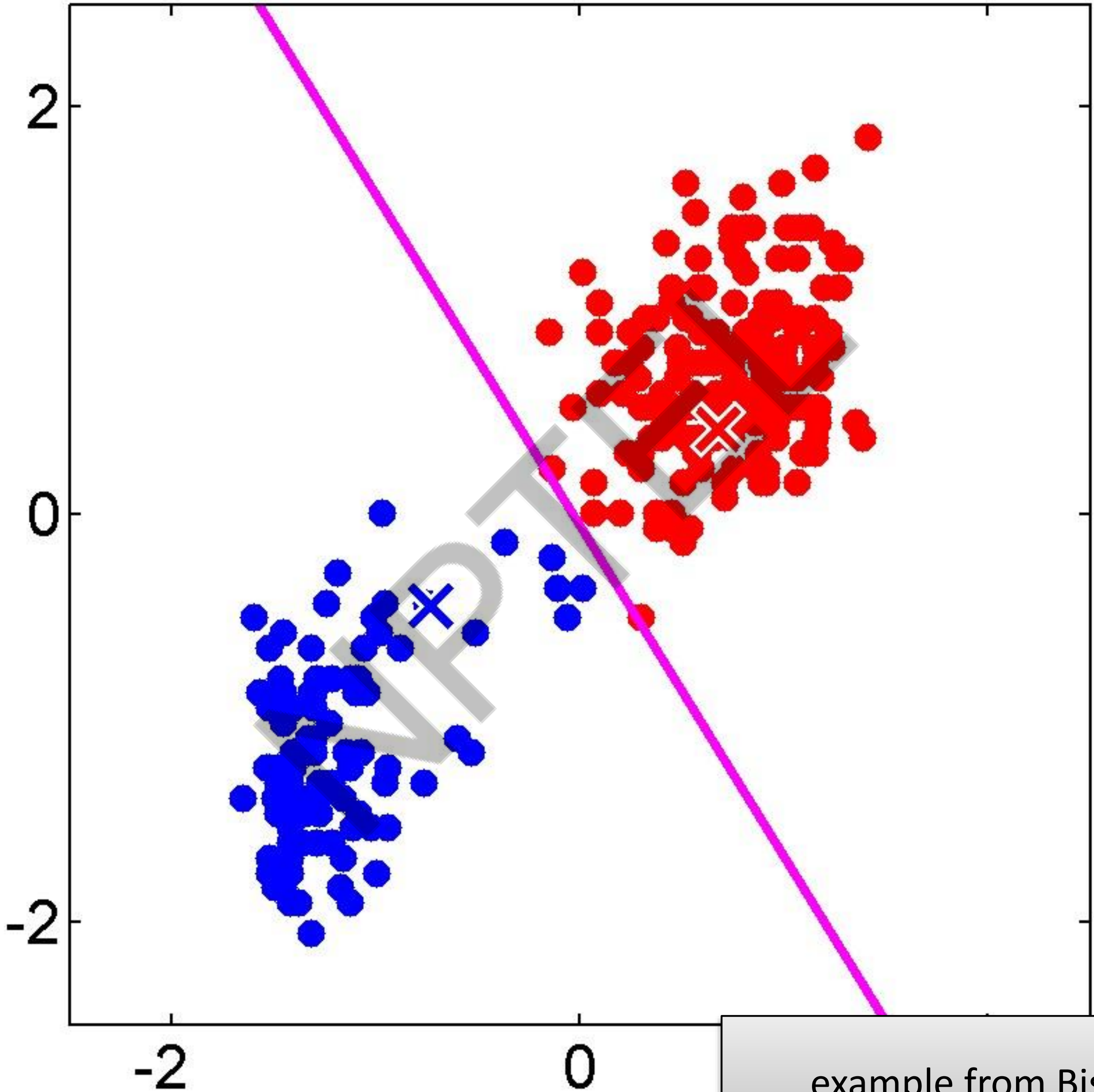
example from Bishop’s Book



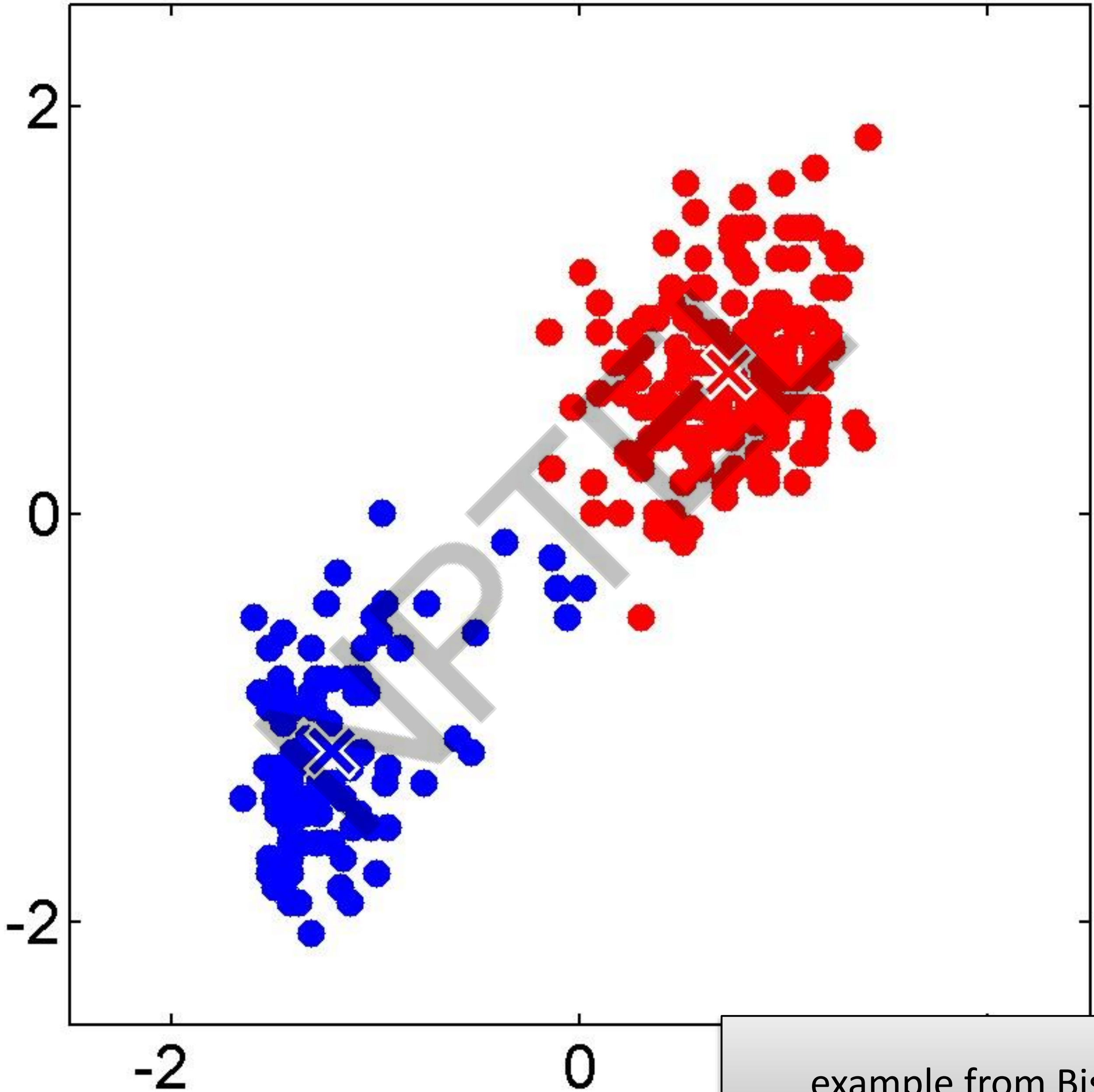
example from Bishop’s Book



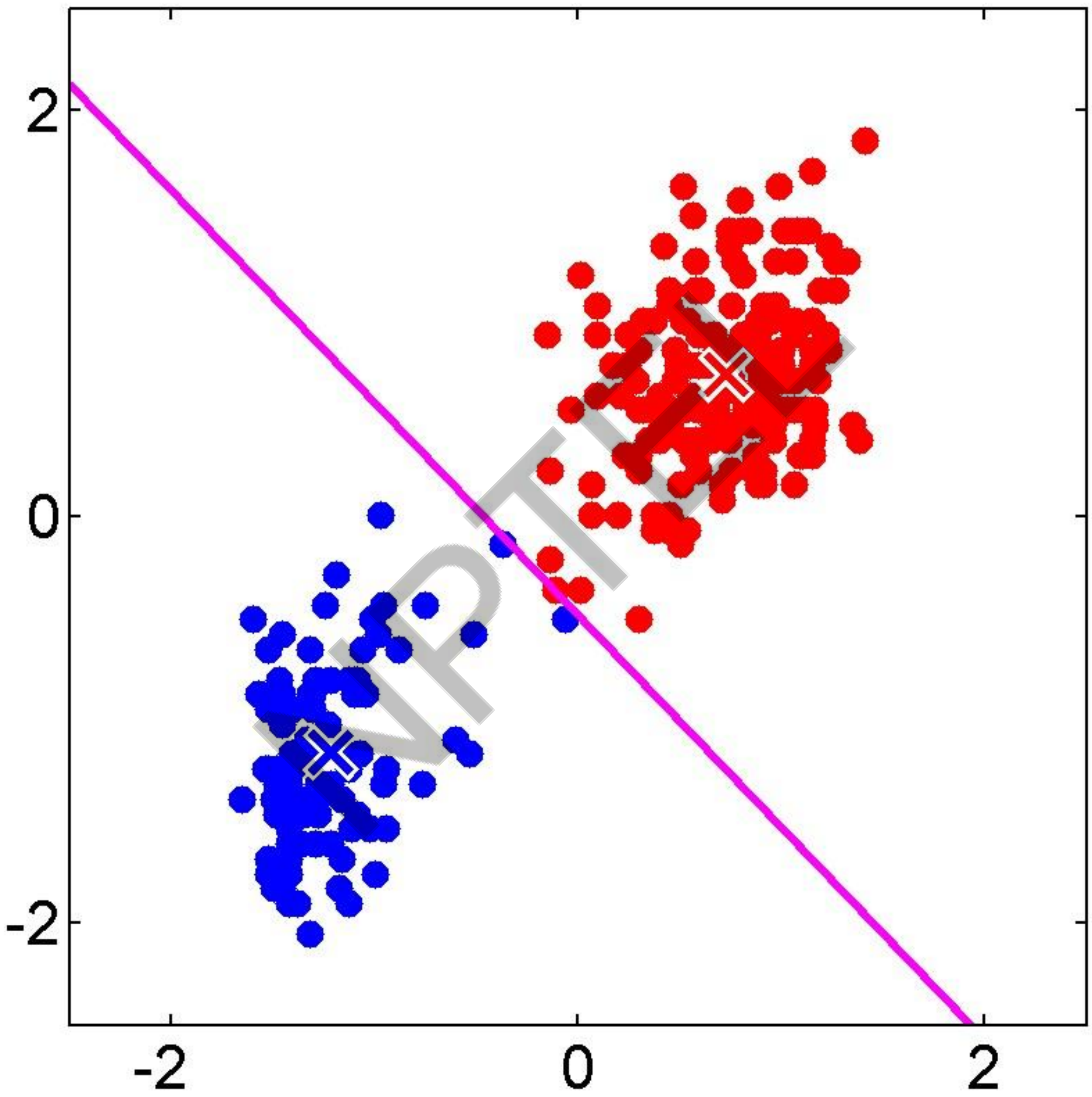
example from Bishop's Book

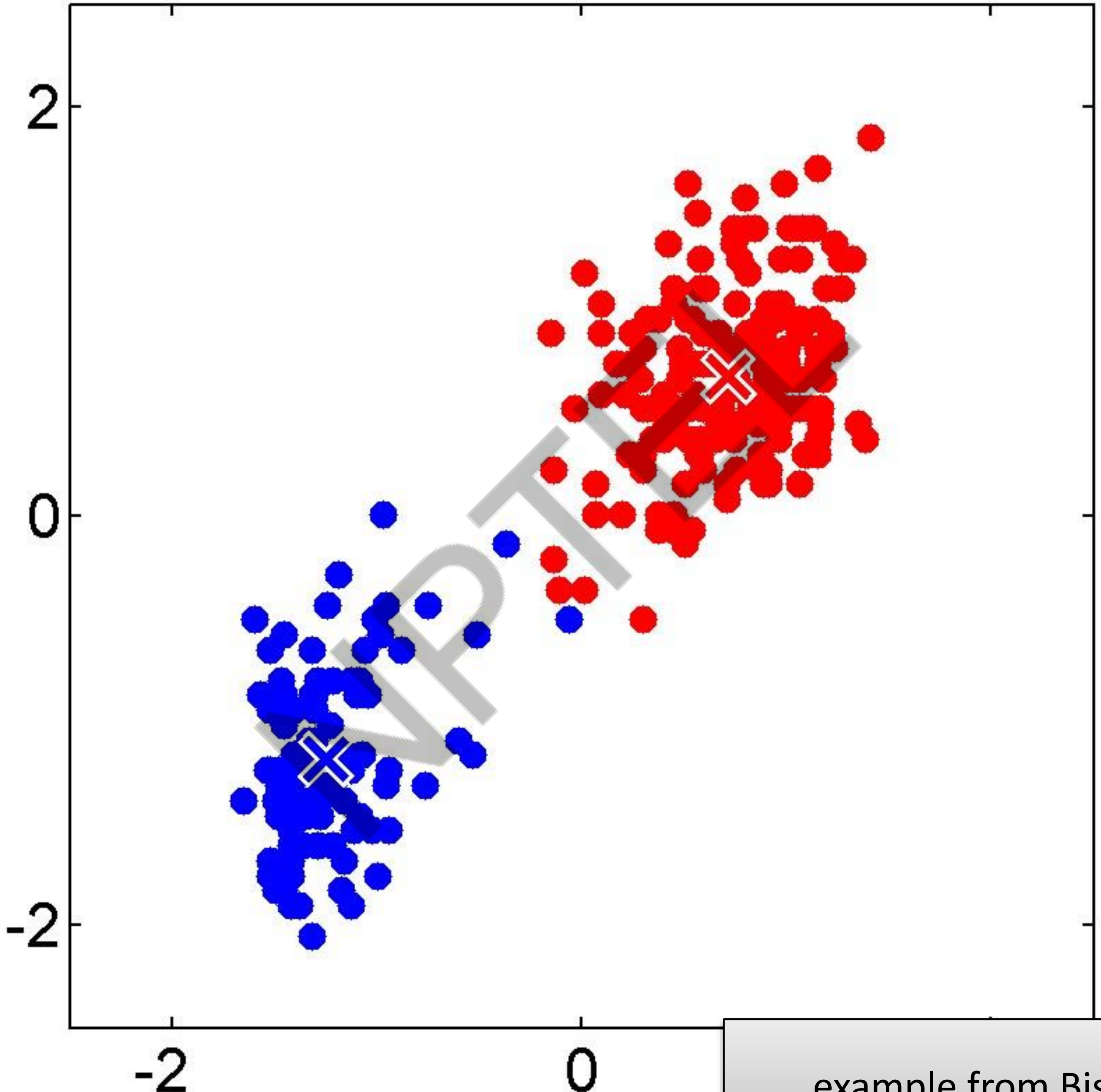


example from Bishop’s Book

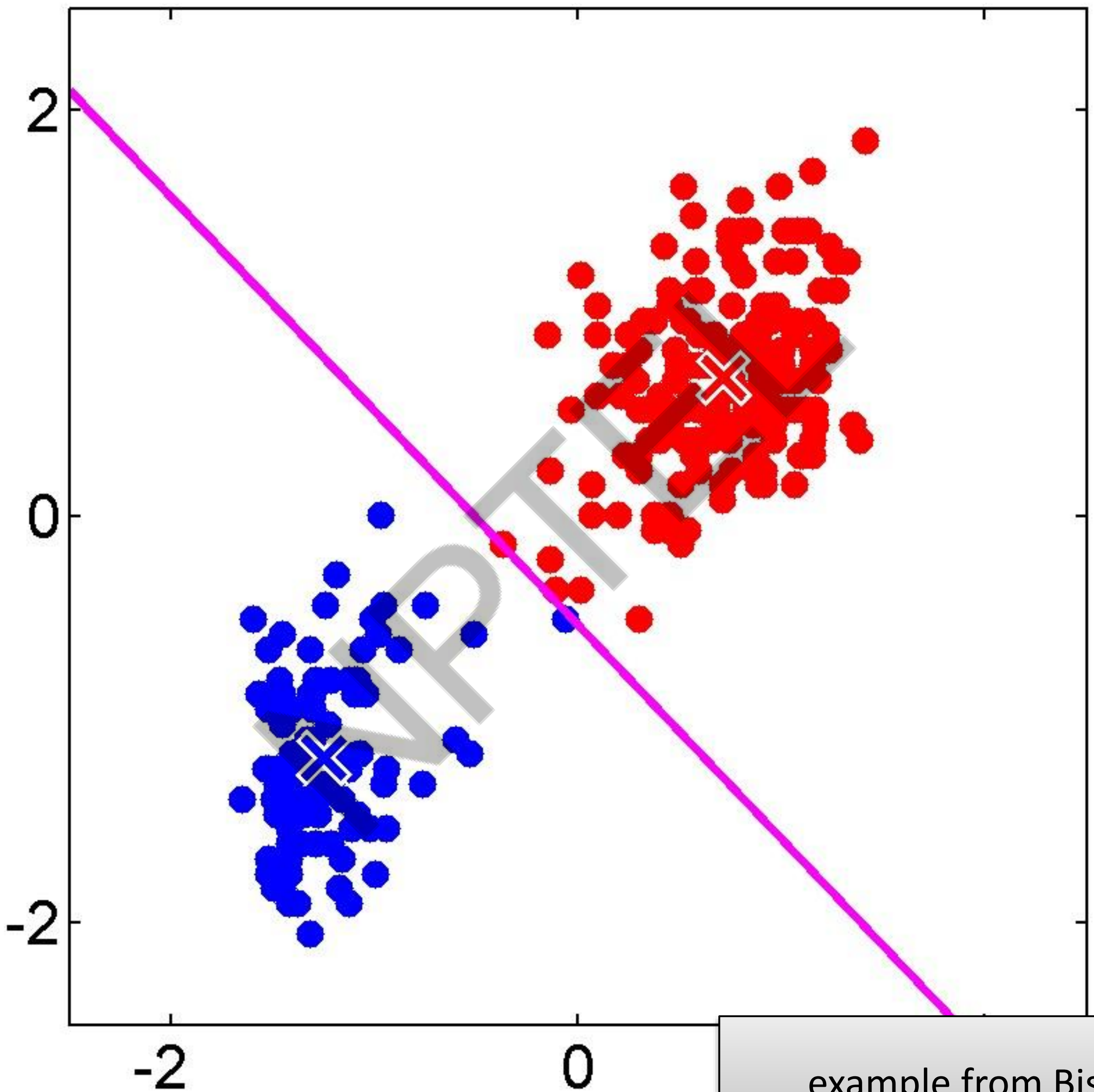


example from Bishop's Book

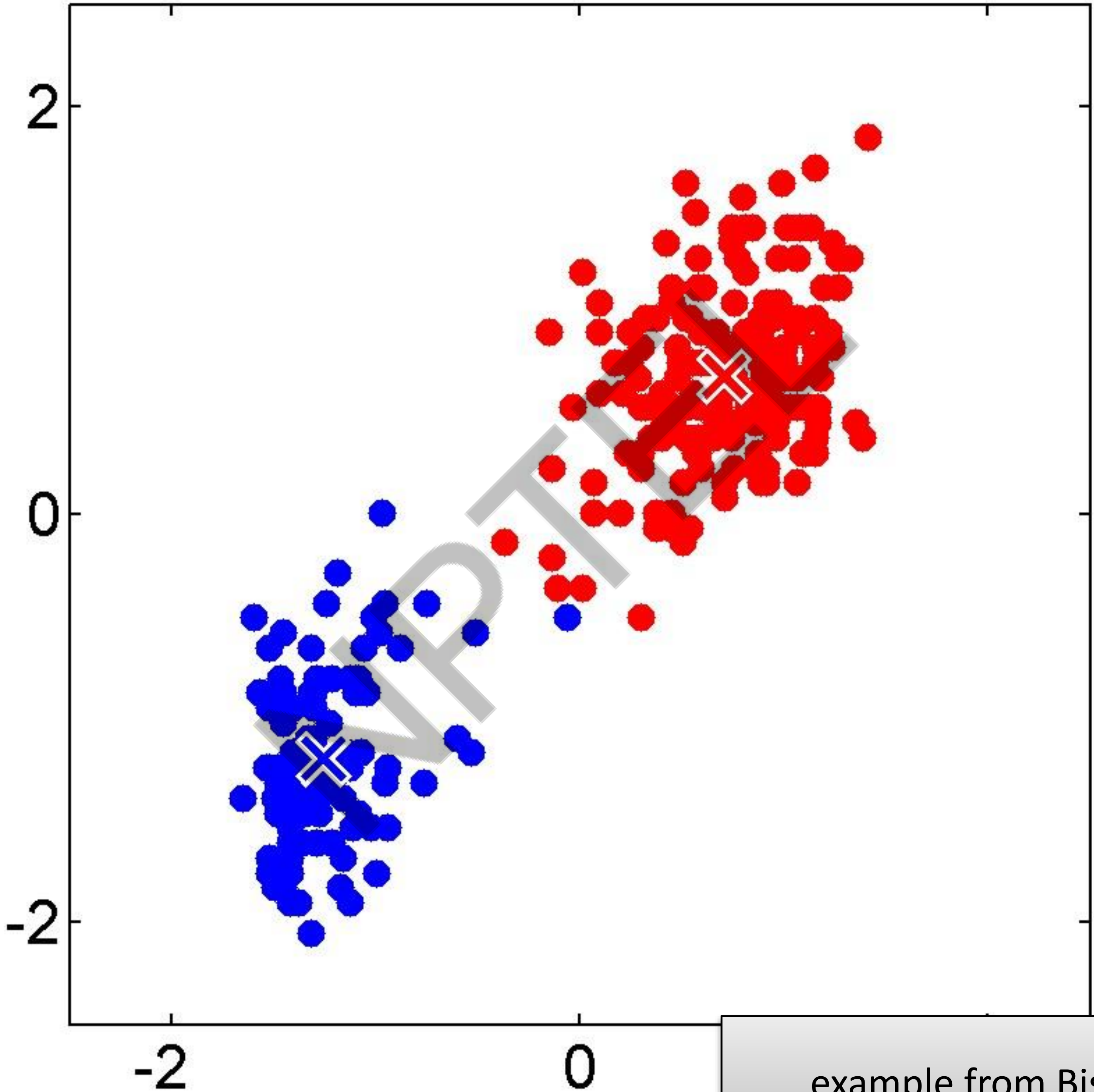

2
0
-2
-2
0
2



example from Bishop’s Book



example from Bishop’s Book



example from Bishop’s Book

Model-based clustering

- Assume k probability distributions with parameters $\theta_1, \theta_2, \ldots, \theta_k$
- Given data X, compute $\theta_1, \theta_2, \ldots, \theta_k$ such that

  $Pr(X|\theta_1, \theta_2, \ldots, \theta_k)$ [likelihood] or

  $\ln Pr(X|\theta_1, \theta_2, \ldots, \theta_k)$ [log likelihood]

is maximized.

- Every point $x \epsilon X$ may be generated by multiple distributions with some probability

EM Algorithm

- Initialize the parameters $\theta_1, \theta_2, \ldots, \theta_k$ randomly
- Let each parameter corresponds to a cluster center (mean)
- Iterate between two steps
  - **E**xpectation step: (probabilistically) assign points to clusters
  - **M**aximation step: estimate model parameters that maximize the likelihood for the given assignment of points

EM Algorithm

Expectation step: (probabilistically) assign points to clusters

compute Prob(point|mean)

Prob(mean|point) =
Prob(mean) Prob(point|mean) / Prob(point)

Maximation step: estimate model parameters that maximize the likelihood for the given assignment of points

Each mean = Weighted avg. of points

Weight = Prob(mean|point)

EM Algorithm

- Initialize k cluster centers
- Iterate between two steps
  - **E**xpectation step: assign points to clusters

$$\Pr(x_i \in C_k) = \frac{\Pr(x_i | C_k)}{\sum_j \Pr(x_i | C_j)}$$

$$w_k = \frac{\sum_i \Pr(x_i \in C_k)}{n}$$

  - **M**aximization step: estimate model parameters

$$r_k = \frac{1}{n} \sum_{i=1}^{n} \frac{\Pr(x_i \in C_k)}{\sum_j \Pr(x_i \in C_j)}$$

K-means Algorithm

- Goal: represent a data set in terms of K clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
  - E-step: assign each data point to nearest prototype
  - M-step: update prototypes to be the cluster means

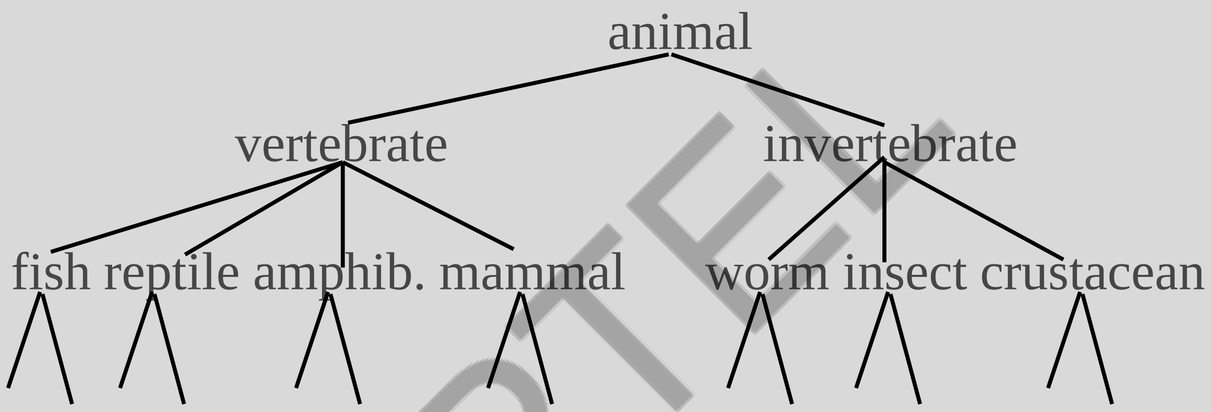
Thank You

Foundations of Machine Learning

Module 9: Clustering
Part C: Hierarchical Clustering

Sudeshna Sarkar
IIT Kharagpur

Hierarchical Clustering

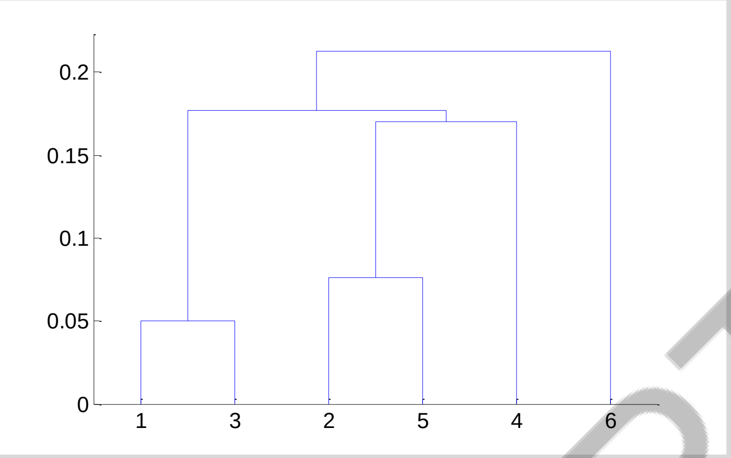


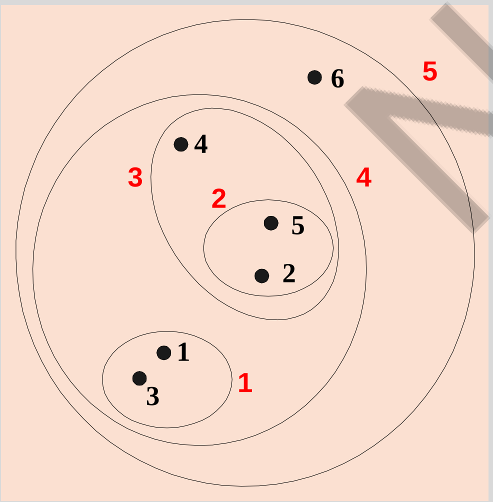
- Produce a nested sequence of clusters.
- One approach: recursive application of a partitional clustering algorithm.

Types of hierarchical clustering

- Agglomerative (bottom up) clustering: It builds the dendrogram (tree) from the bottom level, and
  - merges the most similar (or nearest) pair of clusters
  - stops when all the data points are merged into a single cluster (i.e., the root cluster).
- Divisive (top down) clustering: It starts with all data points in one cluster, the root.
  - Splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

Dendrogram: Hierarchical Clustering

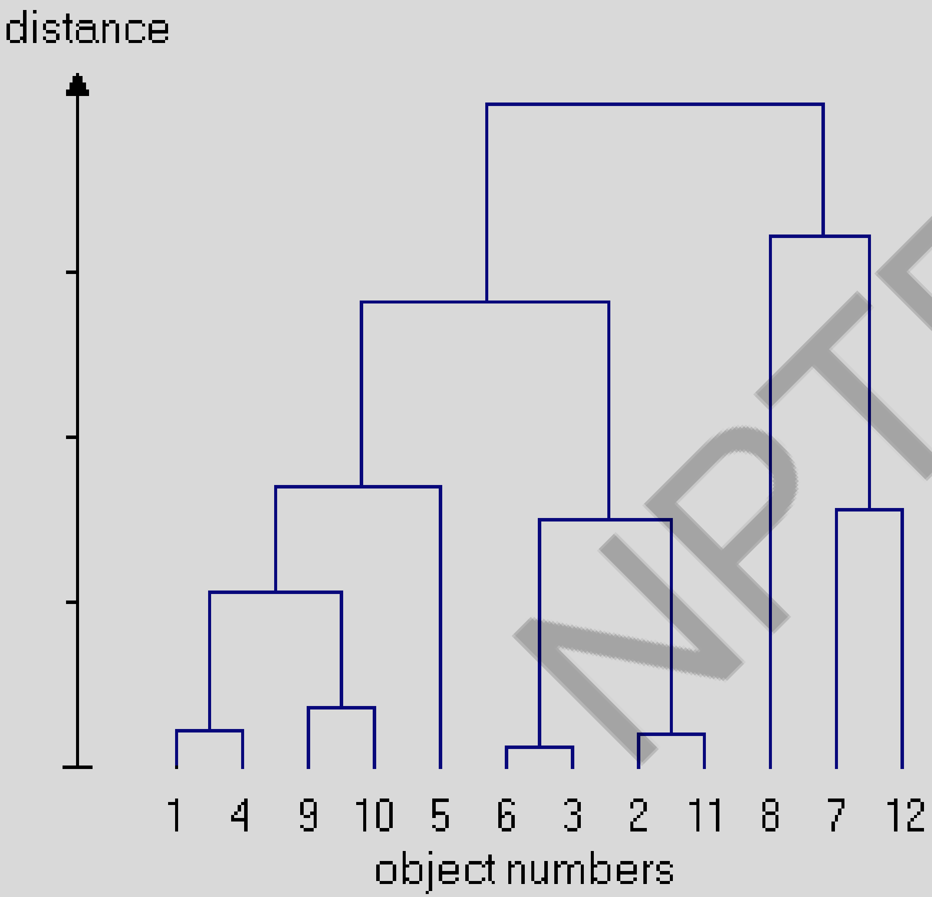




Dendrogram

- Given an input set S
- nodes represent subsets of S
- Features of the tree:
- The root is the whole input set S.
- The leaves are the individual elements of S.
- The internal nodes are defined as the union of their children.

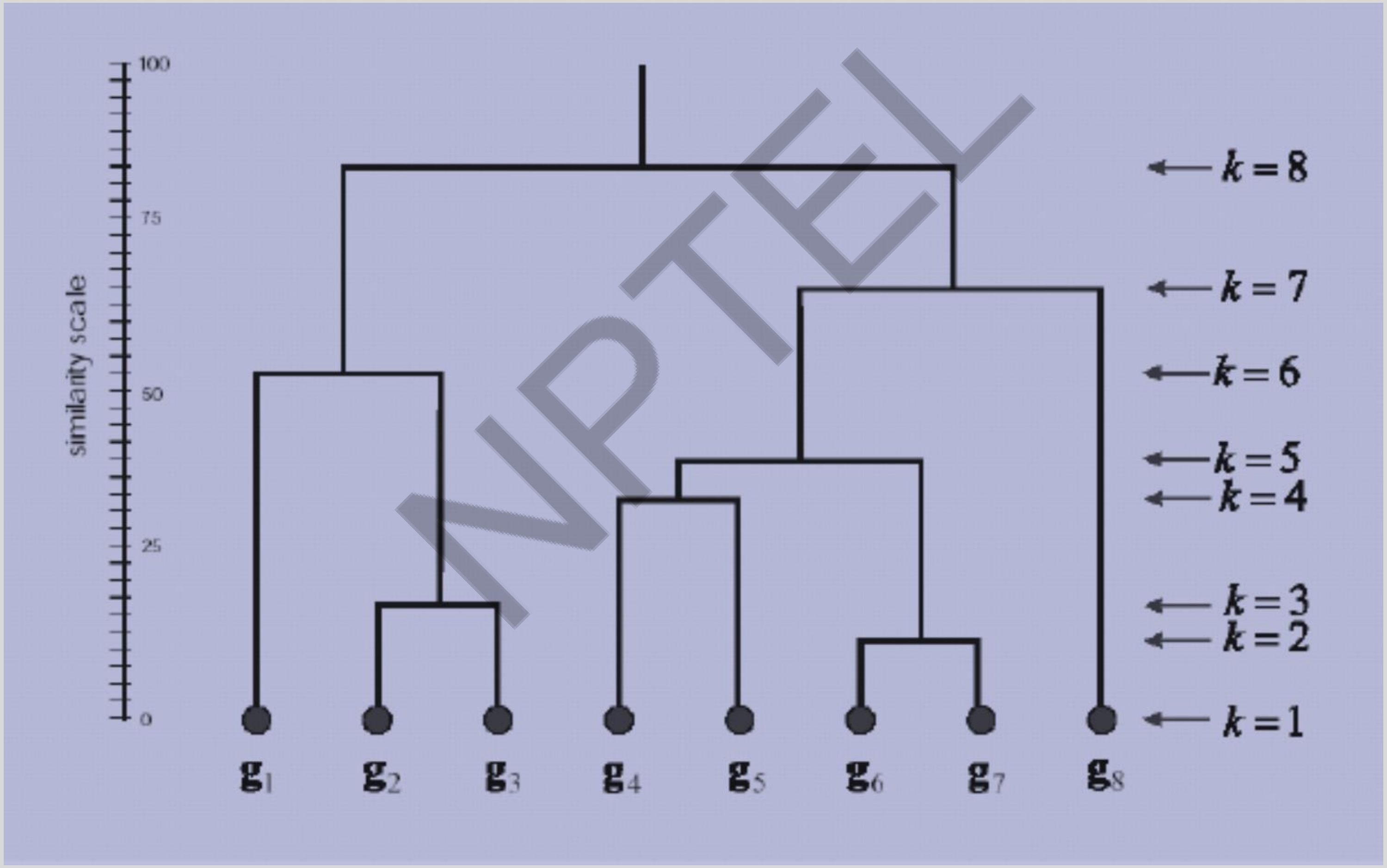
Dendrogram: Hierarchical Clustering



Dendrogram

- Each level of the tree represents a partition of the input data into several (nested) clusters or groups.
- May be cut at any level: Each connected component forms a cluster.

Hierarchical clustering



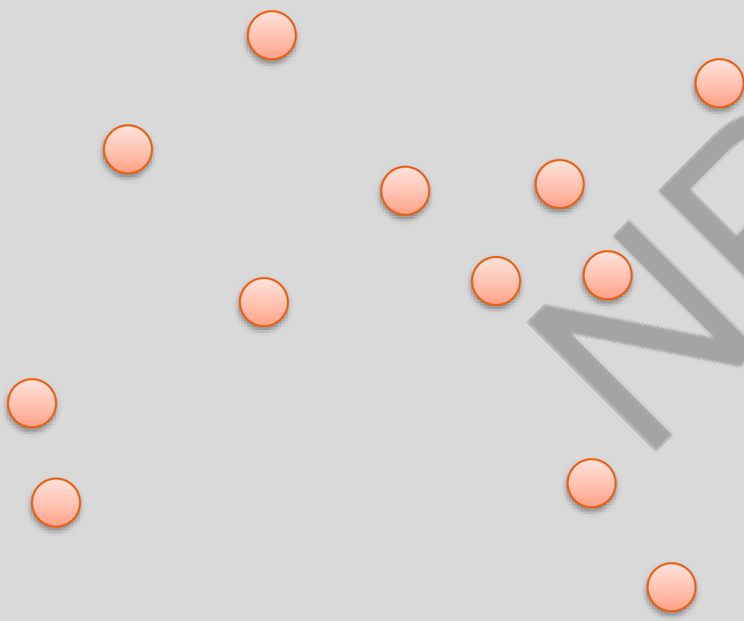
Hierrarchical Agglomerative clustering

- Initially each data point forms a cluster.
- Compute the distance matrix between the clusters.
- Repeat
  - Merge the two closest clusters
  - Update the distance matrix
- Until only a single cluster remains.

Different definitions of the distance leads to different algorithms.

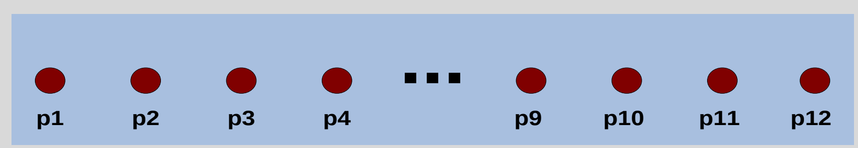
Initialization

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



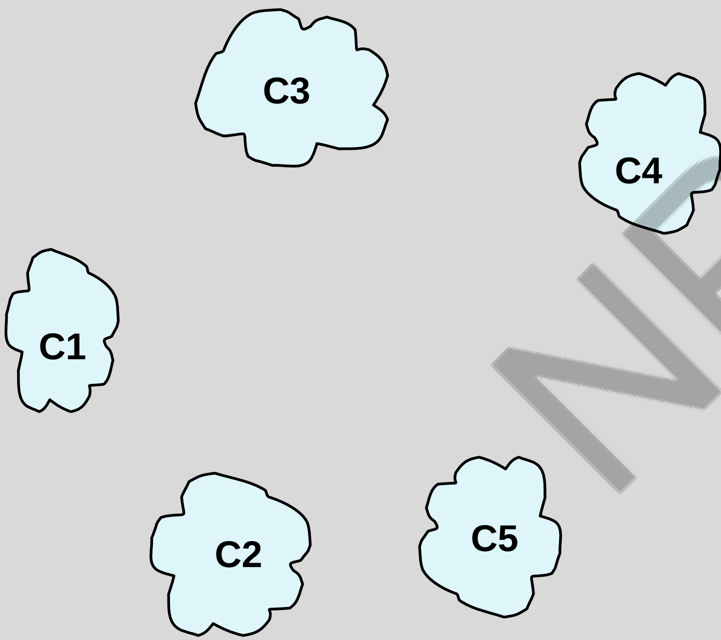
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Distance/Proximity Matrix



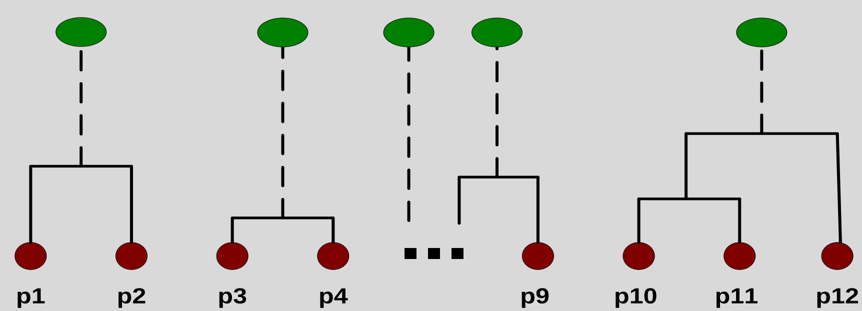
Intermediate State

- After some merging steps, we have some clusters



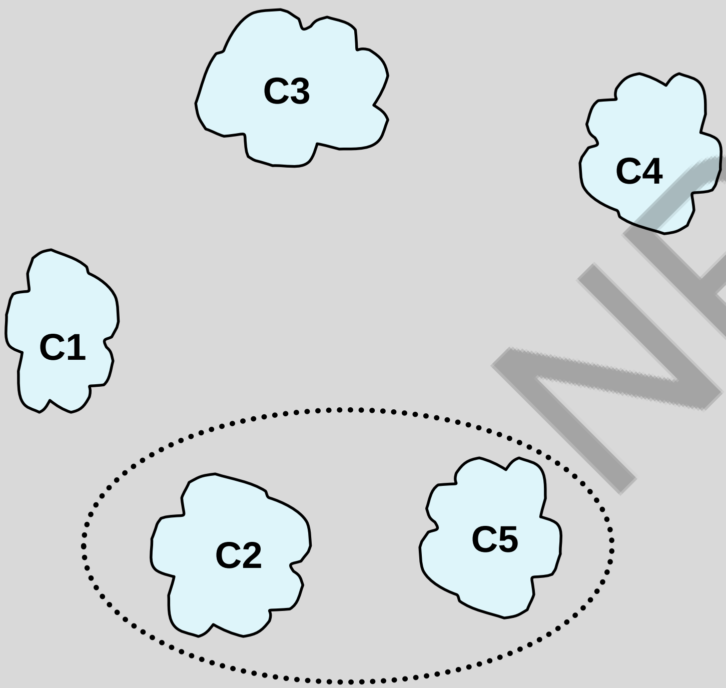
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

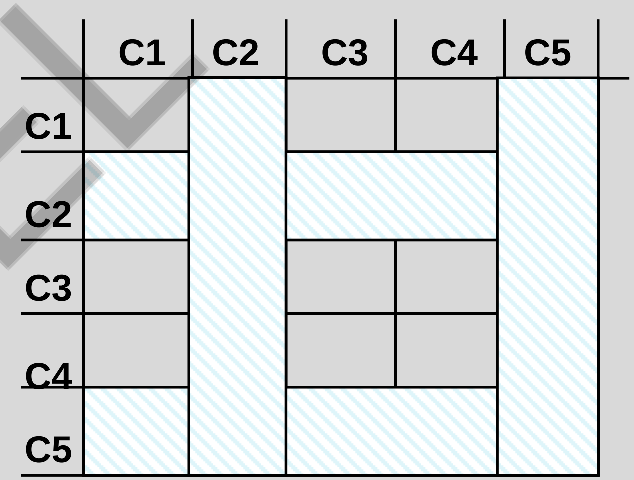
Distance/Proximity Matrix



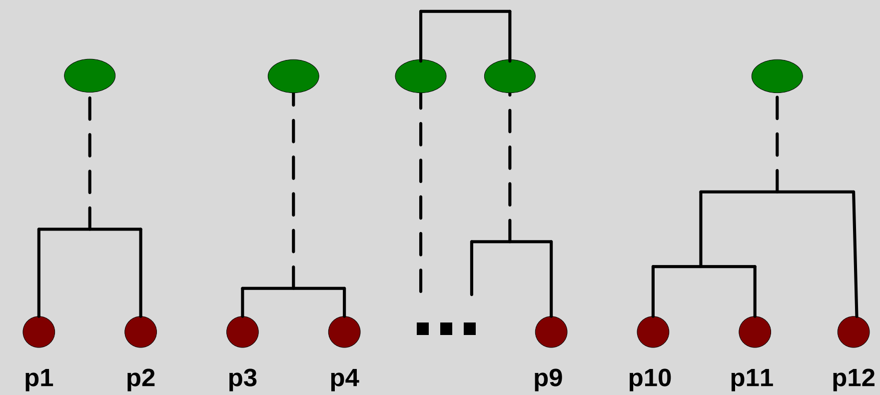
Intermediate State

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



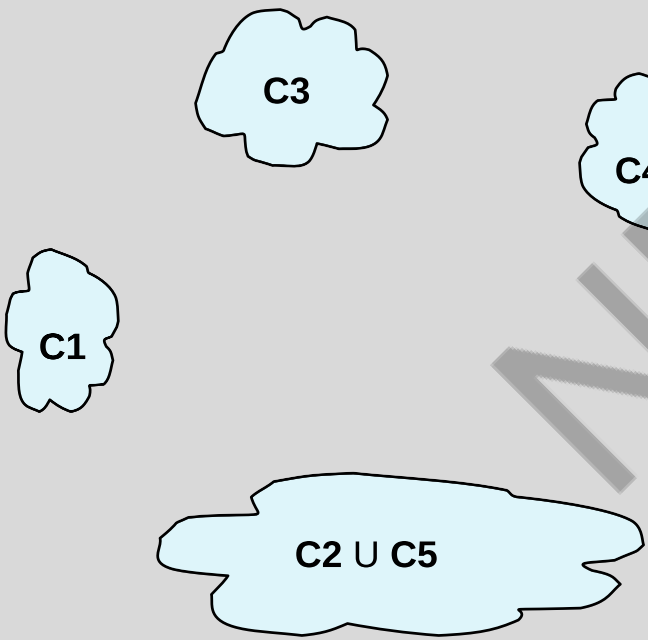


Distance/Proximity Matrix

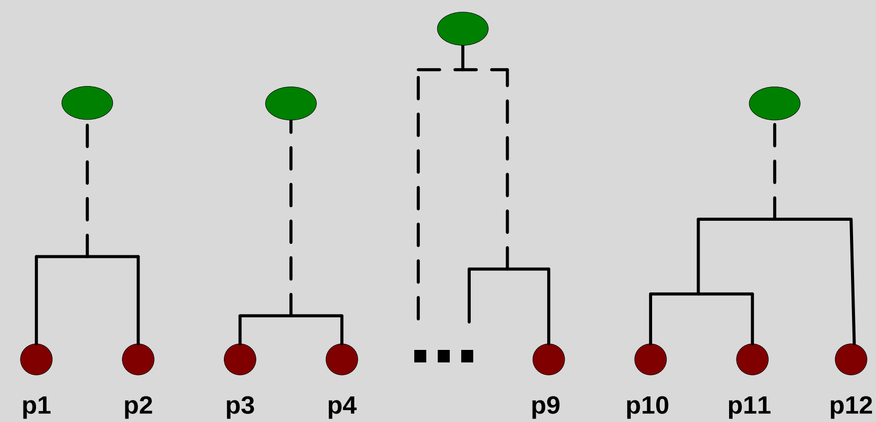


After Merging

- Update the distance matrix



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



Closest Pair

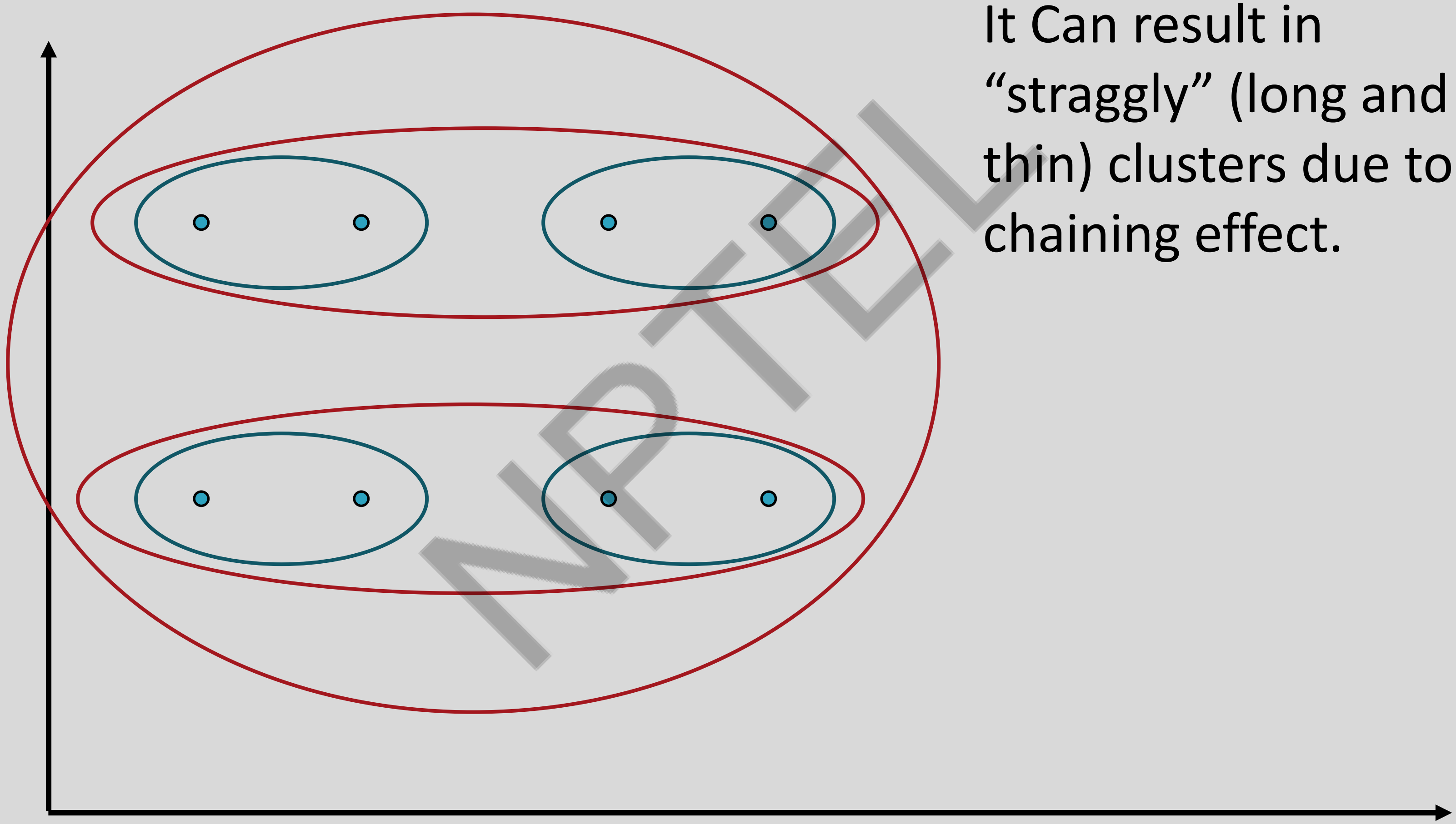
- A few ways to measure distances of two clusters.
- **Single-link**
  - Similarity of the *most* similar (single-link)
- **Complete-link**
  - Similarity of the *least* similar points
- **Centroid**
  - Clusters whose centroids (centers of gravity) are the most similar
- **Average-link**
  - Average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

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

Single Link Example

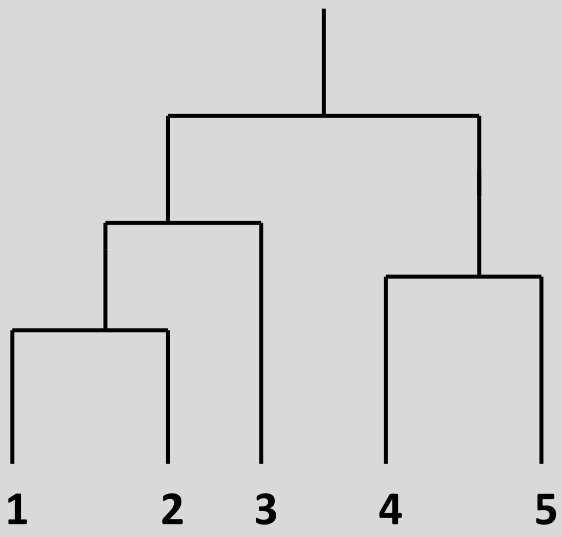


It Can result in "straggly" (long and thin) clusters due to chaining effect.

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.

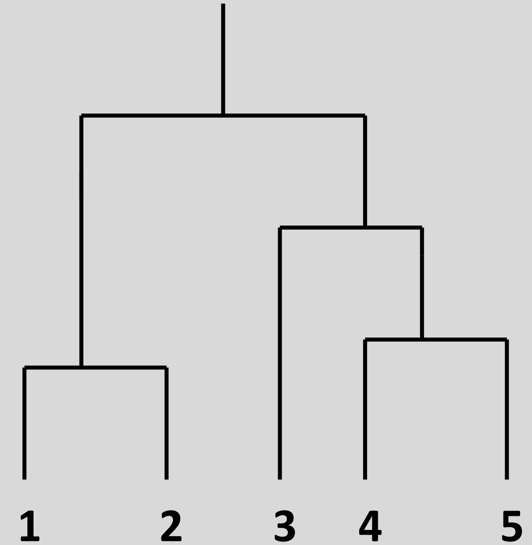
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

- Makes “tighter,” spherical clusters that are typically preferable.
- It is sensitive to outliers because they are far away

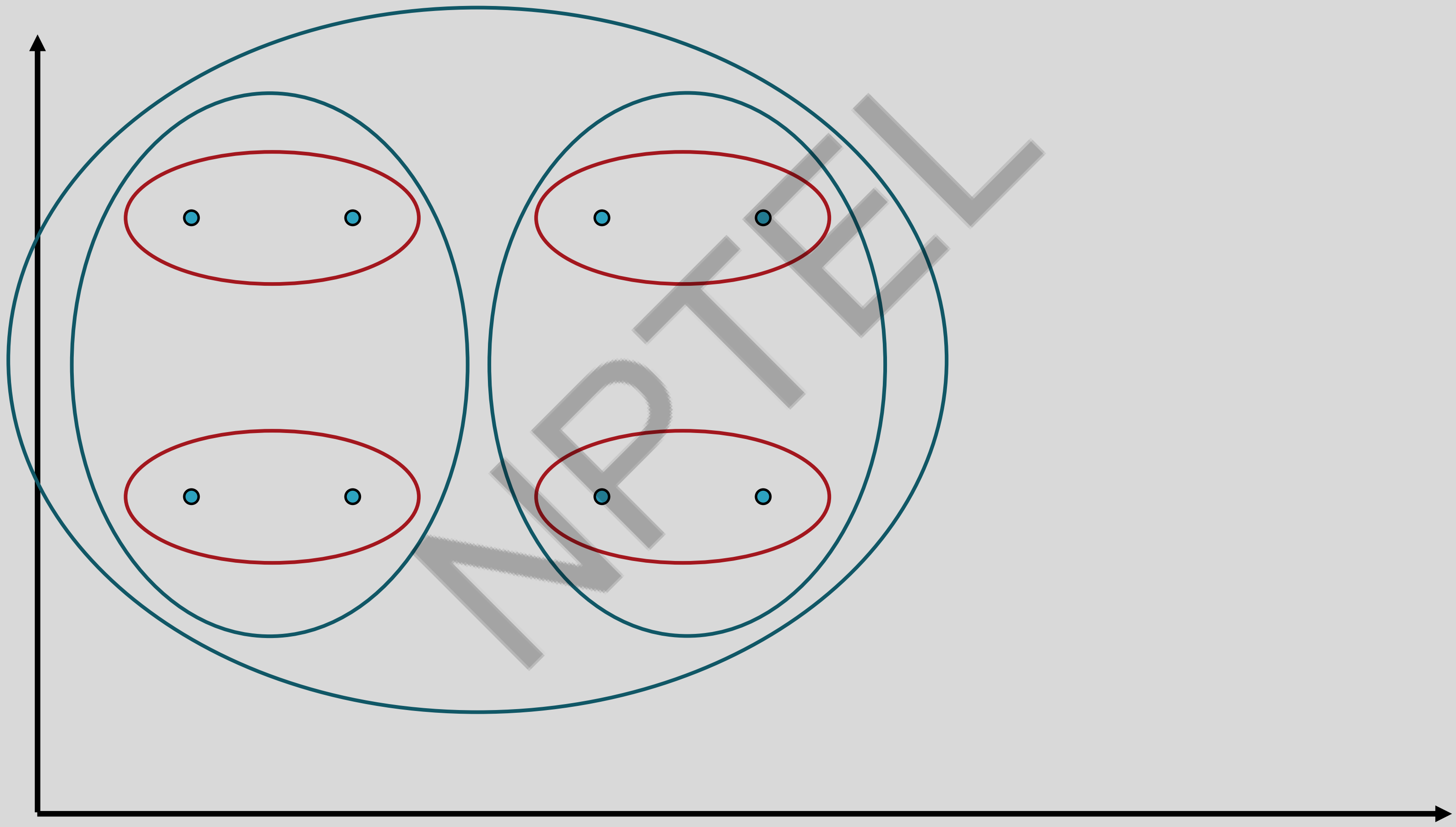
Complete-link clustering: example

- 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



Complete Link Example



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

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

- Compromise between single and complete link. Less susceptible to noise and outliers.
- Two options:
  - Averaged across all ordered pairs in the merged cluster
  - Averaged over all pairs *between* the two original clusters

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.

Model-based clustering

- Assume data generated from k probability distributions
- ***Goal:*** find the distribution parameters
- ***Algorithm:*** Expectation Maximization (EM)
- ***Output:*** Distribution parameters and a soft assignment of points to clusters

Model-based clustering

- Assume k probability distributions with parameters $\theta_1, \theta_2, \dots, \theta_k$
- Given data X, compute $\theta_1, \theta_2, \dots, \theta_k$ such that

  $Pr(X|\theta_1, \theta_2, \dots, \theta_k)$ [likelihood] or

  $\ln Pr(X|\theta_1, \theta_2, \dots, \theta_k)$ [log likelihood]

is maximized.

- Every point $x \epsilon X$ may be generated by multiple distributions with some probability

EM Algorithm

- Initialize the parameters $\theta_1, \theta_2, \dots, \theta_k$ randomly
- Let each parameter corresponds to a cluster center (mean)
- Iterate between two steps
  - **E**xpectation step: (probabilistically) assign points to clusters
  - **M**aximation step: estimate model parameters that maximize the likelihood for the given assignment of points

EM Algorithm

Expectation step: (probabilistically) assign points to clusters

compute Prob(point|mean)

Prob(mean|point) =
Prob(mean) Prob(point|mean) / Prob(point)

Maximation step: estimate model parameters that maximize the likelihood for the given assignment of points

Each mean = Weighted avg. of points

Weight = Prob(mean|point)

EM Algorithm

- Initialize k cluster centers
- Iterate between two steps
  - **E**xpectation step: assign points to clusters

$$\Pr(x_i \in C_k) = \frac{\Pr(x_i|C_k)}{\sum_j \Pr(x_i|C_j)}$$

$$w_k = \frac{\sum_i \Pr(x_i \in C_k)}{n}$$

  - **M**aximization step: estimate model parameters

$$r_k = \frac{1}{n}\sum_{i=1}^{n} \frac{\Pr(x_i \in C_k)}{\sum_j \Pr(x_i \in C_j)}$$

K-means Algorithm

- Goal: represent a data set in terms of K clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
  - E-step: assign each data point to nearest prototype
  - M-step: update prototypes to be the cluster means

Thank You