CLUSTERING

Business Intelligence
Dr Sara Migliorini

CLUSTERING

- Cluster analysis divides data into groups (clusters) that are meaningful, useful, or both.
- Clustering for understanding
  - Clusters are potential classes and cluster analysis is the study of techniques for automatically finding classes.
    - Information retrieval: a query for “movies” in the WWW can return thousands of pages, they can be grouped into categories such as reviews, trailers, etc.
- Clustering for utility
  - Cluster analysis provides an abstraction (cluster prototype).
    - Summarization, compression

APPLICATIONS OF CLUSTER ANALYSIS

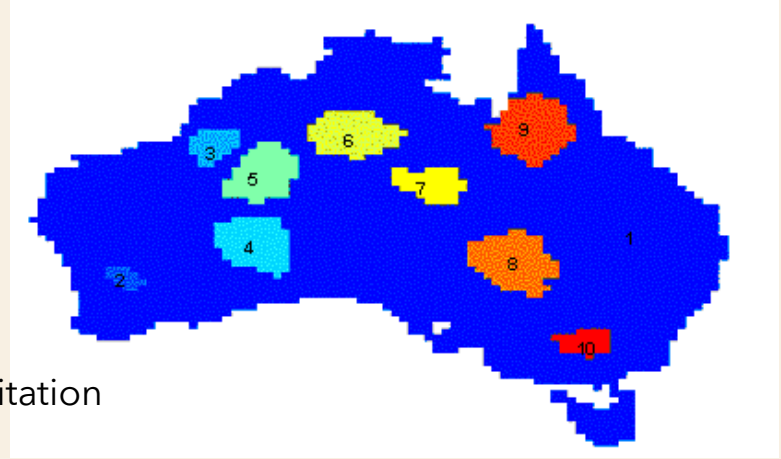
Understanding

Group related documents for browsing, group genes and proteins that have similar functionality, or group stocks with similar price fluctuations

	Discovered Clusters	*Industry Group*
1	Applied-Matl-DOWN,Bay-Network-Down,3-COM-DOWN, Cabletron-Sys-DOWN,CISCO-DOWN,HP-DOWN, DSC-Comm-DOWN,INTEL-DOWN,LSI-Logic-DOWN, Micron-Tech-DOWN,Texas-Inst-Down,Tellabs-Inc-Down, Natl-Semiconduct-DOWN,Oracl-DOWN,SGI-DOWN, Sun-DOWN	Technology1-DOWN
2	Apple-Comp-DOWN,Autodesk-DOWN,DEC-DOWN, ADV-Micro-Device-DOWN,Andrew-Corp-DOWN, Computer-Assoc-DOWN,Circuit-City-DOWN, Compaq-DOWN, EMC-Corp-DOWN, Gen-Inst-DOWN, Motorola-DOWN,Microsoft-DOWN,Scientific-Atl-DOWN	Technology2-DOWN
3	Fannie-Mae-DOWN,Fed-Home-Loan-DOWN, MBNA-Corp-DOWN,Morgan-Stanley-DOWN	Financial-DOWN
4	Baker-Hughes-UP,Dresser-Inds-UP,Halliburton-HLD-UP, Louisiana-Land-UP,Phillips-Petro-UP,Unocal-UP, Schlumberger-UP	Oil-UP

Summarization

Reduce the size of large data sets

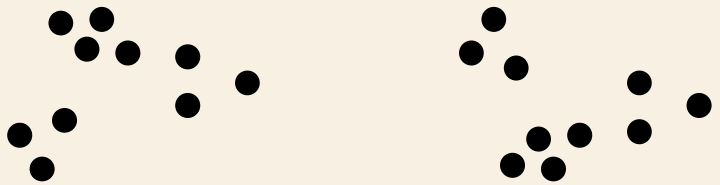


Clustering precipitation in Australia

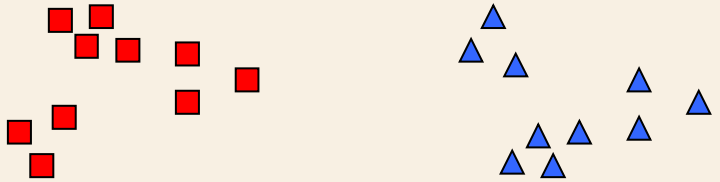
CLUSTERING

- Cluster analysis groups data objects based on information found only in the data that describes the objects and their relationships.
- The objects within a group are similar (or related) to each other, while they are different from (or unrelated to) the objects in other groups.
- Clustering can be regarded as a form of classification in that it creates a labeling of objects with class (cluster) labels.
  - It is a form of unsupervised classification.
  - In Data Mining classification is always supervised!

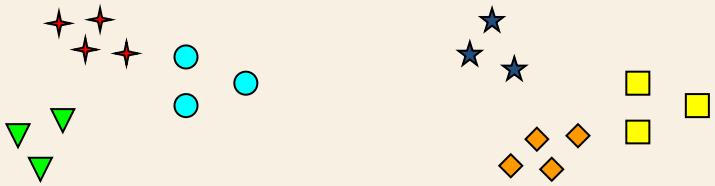
CLUSTERING



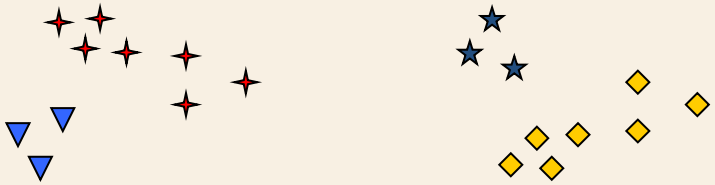
(a) How many clusters?



(b) Two Clusters



(d) Six Clusters



(c) Four Clusters

TYPES OF CLUSTERING

- A clustering is an entire collection of clusters

Hierarchical vs Partitional

- Partitional clustering is simply a division of the set of data objects into non-overlapping clusters. Each object is exactly in one subset.
  - Taken individually, clusters (b)-(d) are partitional clustering.
- Hierarchical clustering is a set of nested clusters that are organized as a tree.
  - The root is the cluster containing all the objects
  - Each node in the tree (except the leaf nodes) is the union of its children.
  - (a) can be seen as the root of a hierarchical clustering, and (b)-(d) the levels.

TYPES OF CLUSTERING

Exclusive vs Overlapping vs Fuzzy

- Exclusive clustering assigns each object to a single cluster.
- Overlapping clustering can assign each object simultaneously to more than one group (cluster).
  - E.g. a person could simultaneously belong to both the group of university students and of workers.
- Fuzzy clustering assigns each object to every cluster but with a membership weight that is between 0 (absolutely does not belong to) to 1 (absolutely belong to).
  - It avoids the arbitrariness of assigning an object to only one cluster when it is close to several.
  - It can be converted to an exclusive clustering by assigning each object to the cluster with the highest membership value.

TYPES OF CLUSTERING

Complete vs Partial

- Complete clustering assigns every object to a cluster
- Partial clustering some objects could be not assigned
  - Noise, outliers, etc…

TYPES OF CLUSTERS

- Well-separated
  - A clusters is a group of objects such that the distance between any two objects in different groups is larger than the distance between any two objects within the group.

3 well-separated clusters

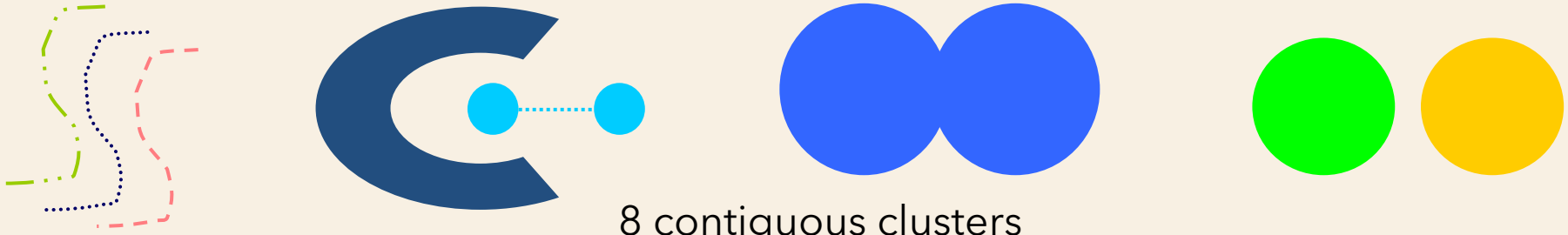
TYPES OF CLUSTERS

- Prototype-based (or center-based)
  - A cluster is a set of objects in which each object is closer (more similar) to the prototype that defines the cluster than the prototype of any other cluster.
  - Prototype = centroid with continuous attributes

4 center-based clusters

TYPES OF CLUSTERS

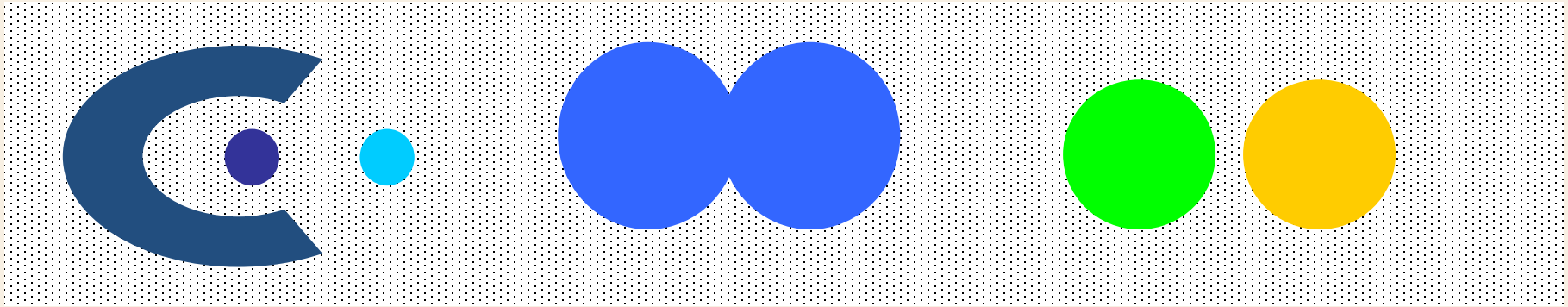
- Graph-based
  - If data are represented as graph, a cluster can be defined as a connected component: a group of objects that are connected to one another, but have no connection outside the group.
  - Contiguity-based cluster: two objects are connected only if they are within a specified distance.
    - Each object in a contiguity-based cluster is closer to some other objects in the cluster than to any object in a different cluster



8 contiguous clusters

TYPES OF CLUSTERS

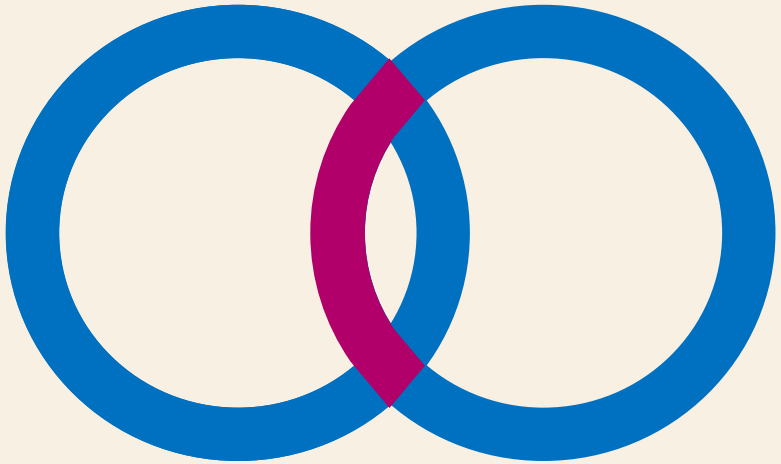
- Density-based
  - A cluster is a dense region of objects that is surrounded by a region of low density
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.



6 density-based clusters

TYPES OF CLUSTERS

- Shared properties (Conceptual clusters)
  - A cluster is a set of objects that share some properties.
  - This notion encompasses all the previous definitions, but also includes new types of clusters.



2 conceptual clusters

K-MEANS

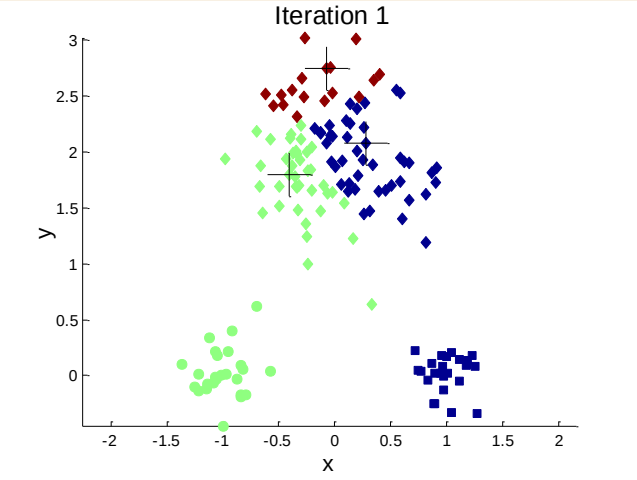
K-MEANS

- It is a *prototype-based* clustering technique which attempt to find a user-specified number of clusters (K) which are represented by their centroids.
- The prototype is defined in terms of a centroid (the mean of a group of points) → continuous n-dimensional space.
- The centroid almost never corresponds to an actual data point.

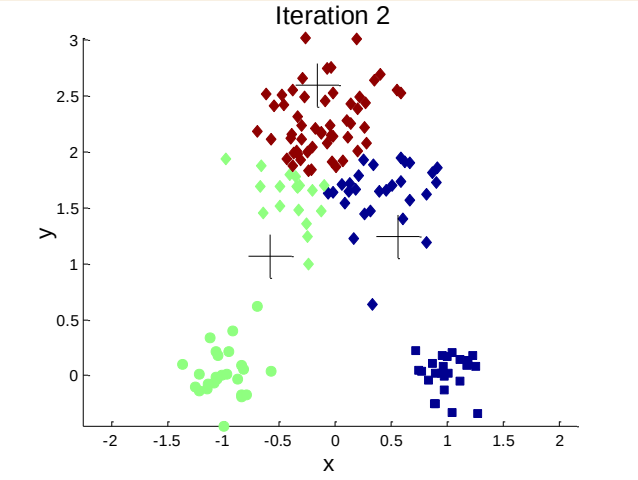
K-MEANS

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

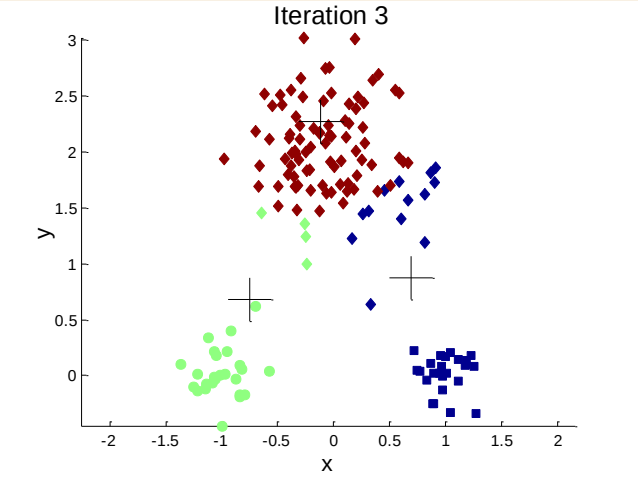
K-MEANS

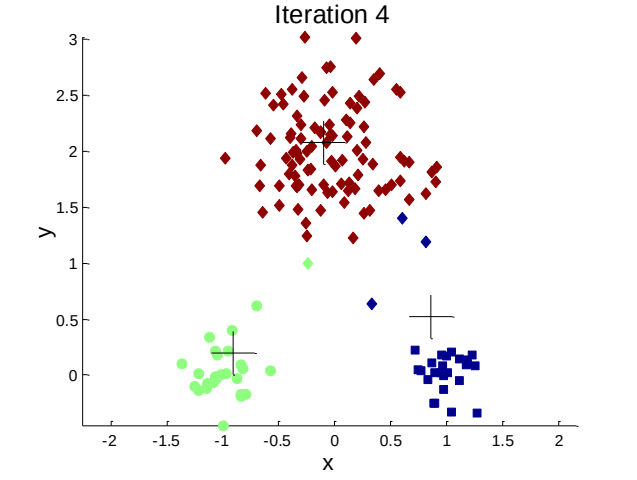


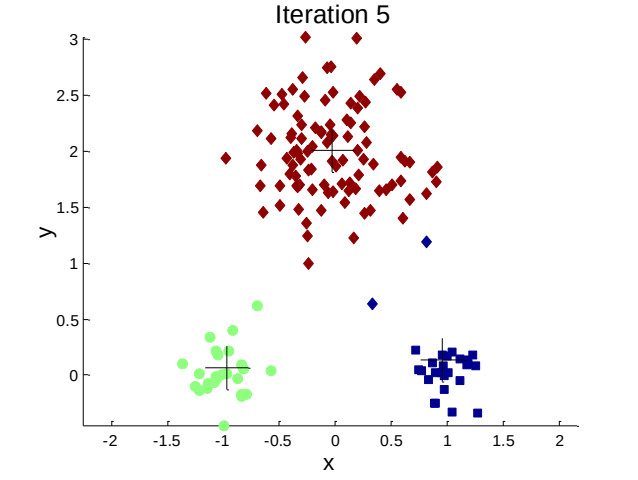
Points are assigned to the initial centroids

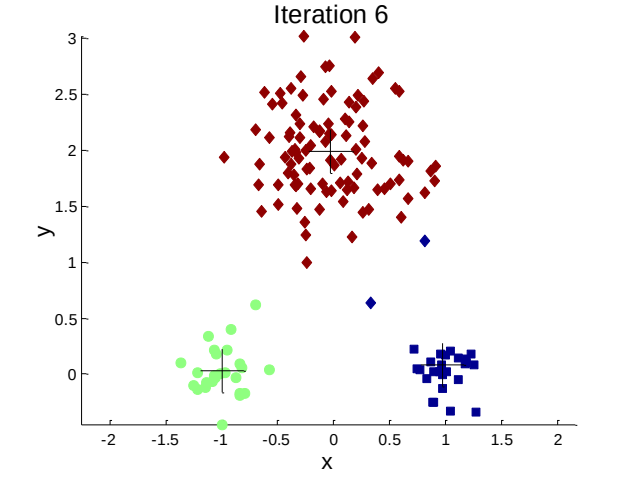


After points are assigned to a centroid, the centroids are update → centroids = the mean









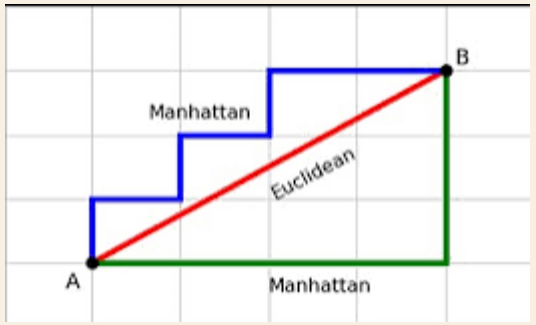
Algorithm terminates since no more changes occur

K-MEANS: TERMINATION

- K-means always converges to a solution.
- It always reaches a state in which no points are shifting from one cluster to another.
  - The centroids do not change.
- The termination condition can be relaxed by using a threshold.

K-MEANS: DISTANCE FUNCTION

- To assign a point to the closest centroid, we need a proximity measure that quantities the notion of closest for the specific data under consideration.
  - Euclidean distance: d(p,q) = $\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$
  - Cosine similarity: CS(A,B) = $\frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}}$
  - Manhattan distance: sum of the absolute differences of their respective Cartesian coordinates: d(p,q) = $\sum_{i=1}^{n} |p_i - q_i|$
  - Jaccard measure: (text documents) similarity between two sets: J(A,B) = $\frac{|A \cap B|}{|A \cup B|}$



CENTROIDS AND OBJECTIVE FUNCTIONS

- The goal of clustering is typically expressed by an objective function that depends on the proximities of the points to another one or to the cluster centroids.
- Quality of the cluster = minimize the distance of each point to its closest centroid
- Data in the Euclidean space: sum of squared errors (SSE)
- Documents: cohesion

SUM OF THE SQUARED ERRORS (SSE)

- For data in the Euclidean space, we use as our objective function the measure the quality of the cluster the Sum of the Squared Errors (SSE).
- We calculate the error in each data point = its Euclidean distance to the closest centroid, and then compute the total sum of the squared errors.
- Given two clusters, we prefer the one with the smallest SSE → its centroids (prototypes) are a better representation of the points in the cluster.
- SSE = $\sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2$
  - x = an object in the i-th cluster C_i, c_i is the centroid of C_i, K is the number of clusters

SUM OF THE SQUARED ERRORS (SSE)

- The centroid that minimizes the SSE of the cluster is the mean
- $c_i = \frac{1}{m_i} \sum_{x \in C_i} x$
- where m_i is the number of objects in the i-th cluster

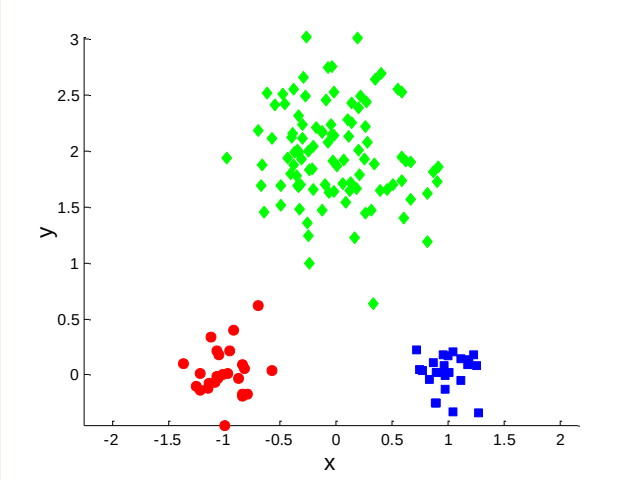
- Suppose a cluster of three points (1,1), (2,3) and (6,2), the centroid c_i = ((1+2+6)/3, (1+3+2)/3) = (3,2)

COHESION

- For document data represented as a document-term matrix (=matrix where we store the frequency of each term), the quality of the cluster is given by the cohesion
- total cohesion = $\sum_{i=1}^{K} \sum_{x \in C_i} cosine(c_i, x)$

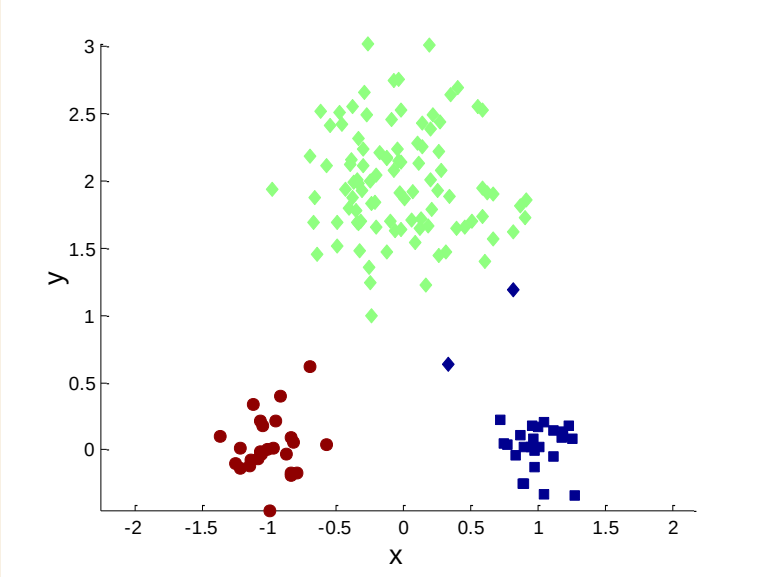
CHOOSING INITIAL CENTROIDS

- We random initialization of centroids is used, different runs of K-means typically produce different total SSE.

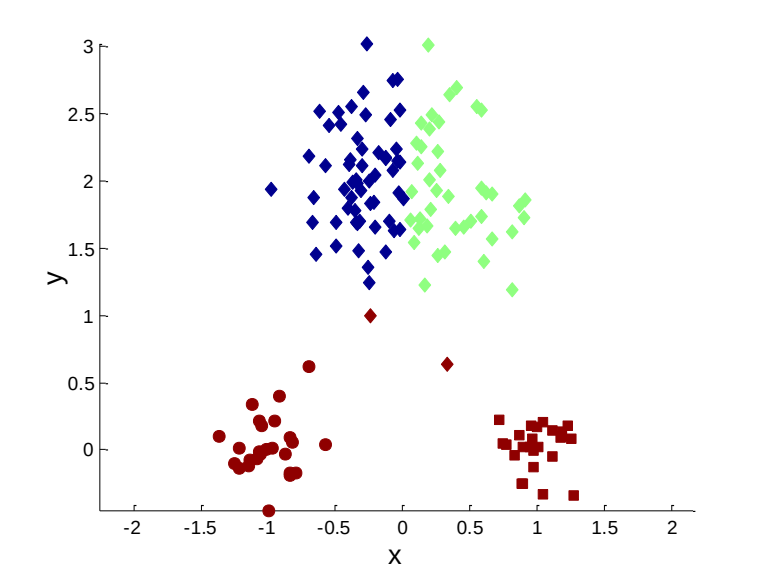


Original Points

CHOOSING INITIAL CENTROIDS



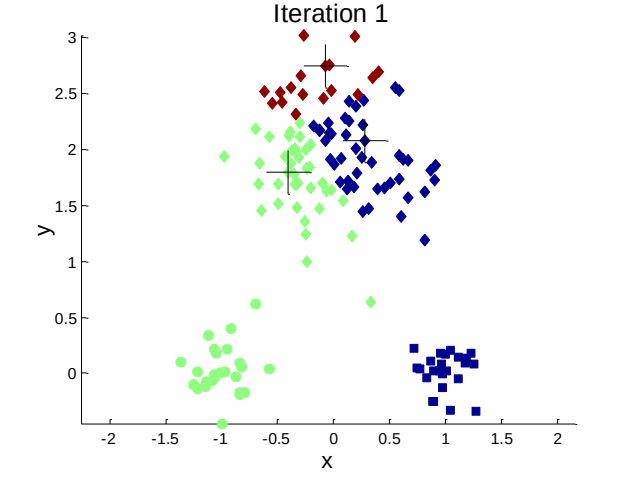
Optimal Clustering

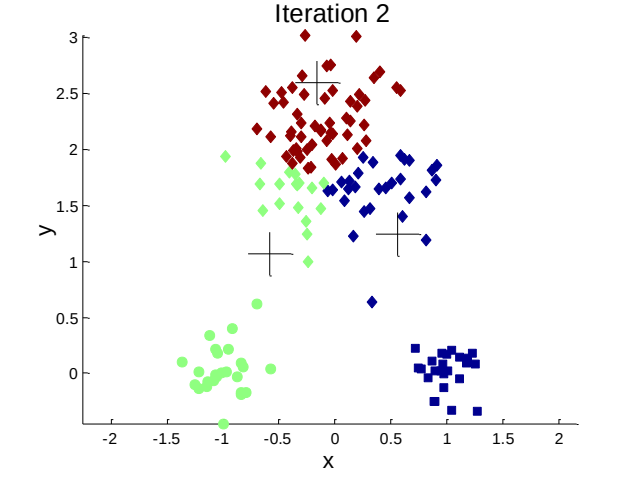


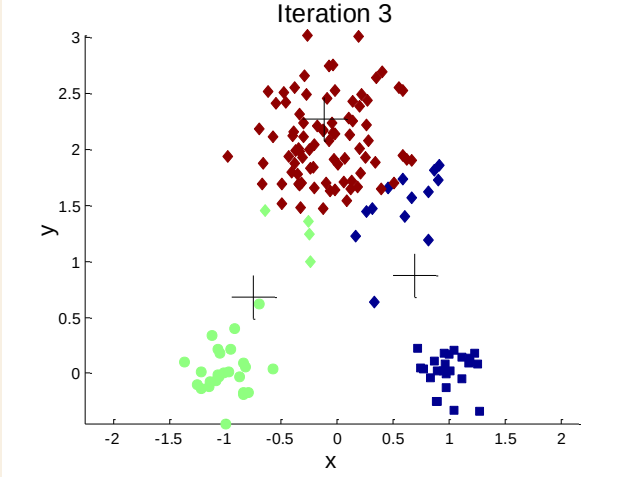
Sub-optimal Clustering

Choosing the proper initial centroids is the key step of the K-means!

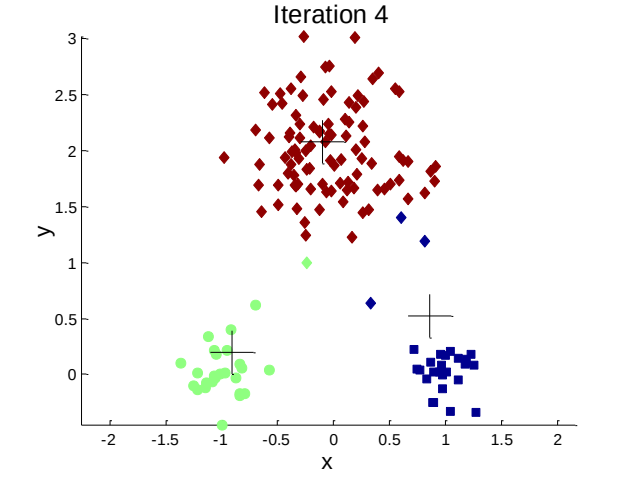
CHOOSING INITIAL CENTROIDS

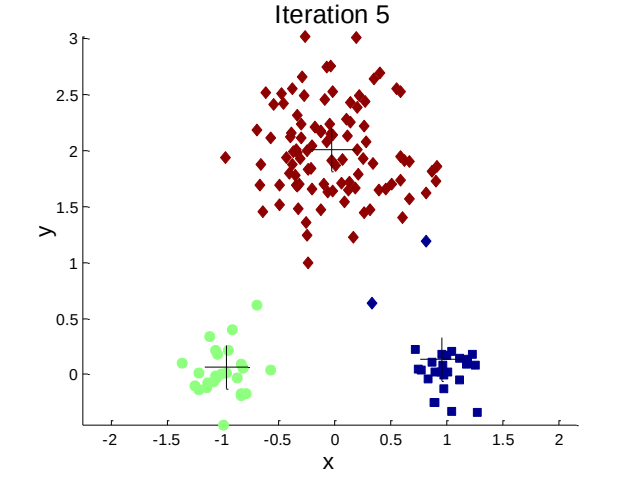


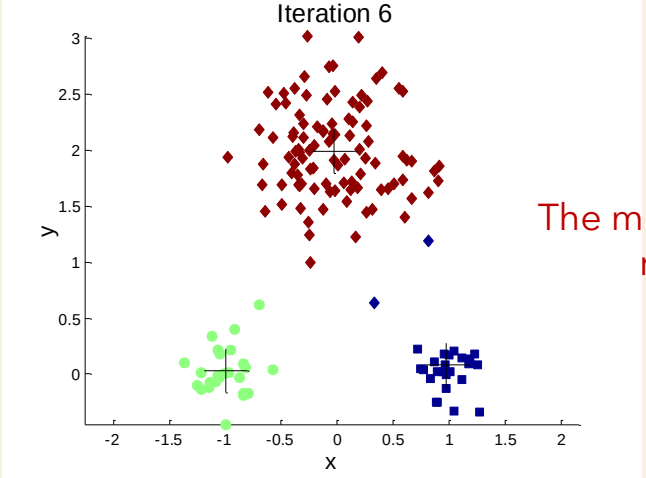




The initial centroids belong to the same natural cluster



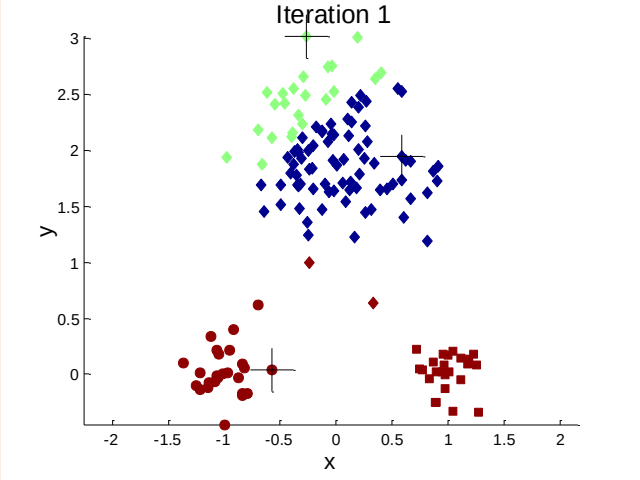


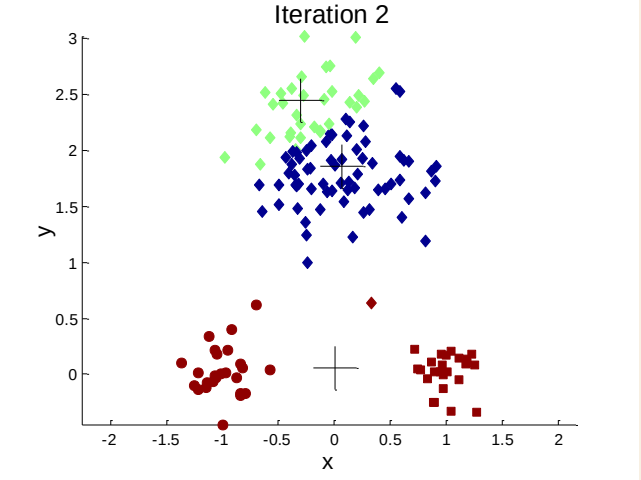


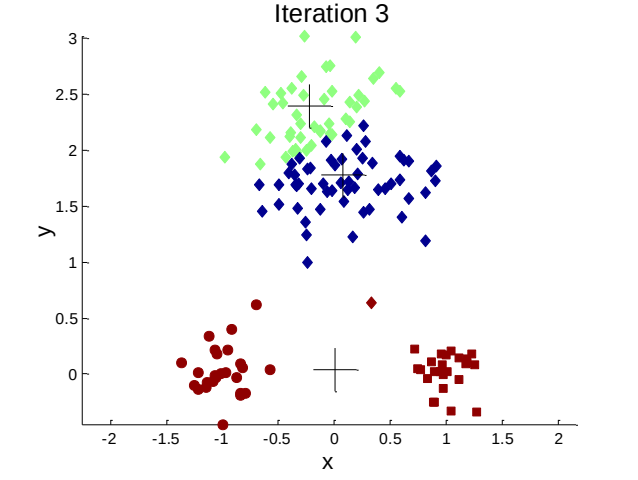
The minimum SSE is reached

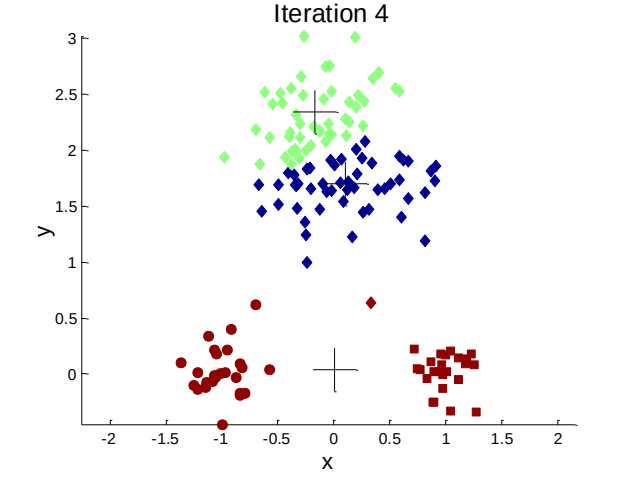
CHOOSING INITIAL CENTROIDS:

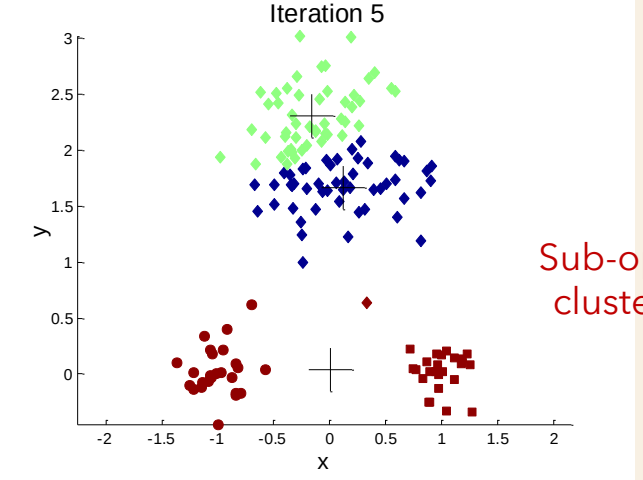
The initial centroids seam to be better distributed











Sub-optimal clustering!

CHOOSING INITIAL CENTROIDS: SOLUTIONS

- Multiple runs: each with a different set of randomly chosen initial centroids and then select the set of cluster with minimum SSE
- Take a sample of points and cluster them using a hierarchical clustering technique to determine initial centroids
  - Only if: (1) the sample is relatively small, (2) K is relatively compared to the sample size
- Select the first point at random, and then select the point that is farthest from any of the initial centroid already selected → K-means++
  - (1) it can select outliers, (2) it can be expensive to compute the farthest points
  - Apply this approach to a sample
- Post-processing
- Bisecting K-means

CHOOSING INITIAL CENTROIDS: K-MEANS++

1. Select an initial point at random to be the first centroid
2. **for** i = 1 to *number of trials* do
3. for each of the N points, x_i, $1 \leq i \leq N$, find the minimum squared distance to the currently selected centroids, $C_1, \ldots, C_j$, $1 \leq j < k$, i.e., $\min_j d^2(C_j, x_i)$ → compute the distance of each point to its closest centroid
4. Randomly select a new centroid by choosing a point with probability proportional to $\frac{\min_j d^2(C_j, x_i)}{\sum_i \min_j d^2(C_j, x_i)}$ → assign to each point a probability proportional to its distance and pick up a centroid from the remaining points using the weighted probabilities.
5. **end for**

K-MEANS ISSUES: EMPTY CLUSTERS

- Empty clusters can be obtained if no points are allocated to a cluster during the assignment step.
- Solutions:
  - Replace the centroid with the point that is farthest away from any current centroid.
  - Replace the centroid by choosing a point at random from the cluster with the highest SSE.

K-MEANS ISSUES: OUTLIERS

- The presence of outliers can influence the clusters that are found when the SSE criterion is used.
- Discover and eliminate outliers beforehand.
  - Specific techniques for identifying outliers.
  - Eliminate points with unusually high contribution to SSE over multiple runs
  - Small clusters can be groups of outliers.

K-MEANS: POST-PROCESSING

- It is possible to reduce the SSE also with post-processing activities (split and merge):
  - Increasing the number of clusters:
    - Splitting: clusters with the largest SSE or the largest standard deviation for a particular attribute are divided
    - Introduce a new cluster centroid: use the point farthest from any cluster center or one random point from the cluster with the highest SSE
  - Reducing the number of clusters:
    - Merging: two clusters with the closest centroids are combined, or that results in a cluster with the smallest increase in the total SSE
    - Disperse a cluster: remove the centroid and reassign the point to another cluster.

BISECTING K-MEANS

- Variant of K-means that produces a better partitioning
- General idea: to obtain K clusters, split the set of all points into two clusters, select one of these clusters to split, and so on, until K clusters have been produced.

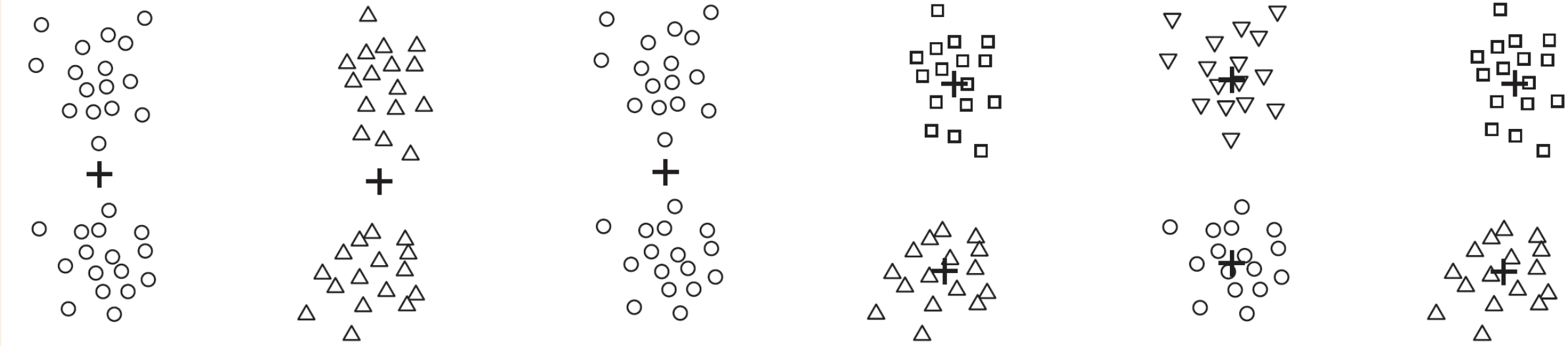
1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

There are several different ways to choose which cluster to split!

BISECTING K-MEANS

- Choice of the cluster to split:
  - The largest cluster
  - The cluster with the largest SSE
  - A criterion based on both the size and the SSE
- Different choices results in different clusters

BISECTING K-MEANS



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

K-MEANS: STRENGTHS AND LIMITATIONS

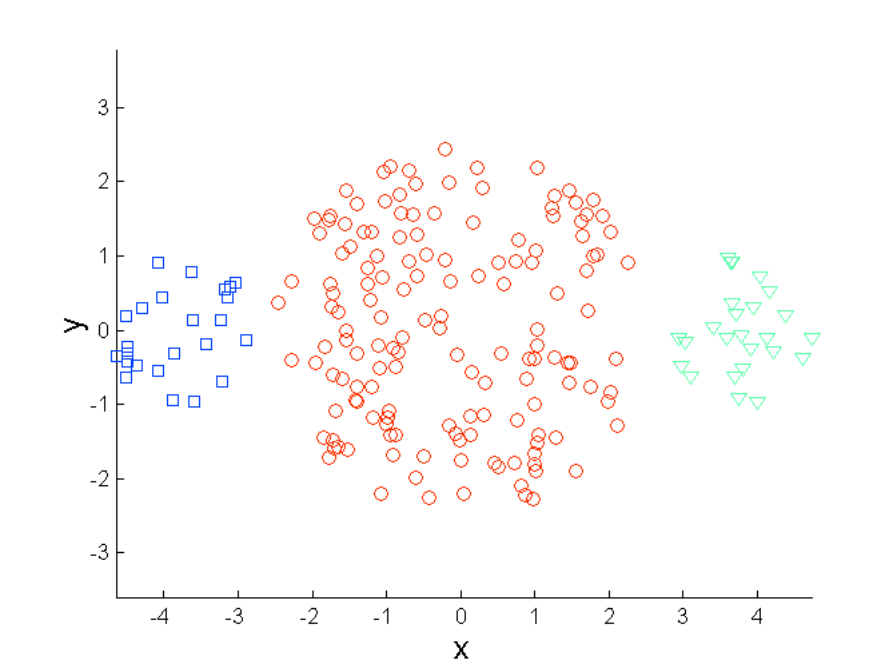
Strengths:

- K-means is simple and can be used for a wide variety of data types.
- K-means is quite efficient.

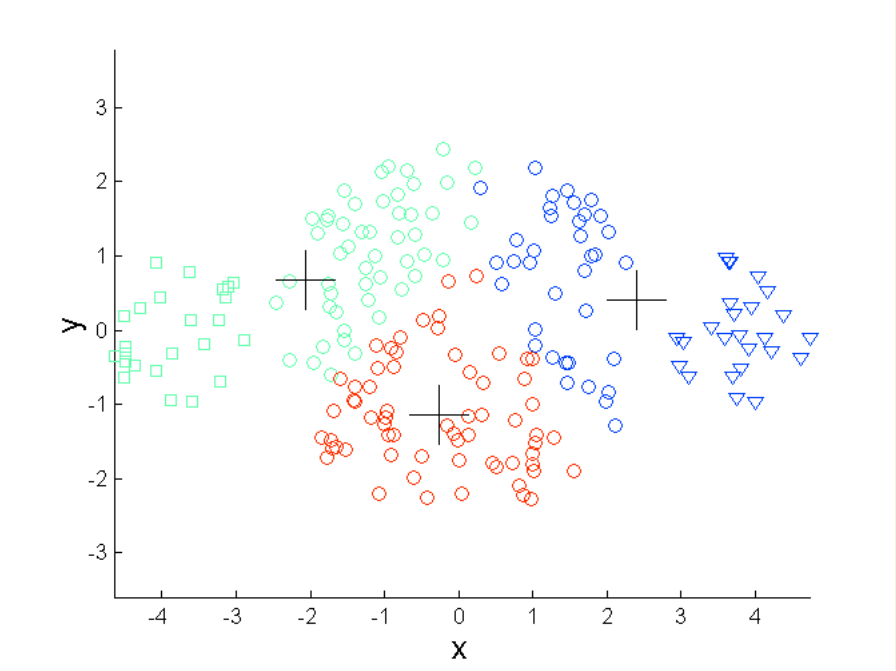
Limitations:

- K-means has trouble in clustering data which contains outliers.
- K-means is restricted to data for which there is a notion of center.
- K-means has problems when clusters have non-spherical (or globular) shapes or widely different sizes or densities.

LIMITATIONS OF K-MEANS: DIFFERENT SIZES



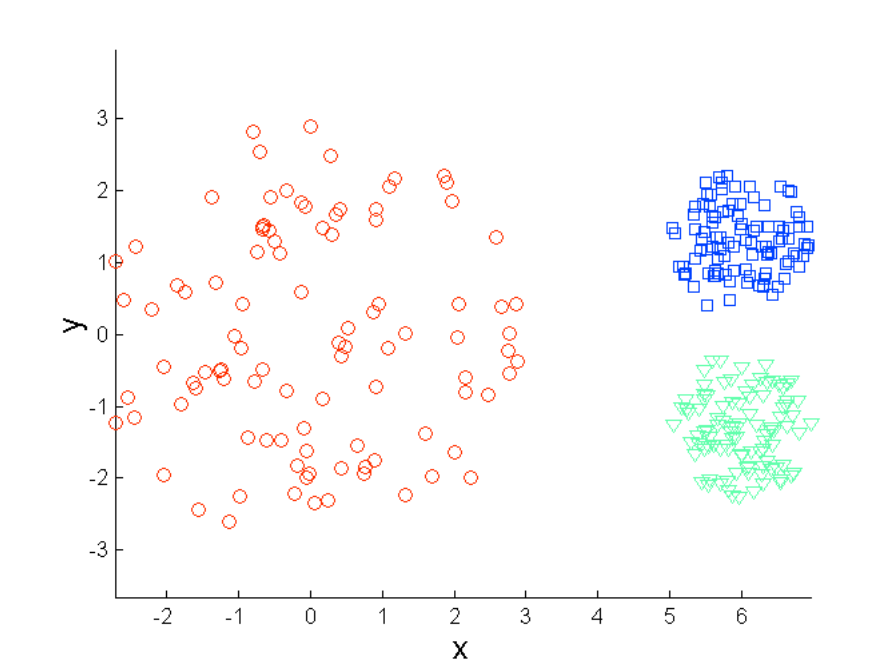
Original Points



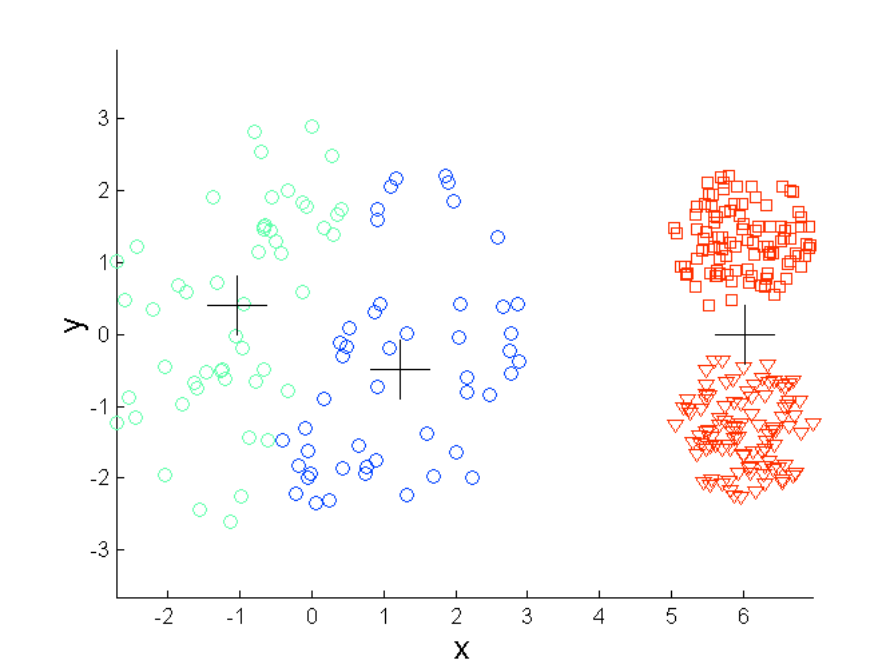
K-means (3 Clusters)

The largest cluster is broken

LIMITATIONS OF K-MEANS: DIFFERING DENSITY



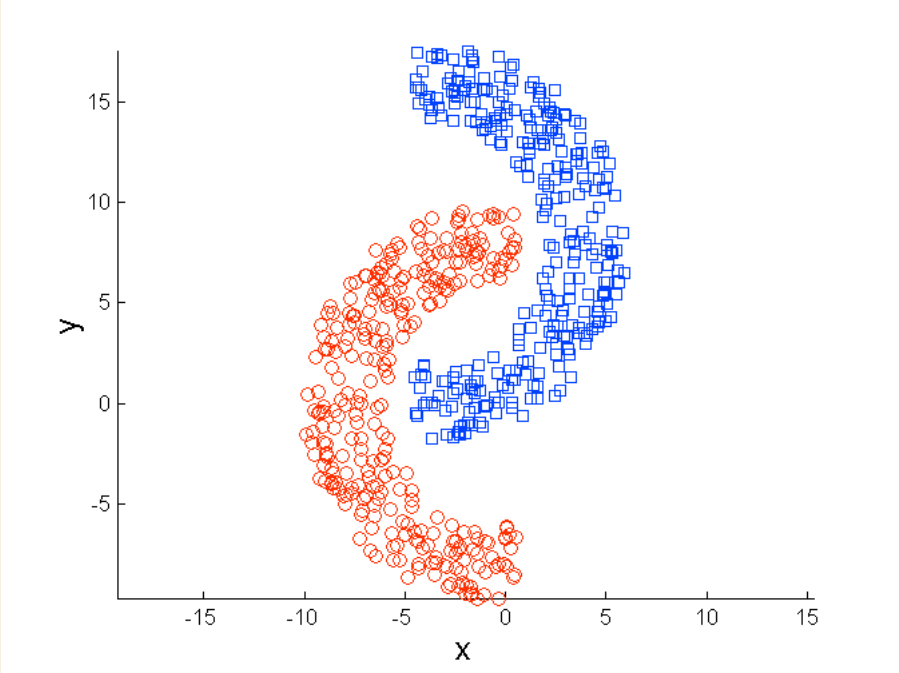
Original Points



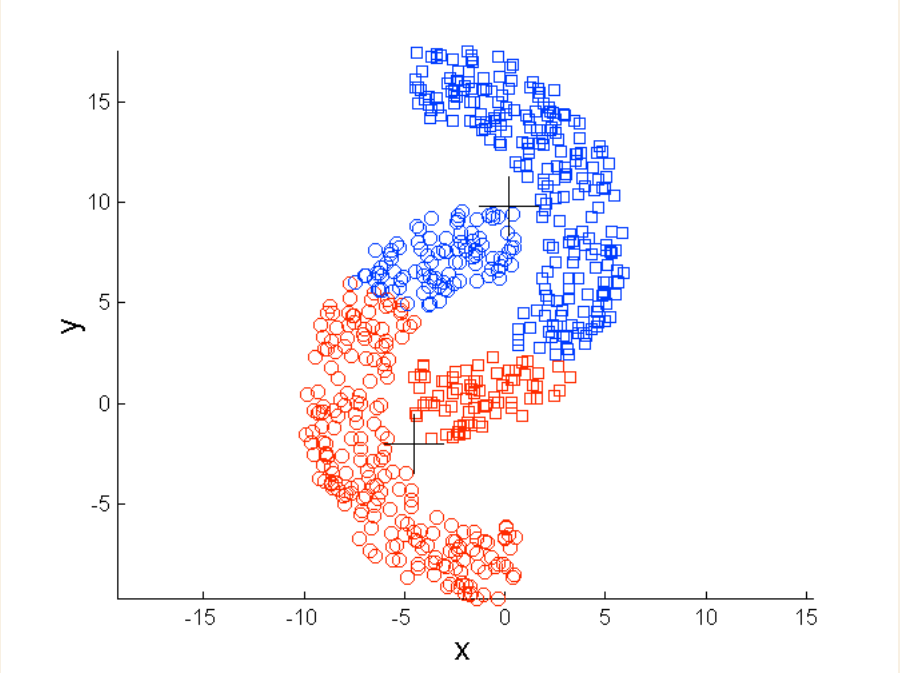
K-means (3 Clusters)

The two small clusters are much denser than the largest one

LIMITATIONS OF K-MEANS: SHAPE NOT GLOBULAR



Original Points



K-means (2 Clusters)

HIERARCHICAL CLUSTERING

HIERARCHICAL CLUSTERING

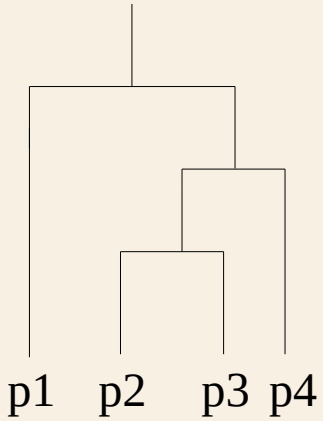
- A hierarchical clustering produces a set of nested clusters organized as a hierarchical tree.
- It does not assume any particular number of clusters
  - The desired number of clusters can be obtained by “cutting” the hierarchy at the proper level.
- It can correspond to meaningful taxonomies

HIERARCHICAL CLUSTERING

- Agglomerative:
  - Start with the points as individual clusters and, at each step, merge the closest pair of clusters.
  - Require a notion of cluster proximity.
  - The most common techniques
- Divisive:
  - Start with one, all-inclusive cluster and, at each step, split a cluster until only a singleton cluster of individual points remain.
  - Require a splitting criterion.

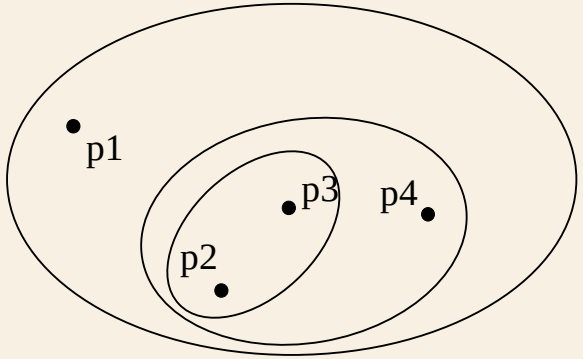
HIERARCHICAL CLUSTERING

- A hierarchical clustering is often displayed graphically by using a tree-like diagram called **dendrogram** which displays the cluster-subcluster relationships and the order in which the clusters are merged or split.



HIERARCHICAL CLUSTERING

- For sets of two-dimensional points, a hierarchical clustering can also be displayed with a nested cluster diagram.



BASIC AGGLOMERATIVE

- Starting with individual points as clusters, merge the two closest cluster until only one cluster remains.
- Basic algorithm

1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4. Merge the two closest clusters
5. Update the proximity matrix
6. Until only a single cluster remains

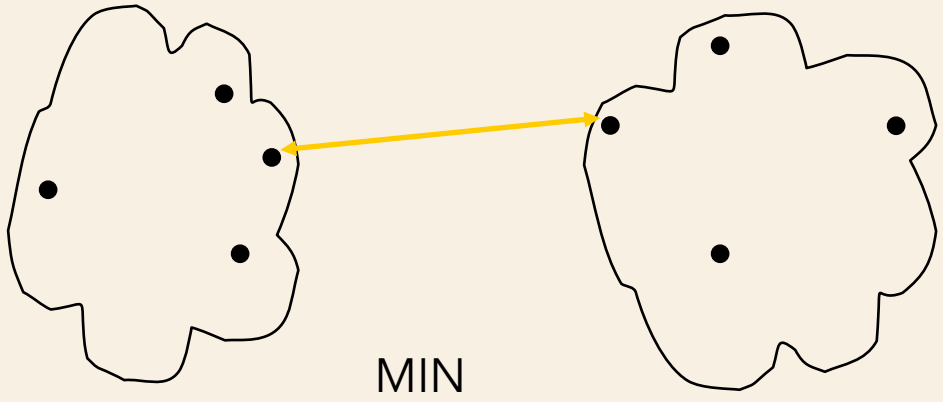
BASIC AGGLOMERATIVE

- The key operation is the computation of the proximity between two clusters.
- The definition of cluster proximity is what differentiates the various agglomerative hierarchical techniques.

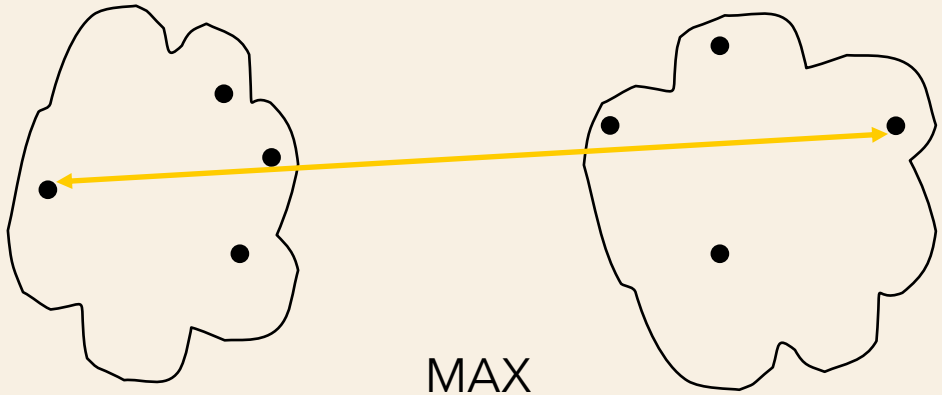
PROXIMITY BETWEEN CLUSTERS

- Graph-based view of the clusters:
  - MIN (or single LINK): defines cluster proximity as the proximity between the closest two points that are in different clusters.
    - Shortest edge
  - MAX (or complete LINK): defines cluster proximity as the proximity between the farthest two points in different clusters.
    - Longest edge
  - GROUP AVERAGE: defines cluster proximity to be the average pairwise proximities of all pairs of points from different clusters.

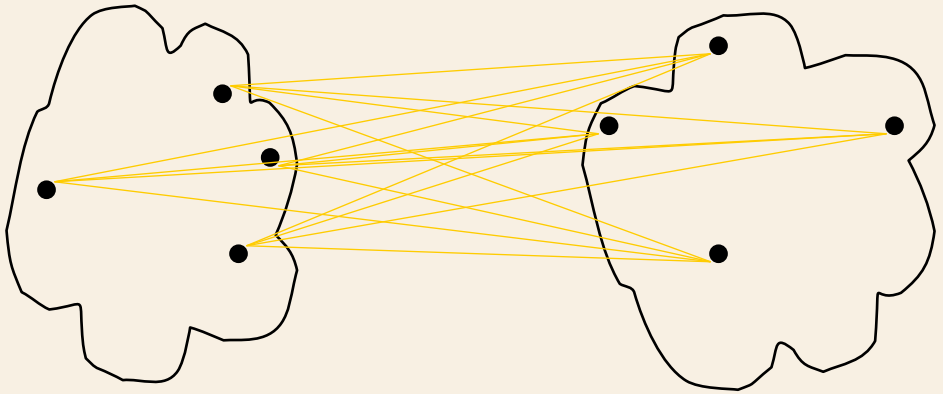
PROXIMITY BETWEEN CLUSTERS



MIN



MAX

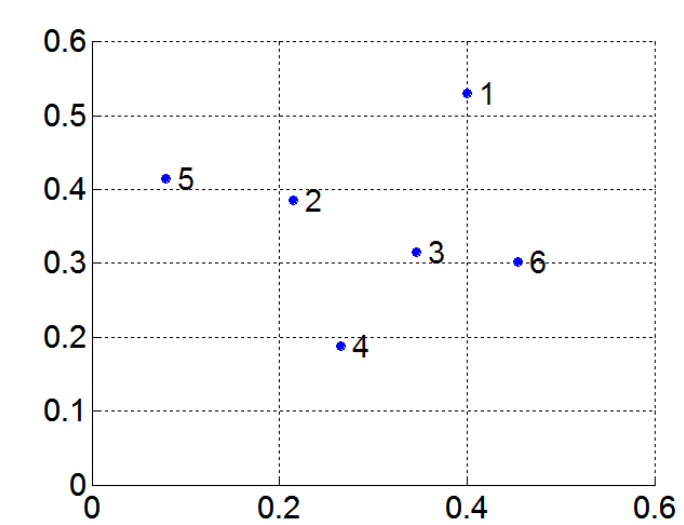


GROUP AVERAGE

PROXIMITY BETWEEN CLUSTERS

- Prototype-based view (each cluster is represented by its centroid)
  - Proximity between centroids
  - Other methods based on an objective functions
    - Ward's method uses the SSE: it measures the proximity between two clusters in terms of the increase in the SSE that results from merging two clusters.

EXAMPLE



Set of six two-dimensional points

Point	x Coordinate	y Coordinate
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022

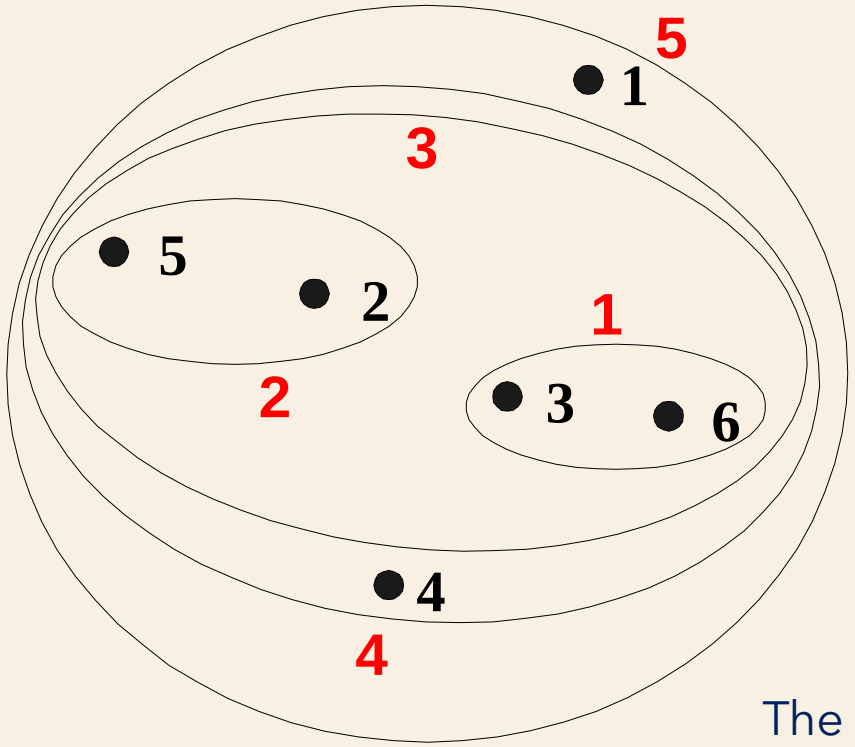
xy-coordinates

Distance Matrix (Euclidean distance)

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

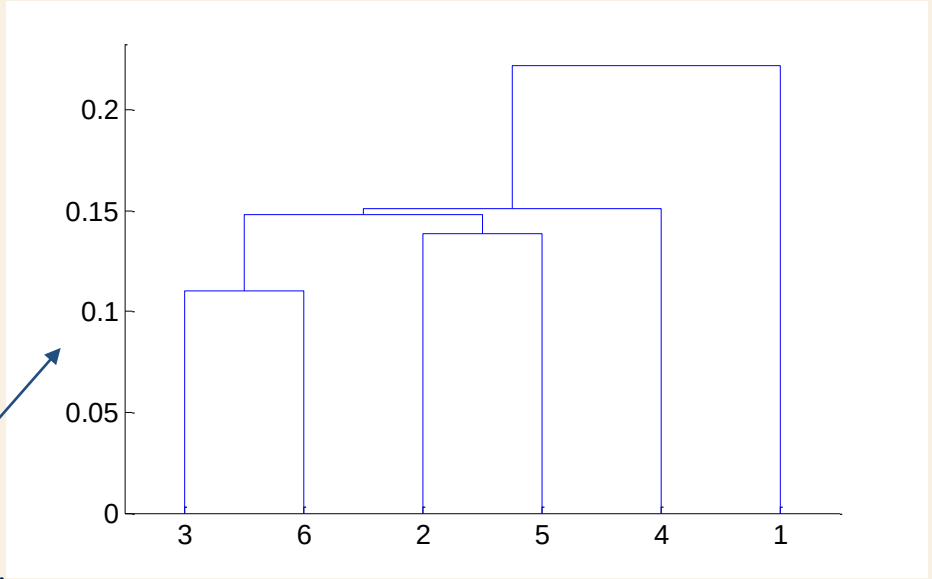
EXAMPLE: MIN OR SINGLE LINK

Nested Clusters



- Proximity = minimum of the distance between any two points in the two different cluster

Dendrogram



The height is the distance.

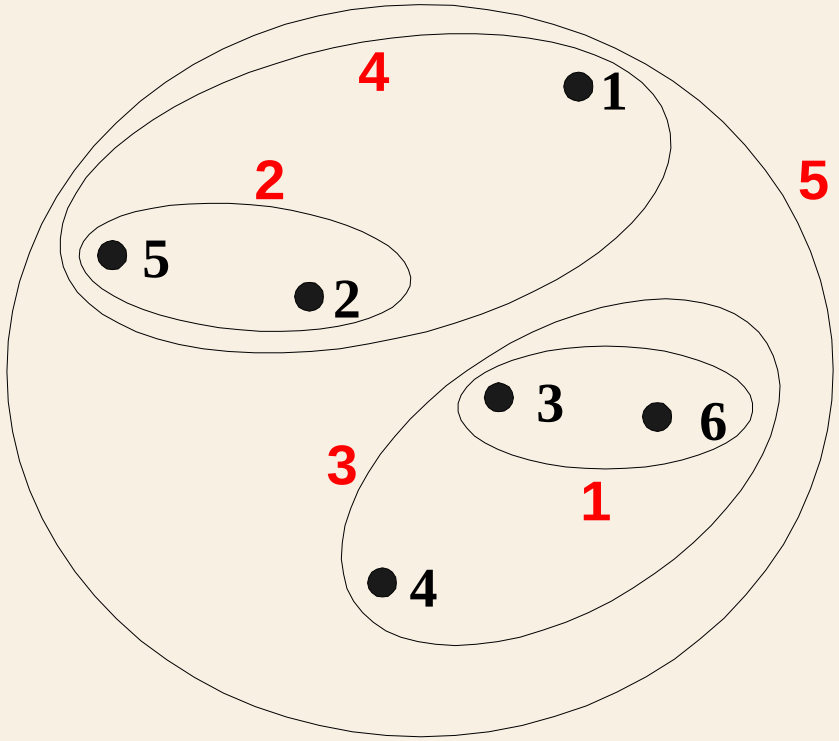
dist({3}, {6}) = 0.11

dist({3,6}, {2,5}) = min(dist(3,2), dist(3,5), dist(6,2), dist(6,5))
= min(0.15, 0.25, 0.28, 0.39) = 0.15

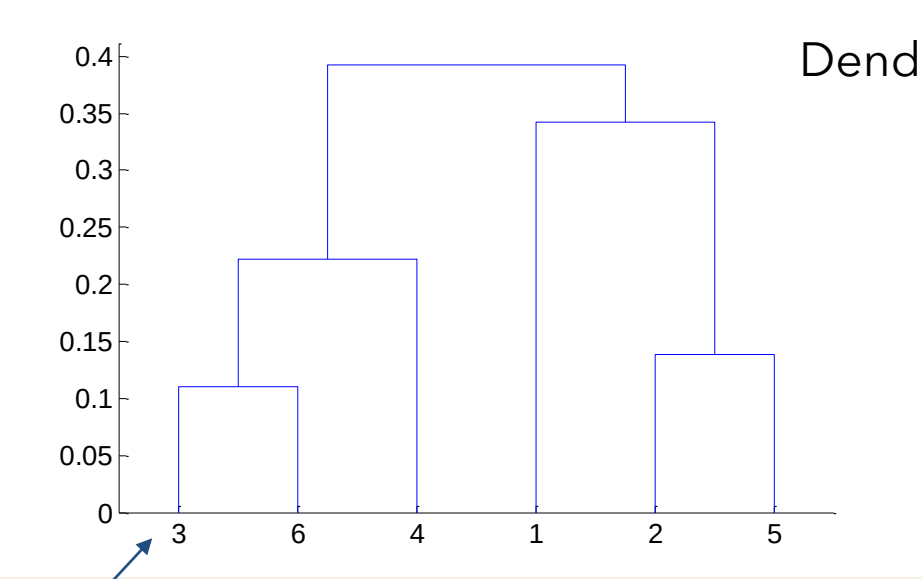
EXAMPLE: MAX OR COMPLETE LINK OR CLIQUE

- Proximity = maximum of the distance between any two points in the two different clusters

Nested Clusters



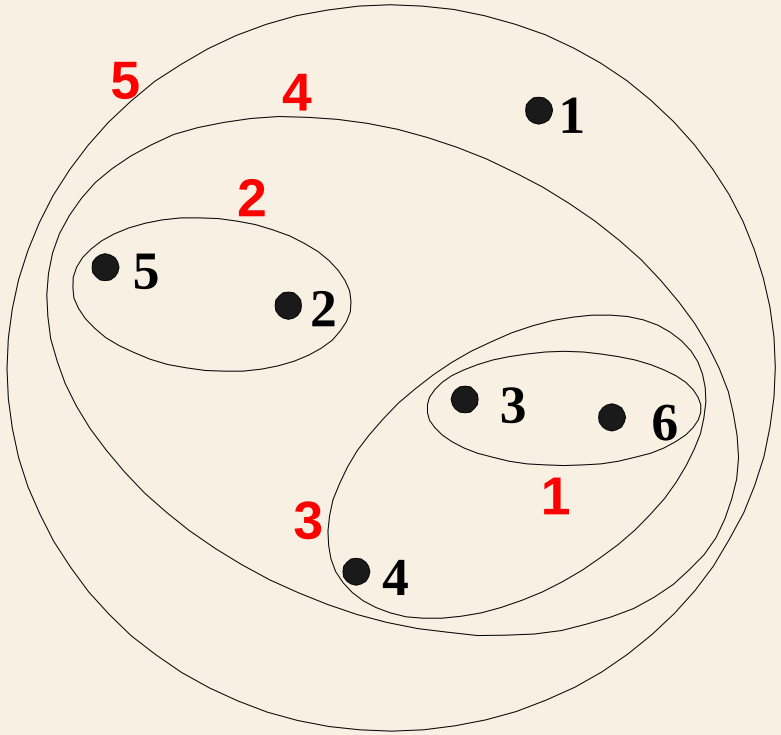
Dendrogram



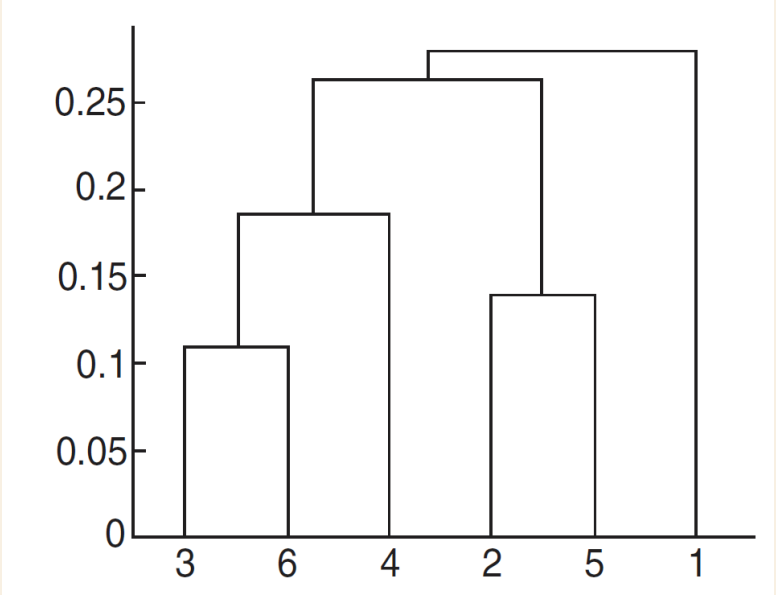
dist({3,6}, {4}) = max(dist(3,4), (6,4)} = 0.22
dist({3,6}, {2,5}) = max(dist(3,2), dist(3,5), dist(6,2), dist(6,5)) = 0.39
dist({3,6},{1}) = max(dist(3,6), dist(6,1)) = 0.23

Points 3 and 6 are merged first, as for single link

EXAMPLE: GROUP AVERAGE



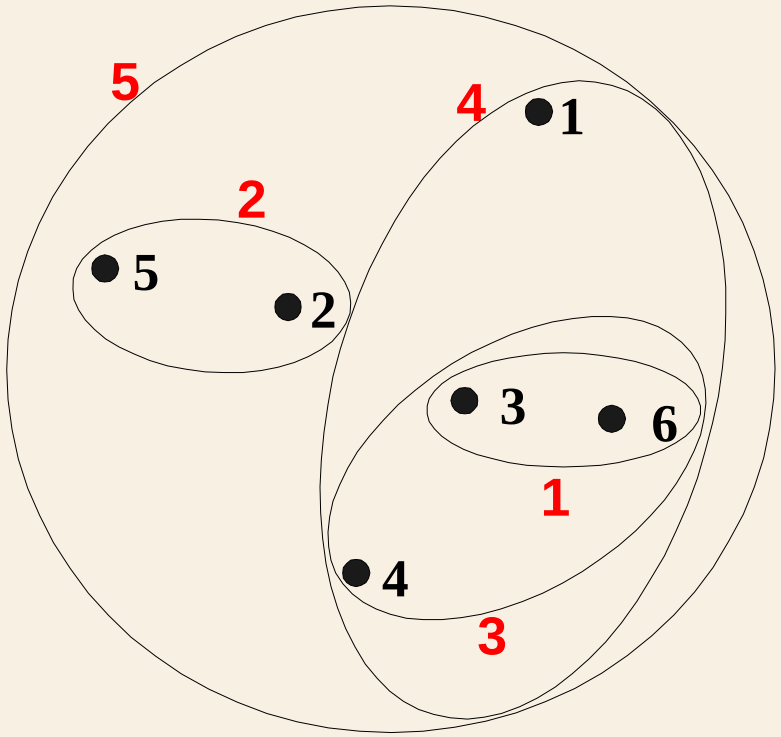
- Proximity = maximum of the distance between any two points in the two different clusters
- $proximity(C_i, C_j) = \frac{\sum_{x \in C_i} \sum_{y \in C_j} proximity(x,y)}{|C_i| \times |C_j|}$



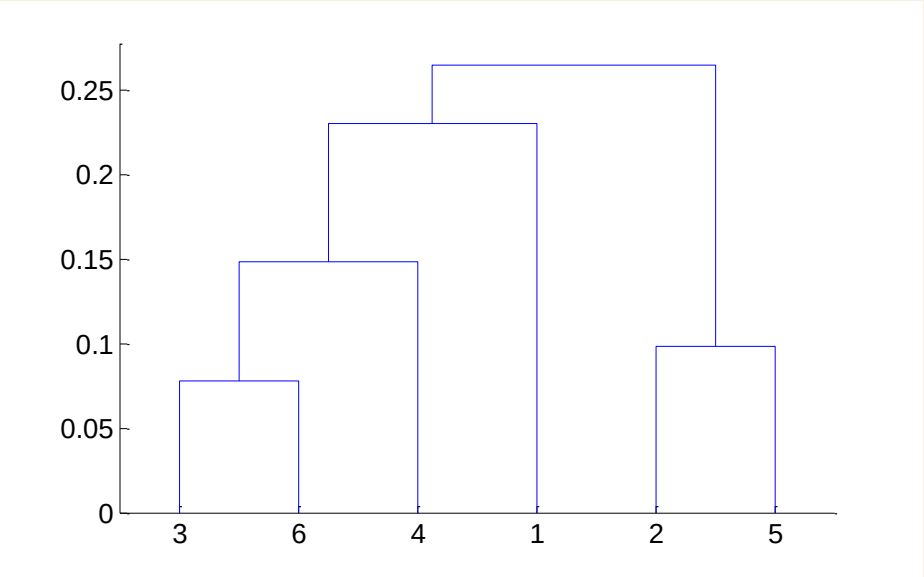
4th stage

dist({3,6,4}, {1}) = (0.22+0.37+0.23)/(3x1) = 0.28
dist({2,5}, {1}) = (0.24+0.34)/(2x1) = 0.29
dist({3,6,4},{2,5}) = (0.15+0.28+0.25+0.39+0.20+0.29)/(3x2) = 0.26

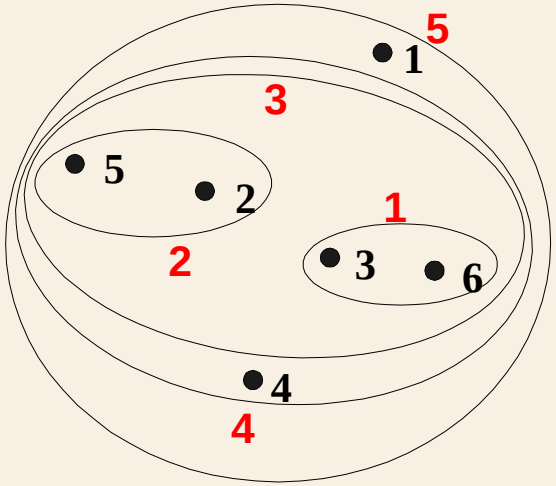
EXAMPLE: WARD'S METHOD



- Proximity = the increase in the SSE when two clusters are merged

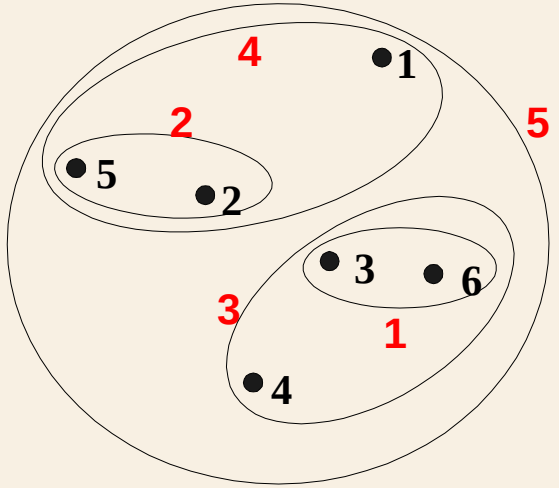


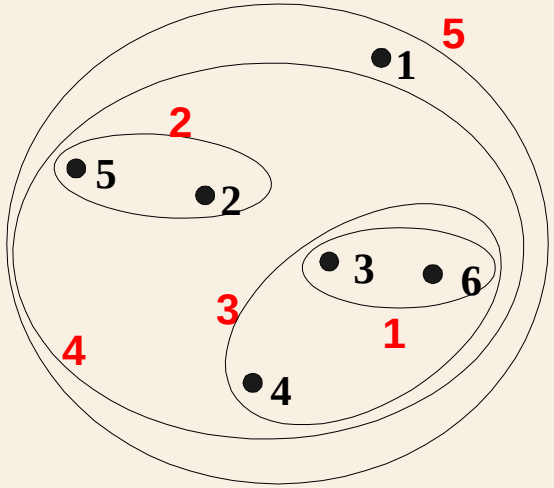
EXAMPLE: COMPARISON



MIN

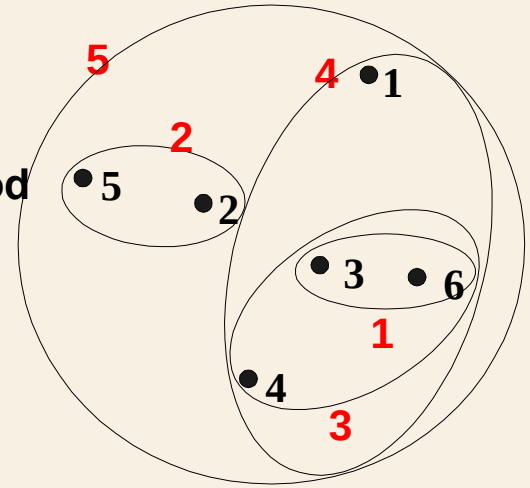
MAX





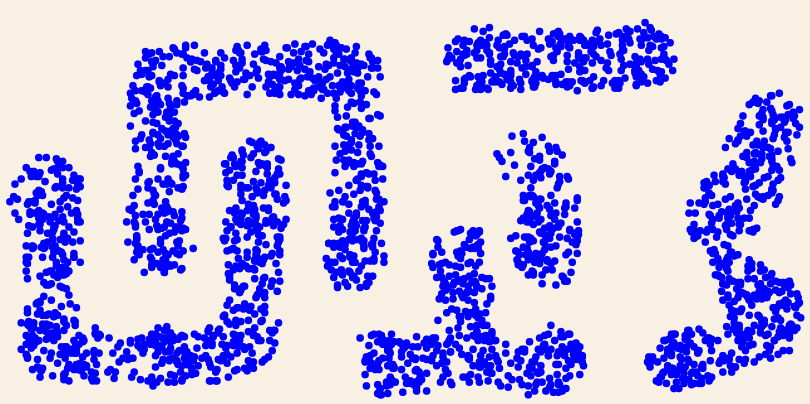
Group Average

Ward's Method

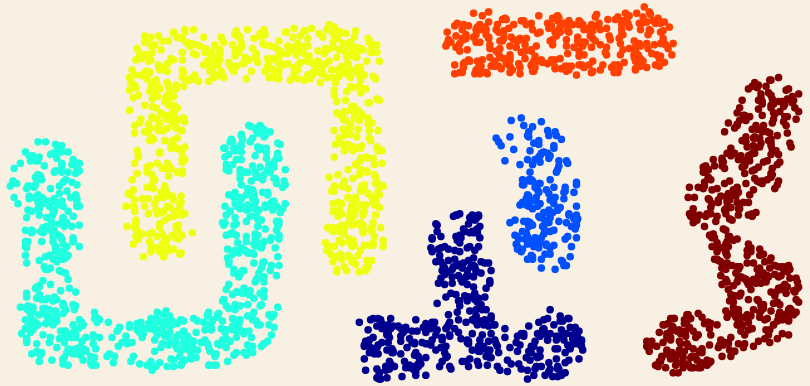


COMPARISON: MIN, MAX, GROUP AVERAGE, WARD'S METHODS

- Strength of MIN: can handle non-elliptical shapes



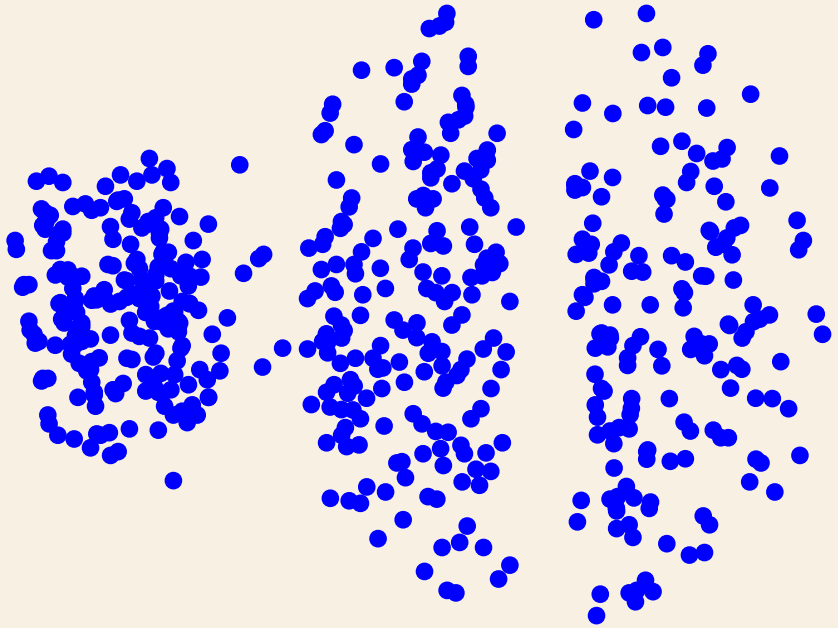
Original Points



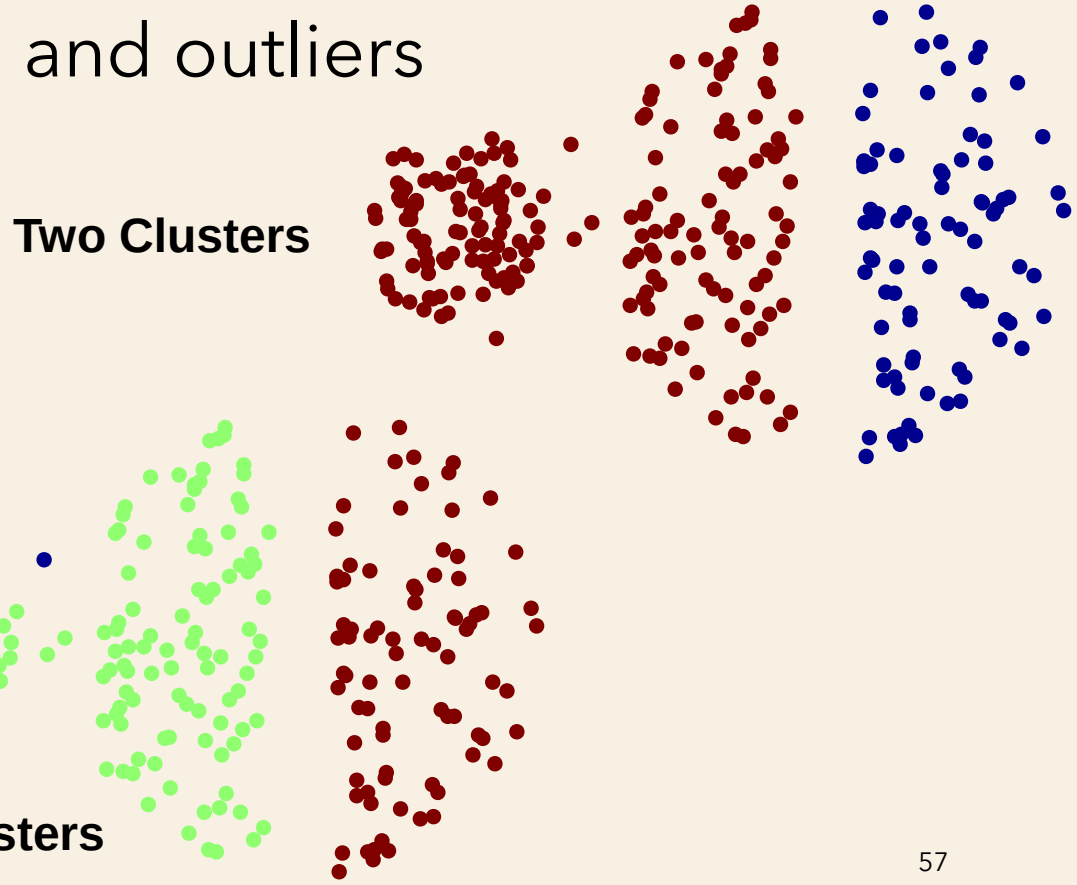
Six Clusters

COMPARISON: MIN, MAX, GROUP AVERAGE, WARD'S METHODS

- Limitation of MIN: sensitive to noise and outliers

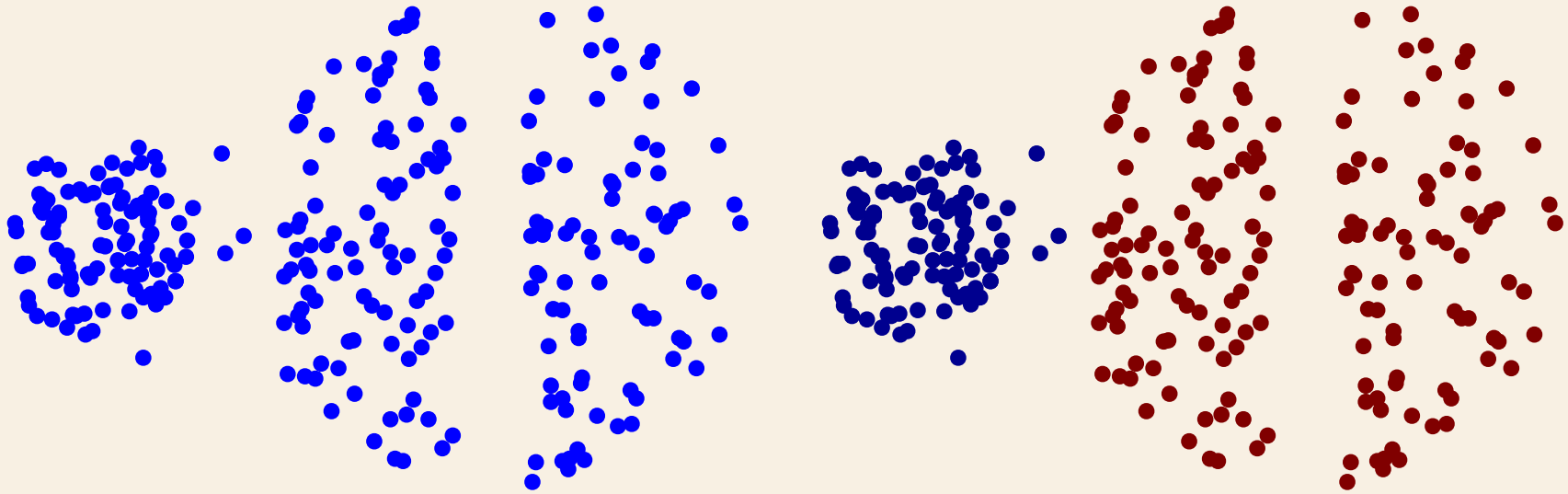


Original Points



COMPARISON: MIN, MAX, GROUP AVERAGE, WARD'S METHODS

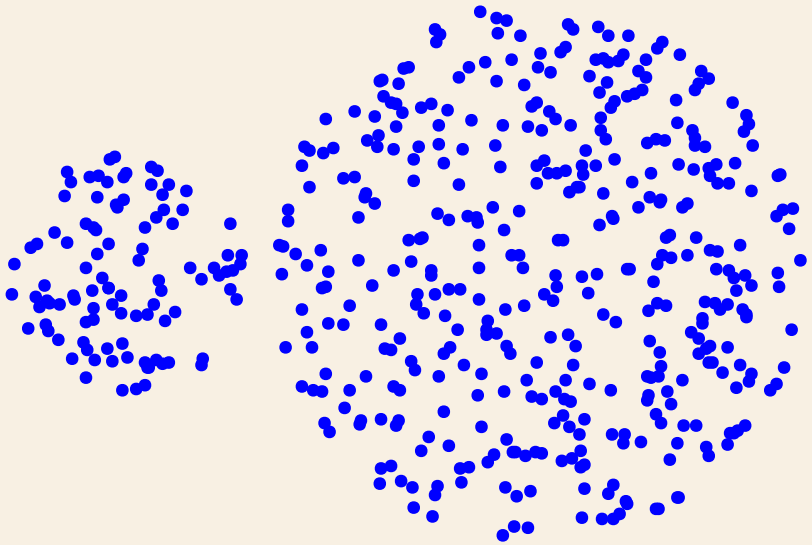
- Strength of MAX: less susceptible to noise and outliers



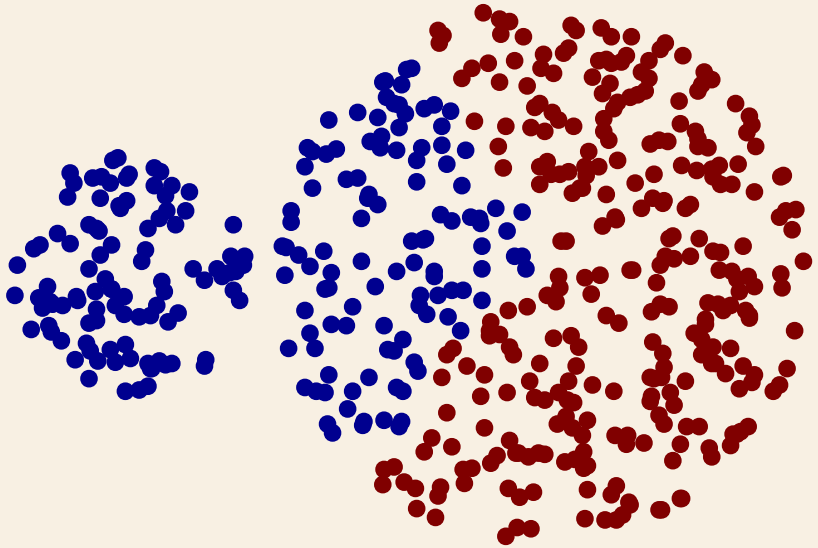
Original Points **Two Clusters**

COMPARISON: MIN, MAX, GROUP AVERAGE, WARD'S METHODS

- Limitation of MAX: it tends to break large clusters and it is biased towards globular clusters



Original Points



Two Clusters

COMPARISON: MIN, MAX, GROUP AVERAGE, WARD'S METHODS

- Group Average is a compromise between single and complete link.
- Strengths:
  - Less susceptible to noise and outliers (as MAX)
- Limitations:
  - Biased towards globular clusters (as MAX)

COMPARISON: MIN, MAX, GROUP AVERAGE, WARD'S METHODS

- Ward's method is similar to group average if distance between
- points is distance squared.
- Strengths:
  - Less susceptible to noise and outliers (as MAX)
- Limitations:
  - Biased towards globular clusters (as MAX)
- Hierarchical analogue to K-means:
  - It can be used to initialize K-means centroids

TIME AND SPACE REQUIREMENTS

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.

- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time with some cleverness

GENERAL PROBLEMS AND LIMITATIONS

- Once a decision is made to combine two clusters, it cannot be undone
- No global objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise
  - Difficulty handling clusters of different sizes and non-globular shapes
  - Breaking large clusters

DBSCAN

DBSCAN

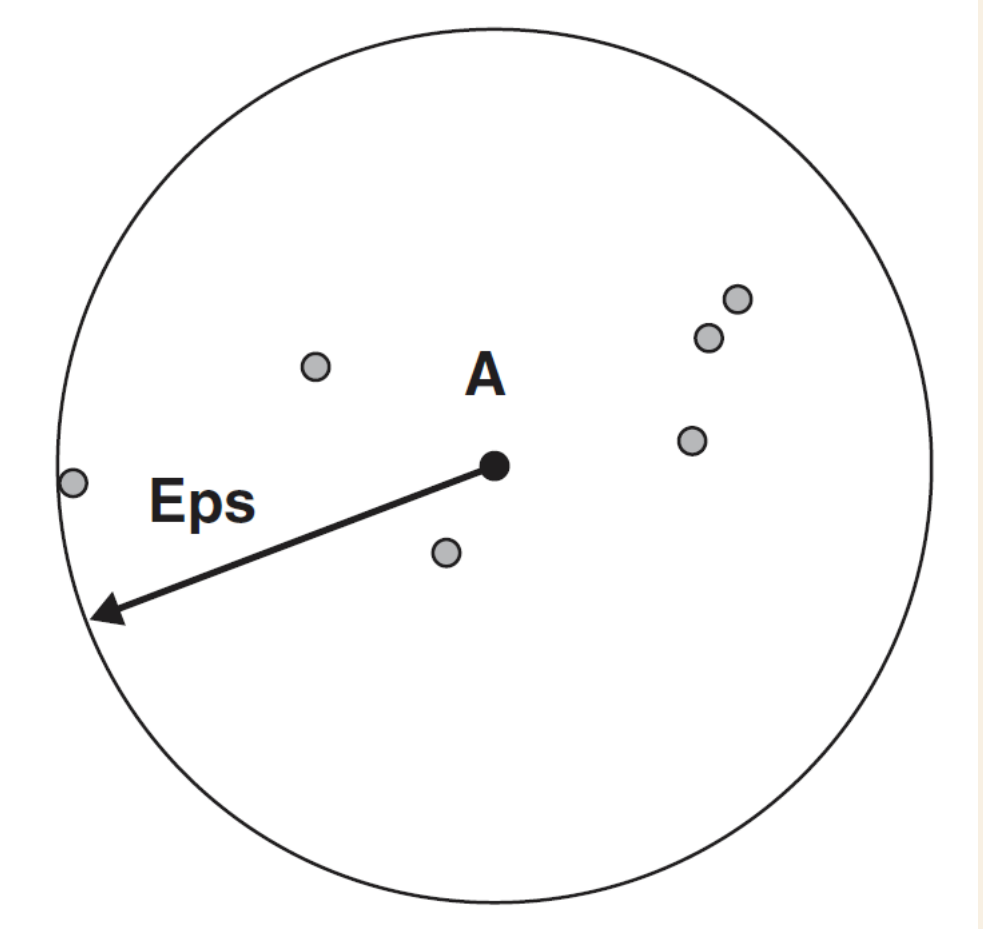
- It is a density-based clustering algorithm in which the number of clusters is automatically determined by the algorithm.
- Density-based clustering locates regions of high density that are separated from one another by regions of low density.
- It requires a notion of density instead of similarity!

DBSCAN – CENTER-BASED DENSITY

- DBSCAN uses a center-based approach in which the density for a particular point in the dataset is estimated as the number of points within a specified radius *eps* of that point.
- The density of any point will depend on the specified radius.
  - If the radius is large enough each point in D has a density equal to |D|.
  - If the radius is small enough each point in D has a density equal to 1.

DBSCAN – CENTER-BASED DENSITY

- The number of points within a radius of *eps* of A is 7.

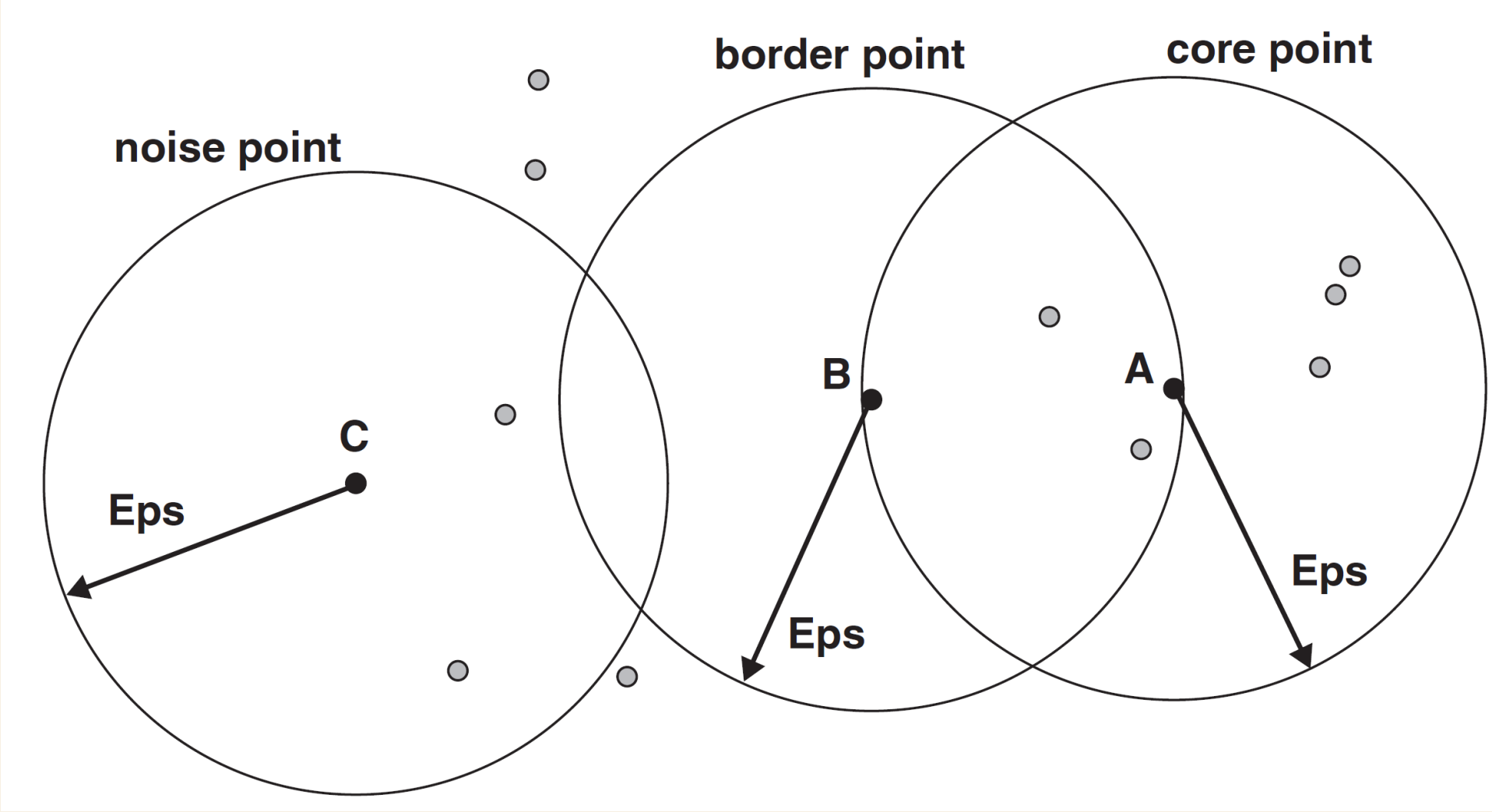


DBSCAN – POINT CLASSIFICATION

The center-based approach to density allows us to classify a point of being a:

- Center point: in the interior of a dense region.
- Border point: on the edge of a dense region.
- Noise or background point: in a sparsely occupied region.

DBSCAN - DENSITY



DBSCAN – CORE POINTS

- A core point is in the interior of a density-based cluster.
- A point p is a core point if there are at least MinPts points within a distance of *eps*, p included, where *MinPts* e *eps* are user-specified parameters.
  - In the previous figure A is a core point for the radius *eps* if *MinPts* ≥ 7.

DBSCAN – BORDER POINTS

- A border point is not a core point (i.e. it has fewer than *MinPts* points within *eps*) but falls within the neighborhood of a core point.
  - In the previous figure B is a border point.
- A border point can fall within the neighborhood of several core points.

DBSCAN – NOISE POINTS

- A noise point is any point that is neither a core point nor a border point.
  - In the previous figure C is a noise point.

DB-SCAN – ALGORITHM

- Any two core points that are close enough (i.e., within a distance *eps* of one another) are put in the same cluster.
- Any border point that is close enough to a core point is put in the same cluster as the core point.
  - Ties need to be solved in case a border point is close to more than one core point.
- Noise points are discarded.

DB-SCAN – ALGORITHM

1: Label all points as core, border, or noise points.

2: Eliminate noise points.

3: Put an edge between all core points within a distance *eps* of each other.

4: Make each group of connected core points into a separate cluster.

5: Assign each border point to one of the clusters of its associated core points

TIME AND SPACE REQUIREMENTS

- The time complexity is O(m * time to find the points in the *eps*-neighborhood).
  - Worst case = $O(m^2)$
  - With indexes (low dimensional space, kd-tree) = O(m log m)
- The space complexity is O(m) because it is necessary to keep a small amount of information for each point: cluster label and point type (core, border, noise).

SELECTION OF PARAMETERS

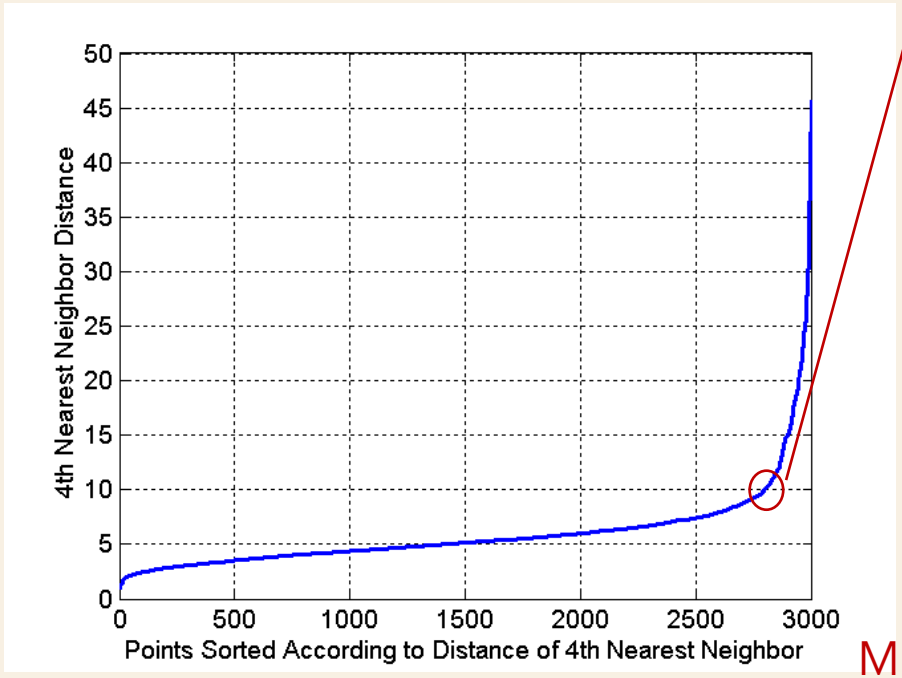
- There is an issue in determine the parameters *eps* and *MinPts*.
- IDEA: look at the distance from a point to its *k*-th nearest neighbor, which is called *k*-dist.
  - For points that belong to the same cluster, the value of *k*-dist will be small, if *k* is no larger than the cluster size.
  - For points that are not in a cluster, such as noise points, the *k*-dist will be relatively large.

SELECTION OF PARAMETERS

1. Compute the *k*-dist value for all points for some *k*.
2. Sort them in increasing order.
3. Plot the sorted values.
4. A sharp change will be seen at the value of *k*-dist that correspond as a suitable value for *eps*.
5. *eps* = *k*-dist for which a sharp change happens
   MinPts = *k*
6. The points for which the *k*-dist is less than *Eps* are labelled as core points, while the other points are labeled as noise or border points.

SELECTION OF PARAMETERS

Original dataset = 300 2D points

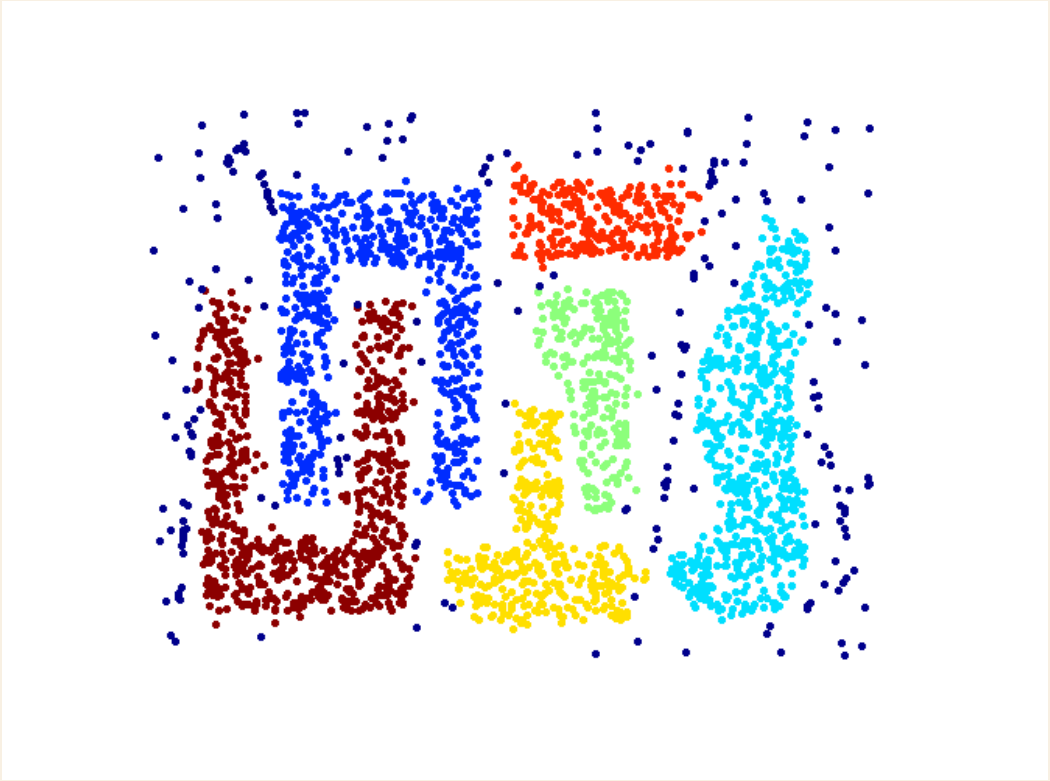


eps = 10

MinPts = 4

K = 4 (default for DBSCAN, good for 2D points)

SELECTION OF PARAMETERS



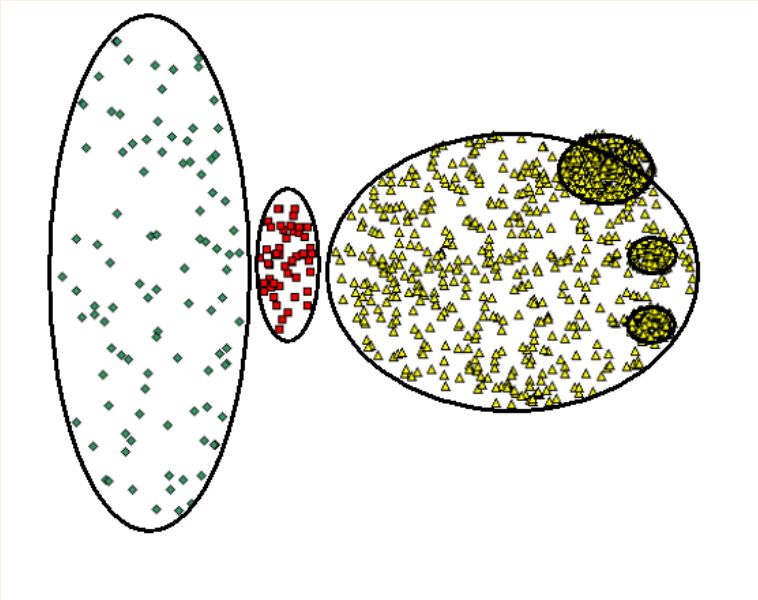
Clusters (dark blue points indicate noise)

DBSCAN works well!

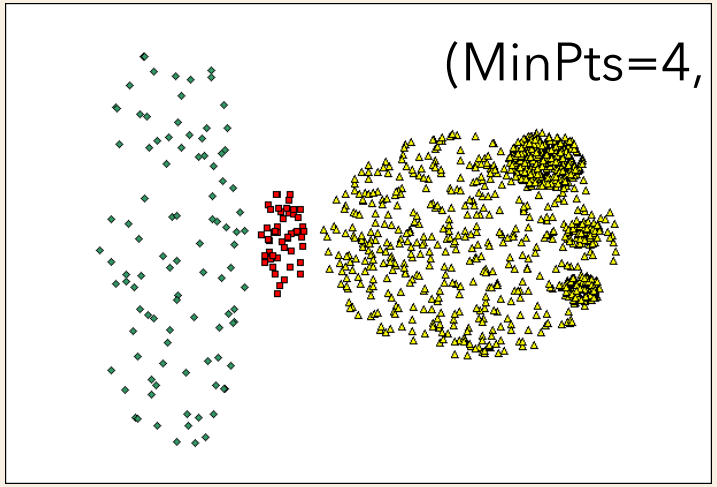
STRENGHTS AND WEAKNESSES

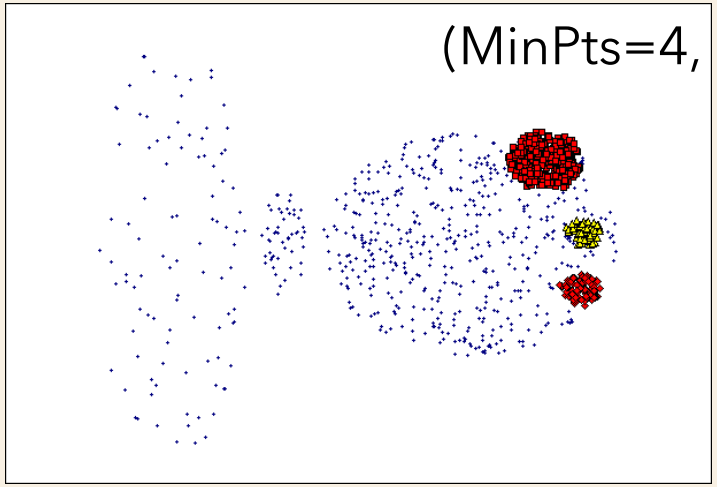
- It is relatively resistant to noise and can handle clusters of arbitrary shapes and sizes.
- It has some trouble when the clusters have widely varying densities.
- It has also trouble with high-dimensional data because density is more difficult to define.
- It could be expensive when the computation of nearest neighbors requires computing all pairwise proximities.

STRENGHTS AND WEAKNESSES



Original Points → clusters with different densities



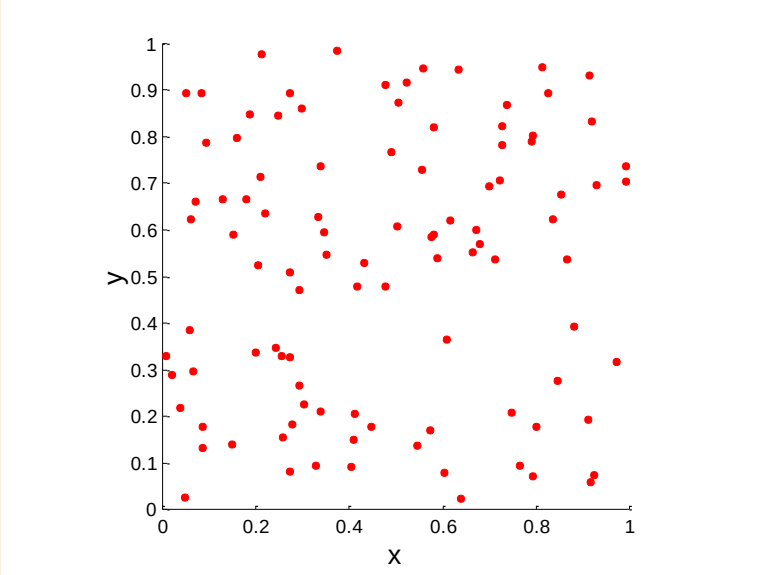


CLUSTER EVALUATION

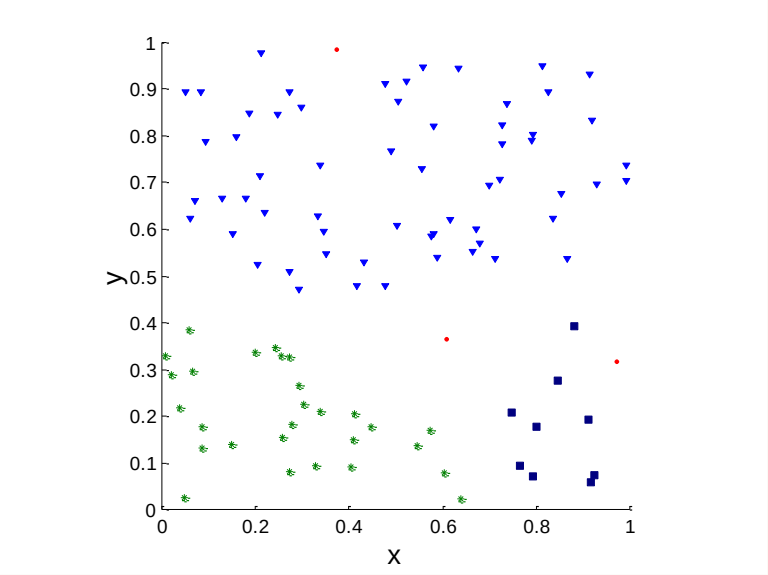
CLUSTER VALIDATION

- For supervised classification we have a variety of measures to evaluate how good our model is.
  - Accuracy, precision, recall
- Cluster evaluation or cluster validation the notion of goodness is not a well-developed or commonly used part of cluster analysis.
  - Clusters are in the eye of the beholder!
  - Almost every clustering algorithm will find clusters in a dataset, even if that data set has no natural cluster structure.

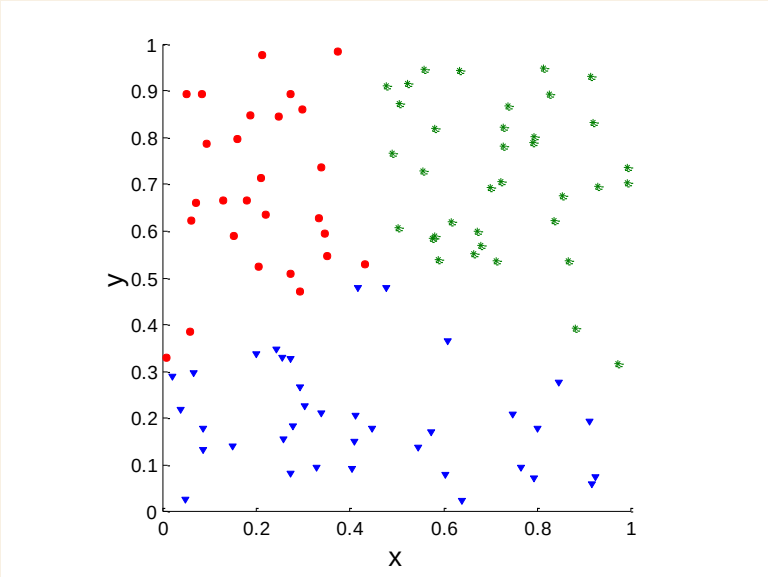
Random Points



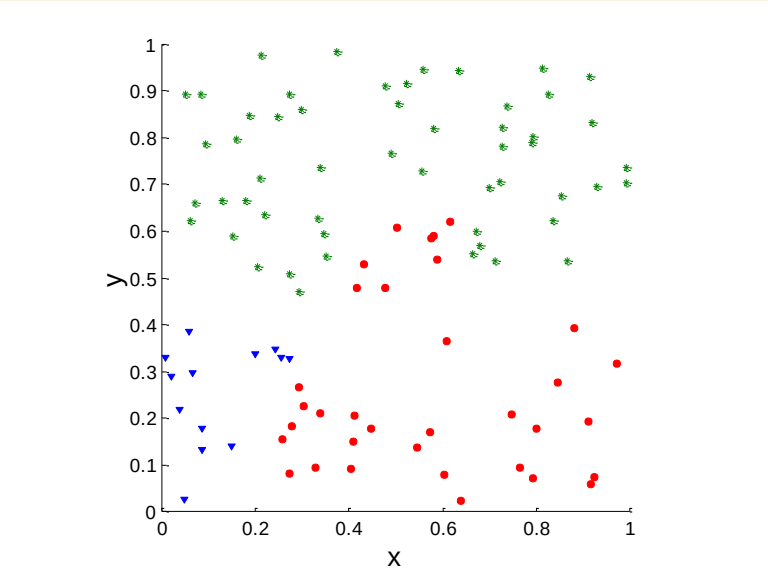
DBSCAN



K-means



Complete Link



CLUSTER VALIDATION

Issues in cluster validation:

1. Determining the clustering tendency of a set of data: distinguishing whether non-random structure actually exists in the data.
2. Determining the correct number of clusters.
3. Evaluating how well the results of a cluster analysis fit the data without reference to external information
4. Comparing the results of a cluster analysis to externally known results (e.g. externally provided labels).
5. Comparing two sets of clusters to determine which is better.

1, 2 and 3 are unsupervised techniques, while 4 requires external information and 5 can be done in both ways.

MEASURE OF CLUSTER VALIDITY

The evaluation measures that are used to judge various aspects of cluster validity are traditionally classified into the following three types:

- Unsupervised
- Supervised
- Relative

UNSUPERVISED MEASURES

- Unsupervised measures determines the goodness of a clustering structure without requiring external information.
  - Internal indexes.
  - E.g. = SSE
- Cluster cohesion (compactness): it determines how closely related are the objects in a cluster.
- Cluster separation (isolation): it determines how distinct or well separated a cluster is from the others.

SUPERVISED MEASURES

- Supervised measures determines the goodness of a clustering structure based on the extent to which such structure matches some external structure.
  - External indexes.
  - E.g. entropy = how well cluster labels match externally supplied labels.

RELATIVE MEASURES

- Relative measures compares different clusterings or clusters.
- They represent a specific use of supervised or unsupervised measure.
  - E.g. two K-means clustering can be compared using either SSE or entropy.

UNSUPERVISED EVALUATION

- The overall validity of clustering composed of K cluster is a weighted sum of the validity of individual clusters.
- $overall\ validity = \sum_{i=1}^{K} w_i\ validity(C_i)$
- The validity() function can be cohesion, separation or any combination of them.
- The weights will vary depending on the validity measure.
  - In the simplest case = 1.
  - It can depend on some properties of the cluster.

UNSUPERVISED EVALUATION

- $cohesion(C_i) = \sum_{\substack{x \in C_i \\ y \in C_i}} proximity(x, y)$
- $separation(C_i, C_j) = \sum_{\substack{x \in C_i \\ y \in C_j}} proximity(x, y)$
- The proximity() function can be a similarity or a dissimilarity.
  - For similarity higher values are better for cohesion while lower values are better for separation.
  - For dissimilarity the opposite is true.

UNSUPERVISED EVALUATION

- $cohesion(C_i) = \sum_{x \in C_i} proximity(x, c_i)$
- where c_i is the prototype (centroid) of the cluster C_i
- If we use the SSE as proximity() function, the overall cohesion of a clustering structure is measured by the within cluster sum of squares (SSE):
- $total\ cohesion = SSE = \sum_i \sum_{x \in C_i} (x - c_i)^2$

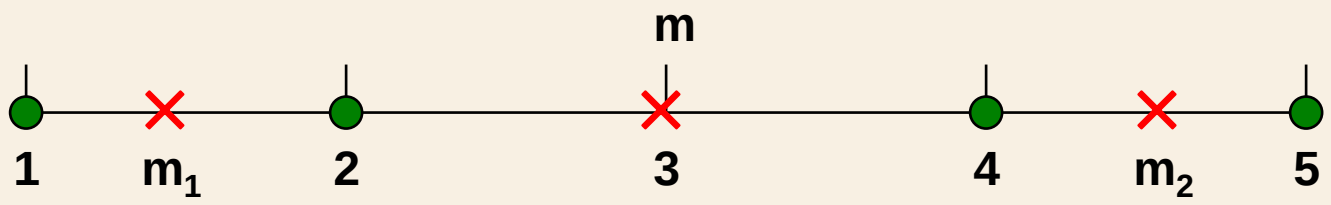
UNSUPERVISED EVALUATION

- $separation(C_i, C_j) = proximity(c_i, c_j)$
- The separation of cluster prototypes from one another is directly related to the separation of cluster prototypes from an overall prototype
- $separation(C_i) = proximity(c_i, c)$
- where c is the overall prototype (centroid)

PROTOTYPE-BASED VIEW OF CLUSTER – COHESION AND SEPARATION

- If we use the SSE as proximity() function, the overall separation of a clustering structure is measured by the between cluster sum of squares
- $total\ separation = SSB = \sum_i |C_i|(c - c_i)^2$
- where c_i is the prototype (centroid) of the cluster C_i and c is the overall prototype (centroid)

PROTOTYPE-BASED VIEW OF CLUSTER COHESION AND SEPARATION



K = 1 cluster

- $SSE = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$
- $SSB = 4 \times (3-3)^2 = 0$

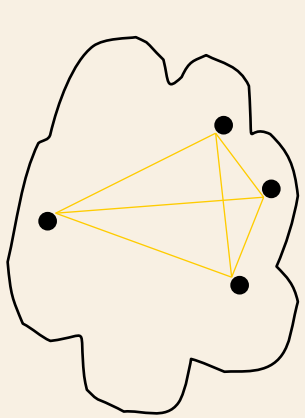
K = 2 clusters

- $SSE = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$
- $SSB = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$

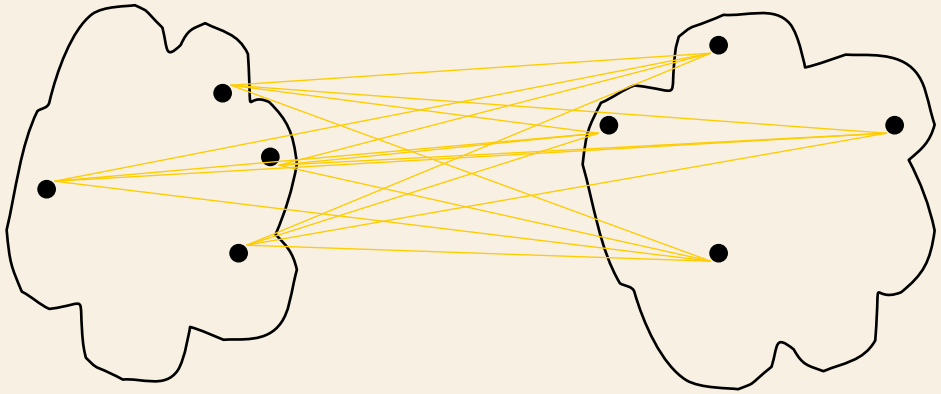
GRAPH-BASED VIEW OF CLUSTER COHESION AND SEPARATION

A proximity graph-based approach can also be used for cohesion and separation.

- Cluster cohesion is the sum of the weight of all links within a cluster.
- Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.



cohesion



separation

UNSUPERVISED EVALUATION: SILHOUETTE

- The Silhouette coefficient combines both cohesion and separation into a unique measure.
- It succinctly evaluate how well each object lies within its cluster.
- It is defined for single points and can be used to compute a silhouette measure for a cluster or for the entire clustering structure.

UNSUPERVISED EVALUATION: SILHOUETTE

- For each object *o*
- Calculate its average distance to all other objects in its cluster C_i: a(o)
  - Average dissimilarity of o with all the other objects in C_i
- Calculate for any other cluster C_j not containing *o*, the average distance between *o* and all the other objects in C_j. Find the minimum value: b(o)
  - Min (average dissimilarity of o with all the objects in C_j)
- Silhouette(o) = $\frac{b(o)-a(o)}{\max(a(o),b(o))}$

UNSUPERVISED EVALUATION: SILHOUETTE

- The Silhouette value ranges from -1 and +1.
- A negative value in undesirable because it corresponds to the case in which a(o) > b(o).
  - The average distance of the point in the cluster is greater than the minimum average distance between the points in the other clusters.
- The maximum value 1 is obtained for a(o) = 0.

UNSUPERVISED EVALUATION: SILHOUETTE

- Silhouette for a cluster: given the silhouette of the points, the average silhouette of a cluster C is obtained by taking the average of the silhouette coefficients for the points in C.
- Silhouette for a clustering structure: it is obtained by averaging the silhouette coefficients of all the points in the dataset.

SUPERVISED EVALUATION

- External available information is typically a class label for the data objects.
- The idea is to measure the degree of correspondence between the cluster labels and the class labels.
- Why?
  - Comparison of a clustering technique with a ground truth.
  - Evaluate the extent of which a manual classification process can be automatically produced
  - Evaluate whether objects in the same cluster tend to have the same label for semi-supervised learning techniques.

SUPERVISED EVALUATION

Two kinds of supervised evaluation approaches:

- Classification oriented:
  - They use measure from classification, such as entropy, purity and F-measure.
  - They evaluate the extent to which a cluster contains objects of a single class.
- Similarity oriented:
  - They use similarity measures, like the Jaccard measure.
  - They measure the extent to which two objects that are in the same class are in the same cluster and vice-versa.

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED

- We measure the degree to which predicted class labels (= cluster labels) correspond to the actual class.
- Entropy: the degree to which each cluster consists of objects of a single class.
  - For each cluster C_i, we compute p_{ij}, the probability that a member of C_i belongs to the class j, as $p_{ij} = m_{ij}/m_i$, where m_i is the number of objects in C_i and m_{ij} is the number of objects of class j in C_i.
  - Entropy(C_i) = $e_i = -\sum_{j=1}^{L} p_{ij} log_2 p_{ij}$ where L is the number of classes
  - Total entropy = e = $\sum_{i=1}^{K} \frac{m_i}{m} e_i$ where K is the number of clusters

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED

- Purity: it measures the extent to which a cluster contains objects of a single class.
  - purity(C_i) = $max_j\ p_{ij}$
  - total purity = $\sum_{i=1}^{K} \frac{m_i}{m} purity(C_i)$ where K is the number of clusters

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED

- Precision: it measures the fraction of a cluster that consists of objects of a specified class.
  - precision(C_i,j) = p_{ij} precision of a cluster C_i w.r.t class j
- Recall: it measures the extent to which a cluster consists all objects of a specified class.
  - recall(C_i,j) = $\frac{m_{ij}}{m_j}$ where m_{ij} is the number of objects in C_i with class j and m_j is the total number of objects of class j

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED

- F-measure: a combination of precision and recall.
- It measures the extent to which a cluster contains *only* objects of a particular class and *all* objects of that class.
  - $F(C_i,j) = \frac{2 \times precision(C_i,j) \times recall(C_i,j)}{precision(C_i,j) + recall(C_i,j)}$

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED (EXAMPLE)

- Cluster 3204 newspaper articles using K-means

6 classes

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED (EXAMPLE)

- Ideally each cluster will contain documents from the same class.
- In reality, each cluster contains documents from many classes!
- Many clusters contain documents primarly from just one class.

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

Exceptionally good cluster

SUPERVISED CLASSIFICATION – CLASSIFICATION ORIENTED (EXAMPLE)

- Precision (C1, Metro) = 506/(3+5+40+506+96+27) = 506/677 = 0.75
- Recall(C1, Metro) = 506/943 = 0.26
- F(C1, Metro) = 0.39

- Precision(C3, Sport) = 671/(1+1+1+7+4+671) = 671/685 = 0.98
- Recall(C3, Sport) = 671/738 = 0.91
- F(C3, Sport) = 0.96

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED

- Idea: any two objects that are in the same cluster should be in the same class and vice versa.
- Rand index = $\frac{f_{00}+f_{11}}{f_{00}+f_{01}+f_{10}+f_{11}}$
  - f_{00} = number of pairs of objects having a different class and a different cluster
  - f_{01} = number of pairs of objects having a different class and the same cluster
  - f_{10} = number of pairs of objects having the same class and a different cluster
  - f_{11} = number of pairs of objects having the same class and the same cluster

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED

- Two-way contingency table

	Same Cluster	**Different Cluster**
Same Class	f_{11}	f_{10}
Different Class	f_{01}	f_{00}

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED (EXAMPLE)

- Five data points p1, …, p5
- Two clusters C1 = {p1, p2, p3} and C2 = {p4, p5}
- Two classes L1 = {p1, p2} and L2 = {p3, p4, p5}

Point	p1	p2	p3	p4	p5
p1	1	1	1	0	0
p2	1	1	1	0	0
p3	1	1	1	0	0
p4	0	0	0	1	1
p5	0	0	0	1	1

Ideal cluster similarity matrix = 1 in the ij-th entry if the two objects i and j are in the same cluster, 0 otherwise

Point	p1	p2	p3	p4	p5
p1	1	1	0	0	0
p2	1	1	0	0	0
p3	0	0	1	1	1
p4	0	0	1	1	1
p5	0	0	1	1	1

Class similarity matrix → defined based on the class labels, 1 in the ij-th entry if two objects i and j belong to the same class, 0 otherwise

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED (EXAMPLE)

- f00 = 4 → pairs with zero in both matrices

Point	p1	p2	p3	p4	p5
p1	1	1	1	0	0
p2	1	1	1	0	0
p3	1	1	1	0	0
p4	0	0	0	1	1
p5	0	0	0	1	1

Ideal cluster similarity matrix

Point	p1	p2	p3	p4	p5
p1	1	1	0	0	0
p2	1	1	0	0	0
p3	0	0	1	1	1
p4	0	0	1	1	1
p5	0	0	1	1	1

Class similarity matrix

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED (EXAMPLE)

- f01 = 2 → pairs with 1 in the cluster and 0 in class similarity

Point	p1	p2	p3	p4	p5
p1	1	1	1	0	0
p2	1	1	1	0	0
p3	1	1	1	0	0
p4	0	0	0	1	1
p5	0	0	0	1	1

Ideal cluster similarity matrix

Point	p1	p2	p3	p4	p5
p1	1	1	0	0	0
p2	1	1	0	0	0
p3	0	0	1	1	1
p4	0	0	1	1	1
p5	0	0	1	1	1

Class similarity matrix

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED (EXAMPLE)

- f10 = 2 → pairs with 0 in the cluster and 1 in class similarity

Point	p1	p2	p3	p4	p5
p1	1	1	1	0	0
p2	1	1	1	0	0
p3	1	1	1	0	0
p4	0	0	0	1	1
p5	0	0	0	1	1

Ideal cluster similarity matrix

Point	p1	p2	p3	p4	p5
p1	1	1	0	0	0
p2	1	1	0	0	0
p3	0	0	1	1	1
p4	0	0	1	1	1
p5	0	0	1	1	1

Class similarity matrix

SUPERVISED CLASSIFICATION – SIMILARITY ORIENTED (EXAMPLE)

- f11 = 4 → pairs with 1 in the cluster and 1 in class similarity

Point	p1	p2	p3	p4	p5
p1	1	1	1	0	0
p2	1	1	1	0	0
p3	1	1	1	0	0
p4	0	0	0	1	1
p5	0	0	0	1	1

Ideal cluster similarity matrix

Point	p1	p2	p3	p4	p5
p1	1	1	0	0	0
p2	1	1	0	0	0
p3	0	0	1	1	1
p4	0	0	1	1	1
p5	0	0	1	1	1

Class similarity matrix

Rand index = (f00+f11)/(f00+f01+f10+f11) = (4+4)/12 = 0.67

FINAL COMMENT ON CLUSTER VALIDITY

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis. Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Algorithms for Clustering Data, Jain and Dubes