Clustering (Cluster Analysis)

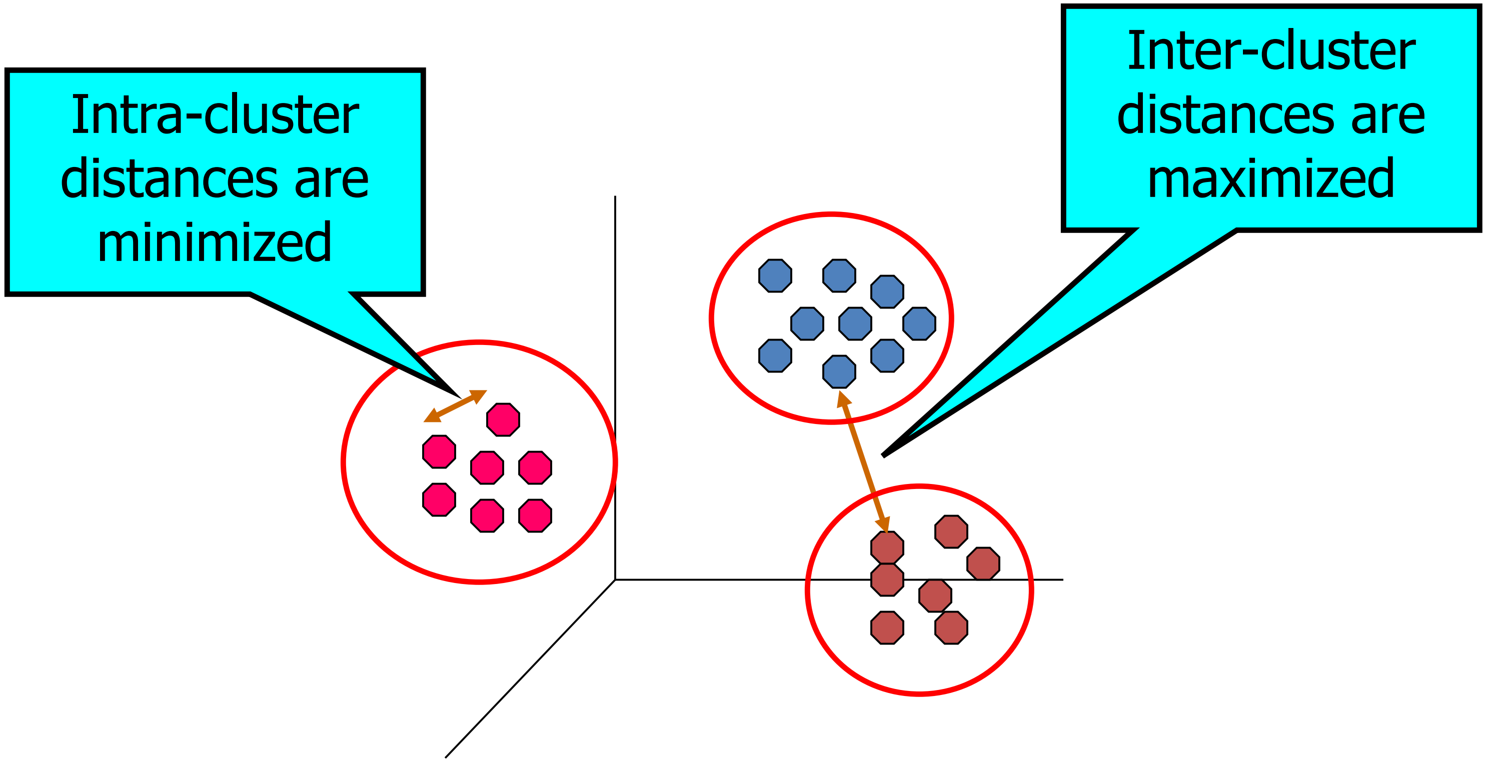
Dr. David C. Anastasiu
San José State University

Outline

- Overview of clustering
- Basic clustering algorithms:
  - K-Means clustering, hierarchical clustering, and density-based clustering
- Other clustering algorithms:
  - Graph-based clustering, topic modeling
- Cluster validity
- Scaling up clustering algorithms
- Choosing the right clustering algorithm

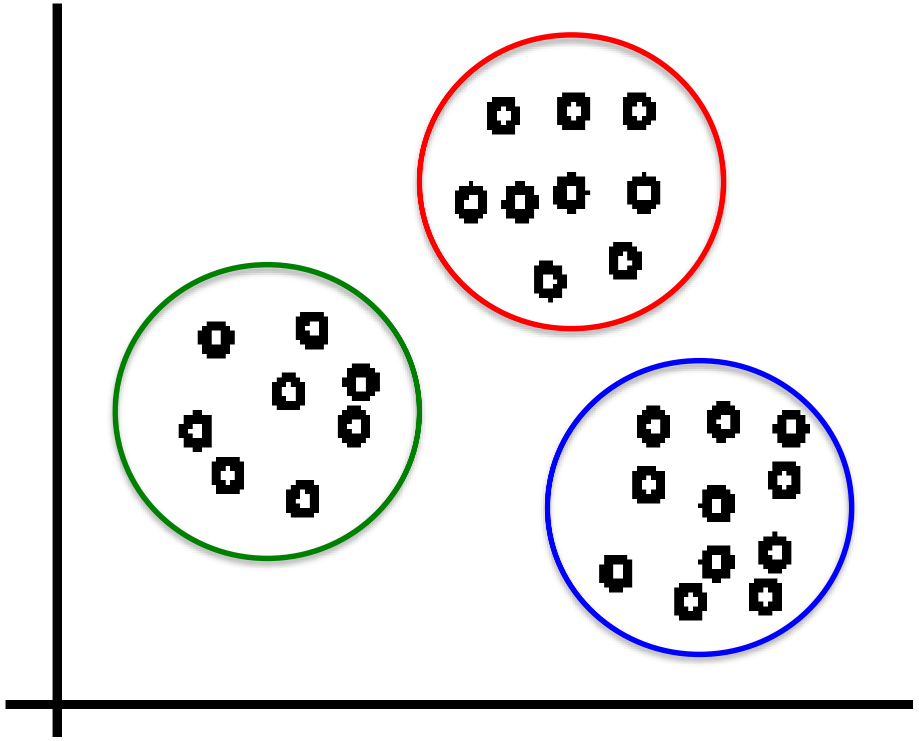
What is cluster analysis?

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


Intra-cluster distances are minimized
Inter-cluster distances are maximized

An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



Slide adapted from slides of B. Liu for “Web Data Mining”

What is clustering for?

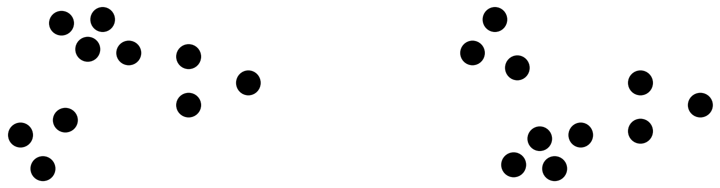
- Example : In marketing, segment customers according to their similarities
  - To do targeted marketing.

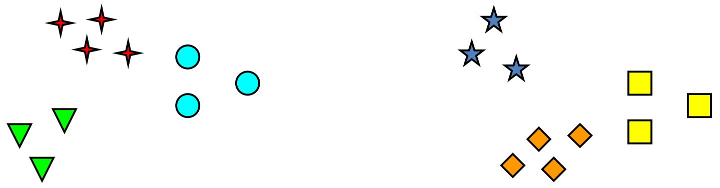
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.
  - Clustering users (e.g. of social networks) is also a very important task

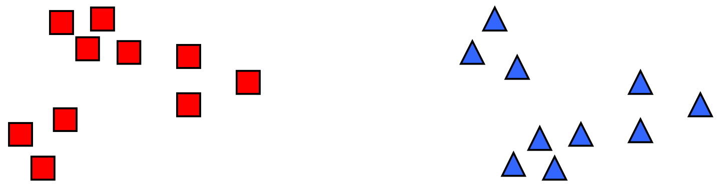
What is not cluster analysis?

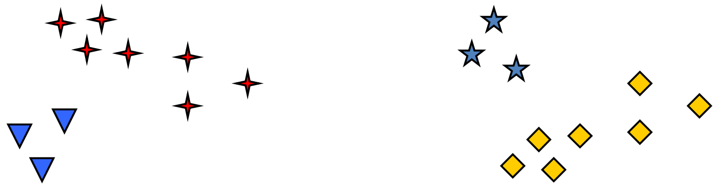
- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name.
- Results of a query
  - Groupings are a result of an external specification.
  - Clustering is a grouping of objects based on the data.
- Supervised classification
  - Have class label information.
- Association analysis
  - Local vs. global connections.

Notion of a cluster can be ambiguous


How many clusters?


Six Clusters


Two Clusters


Four Clusters

Clustering formulations

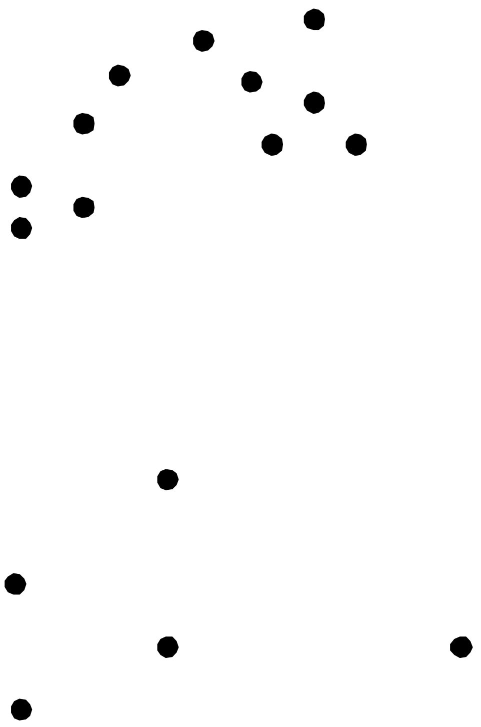
A number of clustering formulations have been developed:

1. We need to find a fixed number of clusters.
   – Well-suited for compression-like applications
2. We need to find clusters of fixed size.
   – Well-suited for neighborhood-discovery (recommendation engine).
3. We need to find the smallest number of clusters that satisfy certain quality criteria.
   – Well-suited for applications in which cluster quality is important
4. We need to find the *natural* number of clusters.
   – This is clustering's holly-grail!
      - Extremely hard, problem dependent, and “quite supervised”.

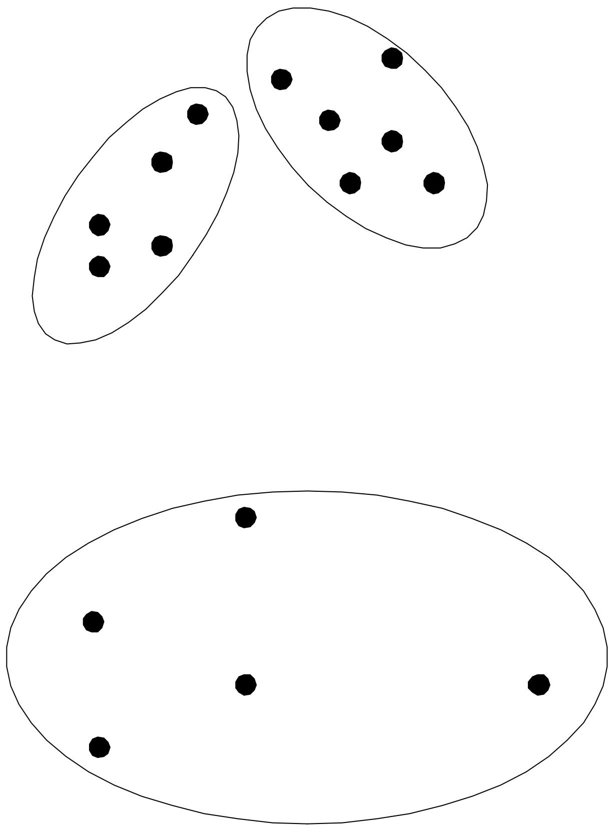
Types of clusterings

- A clustering is a set of clusters.
- Important distinction between hierarchical and partitional sets of clusters.
- Partitional clustering
  - A division of data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset.
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree.

Partitional clustering

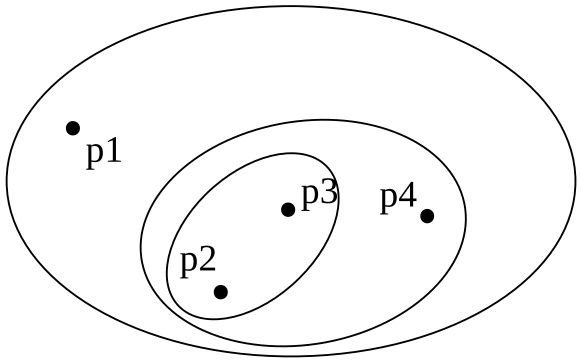


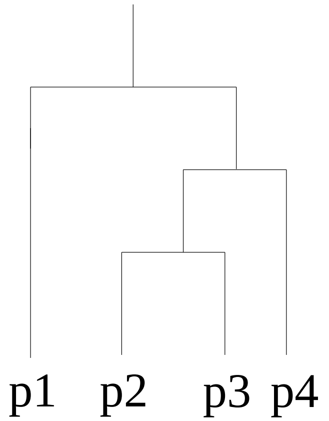
Original Points



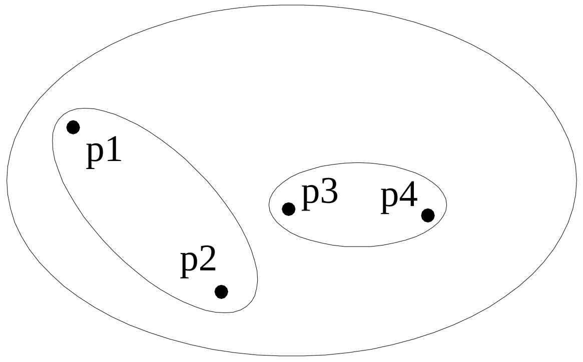
A Partitional Clustering

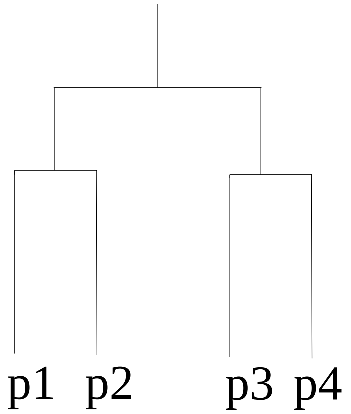
Hierarchical clustering





Traditional Hierarchical Clustering **Traditional Dendrogram**





Non-traditional Hierarchical Clustering **Non-traditional Dendrogram**

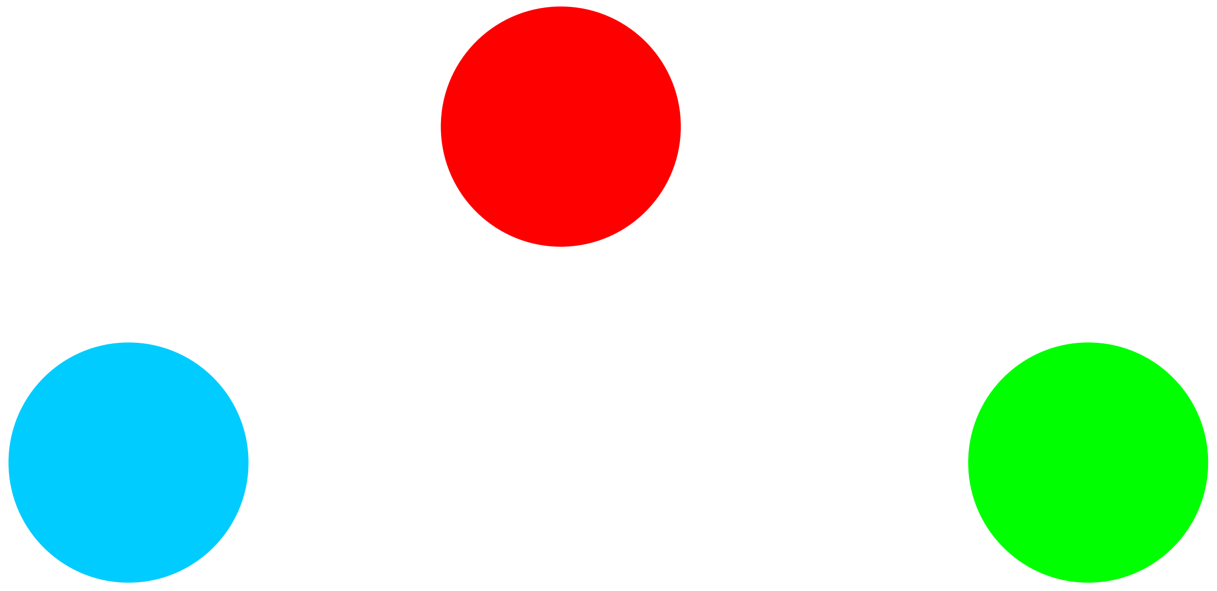
Other distinctions between sets of clusters

- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or “border” points.
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1.
  - Weights must sum to 1.
  - Probabilistic clustering has similar characteristics.
- Partial versus complete
  - In some cases, we only want to cluster some of the data.
- Heterogeneous versus homogeneous
  - Clusters of widely different sizes, shapes, and densities.

Types of clusters: Well-separated

Well-Separated Clusters:

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

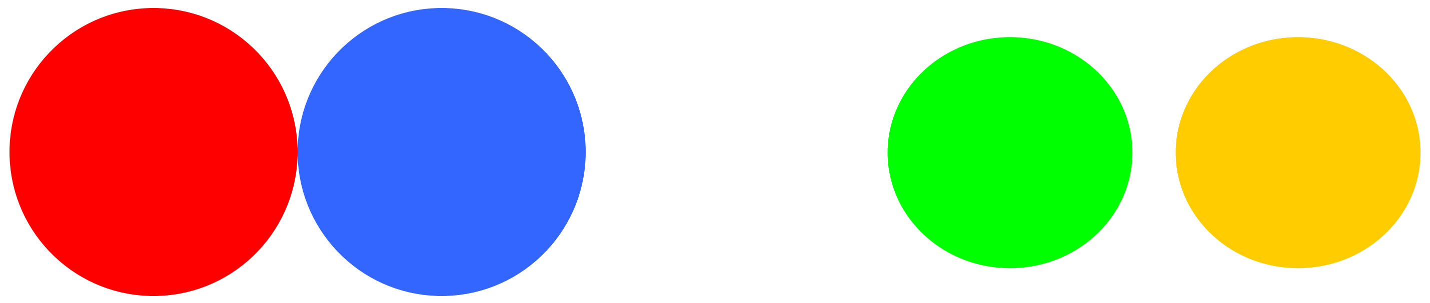


3 well-separated clusters

Types of clusters: Center-based

Center-based

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

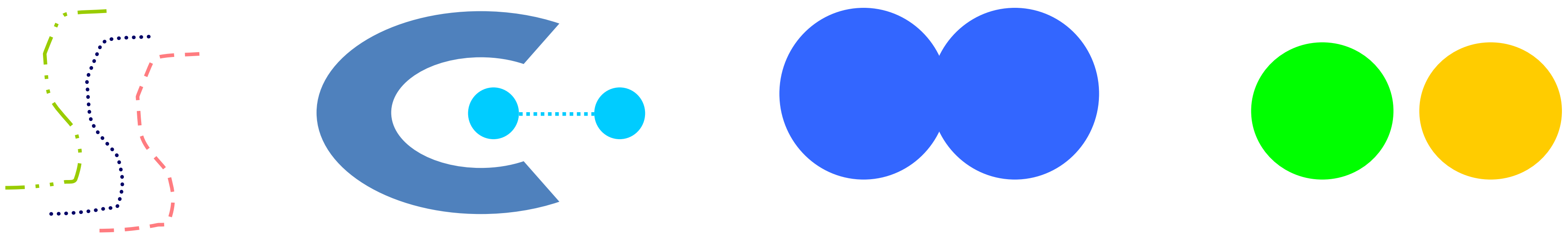


4 center-based clusters

Types of clusters: Contiguity-based

Contiguous cluster (nearest neighbor or transitive)

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

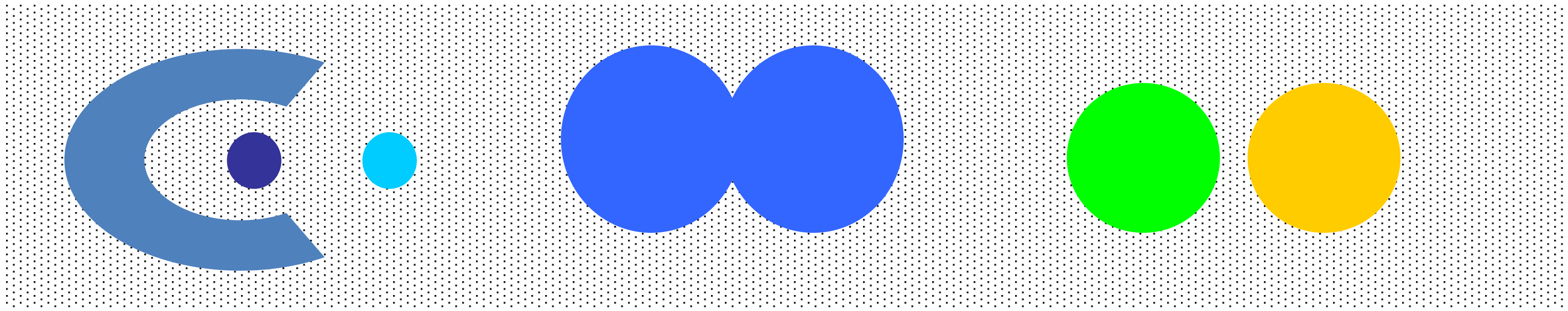


8 contiguous clusters

Types of clusters: Density-based

Density-based

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



6 density-based clusters

Clustering requirements

The fundamental requirement for clustering is the availability of a function to determine the *similarity* or *distance* between objects in the database.

The user must be able to answer some of the following questions:

1. When should two objects belong to the same cluster?
2. How should the clusters look like *(i.e.,* what type of objects should the contain*) ?*
3. What are the object-related characteristics of good clusters?

Clustering the similarity space

Map the clustering problem to a different domain and solve a related problem in that domain

- Proximity matrix defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points.
- Clustering is equivalent to breaking the graph into connected components, one for each cluster.
- Want to minimize the edge weight between clusters and maximize the edge weight within clusters.

1. K-means
2. Hierarchical clustering
3. Density-based clustering

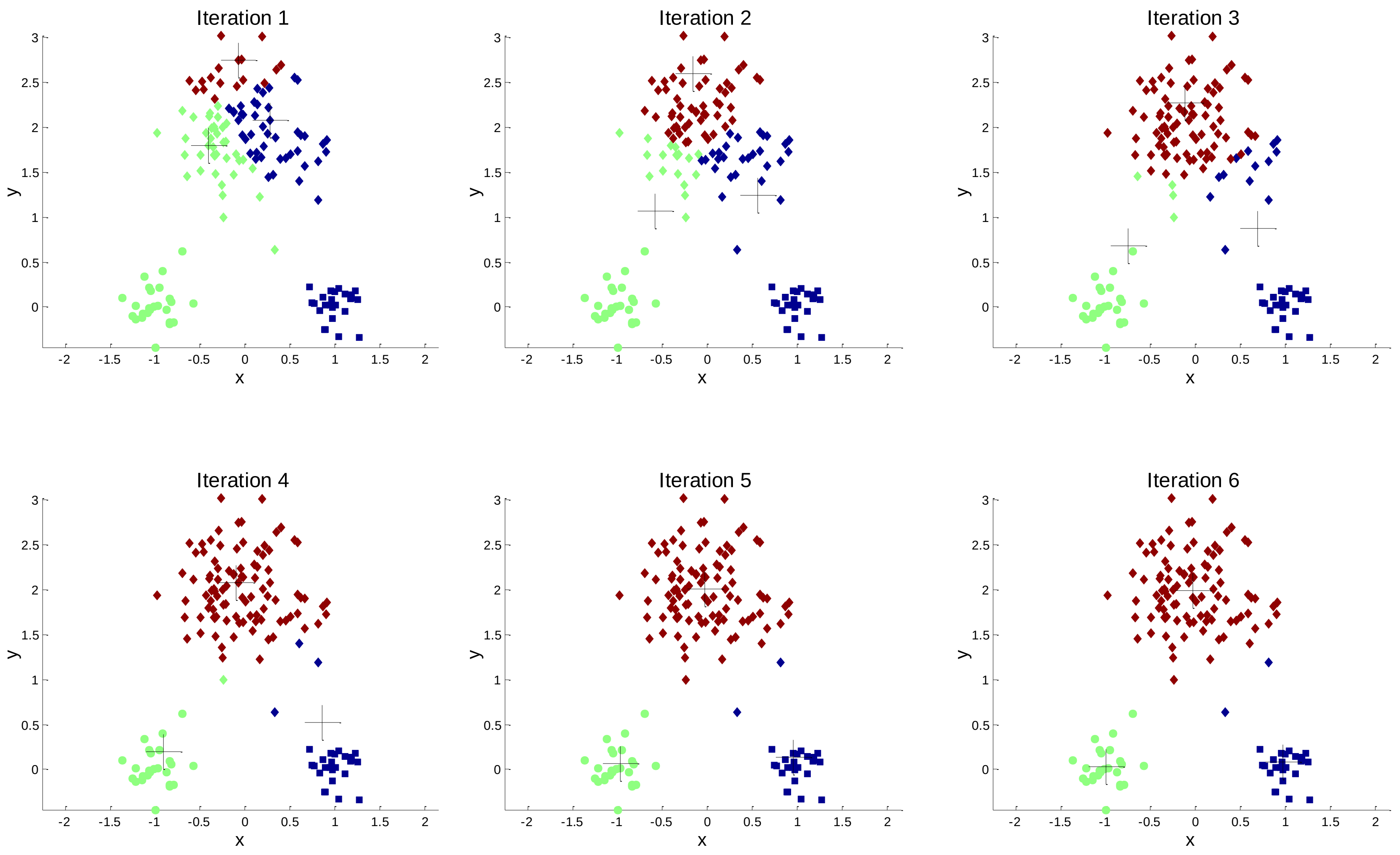
BASIC CLUSTERING ALGORITHMS

K-means clustering

- Partitional clustering approach.
- Number of clusters, K, must be specified.
- Each cluster is associated with a centroid (center point/object).
- Each point is assigned to the cluster with the closest centroid.
- The basic algorithm is very simple.

```
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 do not change.
```

Example of K-means clustering



K-means clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- “Closeness” is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to “Until relatively few points change clusters”.
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes.

K-medoids

- Partitional clustering approach.
- Number of clusters, K, must be specified.
- Each cluster is associated with a medoid (most central object in the cluster).
- Each point is assigned to the cluster with the closest medoid.
- The basic algorithm is very similar to K-means.

```
1: Select K points as the initial medoids.
2: repeat
3:     Form K clusters by assigning all points to the closest medoid.
4:     Assign new medoids as the most central point in each cluster.
5: until The medoids do not change.
```

K-means clustering – Objective

$$\min_{p} \sum_{i=1}^{n} \|d_i - d_{p_i}\|_2^2$$

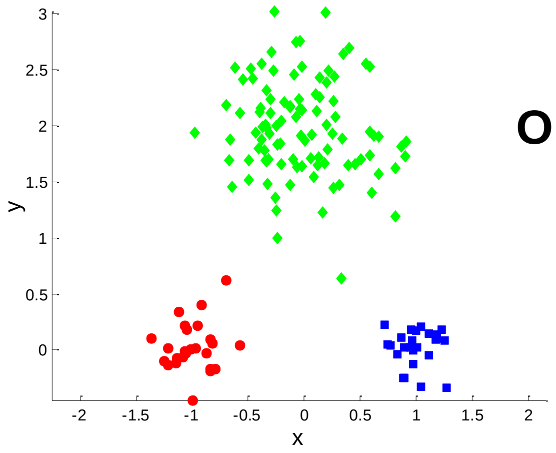
$$\min_{p, c_1, \ldots, c_K} \sum_{i=1}^{n} \|d_i - c_{p_i}\|_2^2$$

These are non-convex optimization problems.

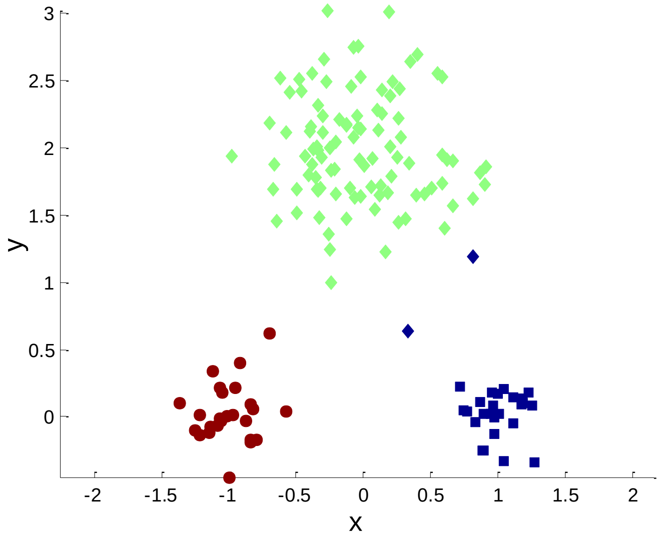
```
1: Select K points as the initial centroids.
2: repeat
3:     Form K clusters by assigning all points to the closest centroid.
4:     Recompute the centroid of each cluster.
5: until The centroids do not change.
```

- The K-means clustering algorithm is a way of solving the optimization problem.
- It uses an iterative alternate least squares optimization strategy.
  a. Optimize cluster assignments (p) given c_{p_i}'s.
  b. Optimizec_{p_i}'s given cluster assignments.
- It guarantees convergence to a local minima solution. However, due to the non-convexity of the problem, it may not be the global minimum.
  - Run K-means multiple times with different initial centroids and return the solution that has the best value.

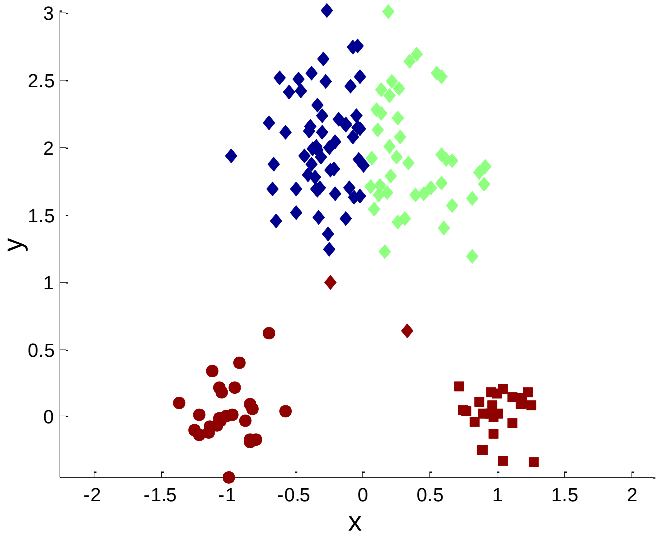
Two different K-means clusterings



Original Points



Optimal Clustering



Sub-optimal Clustering

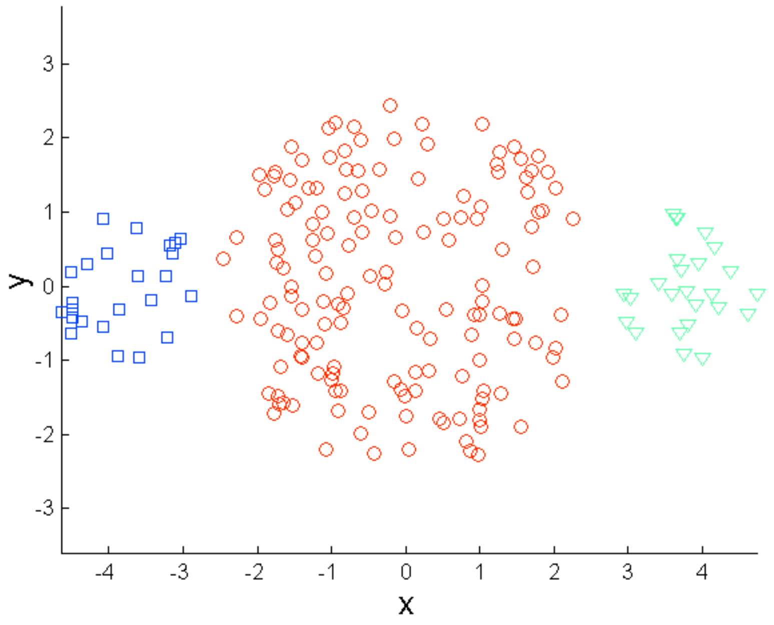
Limitations of K-means

- Def. *problem*: when the clustering solution that you get is not the best, natural, insightful, etc.

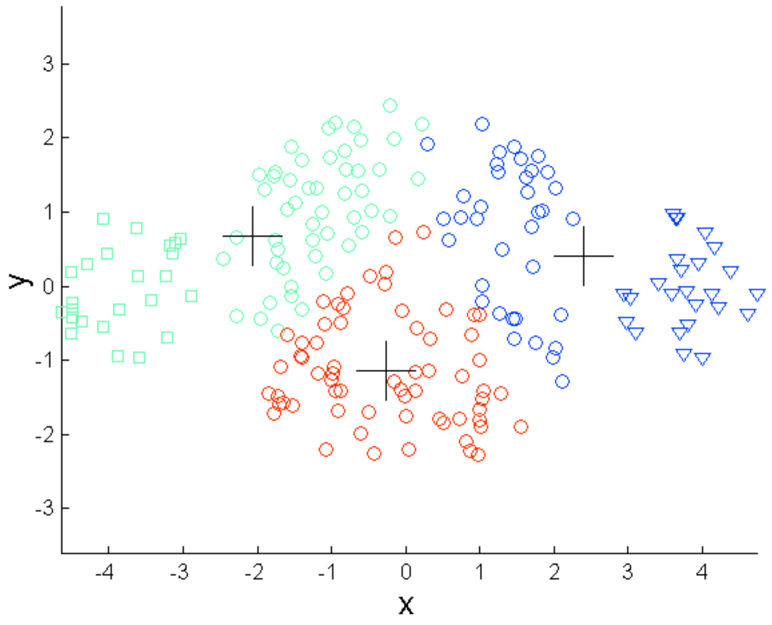
- K-means has *problems* when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has *problems* when the data contains outliers.

Limitations of K-means: Differing sizes

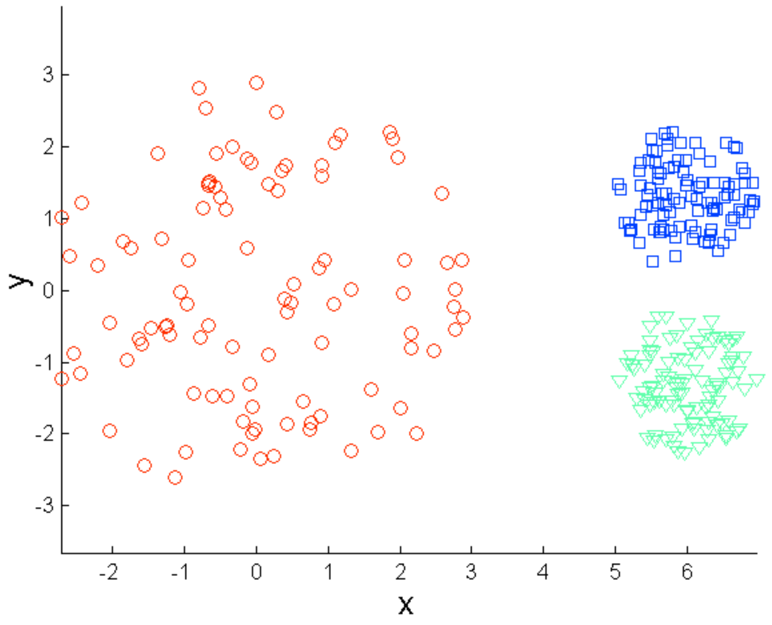


Original Points

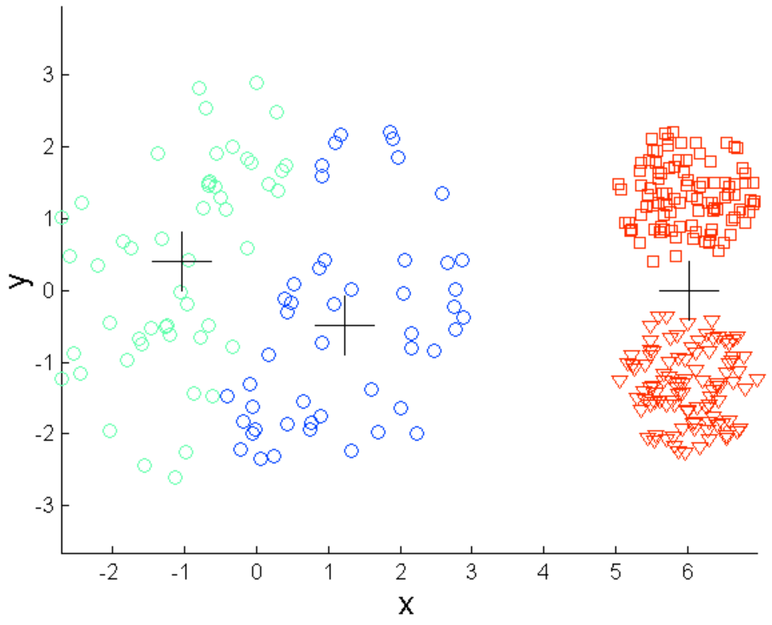


K-means (3 Clusters)

Limitations of K-means: Differing density

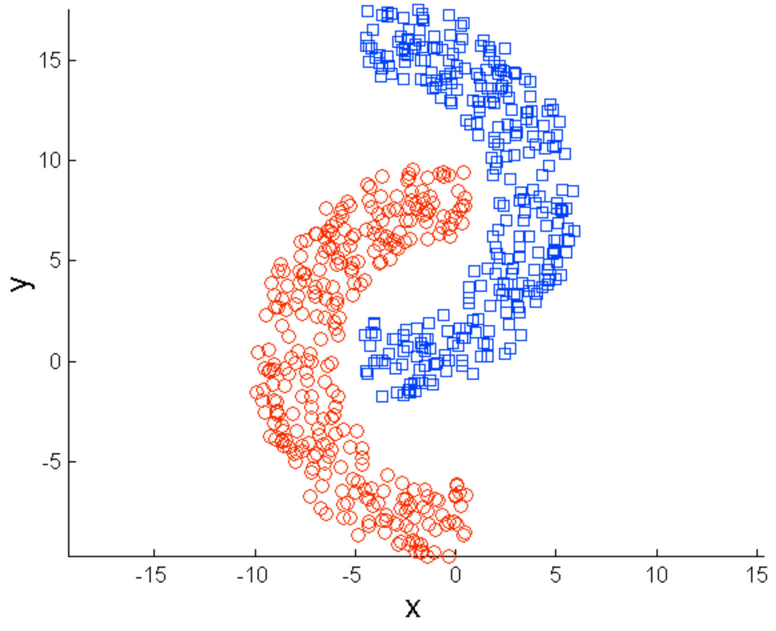


Original Points

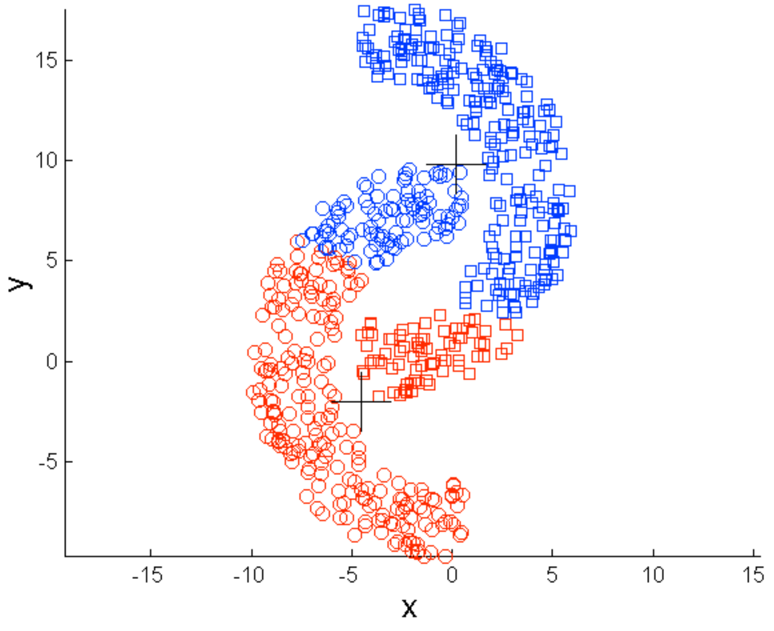


K-means (3 Clusters)

Limitations of K-means: Non-globular shapes

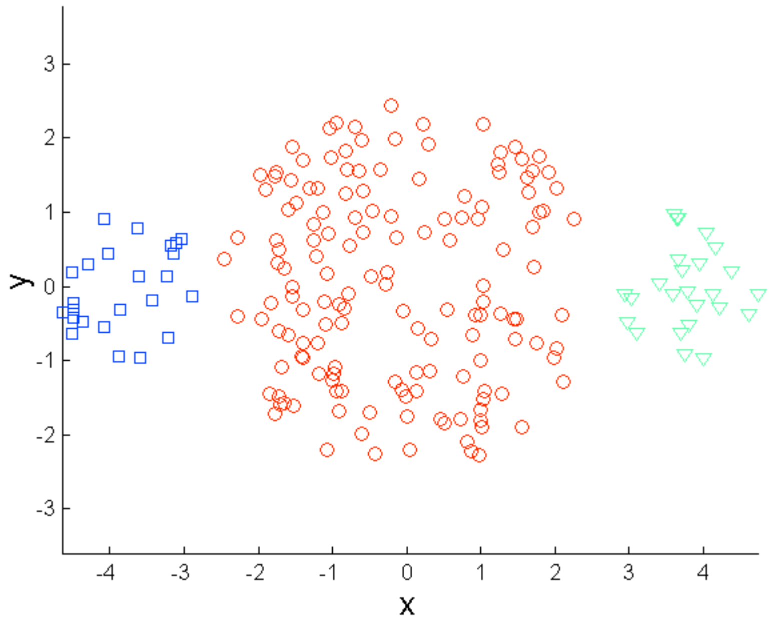


Original Points

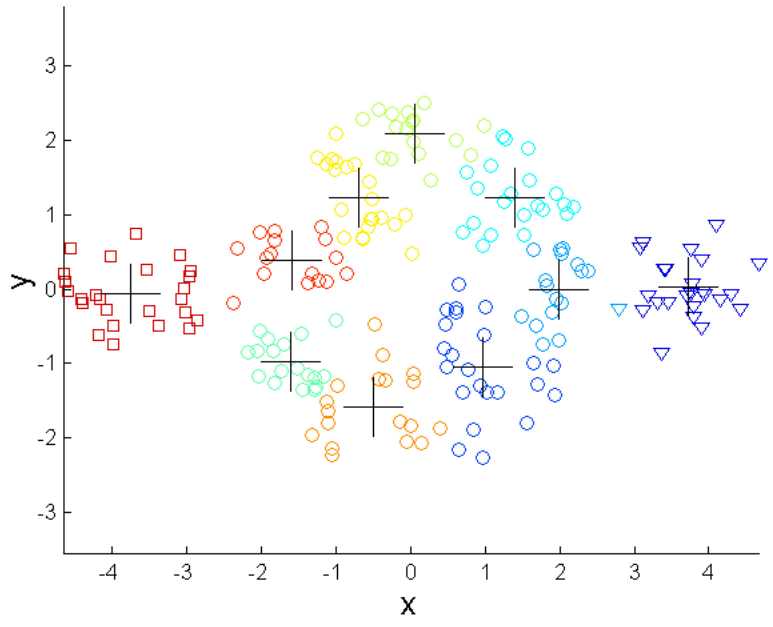


K-means (2 Clusters)

Overcoming K-means Limitations



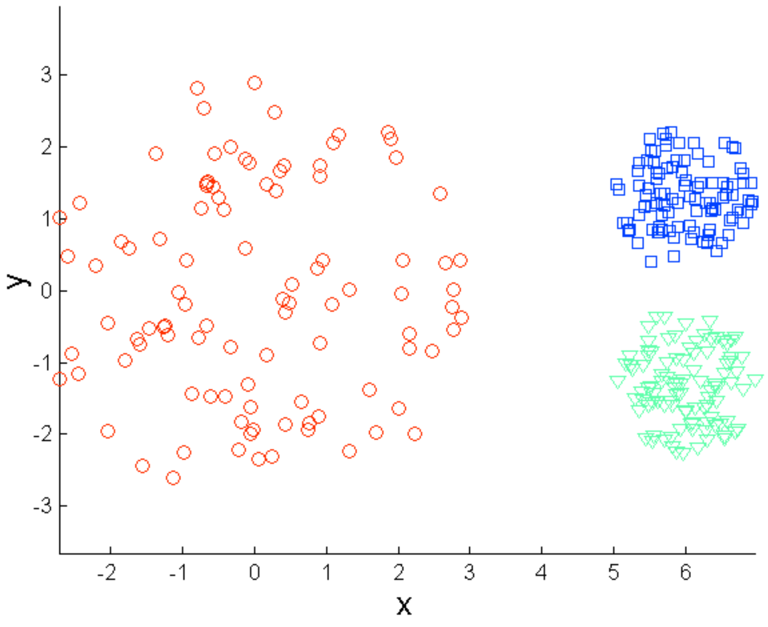
Original Points



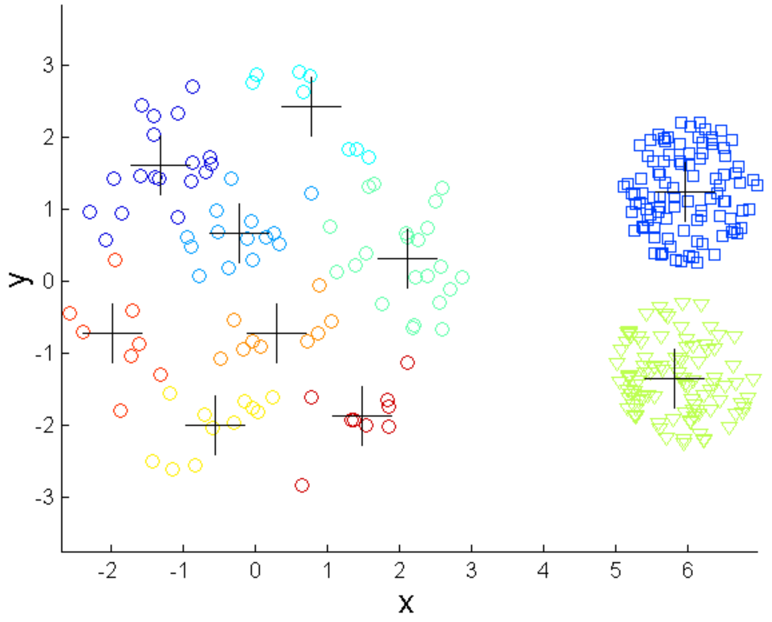
K-means Clusters

One solution is to use many clusters.
Finds parts of clusters, and we may need to put them back together.

Overcoming K-means limitations

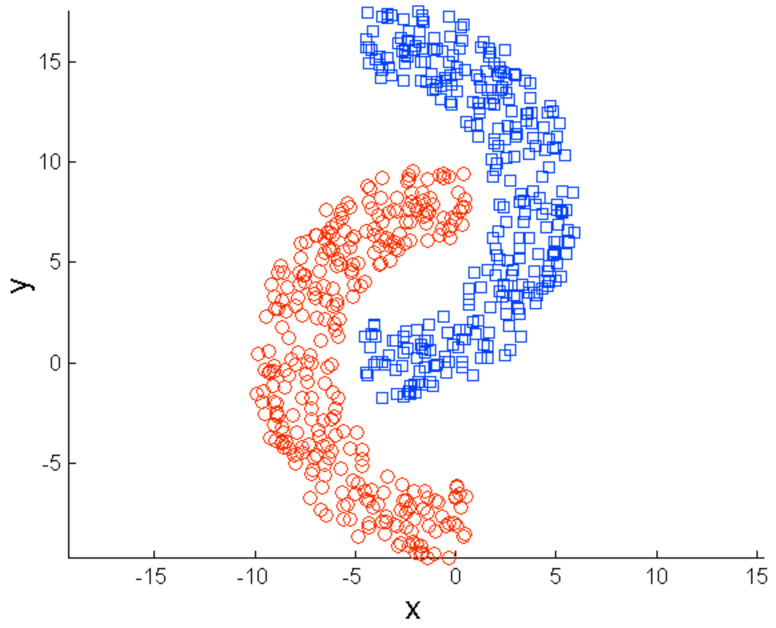


Original Points

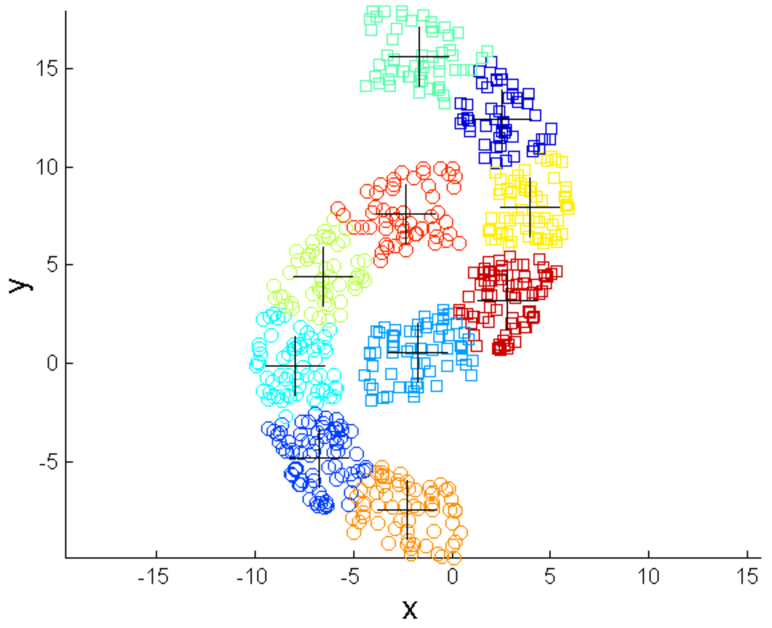


K-means Clusters

Overcoming K-means limitations

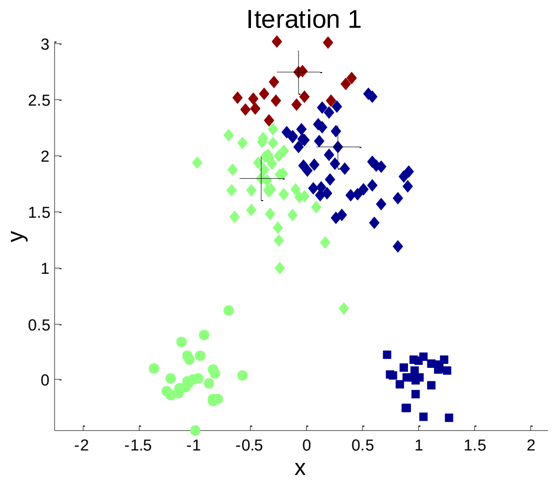


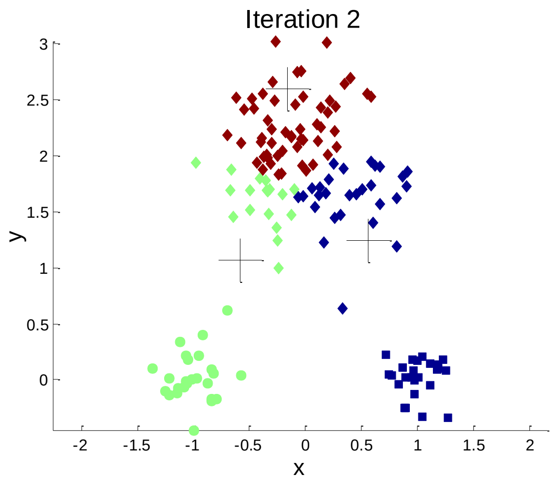
Original Points

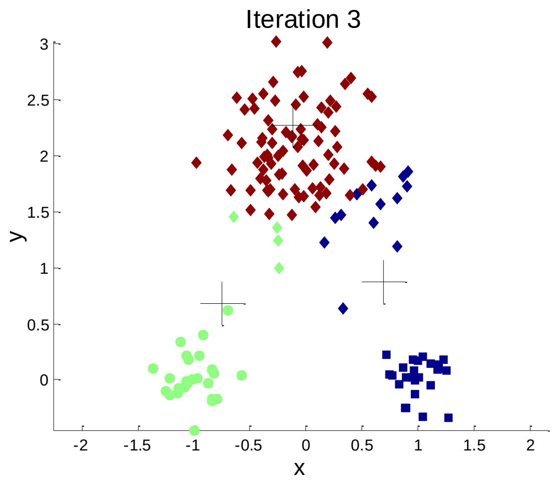


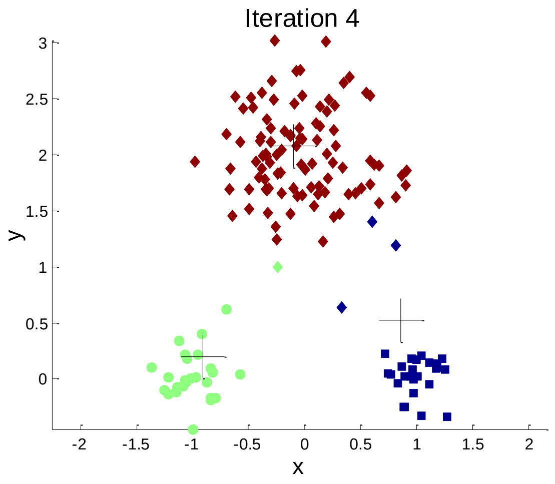
K-means Clusters

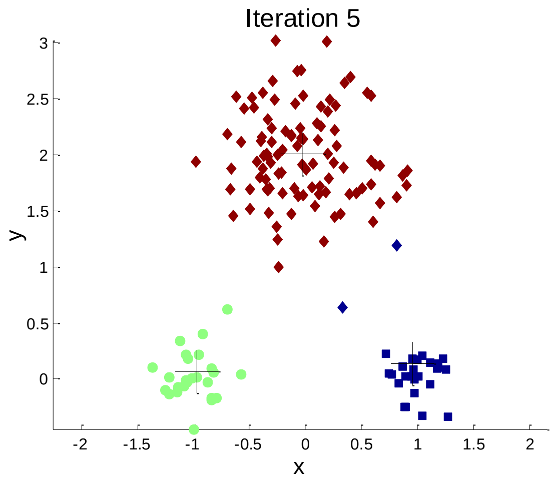
Importance of choosing initial centroids

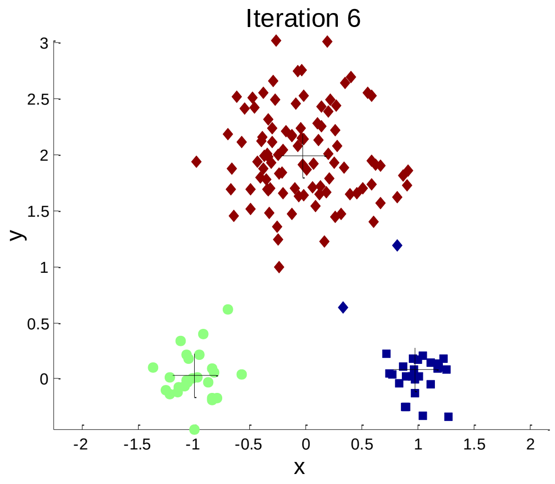




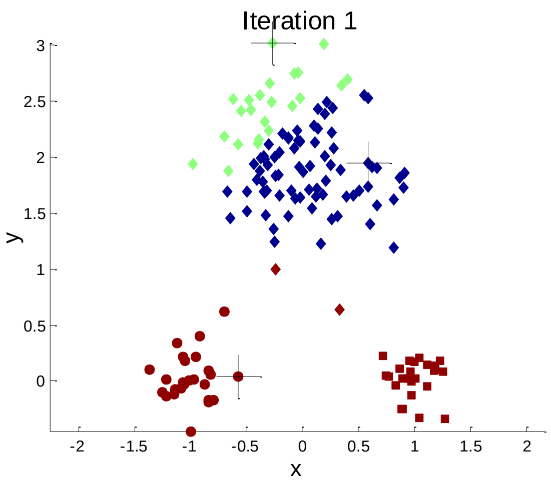


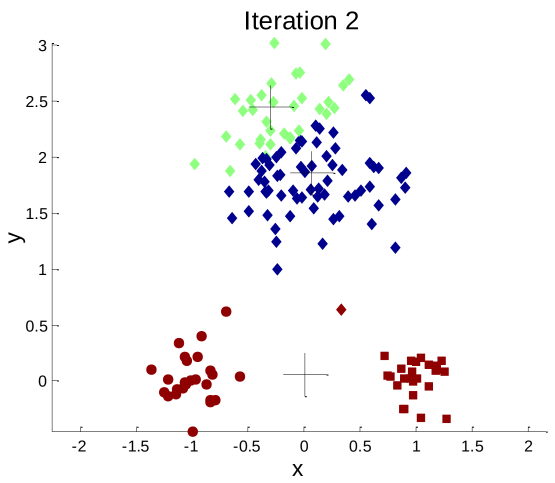


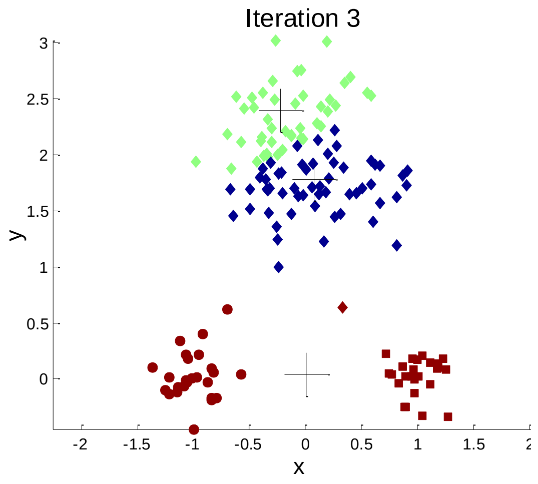


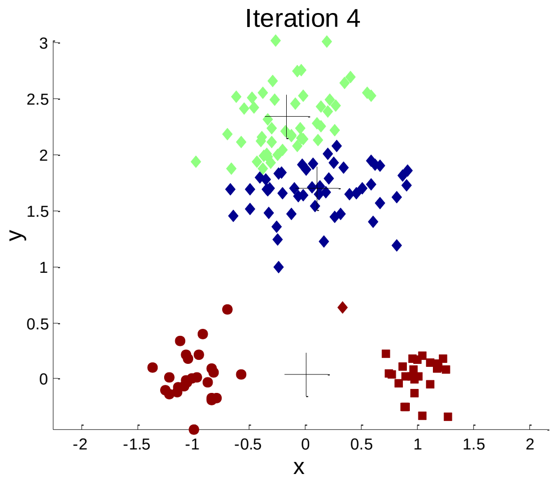


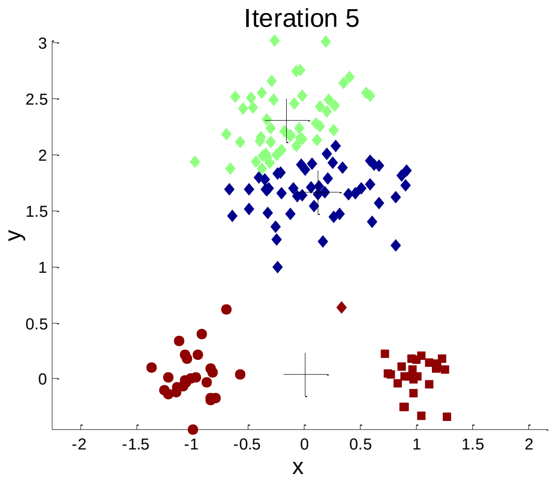
Importance of choosing initial centroids …











Problems with selecting initial points

If there are K “real” clusters then the chance of selecting one centroid from each cluster is small.

- Chance is relatively small when K is large.
- If clusters are of the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

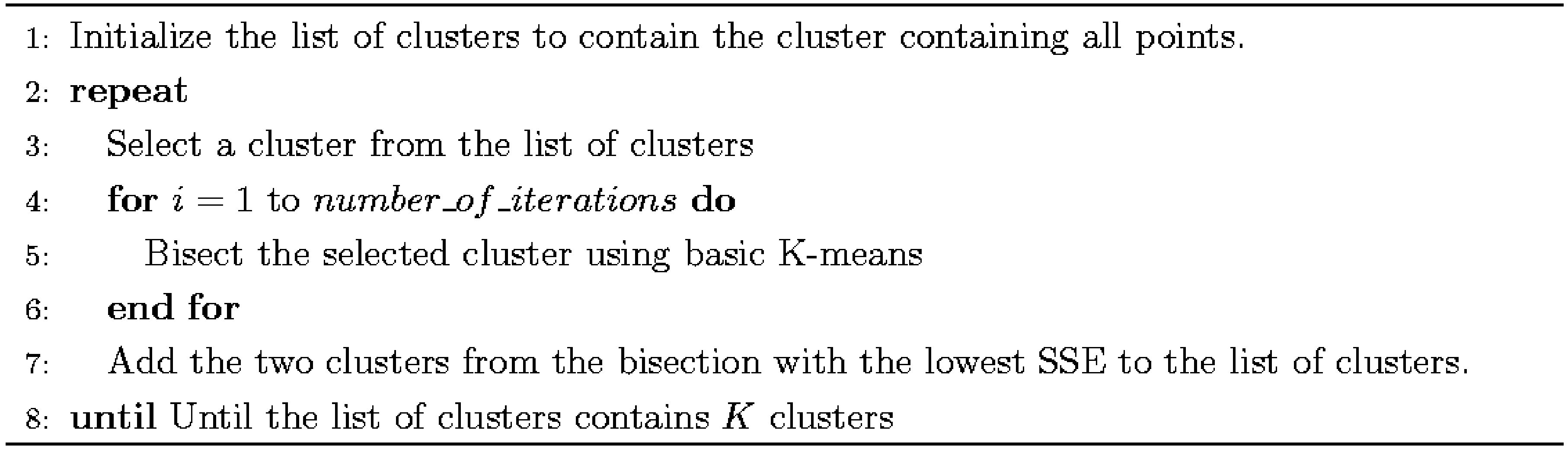
- For example, if K = 10, then probability = $10!/10^{10}$ = 0.00036.
- Sometimes the initial centroids will readjust themselves in the “right” way, and sometimes they will not.

Solutions to initial centroids problem

- Multiple runs
  - Helps, but probability is not on your side.
- Sample and use hierarchical clustering to determine initial centroids.
- Select more than k initial centroids and then select among these initial centroids.
  - Select most widely separated.
- Postprocessing
- Generate a larger number of clusters and then perform a hierarchical clustering.
- Bisecting K-means
  - Not as susceptible to initialization issues.

Bisecting K-means

Variant of K-means that can produce a partitional or a hierarchical clustering.

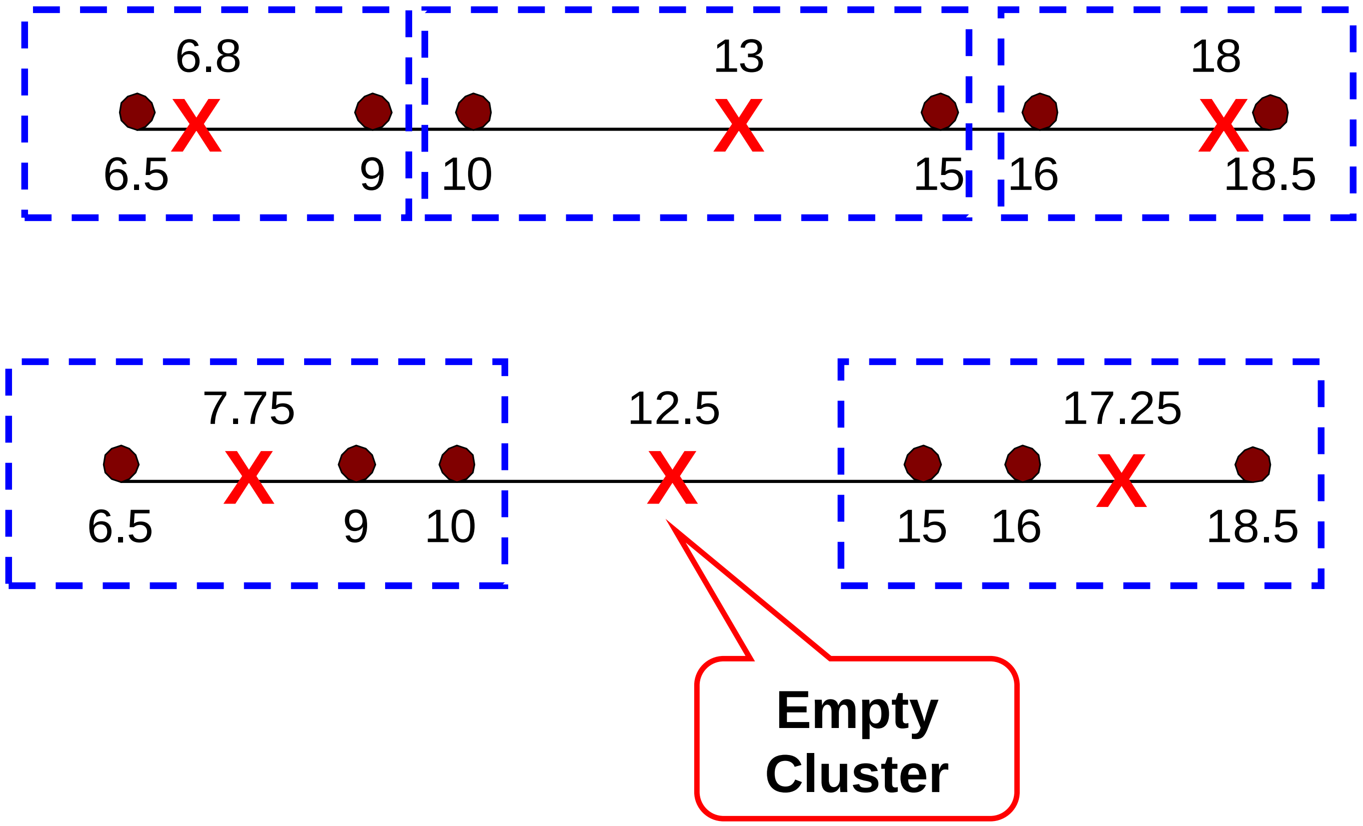


```
1: Initialize the list of clusters to contain the cluster containing all points.
2: repeat
3:    Select a cluster from the list of clusters
4:    for $i = 1$ to $number\_of\_iterations$ do
5:       Bisect the selected cluster using basic K-means
6:    end for
7:    Add the two clusters from the bisection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

CLUTO: http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Empty clusters

- K-means can yield empty clusters



Handling empty clusters

- Basic K-means algorithm can yield empty clusters.
- Select another point and assign it to the empty cluster.
  - Choose the point that contributes most to the K-means' criterion function.
  - Choose a point from the cluster with the highest value of the criterion function.
- If there are several empty clusters, the above can be repeated several times.

Pre-processing and Post-processing

- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers.
  - Split “loose” clusters, i.e., clusters with relatively high objective value (error).
  - Merge clusters that are “close” and that have relatively low error.

Other type of k-means

- Incremental update K-means
  - Update the centroids after each assignment:
    - Each assignment updates zero or two centroids.
    - More expensive.
    - Introduces an order dependency.
    - Never get an empty cluster.
- Robust K-means deals with outliers during the clustering
  - The non-clustered objects are treated as a *penalty* component of the objective function
- Spherical K-means
  - Vectors are normalized, similarity is cosine, minimizes sum of centroid similarities as the objective function
- Spectral K-means
  - DR + K-means

Other criterion functions

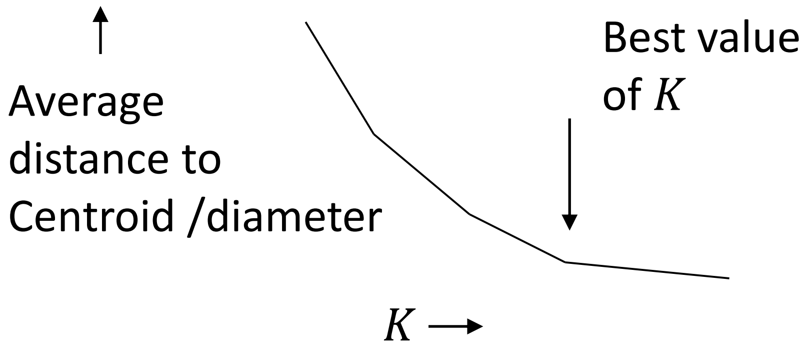
Criterion Function	Optimazition Function
$\mathcal{I}_1$	$$\text{maximize} \quad \sum_{i=1}^{k} \frac{1}{n_i} \left(\sum_{v,u \in S_i} \text{sim}(v, u) \right) \quad (1)$$
$\mathcal{I}_2$	$$\text{maximize} \quad \sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} \text{sim}(v, u)} \quad (2)$$
$\mathcal{E}_1$	$$\text{minimize} \quad \sum_{i=1}^{k} n_i \frac{\sum_{v \in S_i, u \in S} \text{sim}(v, u)}{\sqrt{\sum_{v,u \in S_i} \text{sim}(v, u)}} \quad (3)$$
$\mathcal{G}_1$	$$\text{minimize} \quad \sum_{i=1}^{k} \frac{\sum_{v \in S_i, u \in S} \text{sim}(v, u)}{\sum_{v,u \in S_i} \text{sim}(v, u)} \quad (4)$$
$\mathcal{G}'_1$	$$\text{minimize} \quad \sum_{i=1}^{k} n_i^2 \frac{\sum_{v \in S_i, u \in S} \text{sim}(v, u)}{\sum_{v,u \in S_i} \text{sim}(v, u)} \quad (5)$$
$\mathcal{H}_1$	$$\text{maximize} \quad \frac{\mathcal{I}_1}{\mathcal{E}_1} \quad (6)$$
$\mathcal{H}_2$	$$\text{maximize} \quad \frac{\mathcal{I}_2}{\mathcal{E}_1} \quad (7)$$

CLUTO: http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Getting *K* Right

- Try different K, looking at the change in the average distance to centroid, as K increases.

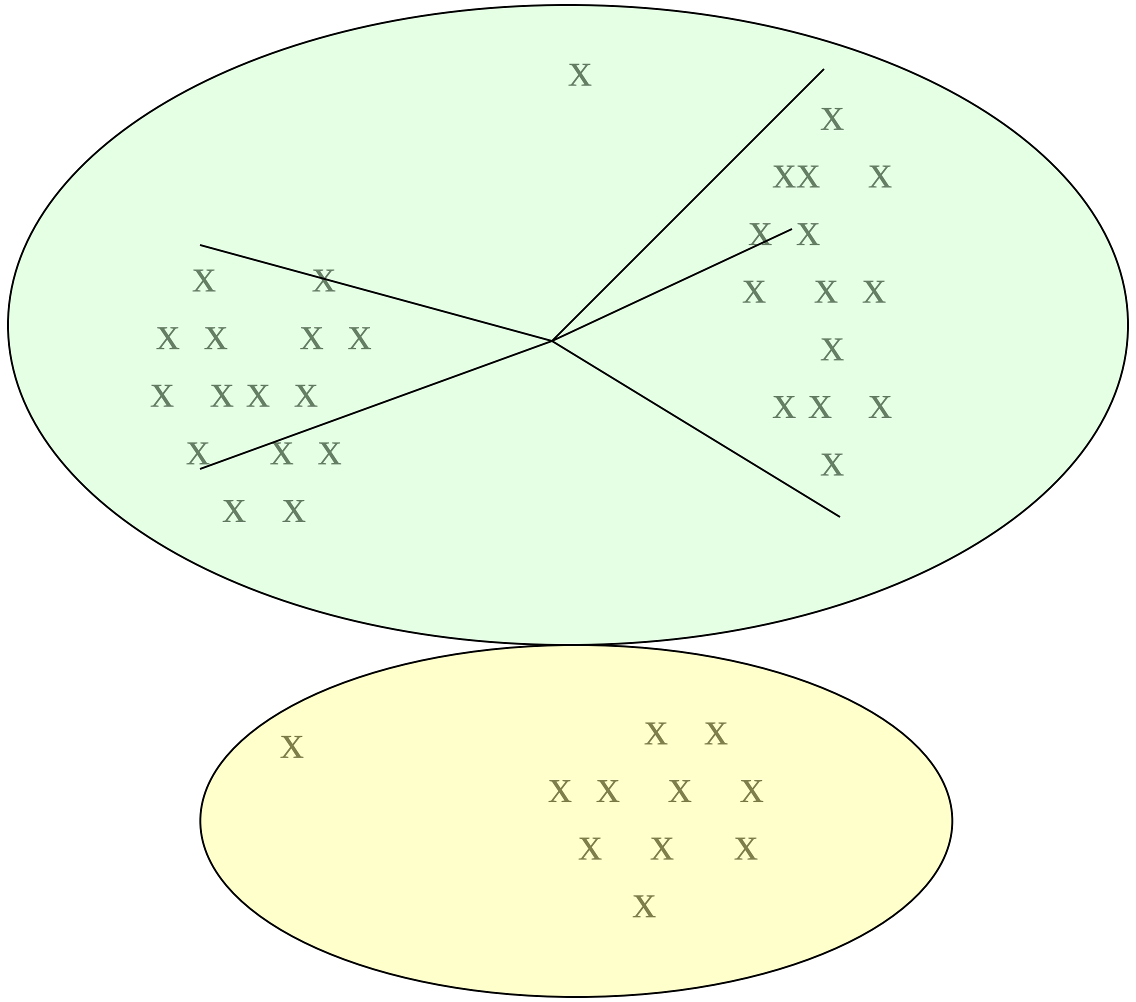
- Average falls rapidly until right K, then changes little.



Slide adapted from slides of J. Ullman and A. Rajamaran for “Mining Massive Datasets”

Example: Picking *k*

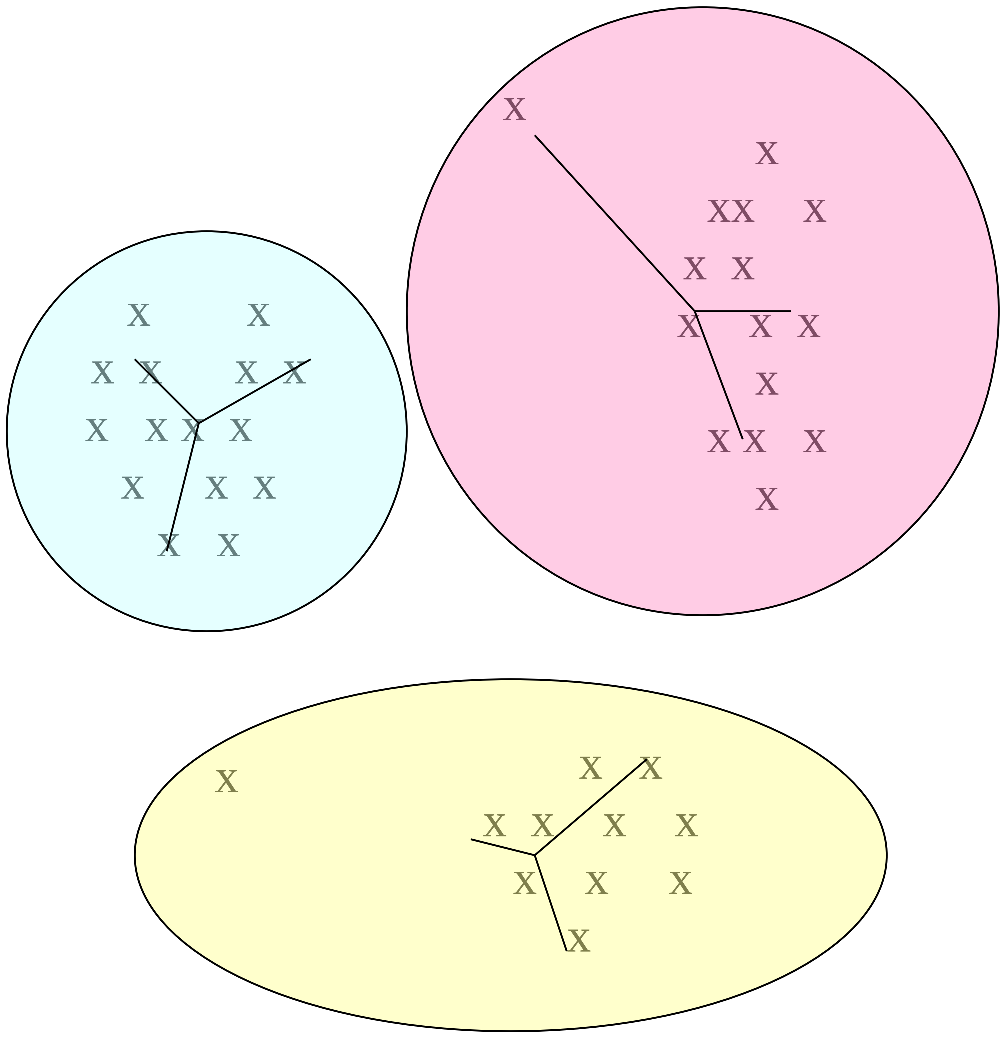
Too few;
many long
distances
to centroid.



Slide adapted from slides of J. Ullman and A. Rajamaran for “Mining Massive Datasets”

Example: Picking *k*

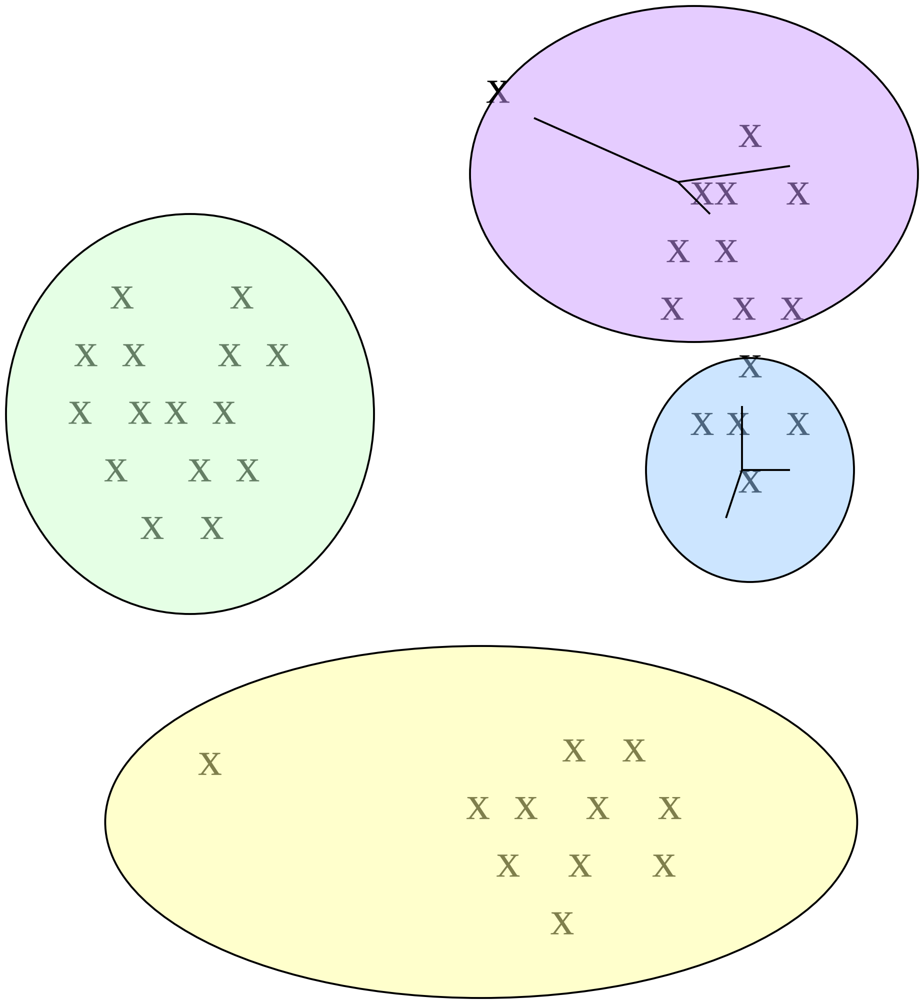
Just right;
distances
rather short.



Slide adapted from slides of J. Ullman and A. Rajamaran for "Mining Massive Datasets"

Example: Picking *k*

Too many;
little improvement
in average
distance.

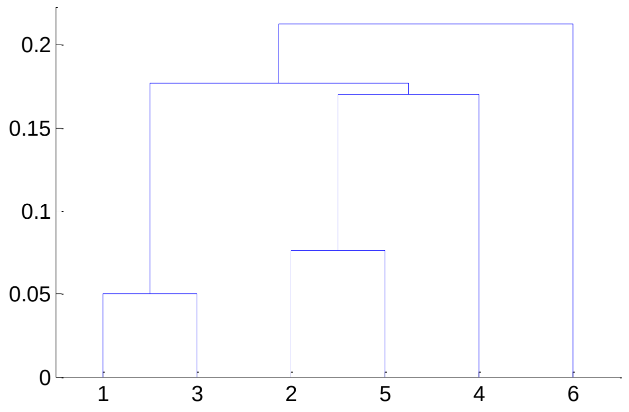


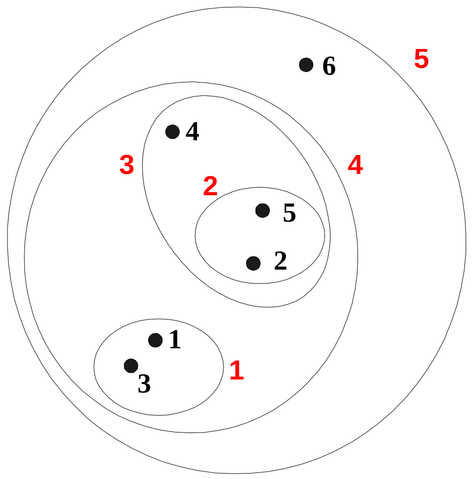
Slide adapted from slides of J. Ullman and A. Rajamaran for “Mining Massive Datasets”

HIERARCHICAL CLUSTERING

Hierarchical clustering

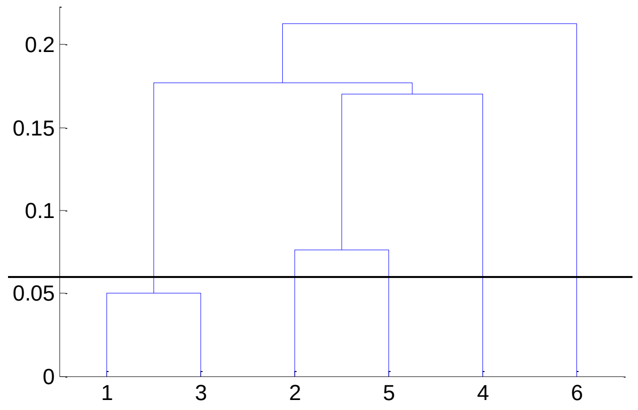
- Produces a set of nested clusters organized as a hierarchical tree.
- Can be visualized as a dendrogram.
  - A tree like diagram that records the sequences of merges or splits.



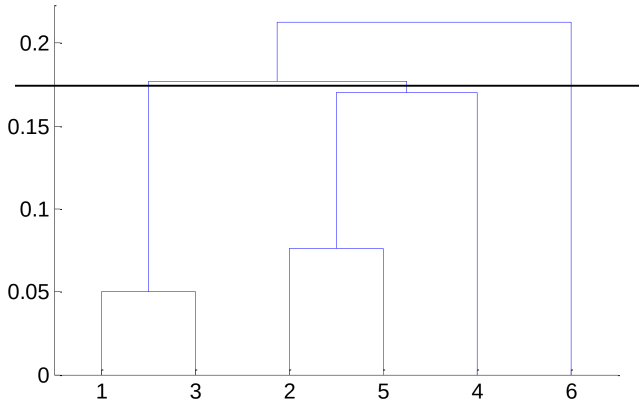


Advantages of hierarchical clustering

- Do not have to assume any particular number of clusters.
  - Any desired number of clusters can be obtained by "cutting" the dendrogram at the proper level.



- They may correspond to meaningful taxonomies.
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …).



Hierarchical clustering

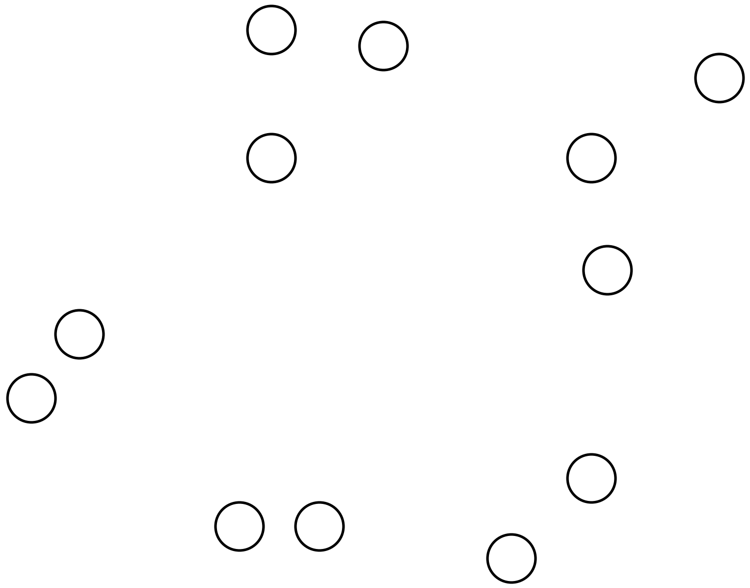
- Two main ways of obtaining hierarchical clusterings:
  - Agglomerative:
    - Start with the points as individual clusters.
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left.
  - Divisive:
    - Start with one, all-inclusive cluster.
    - At each step, split a cluster until each cluster contains a point (or there are k clusters).
- Traditional hierarchical algorithms use a similarity or distance matrix.
  - Merge or split one cluster at a time.

Agglomerative clustering algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix.
  2. Let each data point be a cluster.
  3. <u>Repeat</u>:
  4.   Merge the two closest clusters.
  5.   Update the proximity matrix.
  6. <u>Until</u> only a single cluster remains (or *k* clusters remain).
- Key operation is the computation of the proximity of two clusters.
  - Different approaches to defining the distance between clusters distinguish the different algorithms

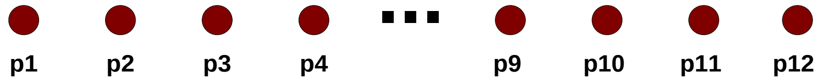
Starting situation

Start with clusters of individual points and a proximity matrix



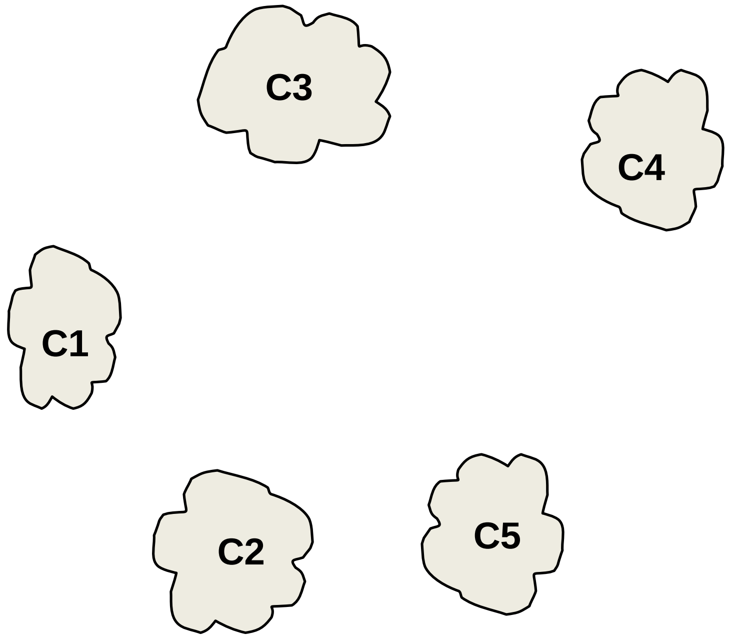
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



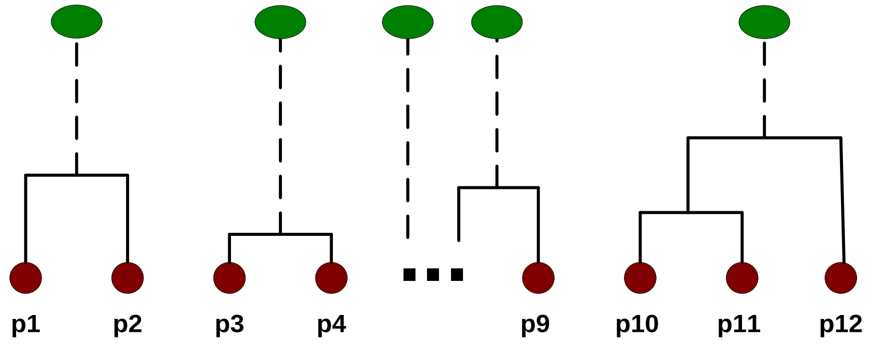
Intermediate situation

After some merging steps, we have some clusters



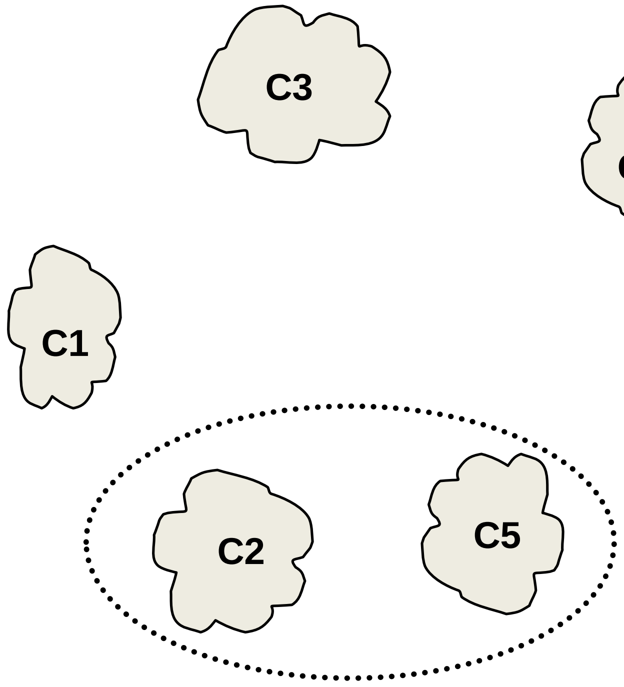
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

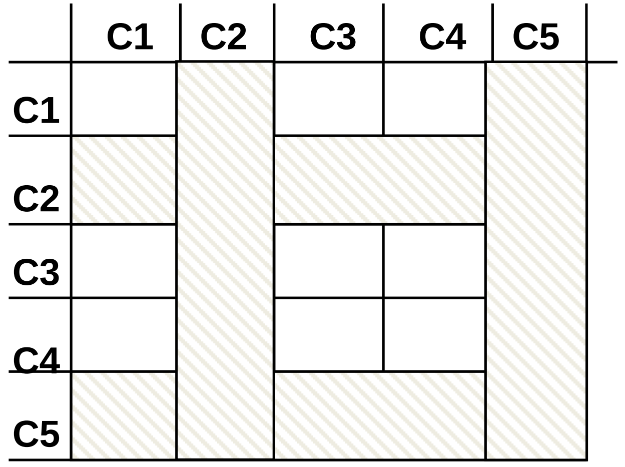
Proximity Matrix



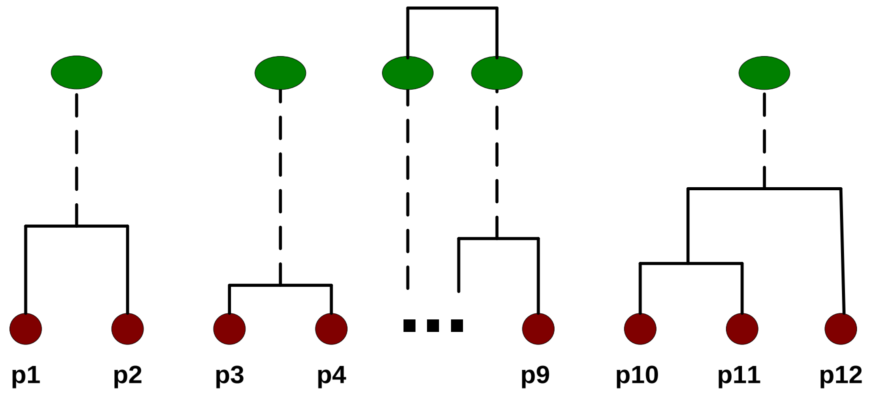
Intermediate situation

We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



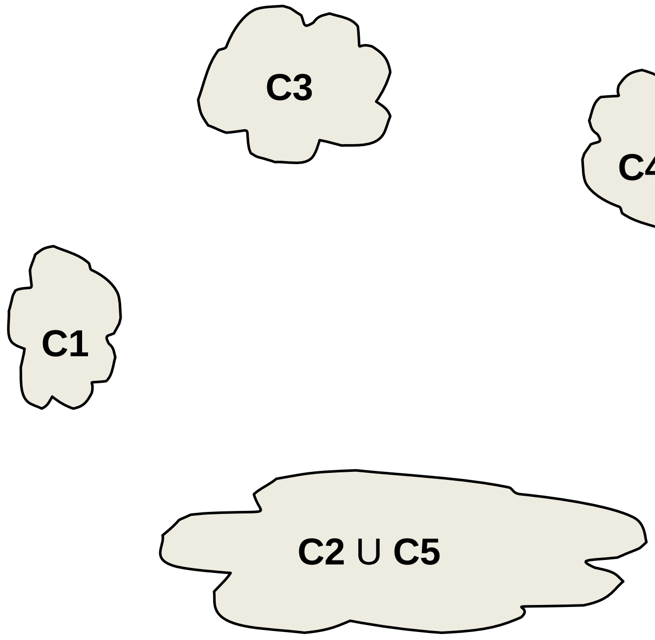


Proximity Matrix



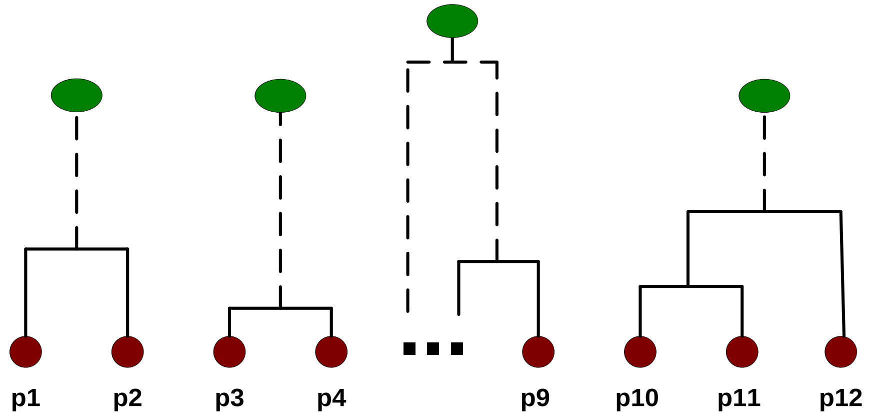
After merging

How do we update the proximity matrix?

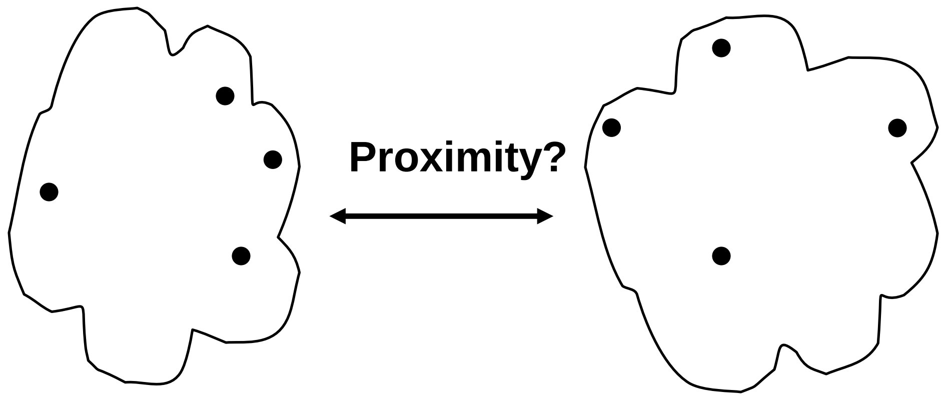


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



Defining inter-cluster proximity



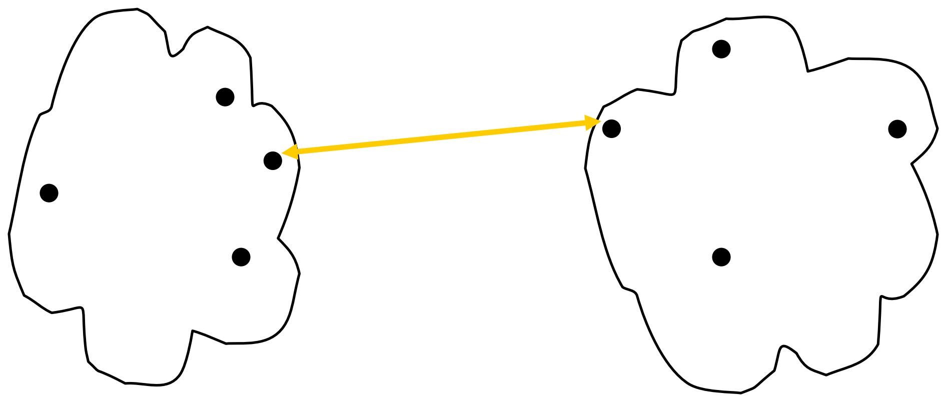
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Minimum distance, maximum distance, average distance, distance between centroids, objective-driven selection, etc.

Defining inter-cluster proximity

Using minimum distance.

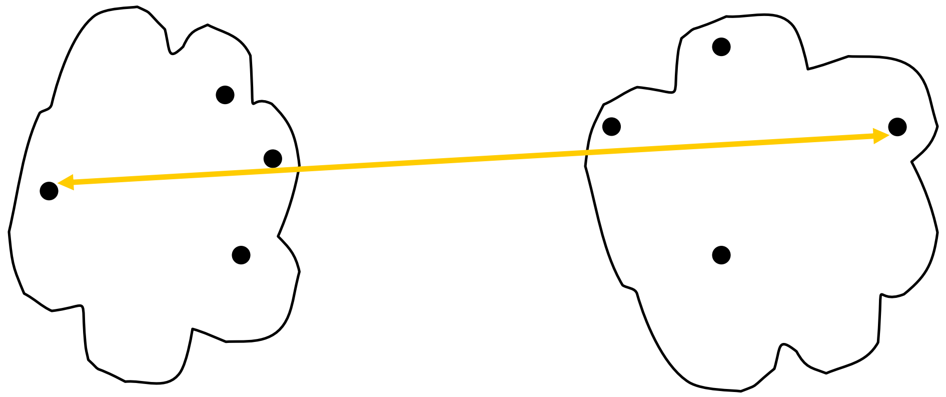


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Defining inter-cluster proximity

Using maximum distance.

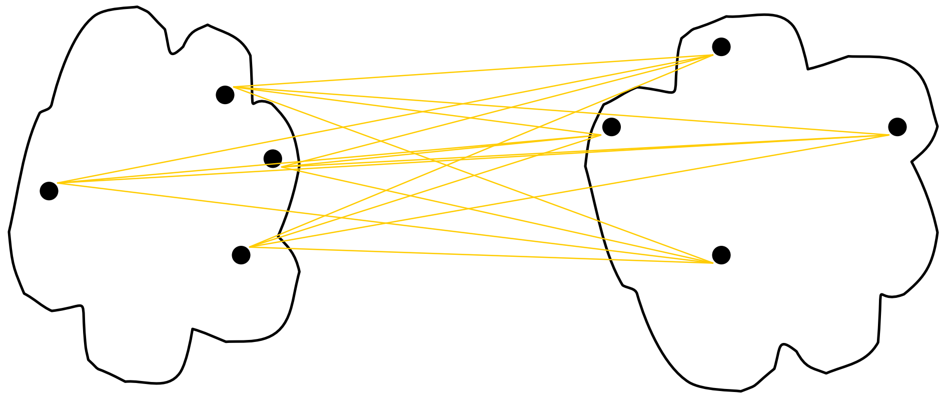


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Defining inter-cluster proximity

Using average distance.

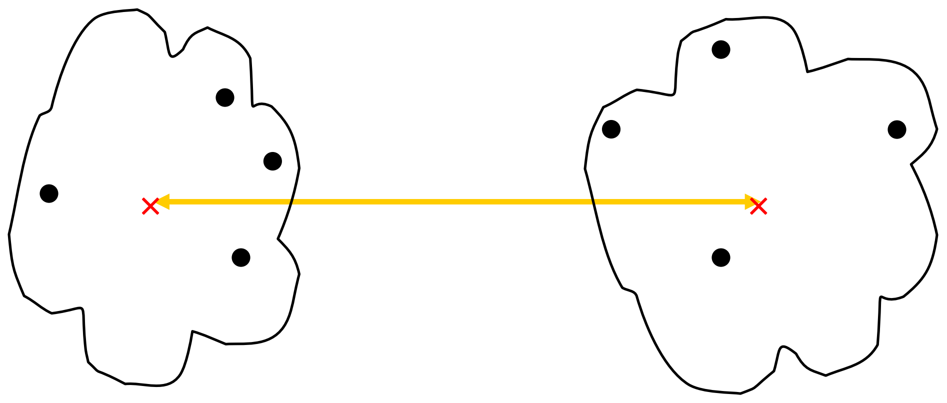


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
. . .						

Proximity Matrix

Defining inter-cluster proximity

Using distance between centroids.



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

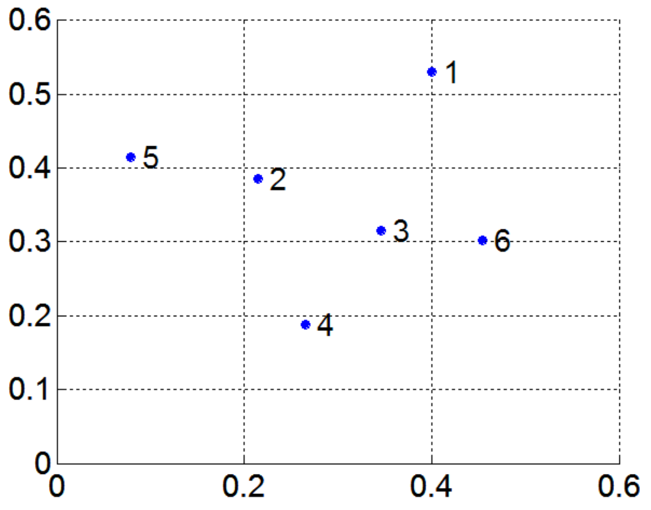
.

.

Proximity Matrix

Minimum distance (aka: single link)

- Proximity of two clusters is based on the two closest points in the different clusters.
  - Determined by one pair of points, i.e., by one link in the proximity graph.
- Example:



Distance Matrix:

	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

Minimum distance (aka: single link)

d(3, 6)=0.11 – link (3), (6)

Clusters: [(1), (2), (3,6), (4), (5)]

Compute d((3,6), 1) = *min*(d(3,1), d(6,1))=0.22

Compute d((3,6), 2) = *min*(d(3,2), d(6,2))=0.15

Compute d((3,6), 4) = *min*(d(3,4), d(6,4))=0.15

Compute d((3,6), 5) = *min*(d(3,5), d(6,5))=0.28

d(2, 5)=0.14 – link (2), (5)

Clusters: [(1), (2,5), (3,6), (4)]

Compute d((2,5), (3,6)) = *min*(d(2,3), d(2,6), d(5,3), d(5,6))=0.15

Compute d((2,5), 1) = *min*(d(2,1), d(5,1))=0.24

Compute d((2,5), 4) = *min*(d(2,4), d(5,4))=0.20

d((2,5), (3,6))=0.15 – link (2,5), (3,6)

Clusters: [(1), (2,3,5,6), (4)]

Compute d((2,3,5,6), 1) = 0.22

Compute d((2,3,5,6), 4) = 0.15

d((2,3,5,6), 4)=0.15 – link (2,3,5,6), (4)

Clusters: [(1), (2,3,4,5,6)]

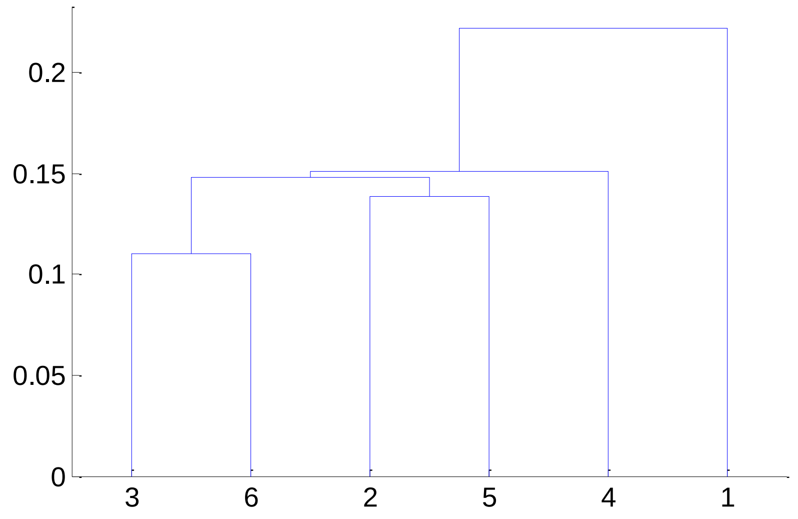
Compute d((2,3,4,5,6), 1) = 0.22

d((2,3,4,5,6), 1)=0.22 – link (2,3,4,5,6), (1)

Clusters: [(1,2,3,4,5,6)]

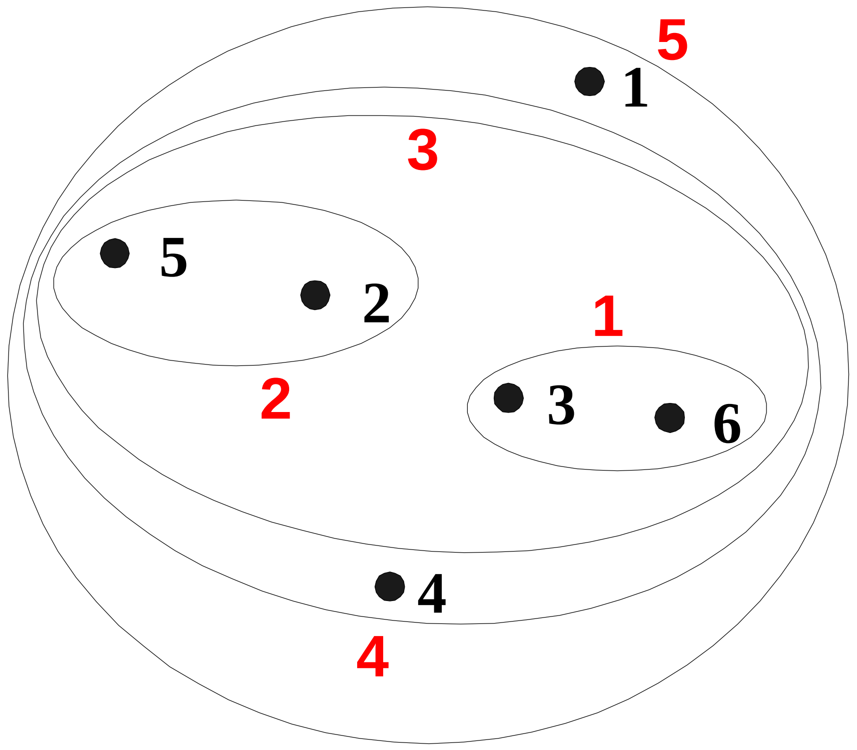
DONE (all assigned)

At each step, choose the link with **MINIMUM** inter-cluster distance

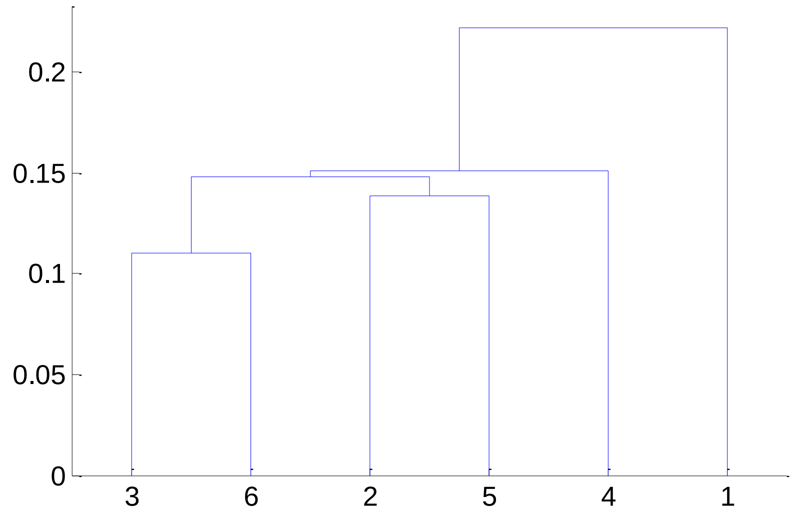


Dendrogram

Minimum distance (aka: single link)

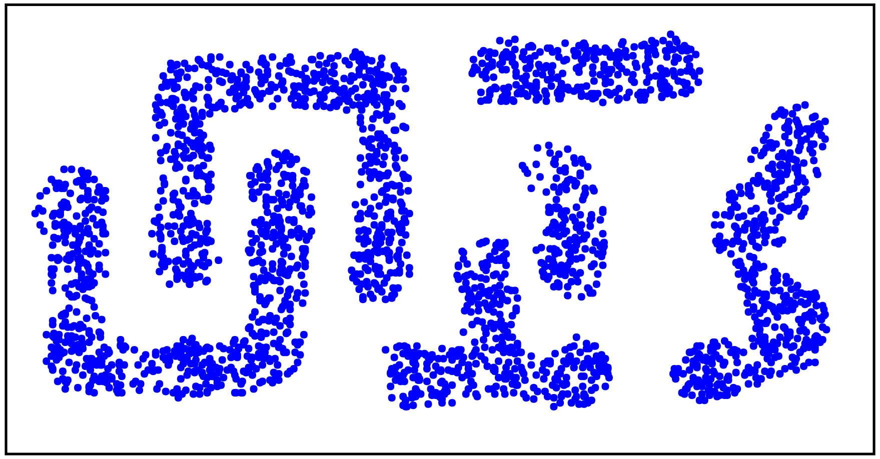


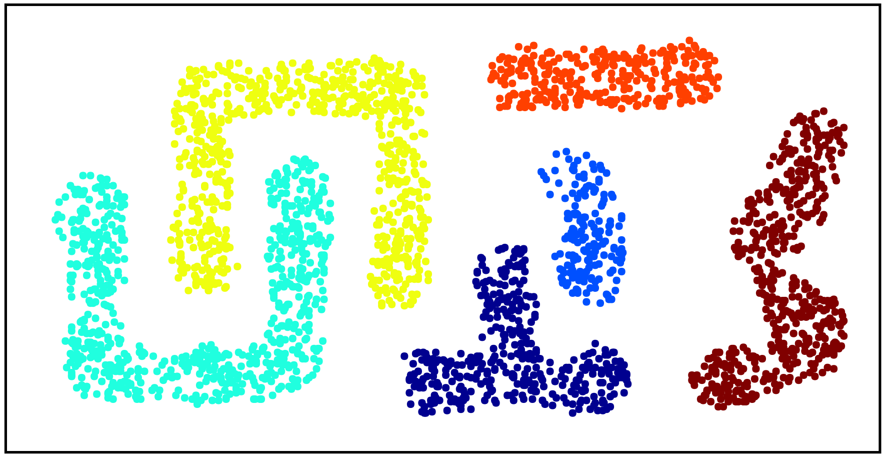
Nested Clusters



Dendrogram

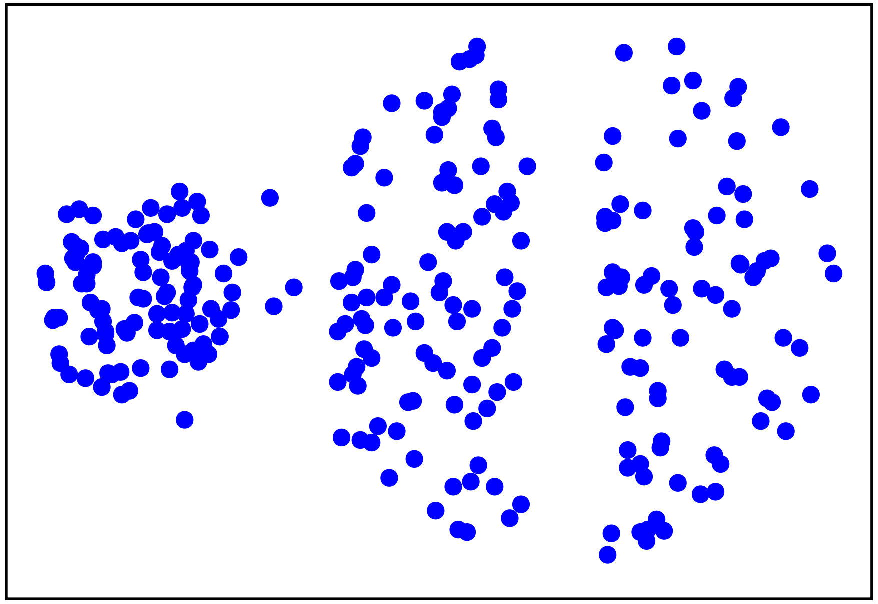
Strength of minimum distance

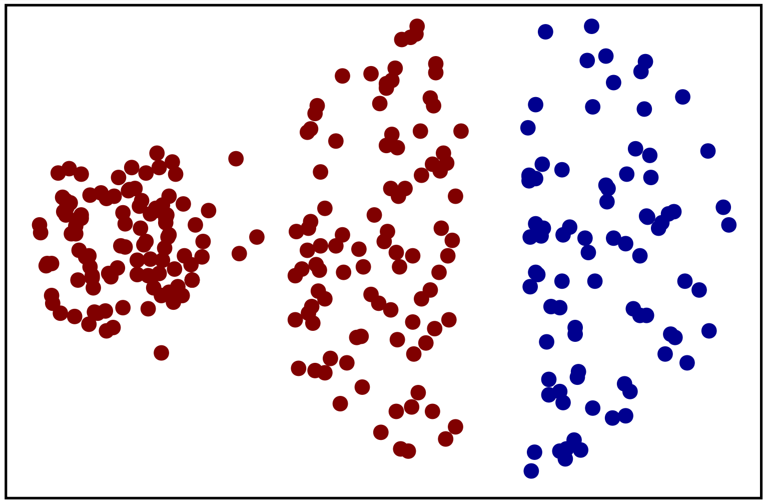

Original Points

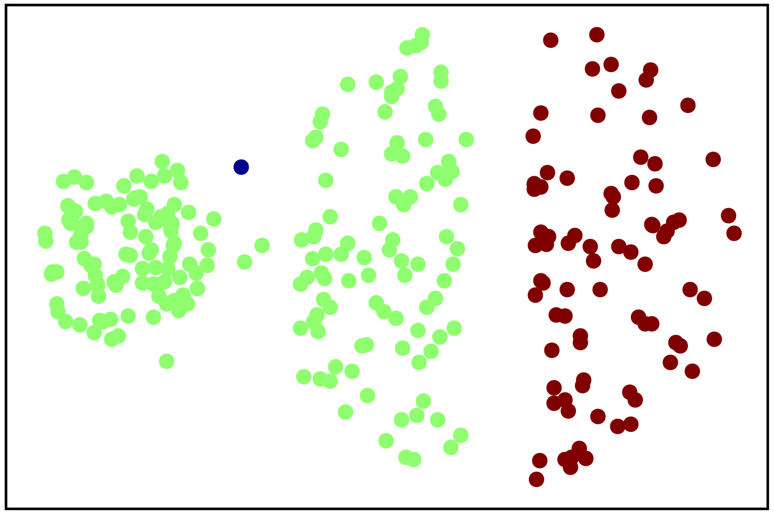

Six Clusters

Can handle non-elliptical shapes.

Limitations of minimum distance


Original Points

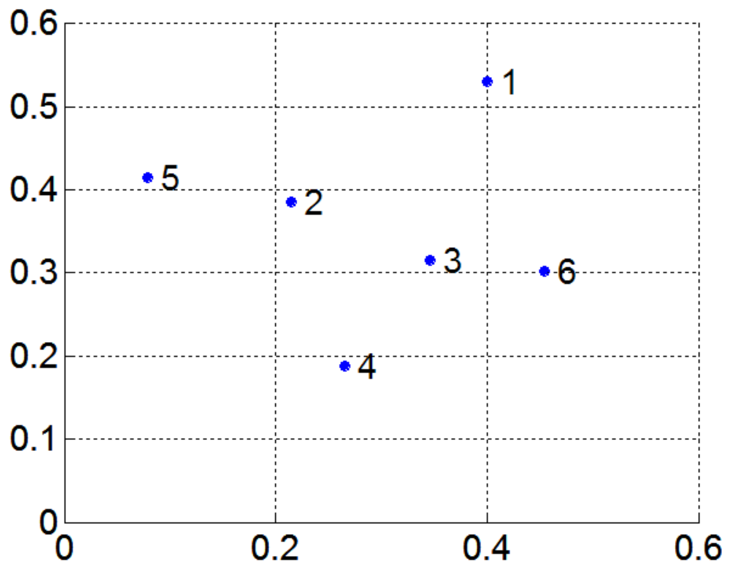

Two Clusters


Three Clusters

Sensitive to noise and outliers.

Maximum distance (aka: complete link)

- Proximity of two clusters is based on the two most distant points in the different clusters.
  - Determined by all pairs of points in the two clusters.



Distance Matrix:

	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

Maximum distance (aka: complete link)

d(3, 6)=0.11 – link (3), (6)

Clusters: [(1), (2), (3,6), (4), (5)]

Compute d((3,6), 1) = max(d(3,1), d(6,1))=0.23

Compute d((3,6), 2) = max(d(3,2), d(6,2))=0.25

Compute d((3,6), 4) = max(d(3,4), d(6,4))=0.22

Compute d((3,6), 5) = max(d(3,5), d(6,5))=0.39

d(2, 5)=0.14 – link (2), (5)

Clusters: [(1), (2,5), (3,6), (4)]

Compute d((2,5), (3,6)) = max(d(2,3), d(2,6), d(5,3), d(5,6))=0.39

Compute d((2,5), 1) = max(d(2,1), d(5,1))=0.34

Compute d((2,5), 4) = max(d(2,4), d(5,4))=0.29

d((3,6), 4)=0.22 – link (3,6), (4)

Clusters: [(1), (2,5), (3,4,6)]

Compute d((3,4,6), 1) = 0.37

Compute d((3,4,6), (2,5)) = 0.39

d((2,5), 1)=0.34 – link (2,5), (1)

Clusters: [(1,2,5), (3,4,6)]

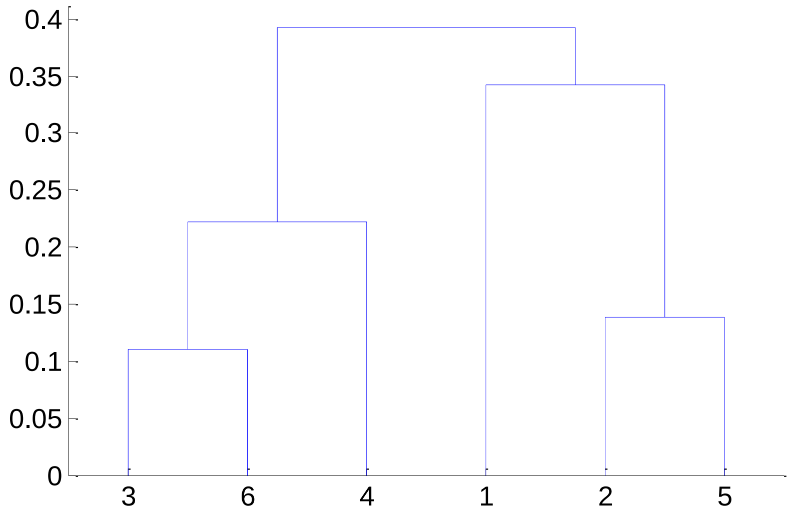
Compute d((1,2,5), (3,4,6)) = 0.39

d((1,2,5), (3,4,6))=0.22 – link (1,2,5), (3,4,6)

Clusters: [(1,2,3,4,5,6)]

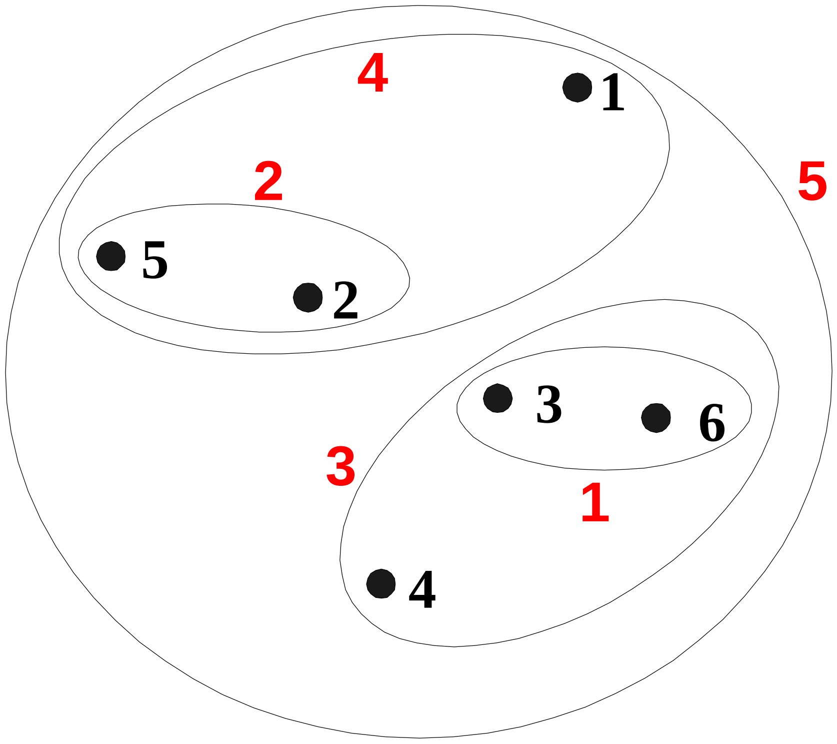
DONE (all assigned)

At each step, choose the link with **MINIMUM** inter-cluster distance

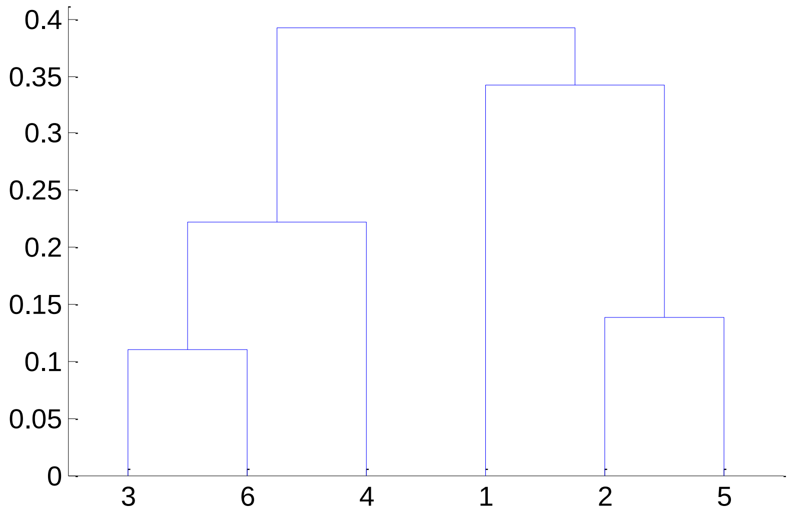


Dendrogram

Maximum distance (aka: complete link)

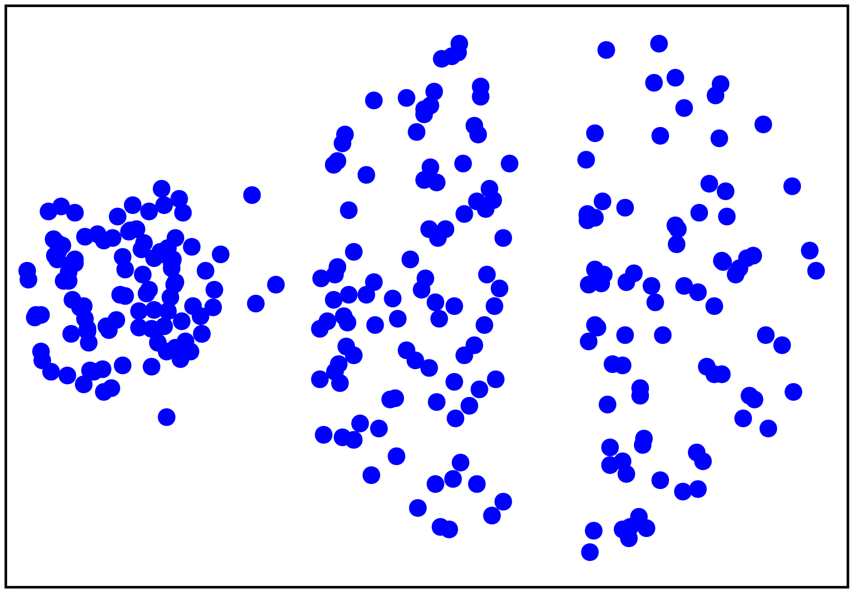


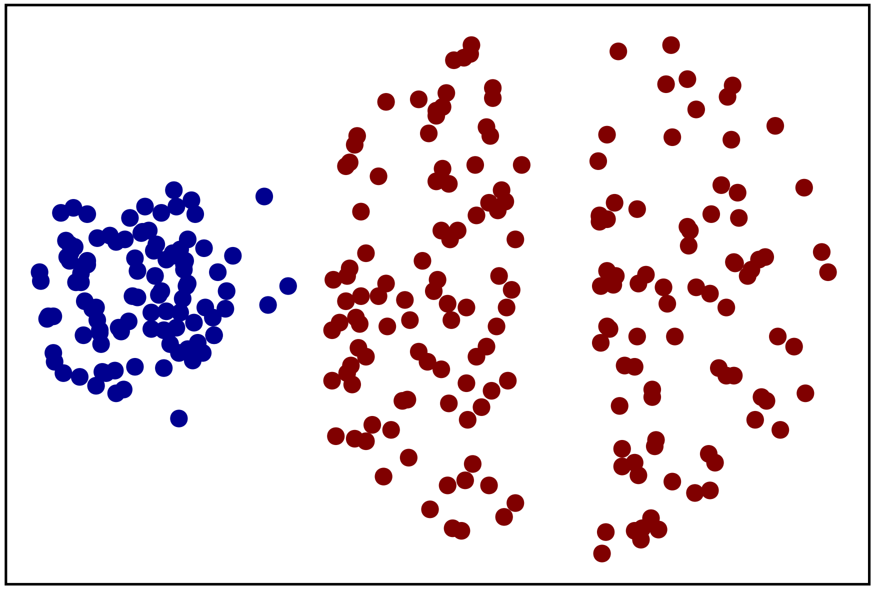
Nested Clusters



Dendrogram

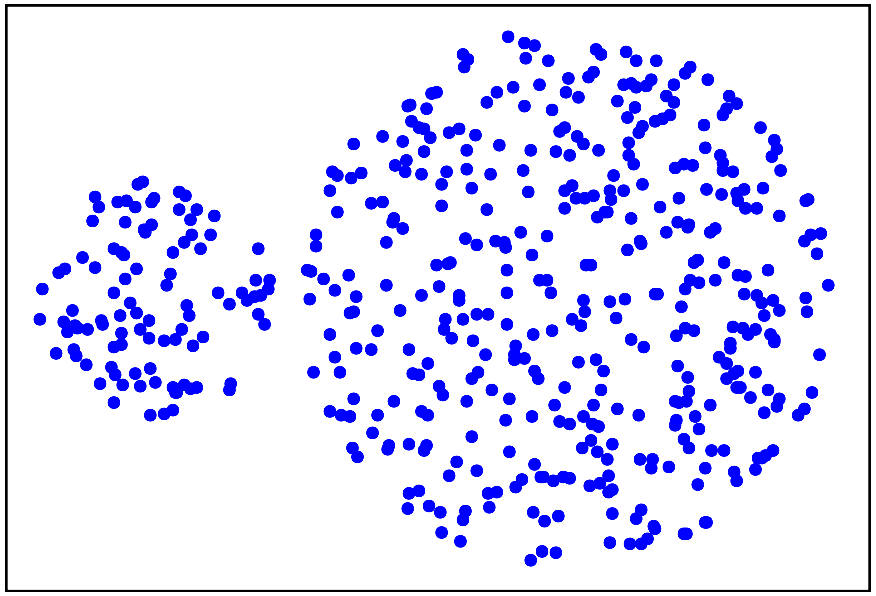
Strength of maximum distance


Original Points

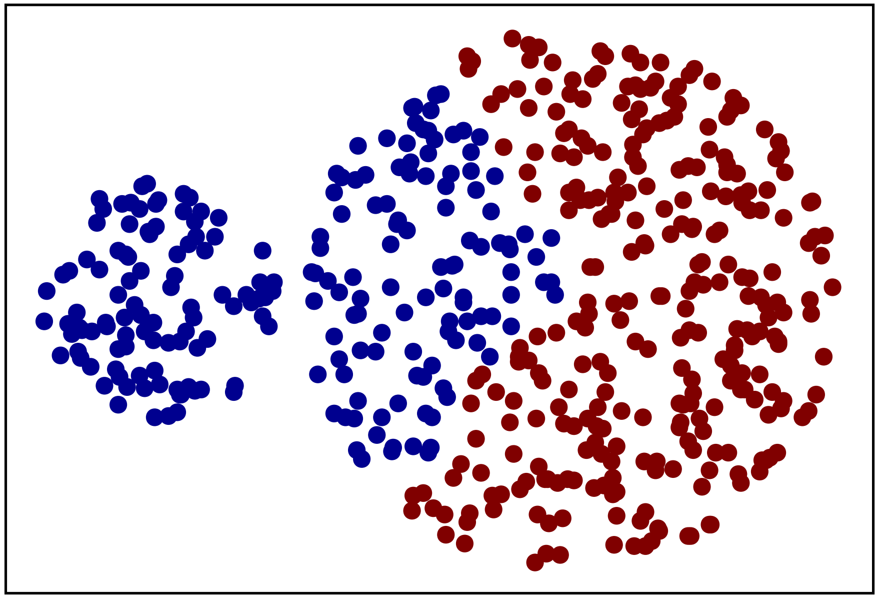

Two Clusters

Less susceptible to noise and outliers.

Limitations of maximum distance



Original Points



Two Clusters

Tends to break large clusters.

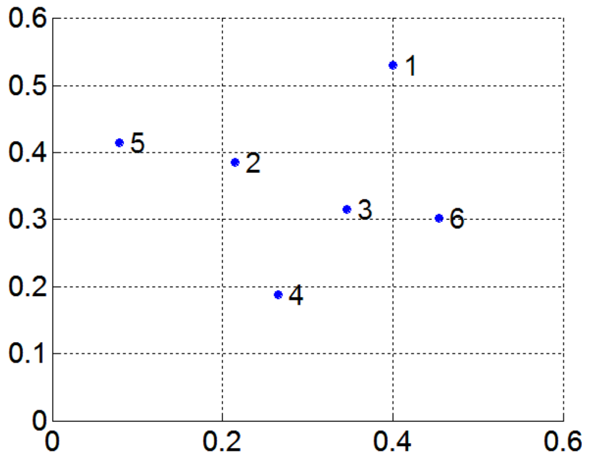
Biased towards globular clusters.

Average distance (aka: UPGMA)

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum\limits_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters.

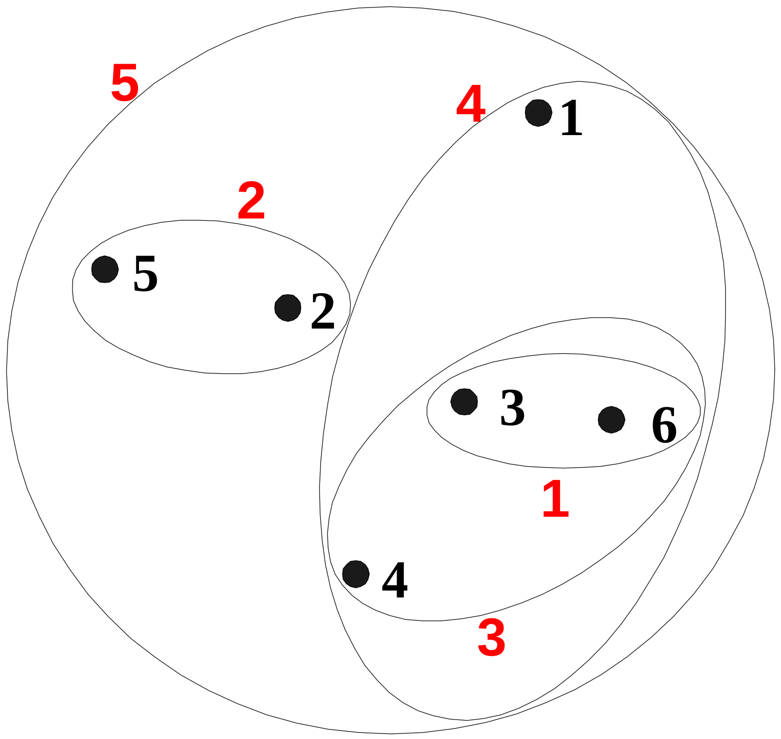


Distance Matrix:

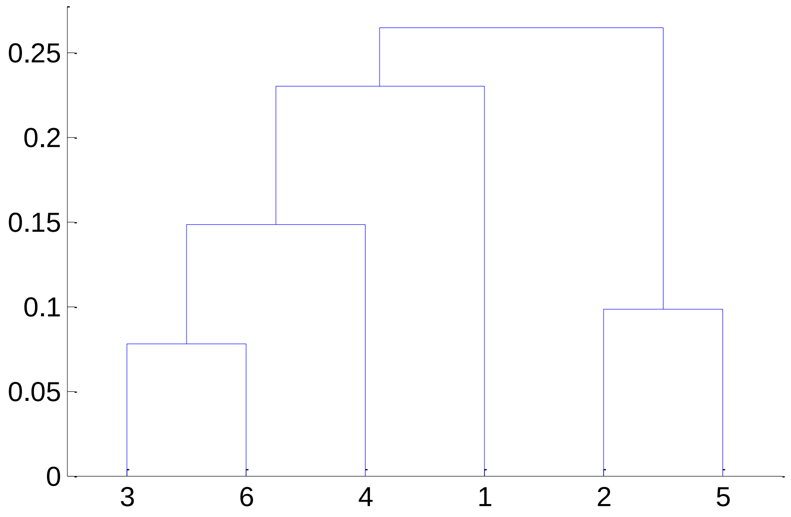
	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

UPGMA: Unweighted Pair Group Method with Arithmetic Mean!

Average distance



Nested Clusters



Dendrogram

Group average

- Compromise between single and complete link.

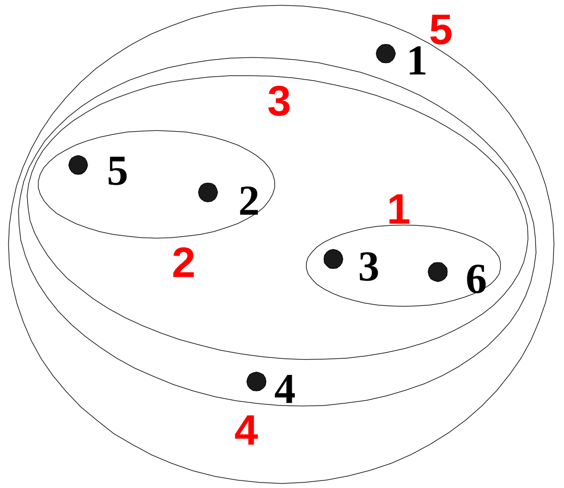
- Strengths:
  - Less susceptible to noise and outliers.

- Limitations:
  - Biased towards globular clusters.

Objective driven: Ward’s Method

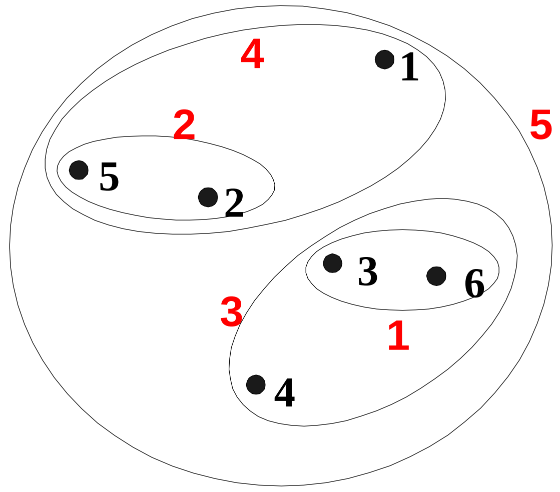
- Similarity of two clusters is based on the increase in squared error when two clusters are merged.
  - Similar to group average if distance between points is distance squared.
- Less susceptible to noise and outliers.
- Biased towards globular clusters.
- Hierarchical analogue of sum-of-squared-error objective (K-means)
  - Can be used to initialize K-means.

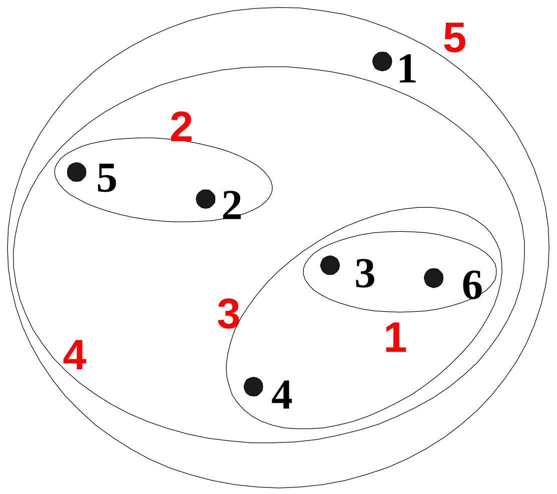
Hierarchical clustering: Comparison



MIN

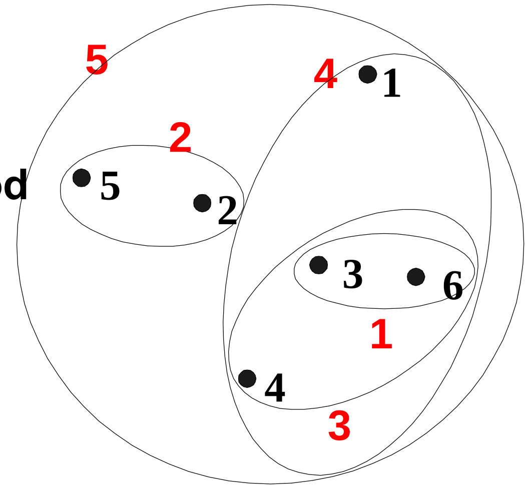
MAX





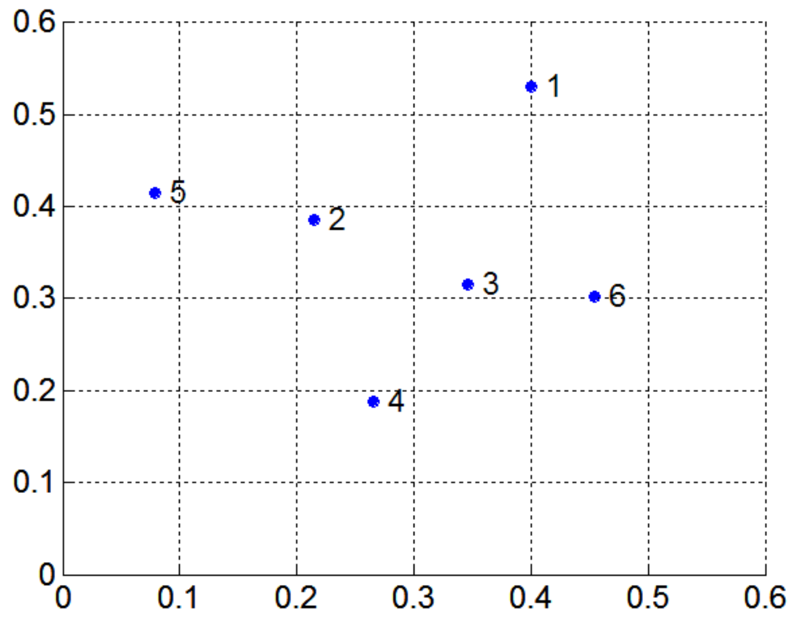
Ward's Method

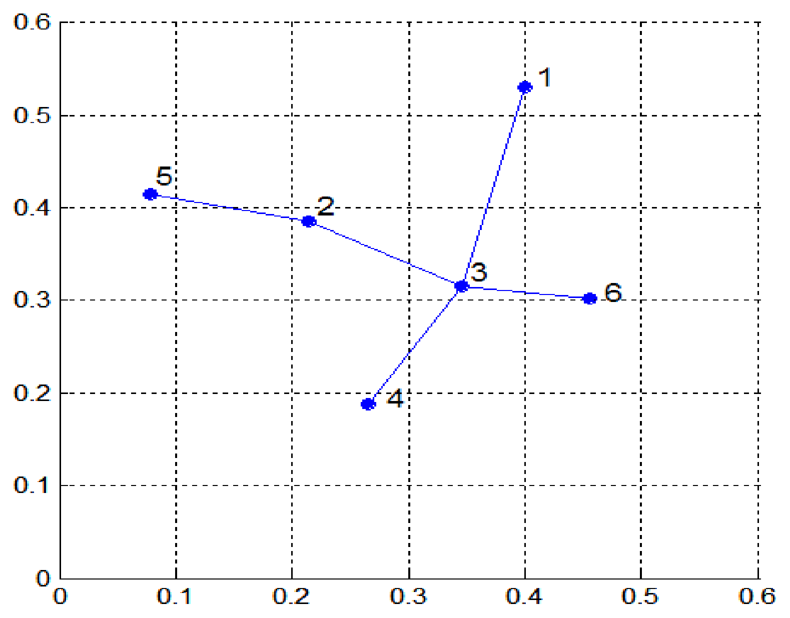
Group Average



MST: Agglomerative hierarchical clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point.
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not.
  - Add q to the tree and put an edge between p and q.





Hierarchical clustering: Time and space requirements

- $\mathrm{O}(n^2)$ space since it uses the proximity matrix.
  - n is the number of points.

- $\mathrm{O}(n^3)$ time in many cases
  - There are n steps and at each step the proximity matrix must be updated and searched (on the average there are n^2 updates on that matrix).
  - Complexity can be reduced to $\mathrm{O}(n^2 \log(n))$ time with some cleverness.

Hierarchical clustering: Problems and limitations

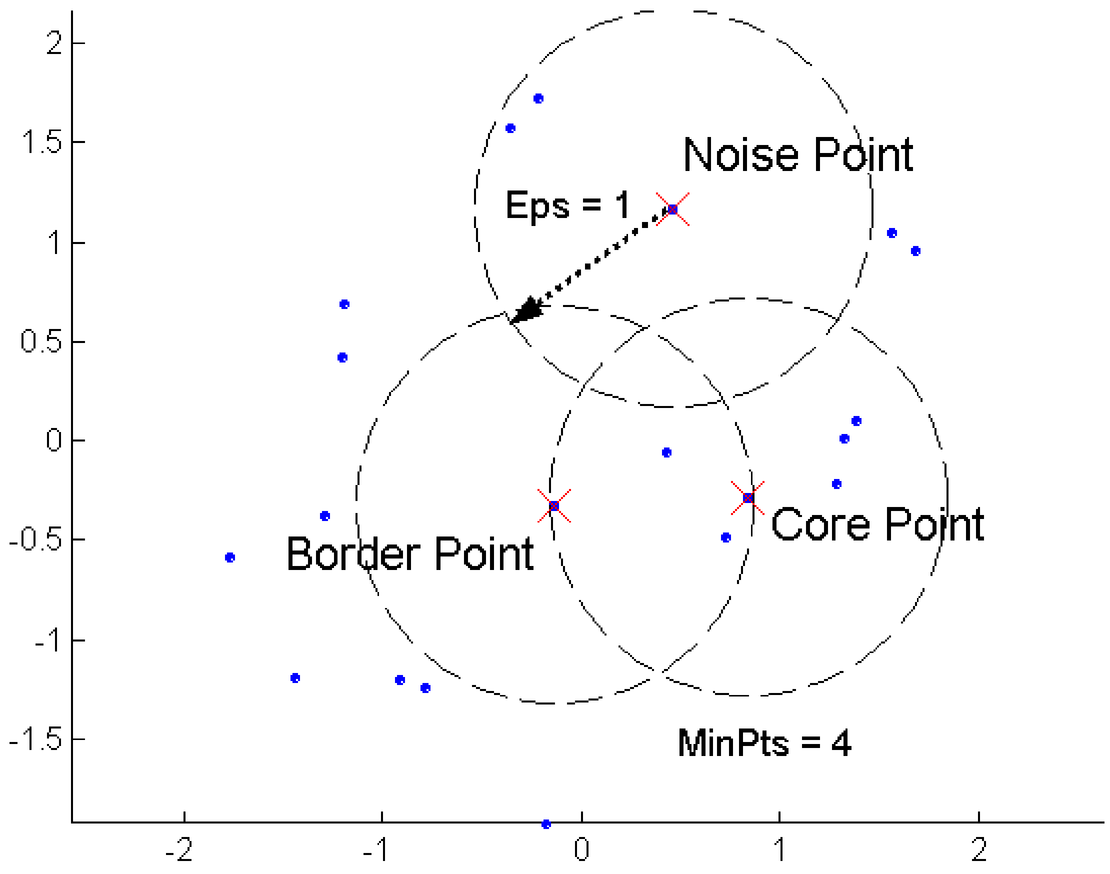
- Once a decision is made to combine two clusters, it cannot be undone.
- Objective function is optimized only locally.
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers.
  - Difficulty handling different sized clusters and convex shapes.
  - Breaking large clusters.

DENSITY-BASED CLUSTERING

DBSCAN

- DBSCAN is a density-based algorithm.
  - The *density* is the number of points within a specified radius (Eps)
  - A point is a *core point* if it has more than a specified number of points (MinPts) within Eps.
    - These are points that are at the interior of a cluster.
  - A *border point* has fewer than MinPts within Eps, but is in the neighborhood of a core point.
  - A *noise point* is any point that is not a *core point* or a *border point*.

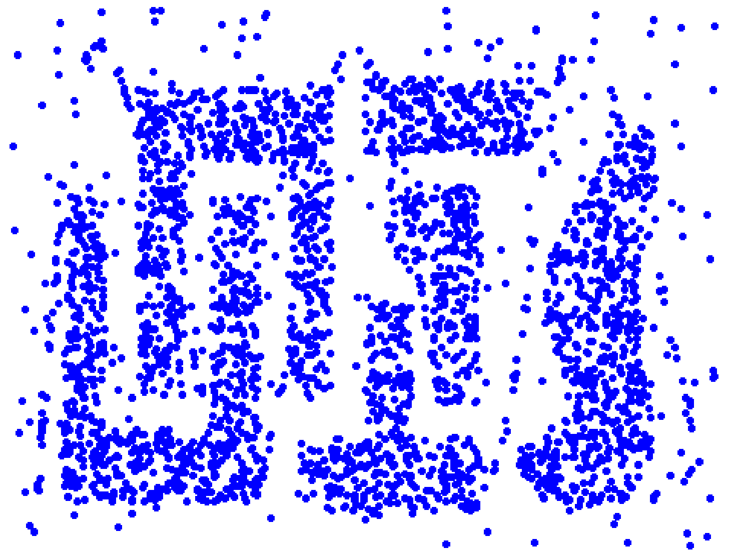
DBSCAN: core, border, and noise points

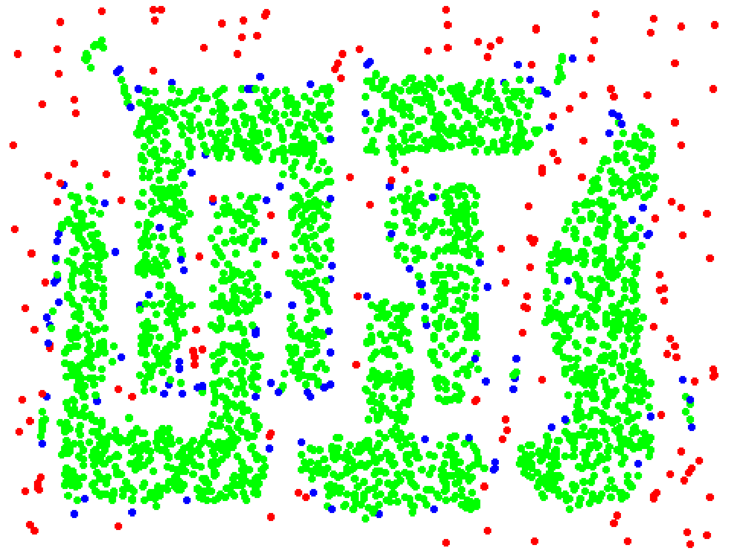


DBSCAN algorithm

```
Algorithm DBSCAN(Data: $\mathcal{D}$, Radius: $Eps$, Density: $\tau$ )
begin
  Determine core, border and noise points of $\mathcal{D}$ at level $(Eps, \tau)$;
  Create graph in which core points are connected
    if they are within $Eps$ of one another;
  Determine connected components in graph;
  Assign each border point to connected component
    with which it is best connected;
  return points in each connected component as a cluster;
end
```

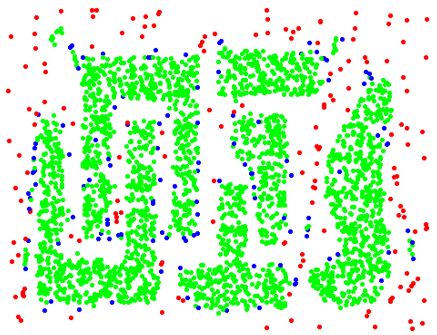
DBSCAN: core, border and noise points

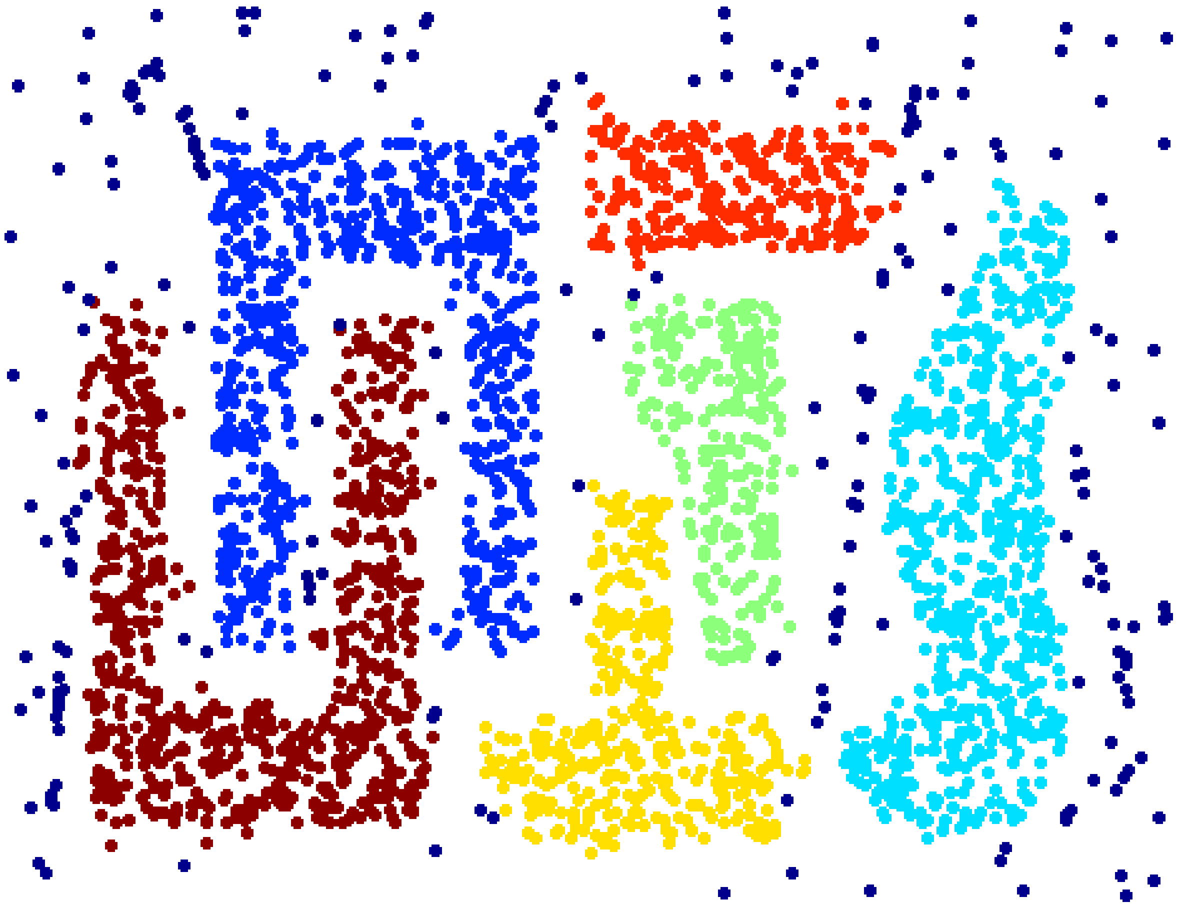

Original Points


Point types: core, border and noise

Eps = 10, MinPts = 4

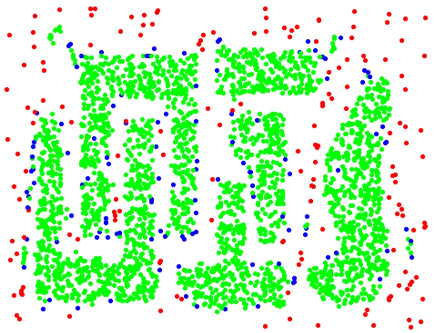
DBSCAN clustering



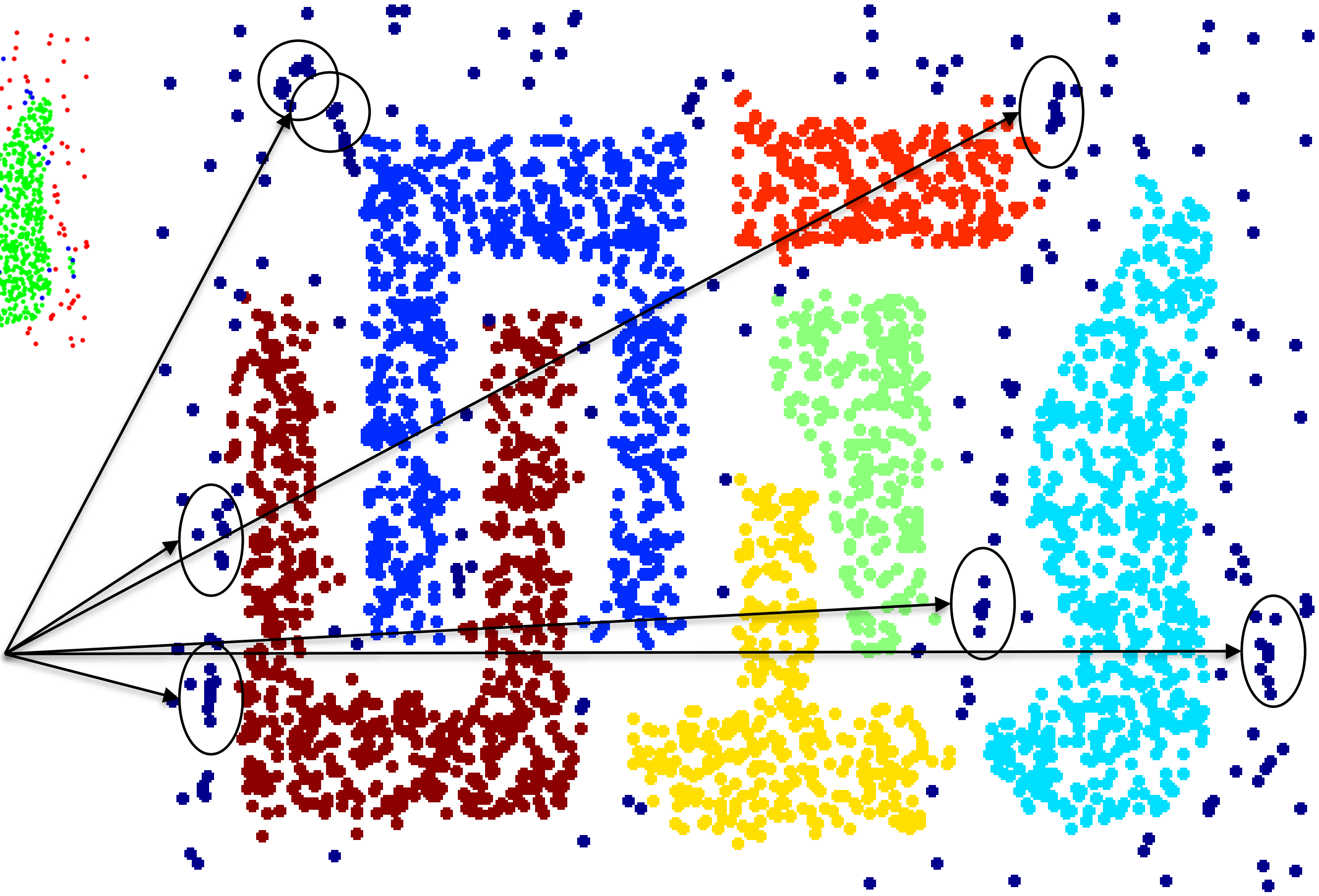


Clusters

DBSCAN clustering

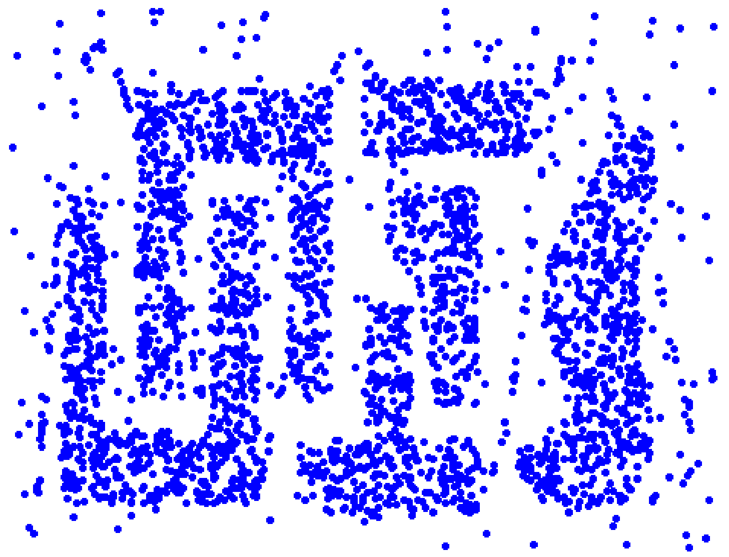


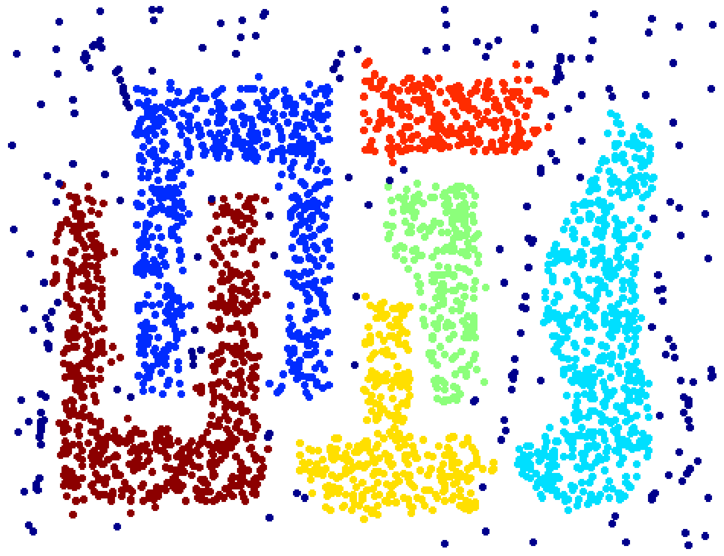
These are also clusters. They are usually eliminated by putting a minimum cluster size threshold.



Clusters

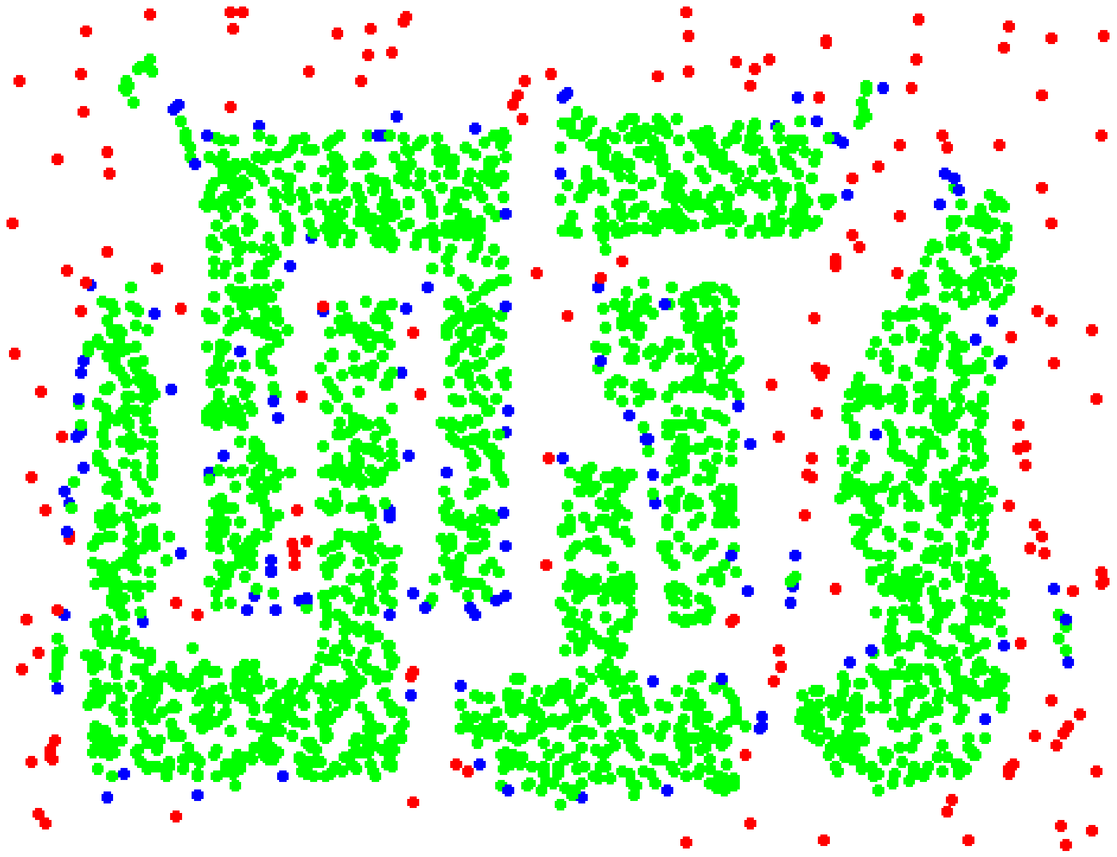
DBSCAN clustering


Original Points

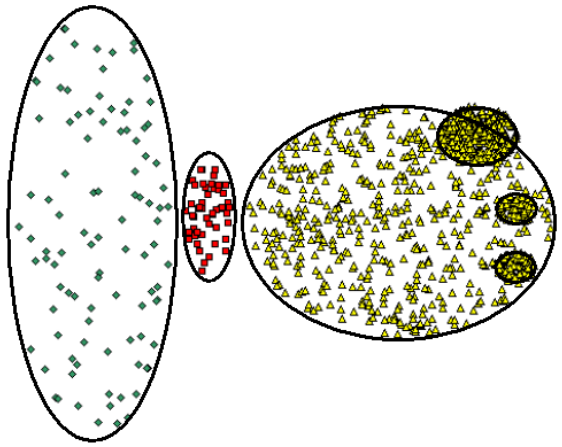

Clusters

- Resistant to (some) noise.
- Can handle clusters of different shapes and sizes.

DBSCAN: How much noise?

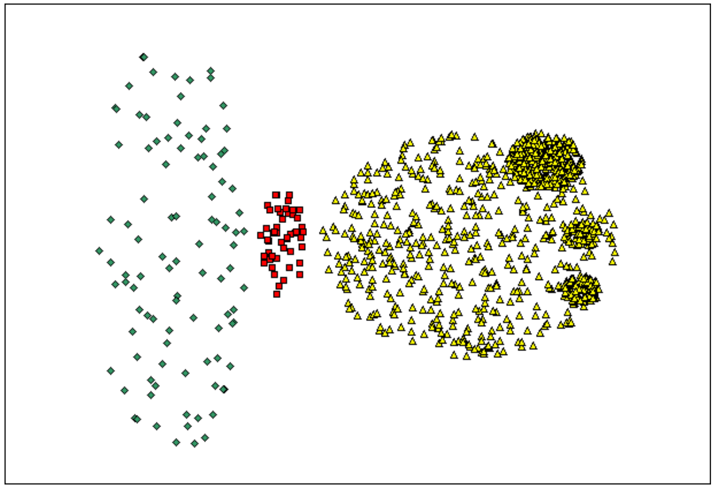


When DBSCAN does not work well

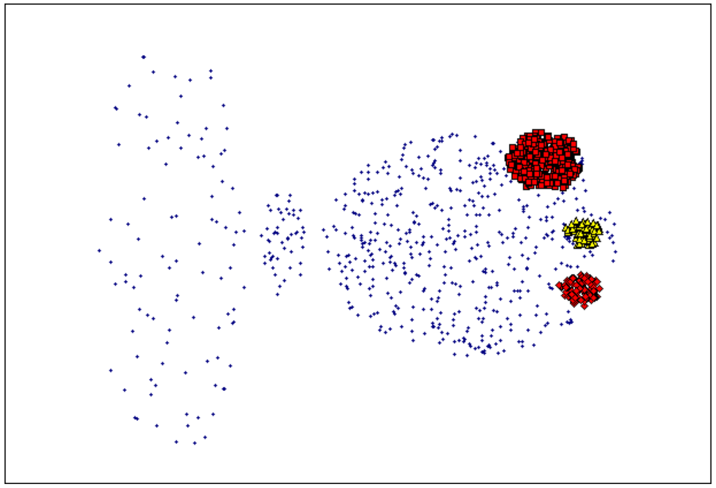


Original Points

- **Varying densities**
- **High-dimensional data**



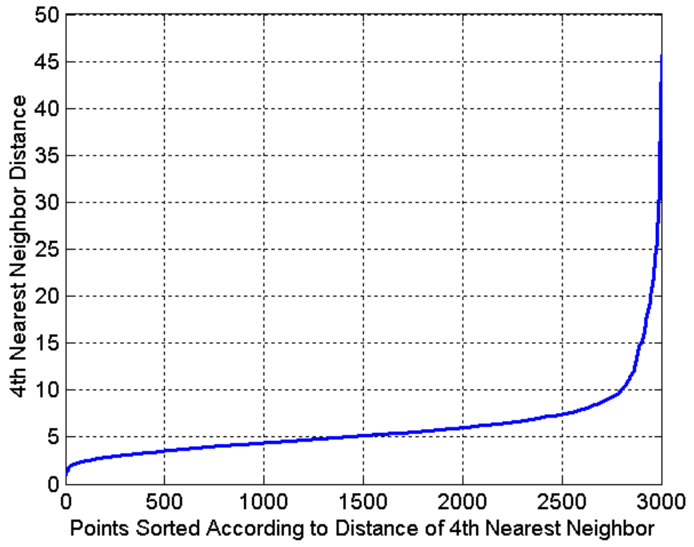
(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their kth nearest neighbors are roughly at the same distance.
- Noise points have the kth nearest neighbor at farther distance.
- So, plot sorted distance of every point to its kth nearest neighbor.



OTHER CLUSTERING ALGORITHMS

Graph-based clustering: General concepts

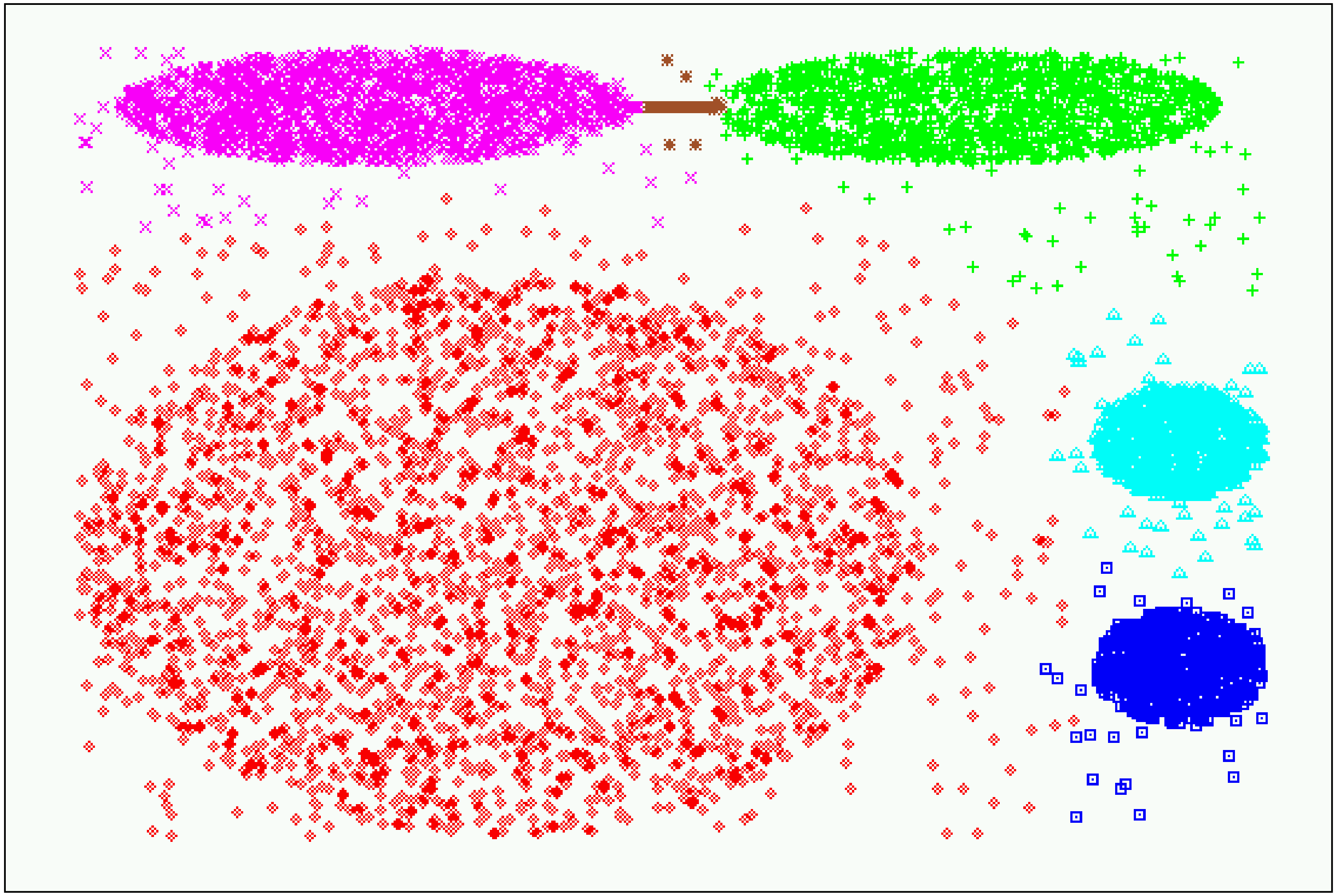
- Graph-based clustering uses the proximity graph.
  - Start with the proximity matrix.
  - Consider each point as a node in a graph.
  - Each edge between two nodes has a weight which is the proximity between the two points.
  - Initially the proximity graph can be fully connected.
  - MIN (single-link) and MAX (complete-link) can be viewed in graph terms.

- It is a general framework and it can be applied to datasets of different and/or mixed attributes.
  - The only requirement is to be able to compute meaningful similarities.

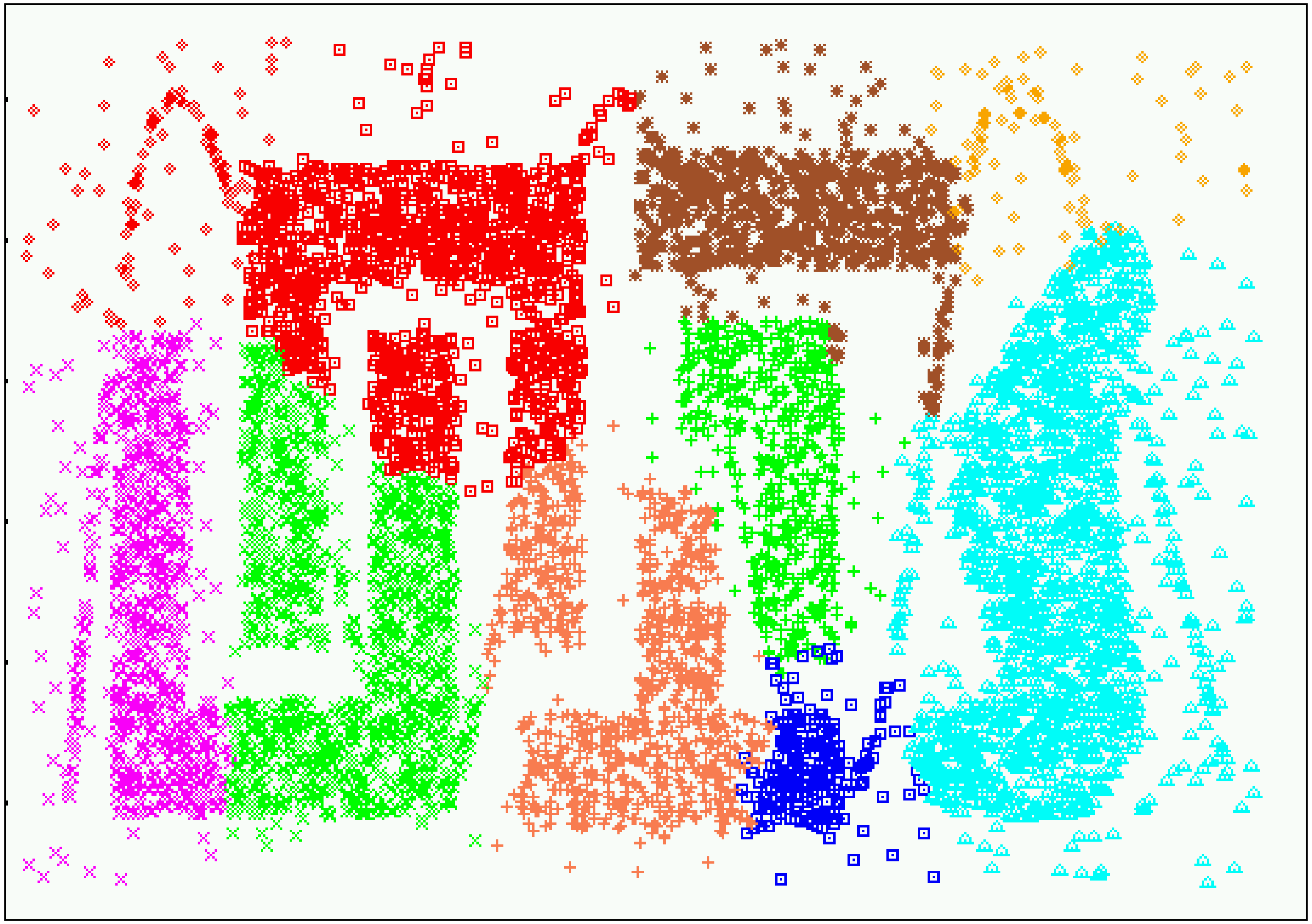
Graph-based clustering: comparison

- CURE
  - Represents a cluster using multiple representative points found by selecting a constant number of points from a cluster.
    - First representative point is chosen to be the point furthest from the center of the cluster.
    - Remaining representative points are chosen so that they are farthest from all previously chosen points.
  - Identifies arbitrarily shaped clusters of various sizes
  - Robust to outliers
- CHAMELEON
  - Tries to dynamically model the “density” and “shape” characteristics of the dataset.
  - Based on several key ideas:
    - Sparsification of the proximity graph.
    - Partitioning the data into clusters that are relatively pure subclusters of the “true” clusters.
    - Merging based on preserving characteristics of clusters.
  - Does not require number of clusters
  - Robust to noise

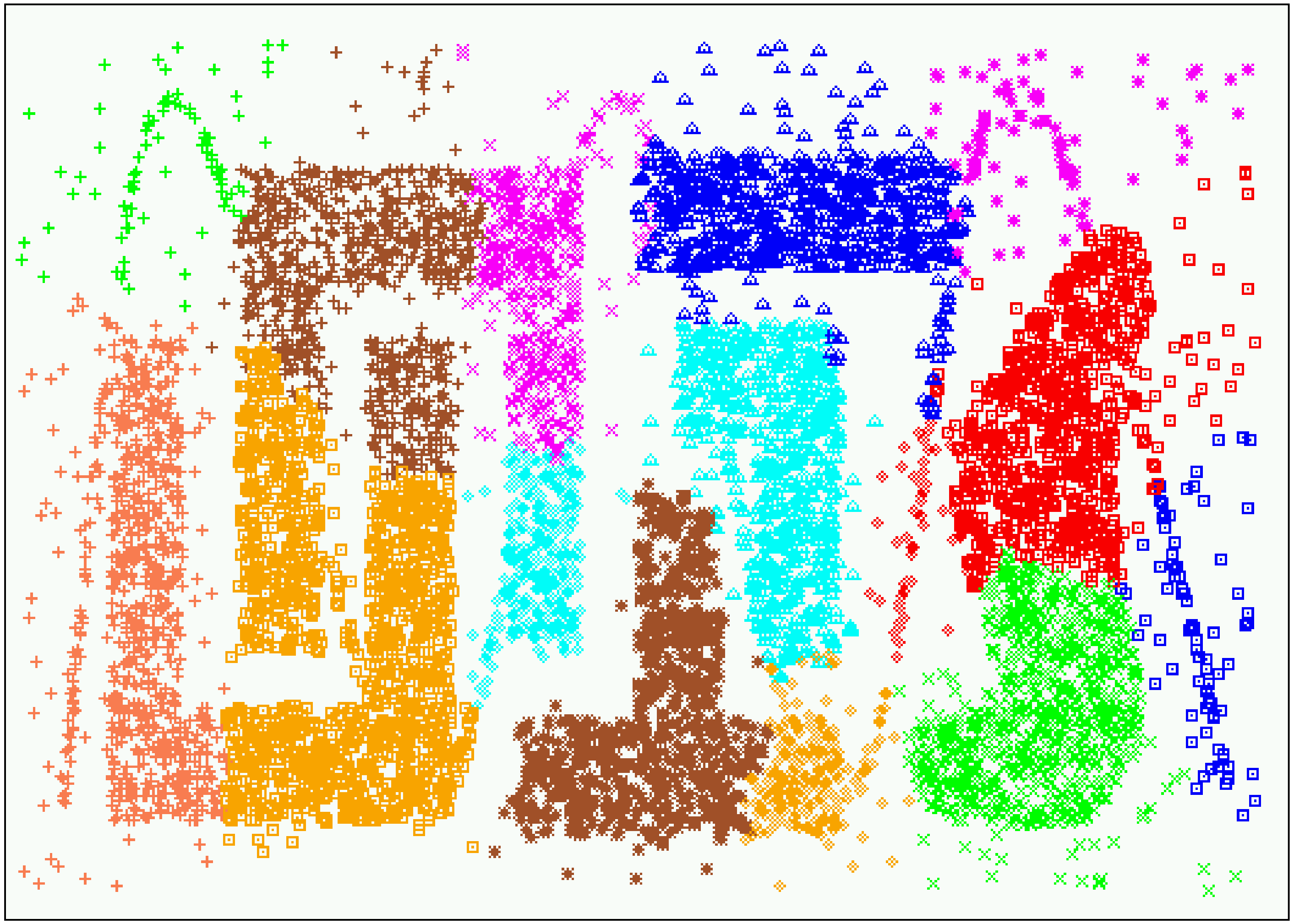
Experimental Results: CHAMELEON



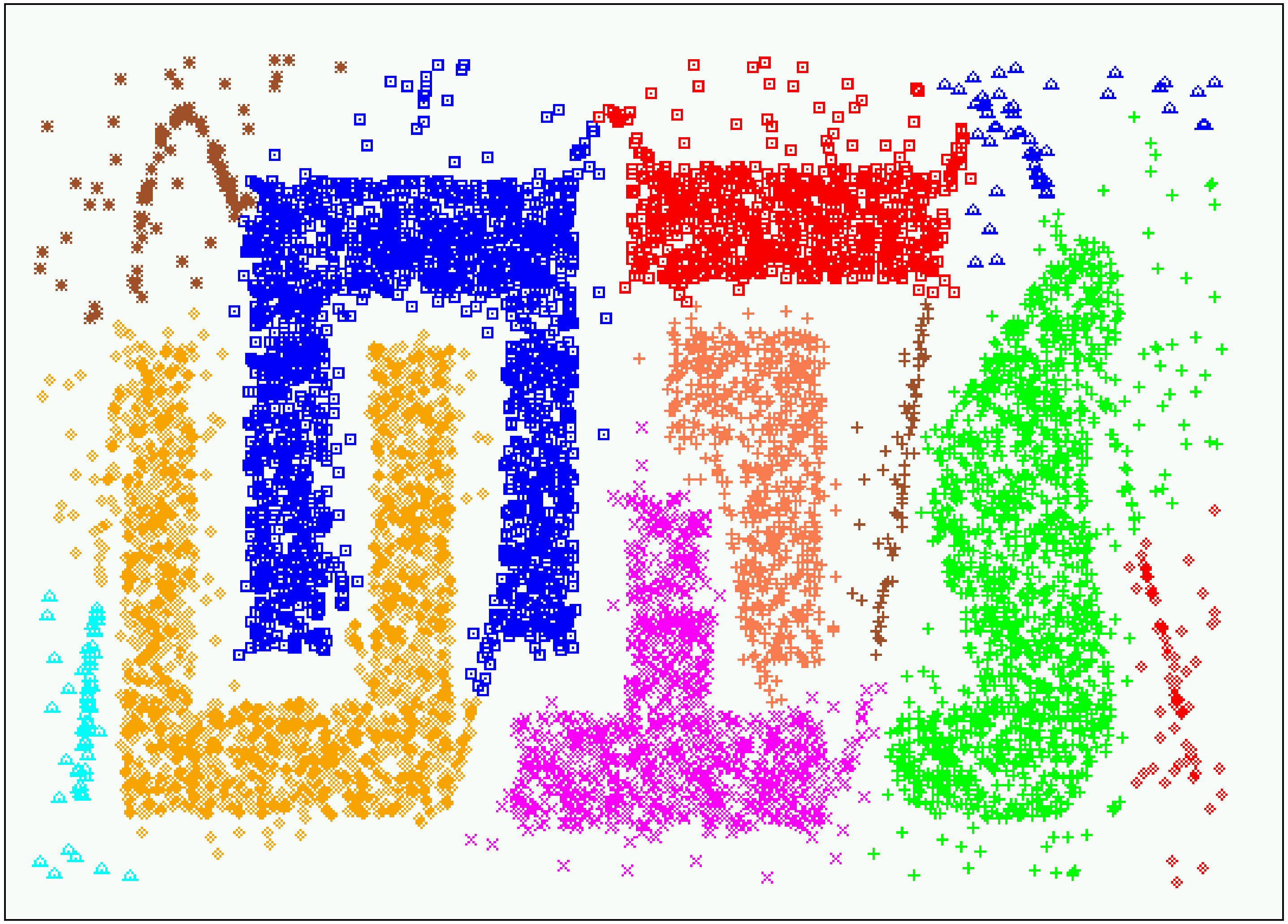
Experimental Results: CURE (*10 clusters*)



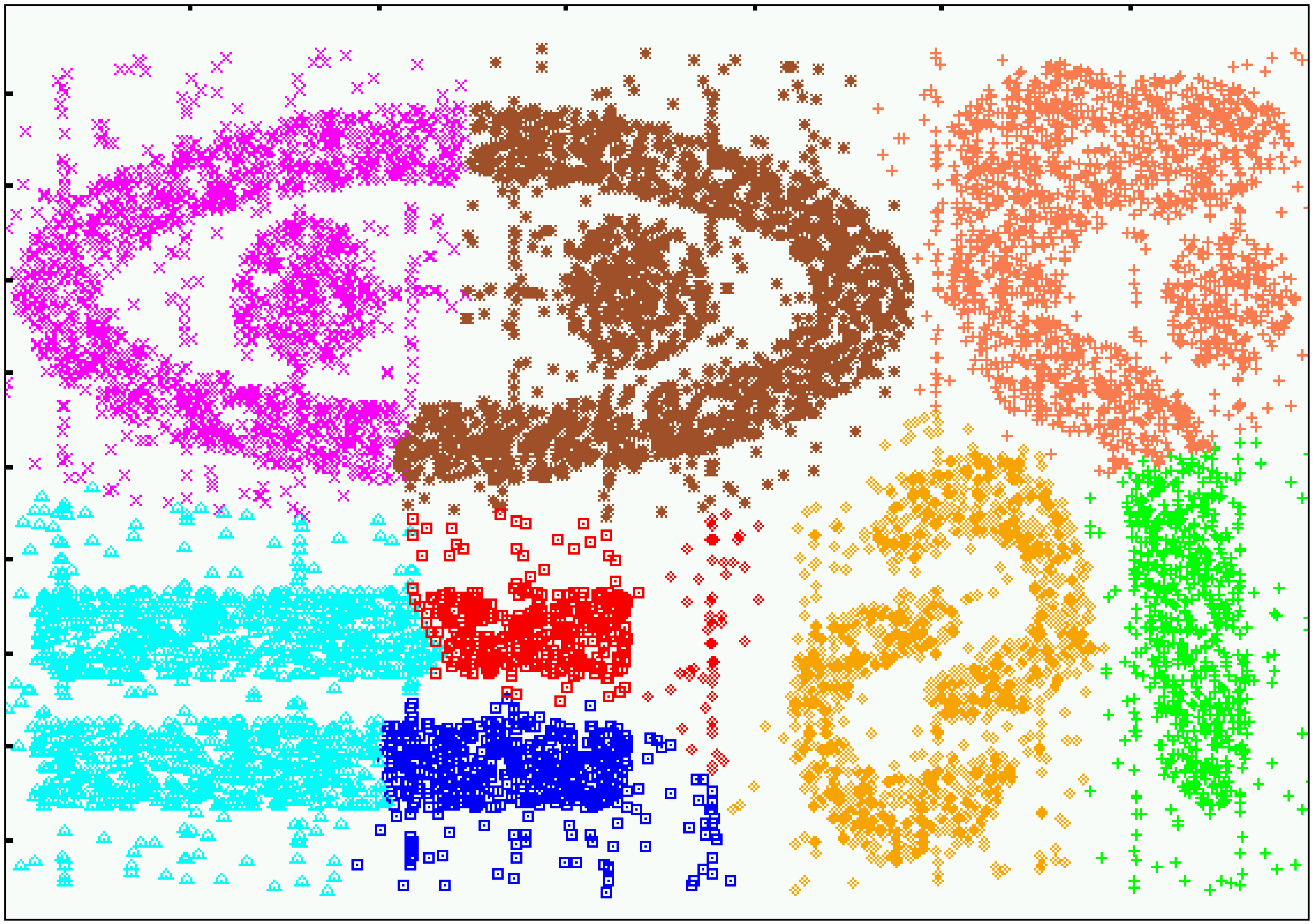
Experimental Results: CURE (*15 clusters*)



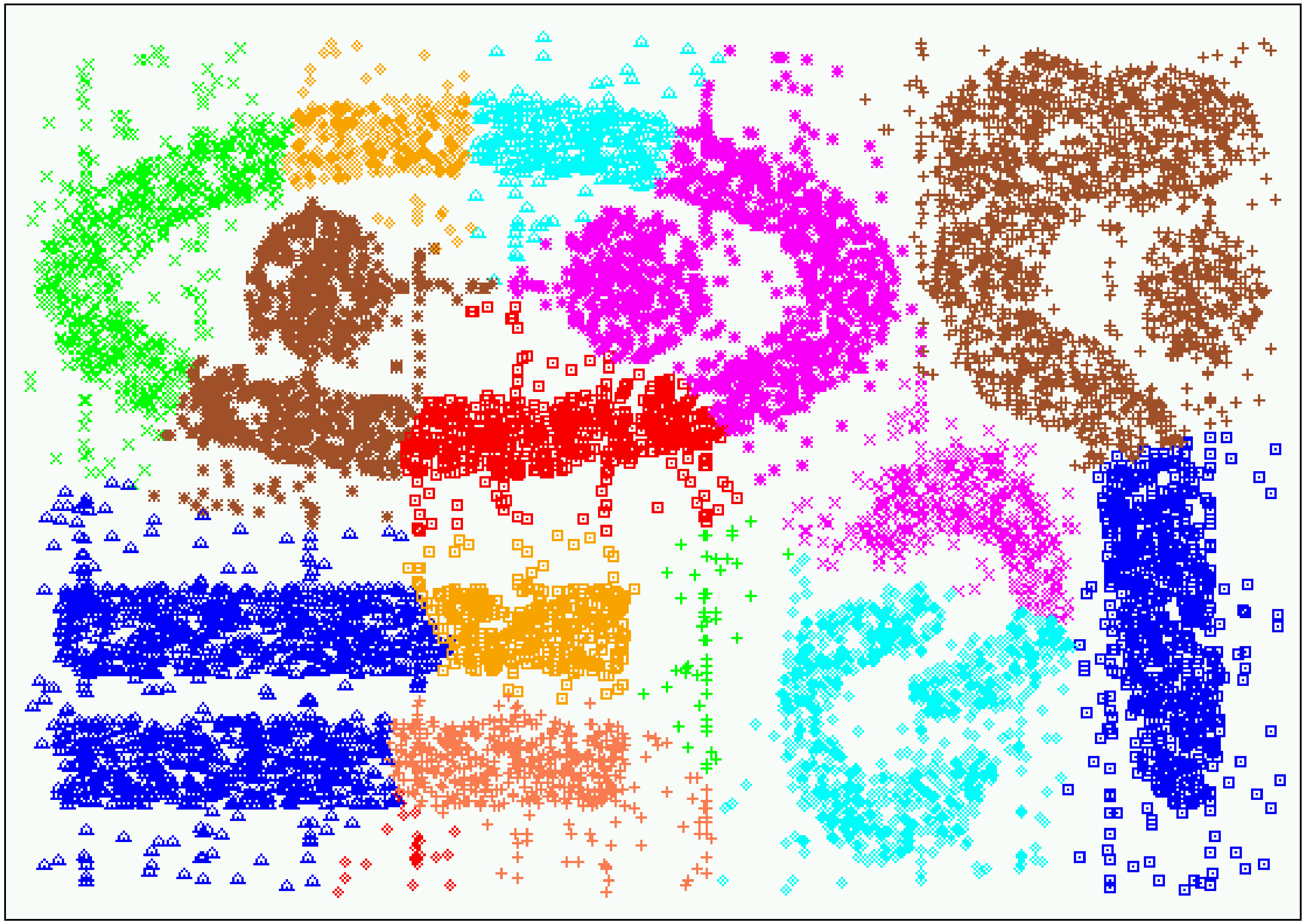
Experimental Results: CHAMELEON



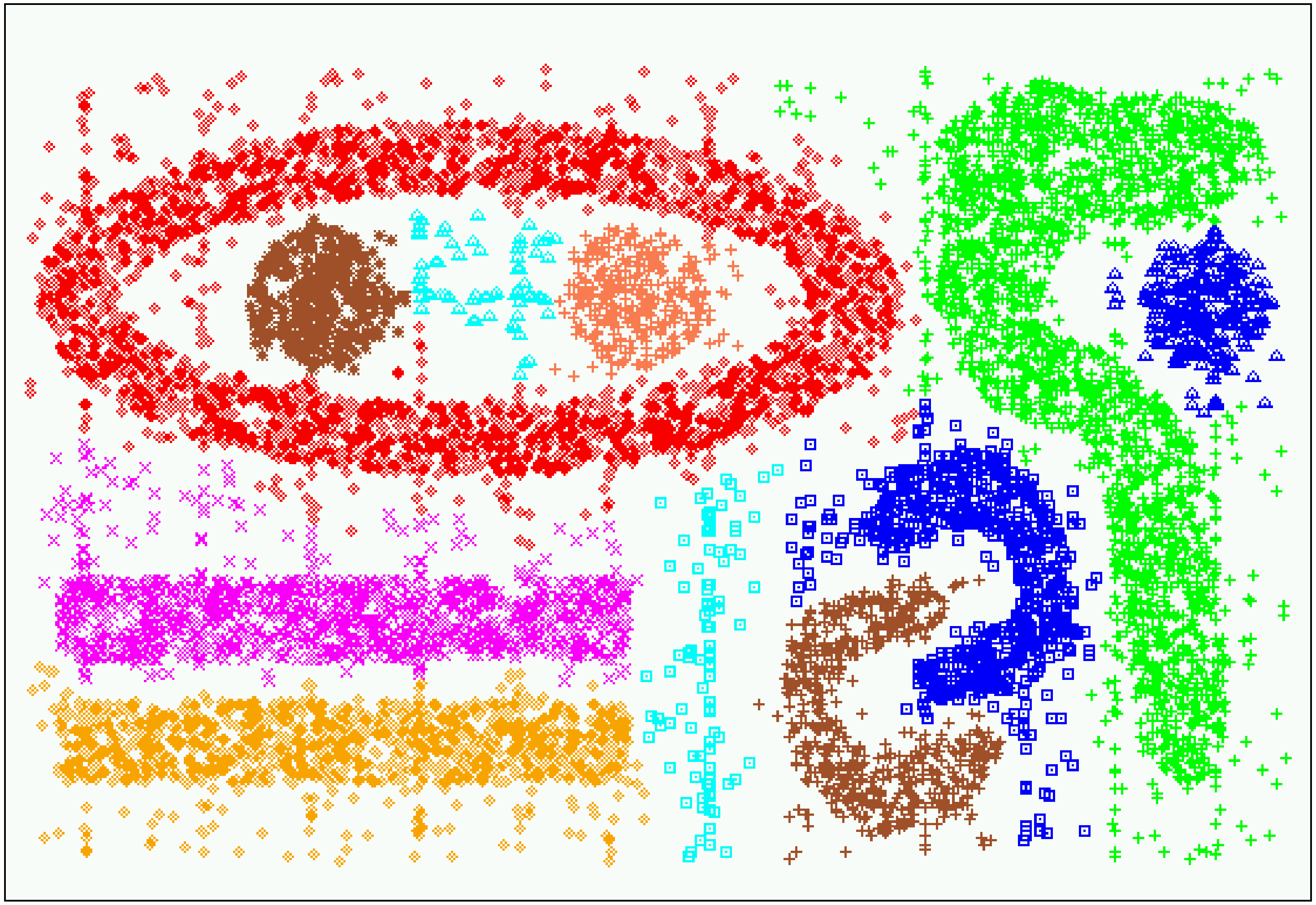
Experimental Results: CURE (*9 clusters*)



Experimental Results: CURE (*15 clusters*)



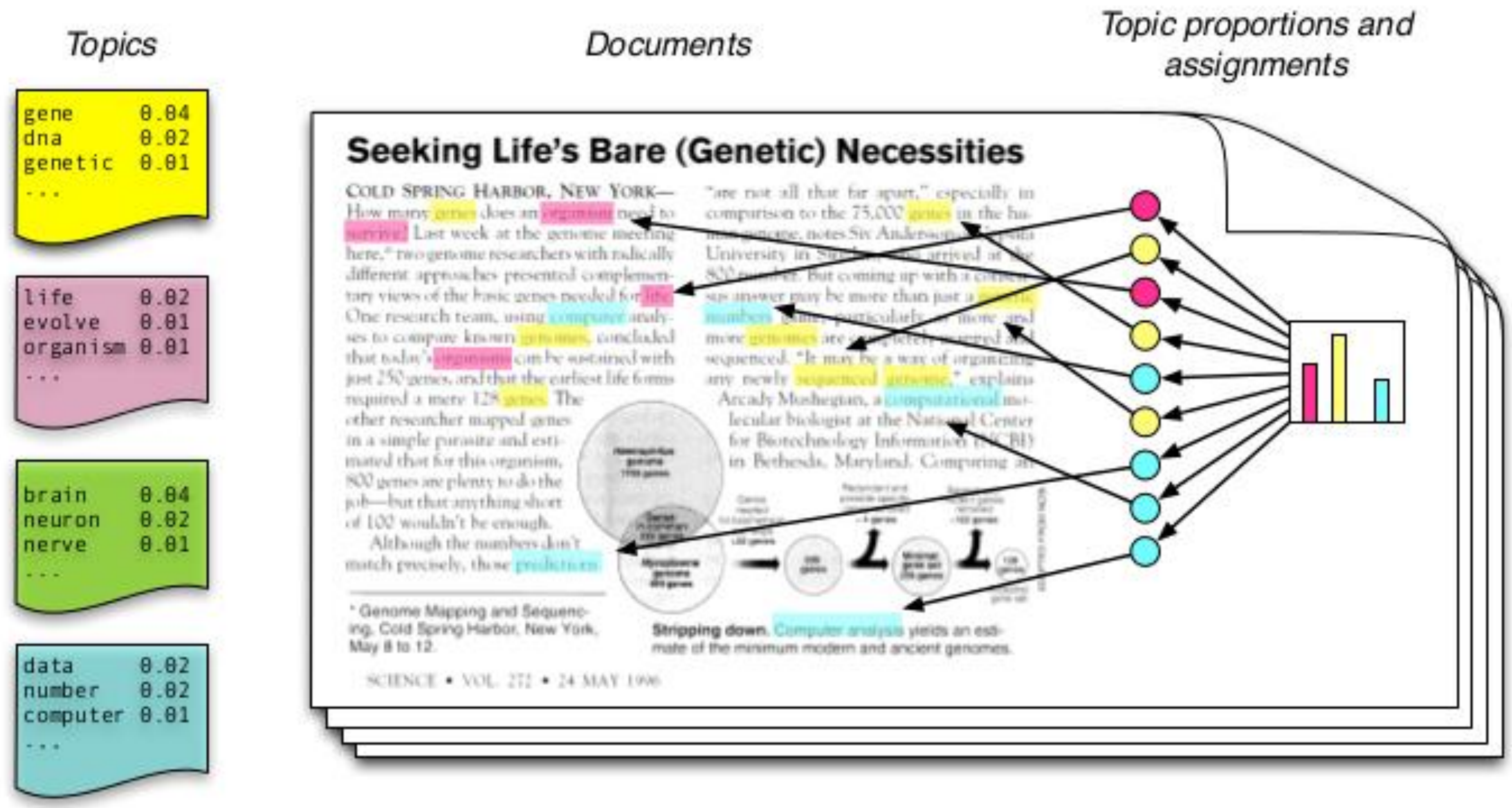
Experimental Results: CHAMELEON



Probabilistic Topic Models

- In general, a document contains multiple ideas/themes/topics
- Assume that a small number of latent (hidden) topics/themes describe the collection of documents. Then:
  - Find the set of latent topics
  - Annotate documents given the topics
  - Group documents based on their themes rather than vocabulary

LDA


Topics
gene 0.04
dna 0.02
genetic 0.01
...
life 0.02
evolve 0.01
organism 0.01
...
brain 0.04
neuron 0.02
nerve 0.01
...
data 0.02
number 0.02
computer 0.01
...
Documents
Topic proportions and assignments
Seeking Life's Bare (Genetic) Necessities
Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.
SCIENCE • VOL. 272 • 24 MAY 1996

- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

Fig. from Blei, David M., Probabilistic Topic Models presentation, Sept. 2, 2012.

Clustering with topic models

$$
\begin{array}{ll|cccccc}
 & & \text{LION} & \text{TIGER} & \text{CHEETAH} & \text{JAGUAR} & \text{PORSCHE} & \text{FERRARI} \\
\hline
\text{CATS} & \overline{X}_1 & 2/41 & 2/41 & 1/41 & 2/41 & 0 & 0 \\
 & \overline{X}_2 & 2/41 & 3/41 & 3/41 & 3/41 & 0 & 0 \\
 & \overline{X}_3 & 1/41 & 1/41 & 1/41 & 1/41 & 0 & 0 \\
\text{BOTH} & \overline{X}_4 & 2/41 & 2/41 & 2/41 & 3/41 & 1/41 & 1/41 \\
\text{CARS} & \overline{X}_5 & 0 & 0 & 0 & 1/41 & 1/41 & 1/41 \\
 & \overline{X}_6 & 0 & 0 & 0 & 2/41 & 1/41 & 2/41
\end{array}
\approx
\begin{array}{l|cc}
 & \text{CATS} & \text{CARS} \\
\hline
\overline{X}_1 & 2/8 & 0 \\
\overline{X}_2 & 3/8 & 0 \\
\overline{X}_3 & 1/8 & 0 \\
\overline{X}_4 & 2/8 & 1/4 \\
\overline{X}_5 & 0 & 1/4 \\
\overline{X}_6 & 0 & 2/4
\end{array}
$$

- Cluster document vectors in the topic space
- Probabilistic proximity measures may be more appropriate, e.g. Kullback-Leibler (KL) or Jensen-Shannon divergences
- Alternatively, compute the conditional probability for a document to belong to each topic → fuzzy clustering

Fig. from Aggarwal, Charu C., Data Mining. Springer, 2015, p.444.

CLUSTER VALIDITY

Cluster validity

- For supervised classification we have a variety of measures to evaluate how good our model is.
  - Accuracy, precision, recall, etc.

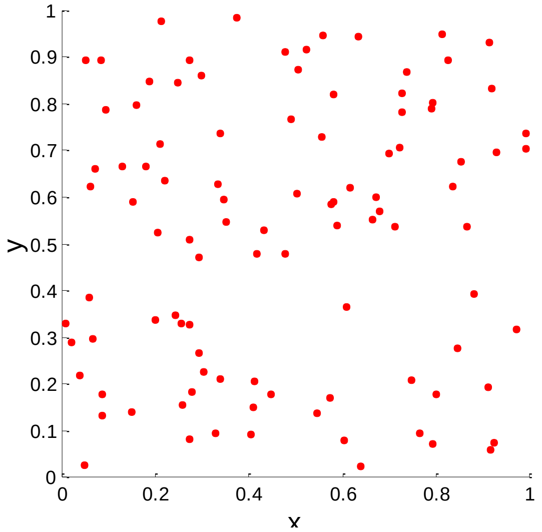
- For cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters?

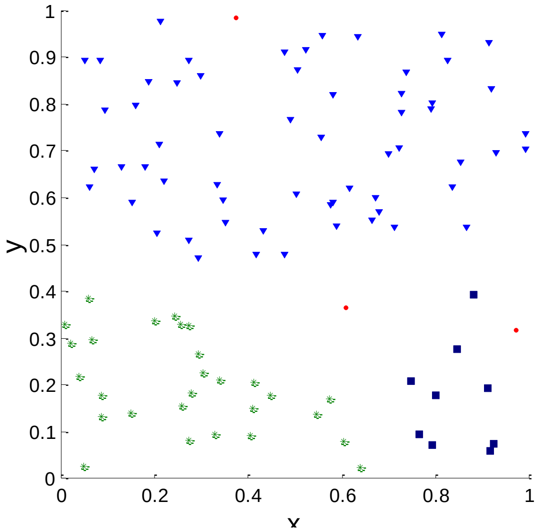
- But “clusters are in the eye of the beholder”!

- Then why do we want to evaluate them?
  - To avoid finding patterns in noise.
  - To compare clustering algorithms.
  - To compare two sets of clusters.
  - To compare two clusters.

Clusters found in random data

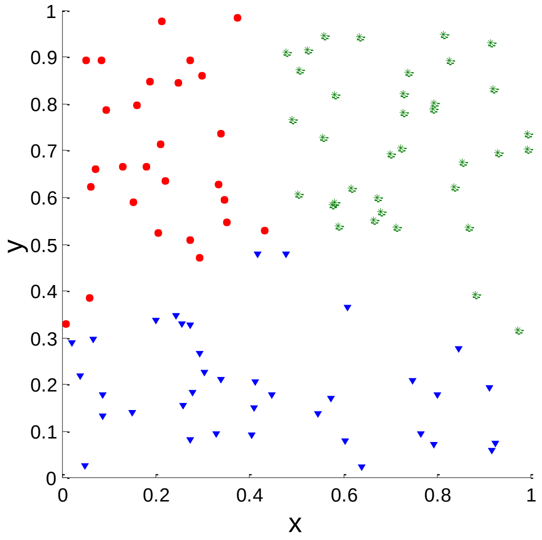
Random Points

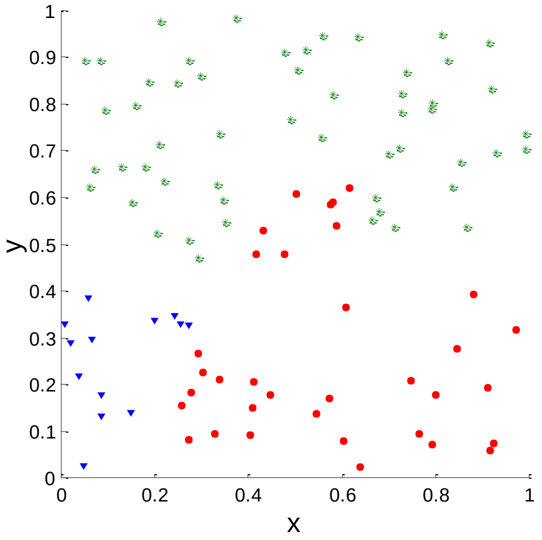




DBSCAN

K-means





Complete Link

Different aspects of cluster validation

- Determining the clustering tendency of a set of data:
  - Is there a non-random structure in the data?
- Comparing the results of a cluster analysis to externally known results.
  - Do the clusters contain objects of mostly a single class label?
- Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
  - Look at various intra- and inter-cluster data-derived properties.
- Comparing the results of two different sets of cluster analyses to determine which is better.
- Determining the “correct” number of clusters.

- The evaluation can be done for the entire clustering solution or just for selected clusters.

Measures of cluster validity

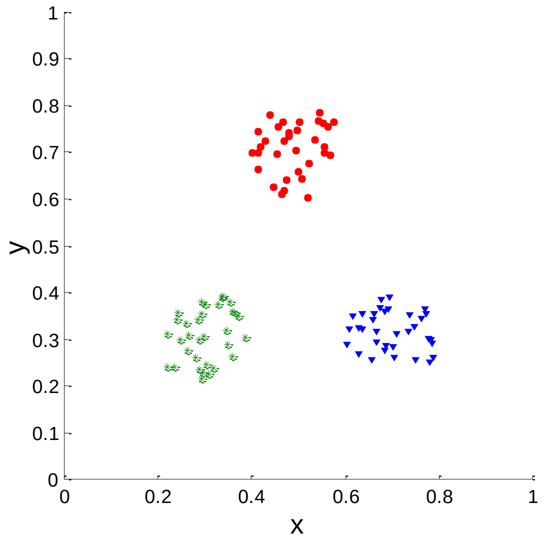
- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy, purity, f-score, etc.
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE) (or any other of the criterion functions that we discussed).
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy.

Measuring cluster validity via correlation

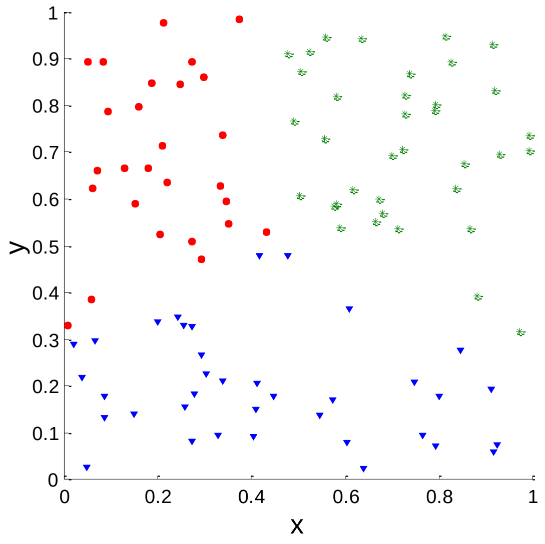
- Two matrices:
  - Proximity (distance) matrix of the data (e.g., pair-wise cosine similarity (Euclidean distance)).
  - Ideal proximity matrix that is implied by the clustering solution.
    - One row and one column for each data point.
    - An entry is 1 if the associated pair of points belong to the same cluster.
    - An entry is 0 if the associated pair of points belongs to different clusters.
- Compute the correlation between the two matrices.
  - i.e., the correlation between the vectorized matrices.
  - (make sure that the ordering of the data points is the same in both matrices)
- High (low) correlation indicates that points that belong to the same cluster are close to each other.
- Not a good measure for some density or contiguity based clusters.

Measuring cluster validity via correlation

Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following two data sets.



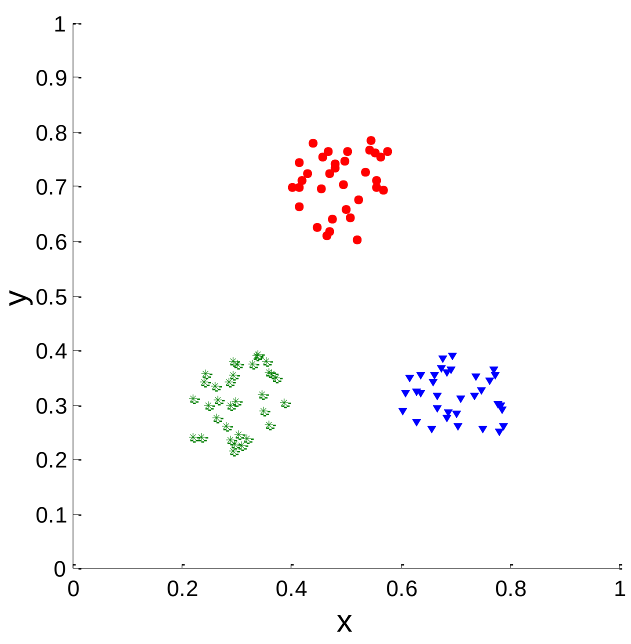
Corr = -0.9235

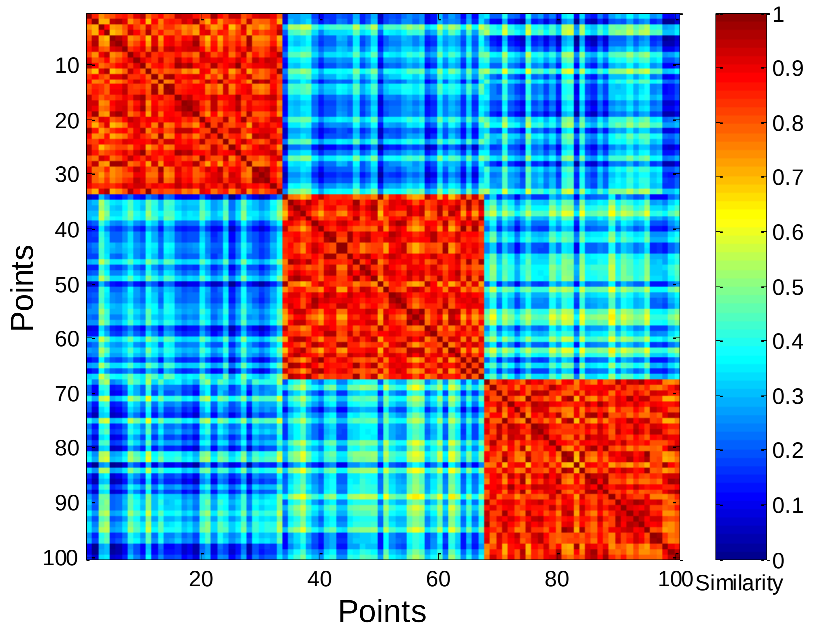


Corr = -0.5810

Using similarity matrix for cluster validation

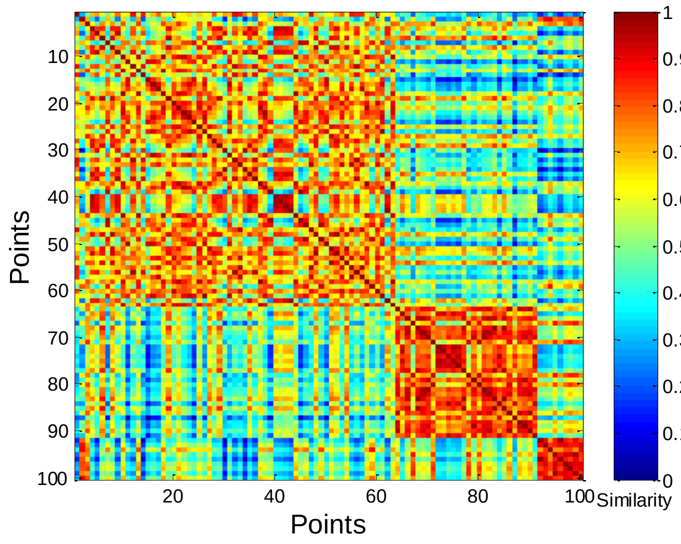
Order the similarity matrix with respect to cluster labels and inspect visually.

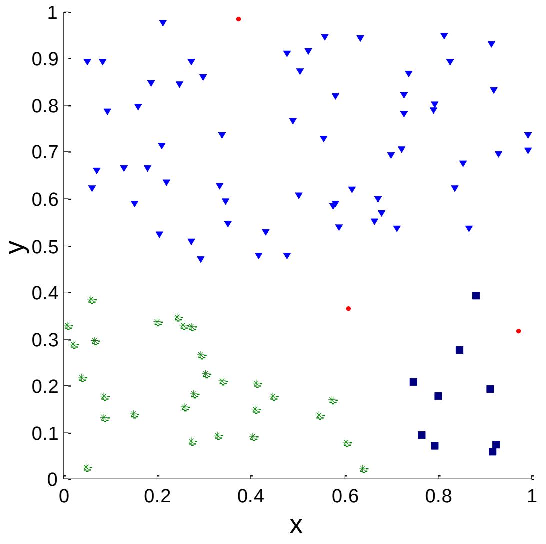




Using similarity matrix for cluster validation

Clusters in random data are not so crisp.

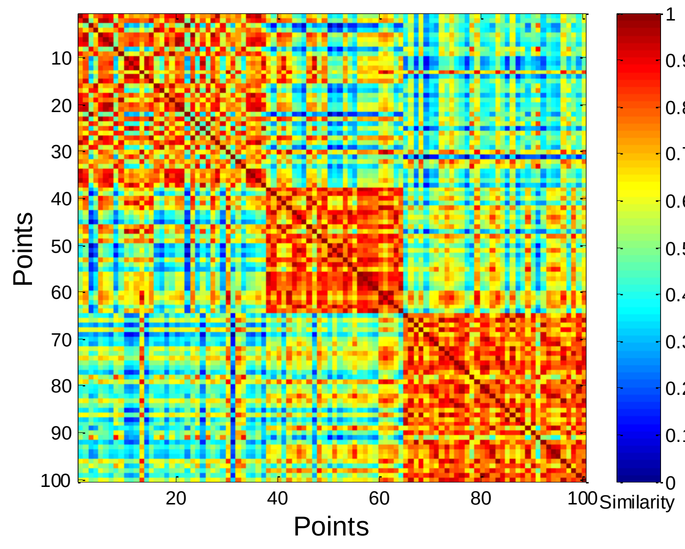


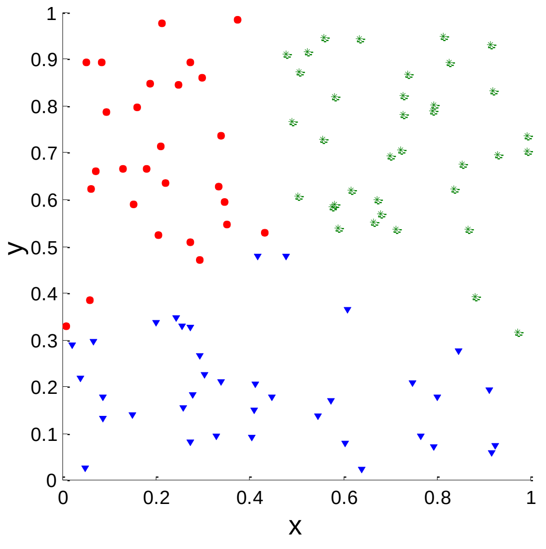


DBSCAN

Using similarity matrix for cluster validation

Clusters in random data are not so crisp.

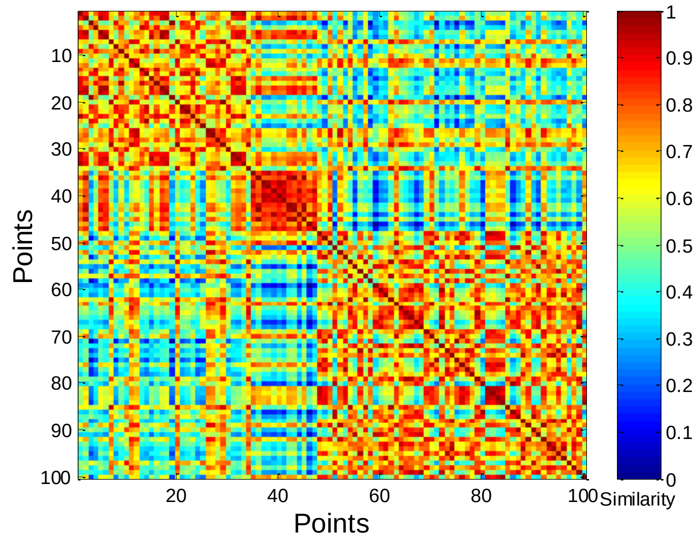


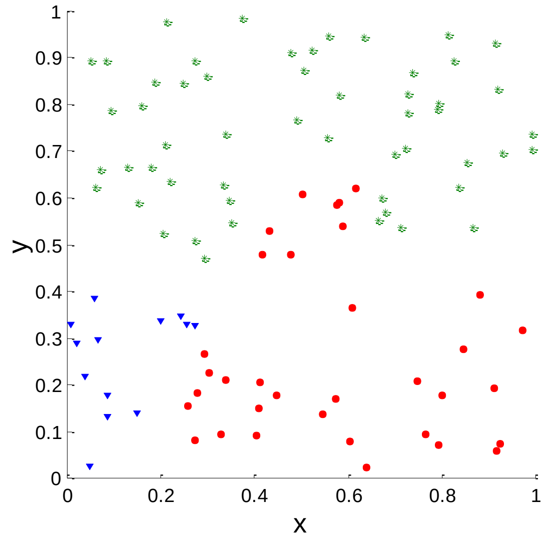


K-means

Using similarity matrix for cluster validation

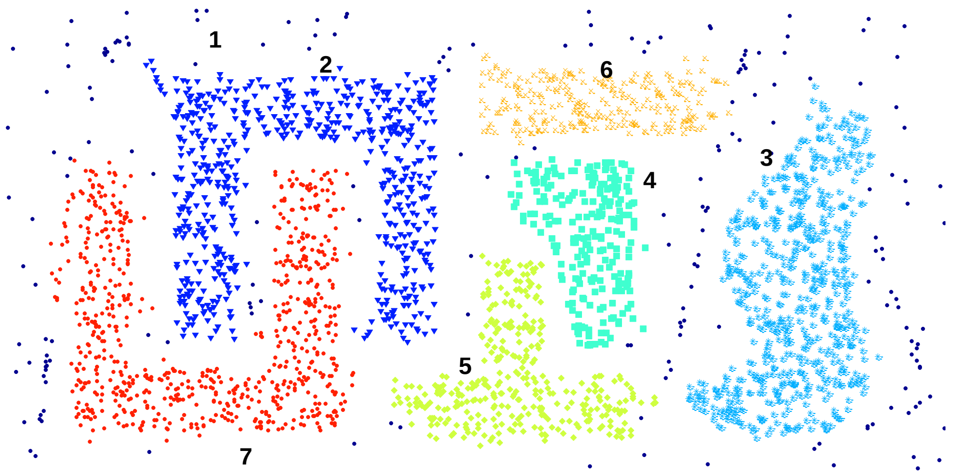
Clusters in random data are not so crisp.

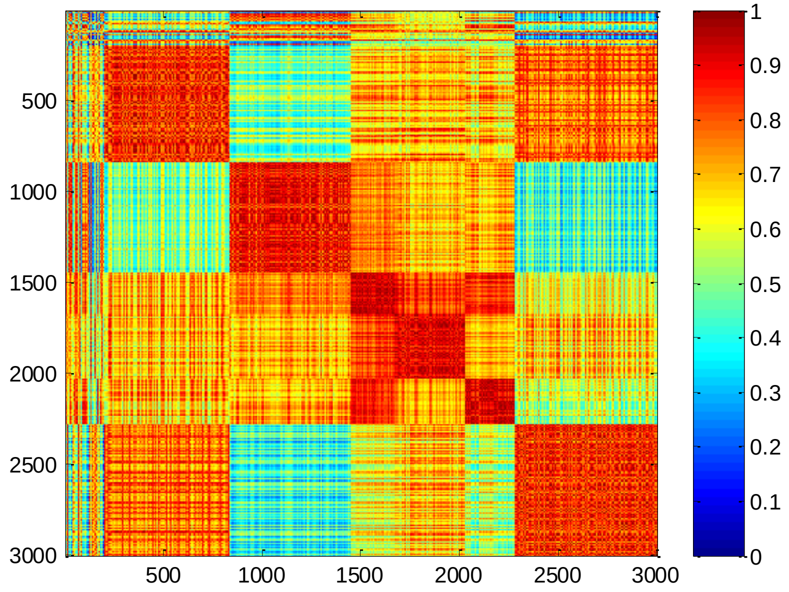




Complete Link

Using similarity matrix for cluster validation

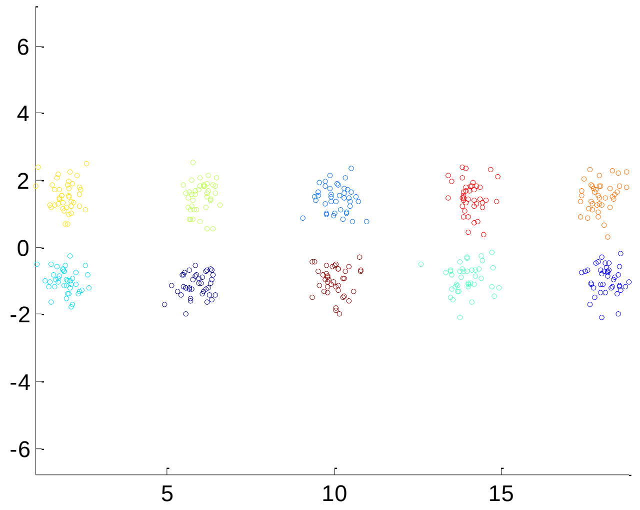


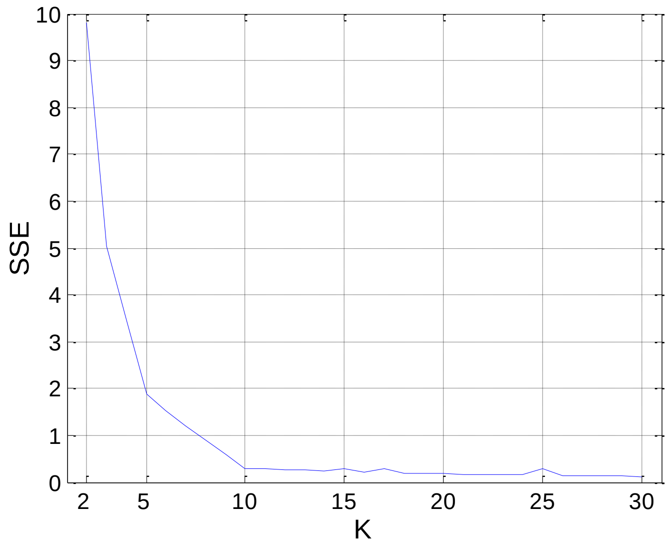


DBSCAN

Internal measures: SSE

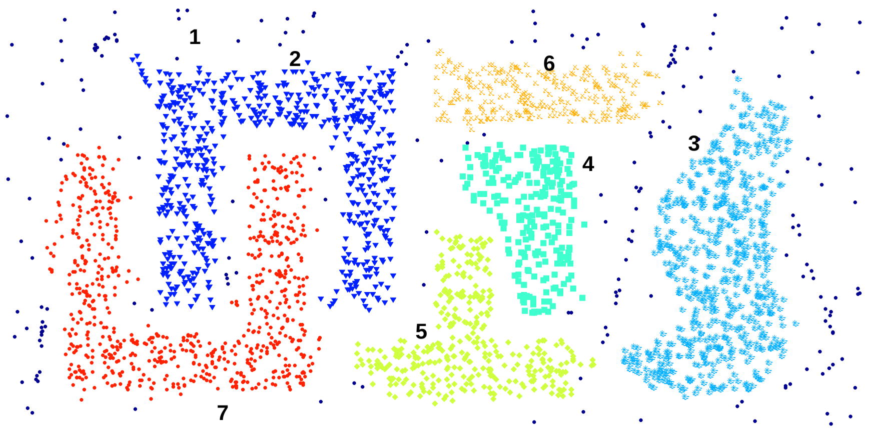
- Clusters in more complicated figures aren't well separated.
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information.
- SSE is good for comparing two clusterings or two clusters (average SSE).
- Can also be used to estimate the number of clusters.

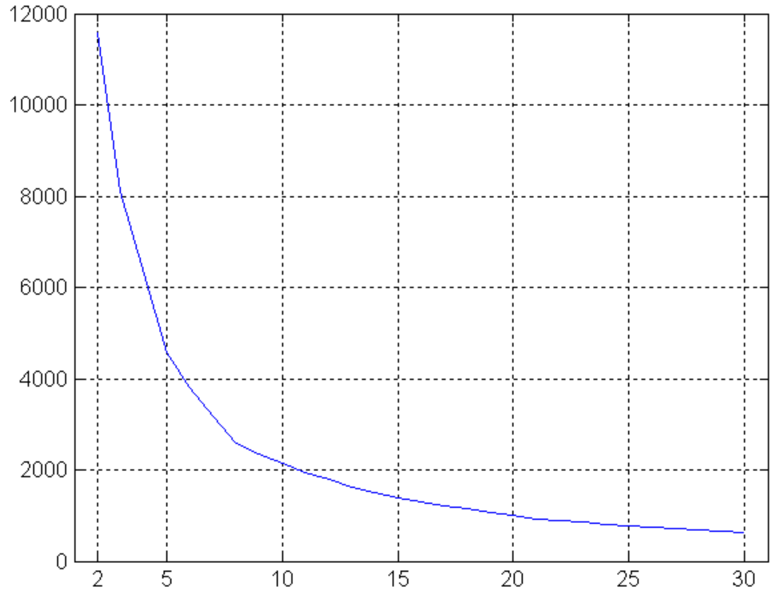




Internal measures: SSE

SSE curve for a more complicated data set.





SSE of clusters found using K-means

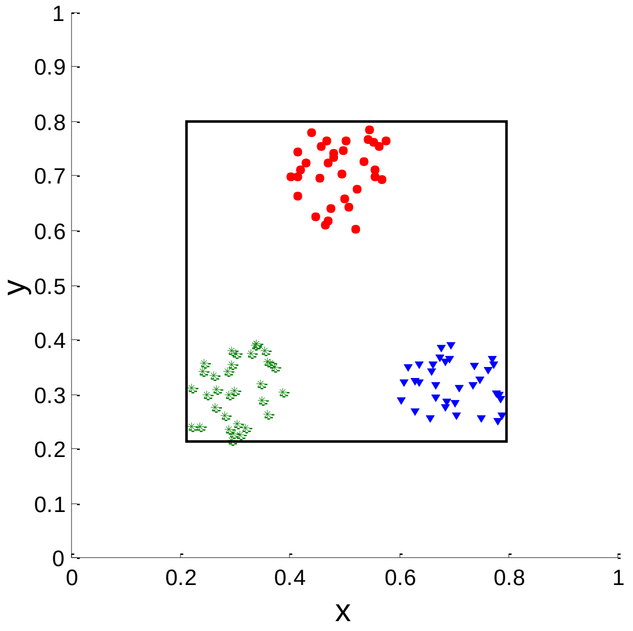
Framework for cluster validity

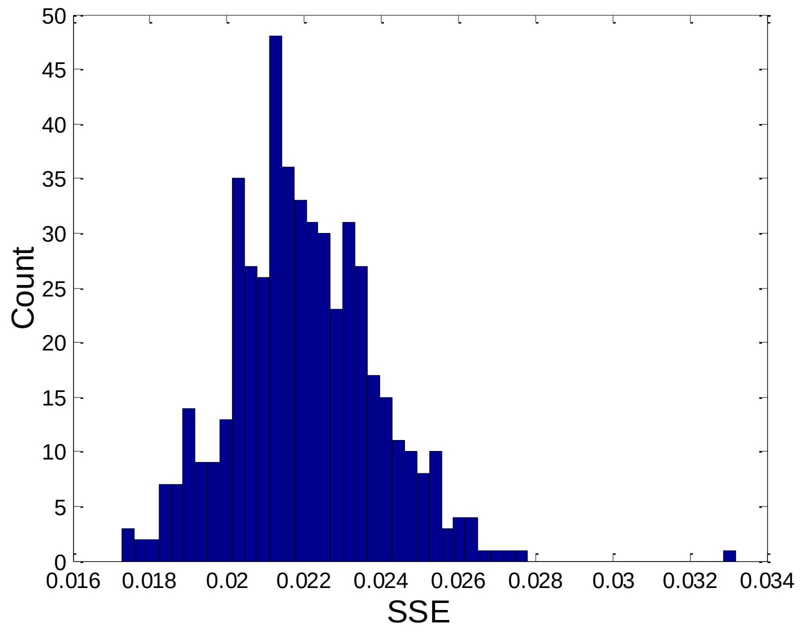
- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has a value of 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity.
  - The more "atypical" a clustering result is, the more likely it represents valid structure in the data.
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - If the value of the index is unlikely, then the cluster results are valid.
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant.

Statistical framework for SSE

Example

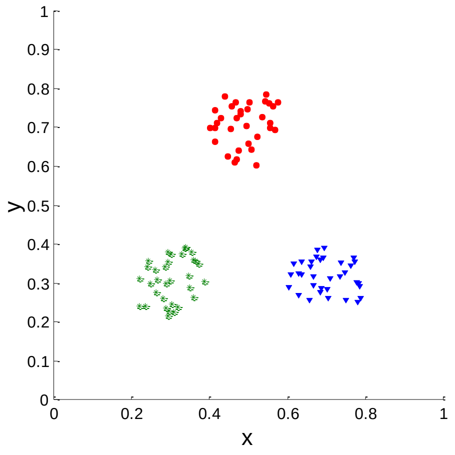
- Compare SSE of 0.005 against three clusters in random data.
- Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values.



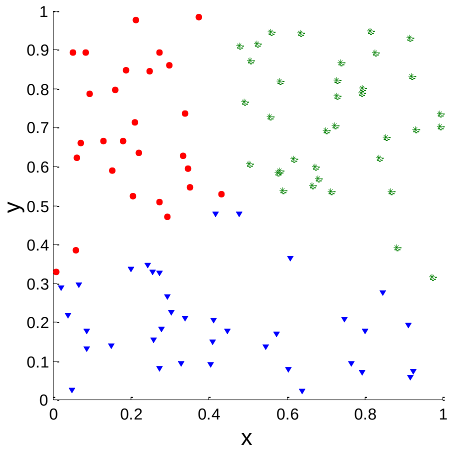


Statistical framework for correlation

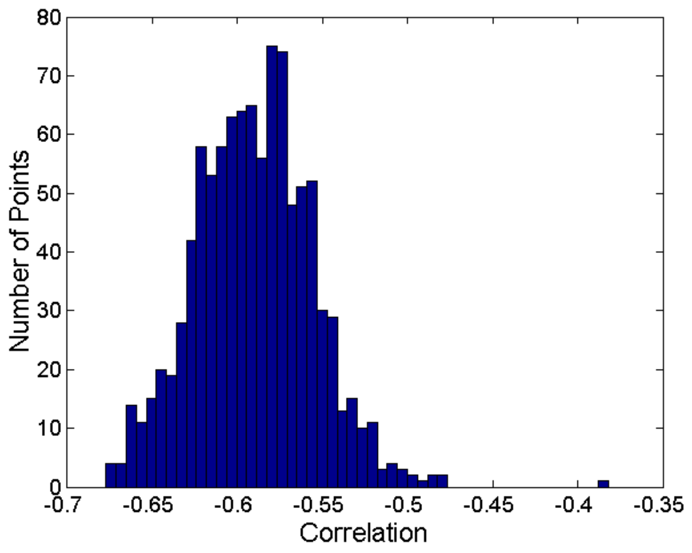
Correlation of ideal similarity and proximity matrices for the K-means clusterings of the following two data sets.



Corr = -0.9235



Corr = -0.5810



Internal measures: Cohesion and separation

- The cluster cohesion measures how closely related are objects in a cluster.
  - Example: SSE
- The cluster separation measures how distinct or well-separated a cluster is from other clusters.
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

$$WSS = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

  - Separation is measured by the between cluster sum of squares

$$BSS = \sum_{i} |C_i| (m - m_i)^2$$

where $|C_i|$ is the size of cluster i.

Internal measures: Cohesion and separation

- The cluster cohesion measures how closely related are objects in a cluster.
  - Example: SSE
- The cluster separation measures how distinct or well-separated a cluster is from other clusters.
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

  - Separation is measured by the between cluster sum of squares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

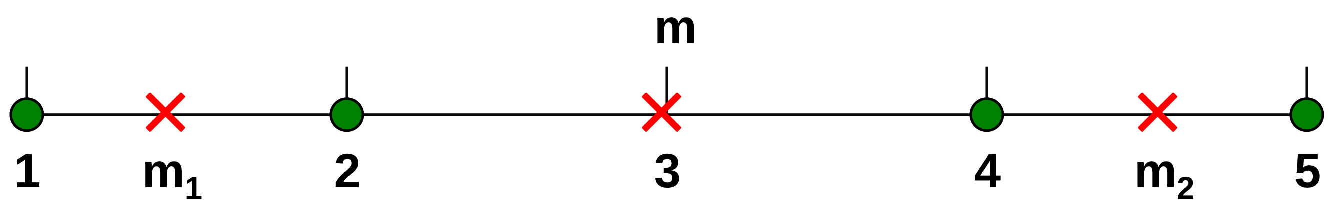
where $|C_i|$ is the size of cluster i.

What can you say about WSS+BSS?

Internal measures: Cohesion and separation

Example: SSE

BSS + WSS = constant



K=1 cluster:

$$WSS = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$$

$$BSS = 4 \times (3-3)^2 = 0$$

$$Total = 10 + 0 = 10$$

K=2 clusters:

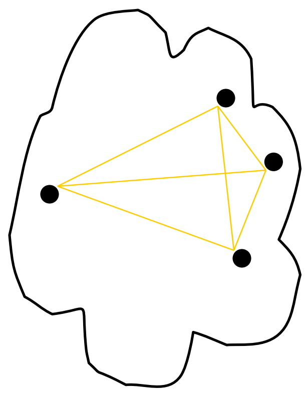
$$WSS = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$

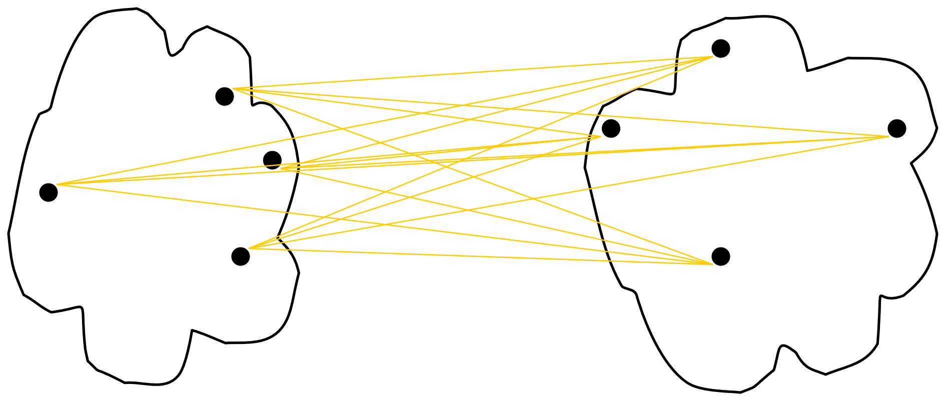
$$BSS = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$$

$$Total = 1 + 9 = 10$$

Internal measures: 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

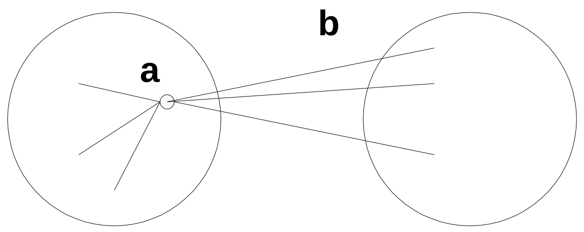

separation

Internal measures: Silhouette coefficient

- The silhouette coefficient combines ideas of both cohesion and separation, but for individual points, as well as clusters and clusterings.
- For an individual point, *i*
  - Calculate *a* = average distance of *i* to the points in its cluster.
  - Calculate *b* = minimum (average distance of *i* to points in another cluster).
  - The silhouette coefficient for a point is then given by

    s = (b – a)/max(a,b)

  - Typically between 0 and 1.
  - The closer to 1 the better.



- Can calculate the average silhouette coefficient for a cluster or a clustering.

External measures of cluster validity: Entropy and purity

Table 5.9. K-means Clustering Results for LA Document Data Set

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

entropy For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the 'probability' that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula $e_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$, where the L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^{K} \frac{m_i}{m} e_j$, where m_j is the size of cluster j, K is the number of clusters, and m is the total number of data points.

purity Using the terminology derived for entropy, the purity of cluster j, is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$.

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

SCALING CLUSTERING ALGORITHMS

Computational & memory complexity

- What should be the computational and memory complexity of a “scalable” clustering algorithm?
- How do the different algorithms that we saw meet those requirements?
- How do I improve the complexity of clustering algorithms?

Reducing complexity

- Use sampling to compute an initial clustering and then deal with the non-sampled objects.
  - Sample objects and/or dimensions.
  - Use ensemble based methods for putting things back together.
- Use methods that rely on sparse graphs.
  - Utilize the sparsity of the graph to prune the amount of computations that needs to be done in finding the “edges” of that graph.
  - Utilize approximate similarity measures.

WHICH CLUSTERING ALGORITHMS SHOULD I USE?

How to choose a clustering algorithm

- Clustering research has a long history. A vast collection of algorithms are available.
  - We only introduced several main algorithms.
- Choosing the "best" algorithm is a challenge.
  - Every algorithm has limitations and works well with certain data distributions.
  - It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any "ideal" structure or distribution required by the algorithms.
  - One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

Choose a clustering algorithm (cont …)

- Due to these complexities, the common practice is to
  - run several algorithms using different distance functions and parameter settings, and
  - then carefully analyze and compare the results.
- The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.
- Clustering is highly application dependent and to certain extent subjective (personal preferences).

Indirect evaluation

- In some applications, clustering is not the primary task, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
  - E.g. in an application, the primary task is to provide recommendations on book purchasing to online shoppers.
    - If we can cluster books according to their features, we might be able to provide better recommendations.
    - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
    - Here, we assume that the recommendation can be reliably evaluated.

Summary

- Clustering has a long history and still active
  - There are a huge number of clustering algorithms
  - More are still being proposed each year.
- We only introduced some main algorithms. There are many others, e.g.,
  - Density-based algorithms, sub-space clustering, neural networks-based methods, fuzzy clustering, co-clustering, etc.
- Clustering is hard to evaluate, but very useful in practice. This partially explains why there are still a large number of clustering algorithms being devised every year.
- Clustering is highly application-dependent and to some extent subjective.