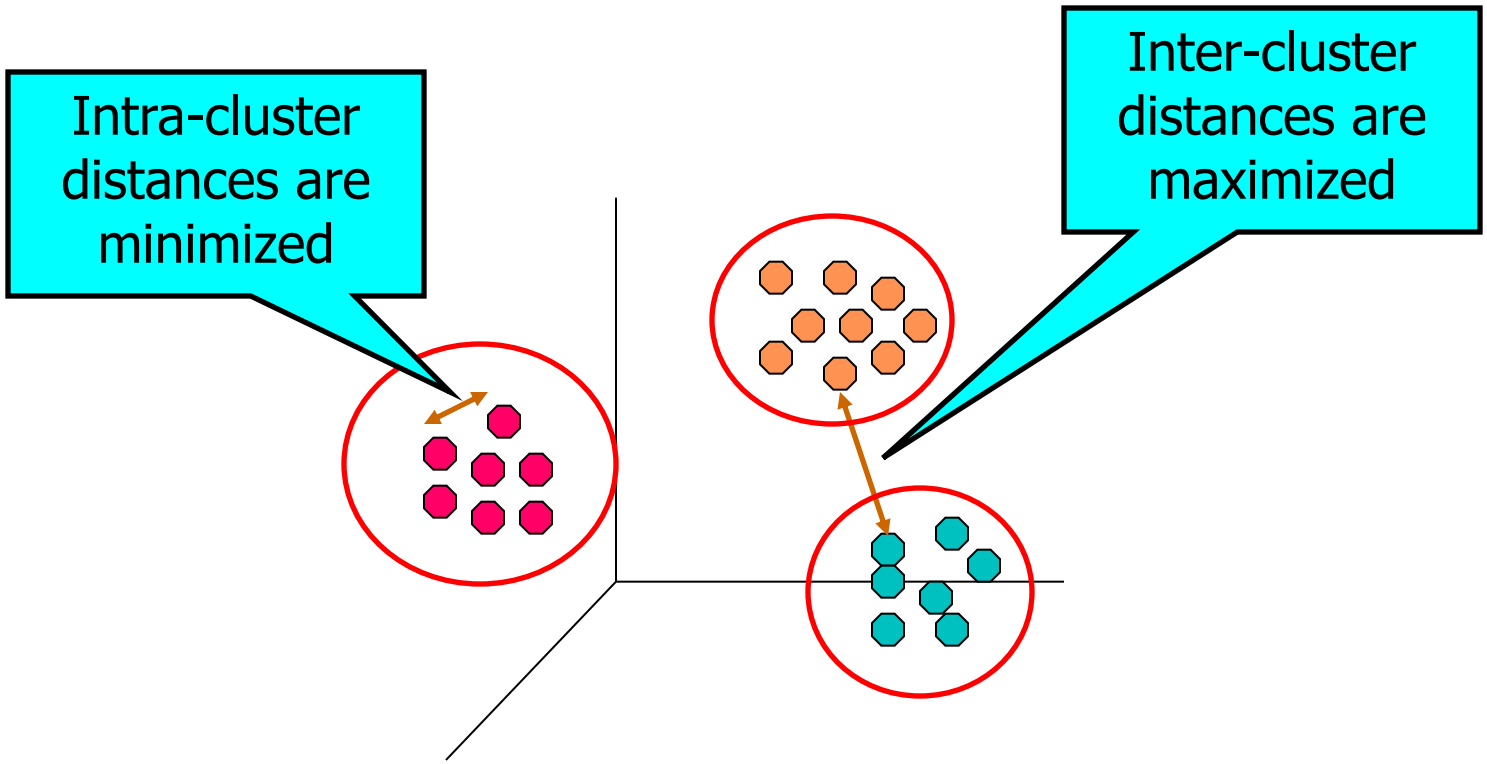
Clustering:
Basic Concepts and Algorithms

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



Applications of Cluster Analysis

- **Biology**: taxonomy of living things
- **Information retrieval**: document clustering
- **Land use**: Identification of areas of similar land use in an earth observation database
- **Marketing**: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs (market segmentation);
- **City-planning**: Identifying groups of houses according to their house type, value, and geographical location;
- **Earth-quake studies**: Observed earth quake epicenters should be clustered along continent faults;
- **Climate**: understanding earth climate, find patterns of atmospheric and ocean;
- **Economic Science**: market analisys

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 as a Preprocessing Tool

- **Summarization**:
  - Preprocessing for regression, Principal c<omponent Analisys (PCA), classification, and association analysis
- **Compression**:
  - Image processing: vector quantization
- **Finding K-nearest Neighbors**
  - Localizing search to one or a small number of clusters
- **Outlier detection**
  - Outliers are often viewed as those “far away” from any cluster

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - **high intra-class similarity**: **cohesive** within clusters
  - **low inter-class similarity**: **distinctive** between clusters
- The quality of a clustering method depends mainly on the similarity measure used by the method

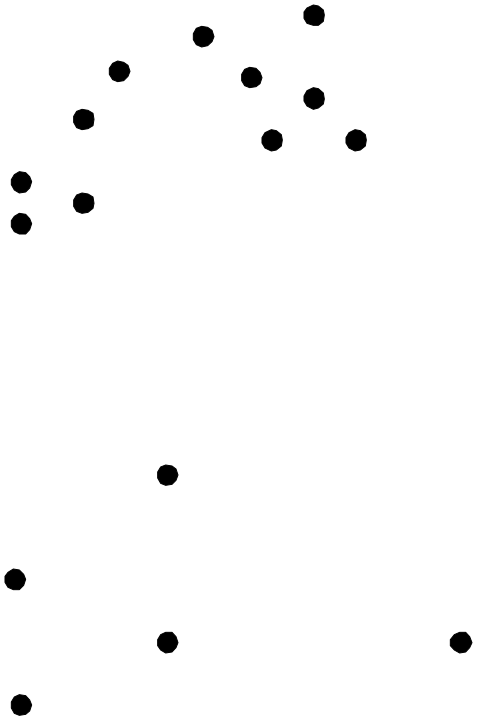
Major Clustering Approaches

- **Partitioning approach:**
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- **Hierarchical approach:**
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- **Density-based approach:**
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue

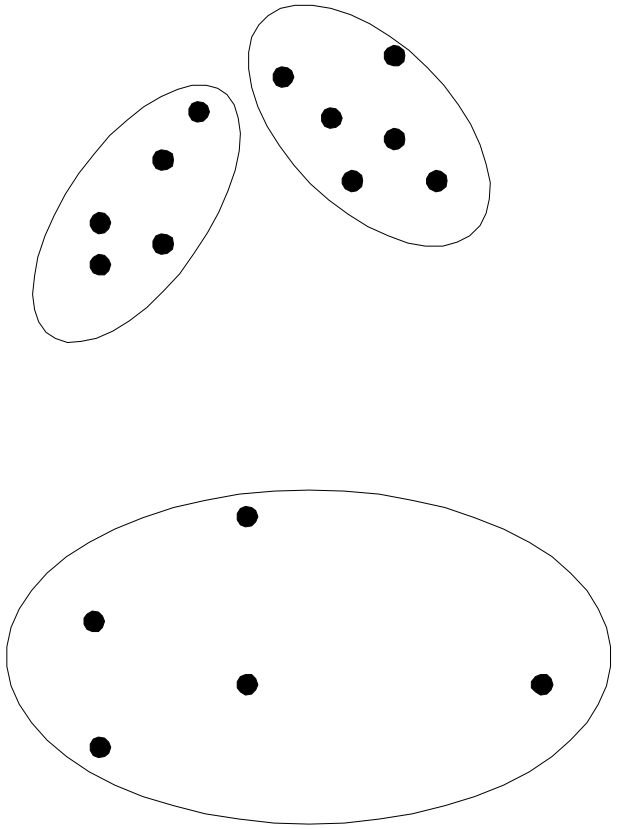
Other Clustering Approaches

- **Grid-based approach**:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE
- **Model-based:**
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- **Frequent pattern-based:**
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- **User-guided or constraint-based**:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- **Link-based clustering**:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

Partitional Clustering

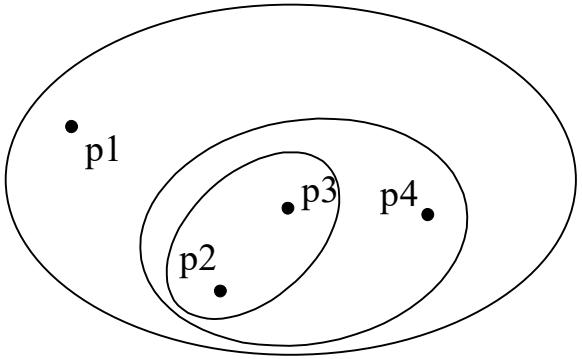


Original Points

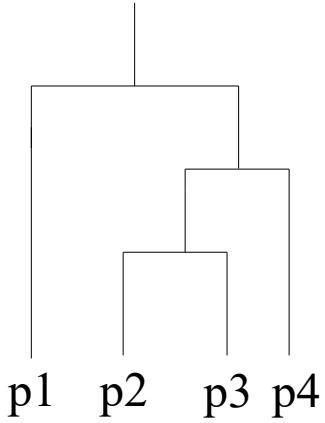


A Partitional Clustering

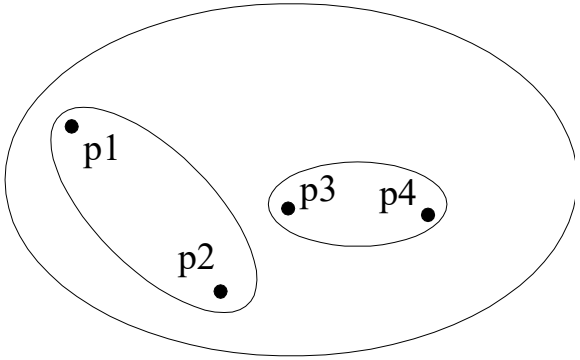
Hierarchical Clustering



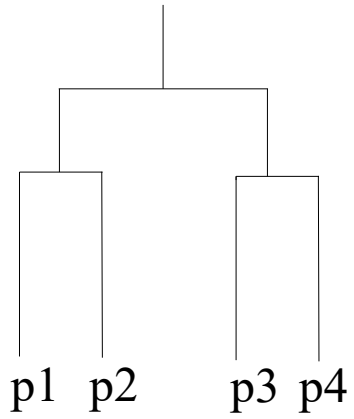
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Other Distinctions Between Sets of Clusters

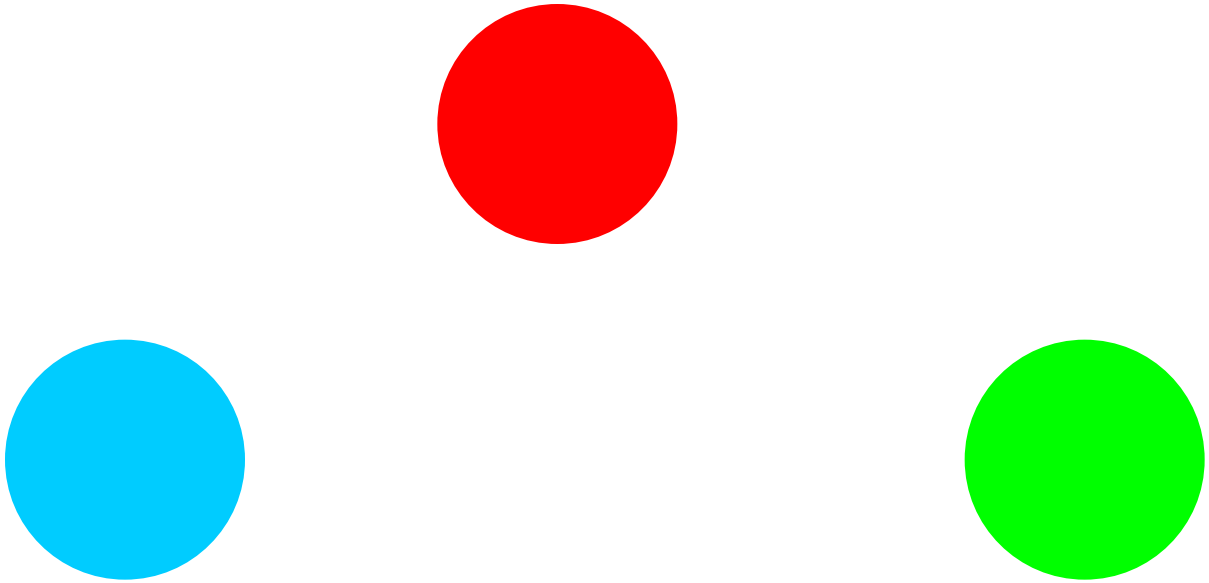
- **Exclusive versus non-exclusive**
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or ‘border’ points
- **Fuzzy versus non-fuzzy**
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must 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 clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

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

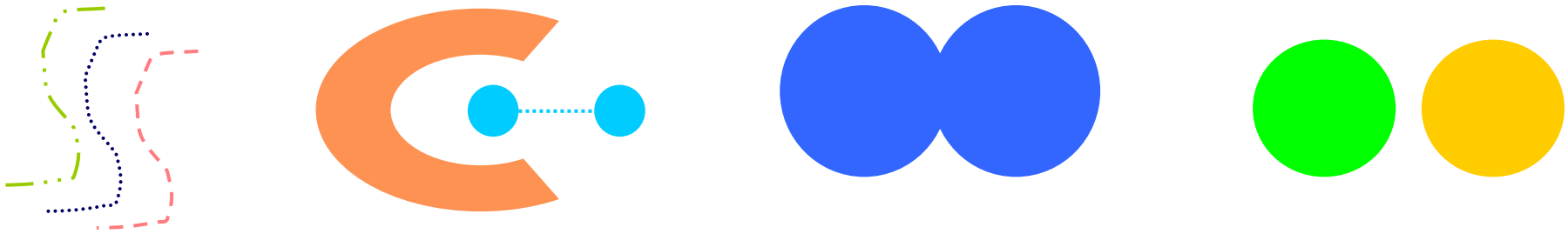
Types of Clusters: Center-Based

- 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

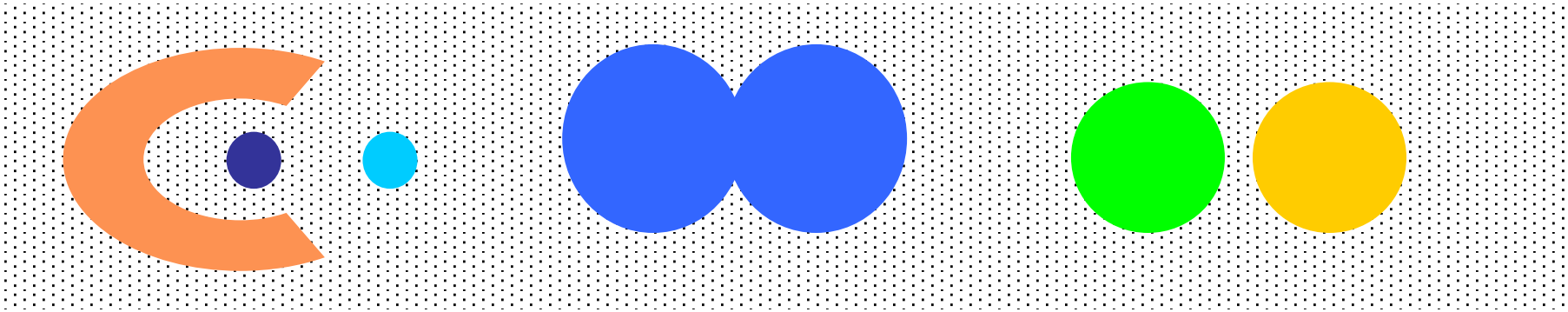
- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is **closer (or more similar) to one or more other points in the cluster than to any point not in the cluster**.



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.

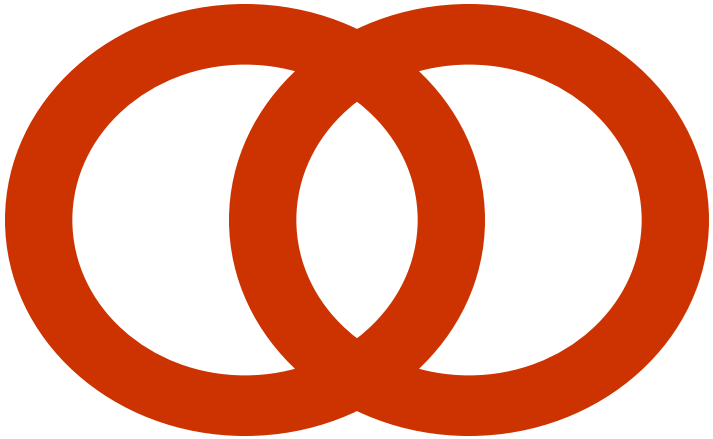


6 density-based clusters

Types of Clusters: Conceptual Clusters

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

.



2 Overlapping Circles

Types of Clusters: Objective Function

- Clusters Defined by an Objective Function
  - Finds clusters that minimize or maximize an objective function.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters by using the given objective function. (NP Hard)
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives
  - A variation of the global objective function approach is to fit the data to a parameterized model.
    - Parameters for the model are determined from the data.
    - Mixture models assume that the data is a ‘mixture' of a number of statistical distributions.

Map Clustering Problem to a Different Problem

- 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

Characteristics of the Input Data Are Important

- Type of proximity or density measure
  - Central to clustering
  - Depends on data and application
- Data characteristics that affect proximity and/or density are
  - Dimensionality
    - Sparseness
  - Attribute type
  - Special relationships in the data
    - For example, autocorrelation
  - Distribution of the data
- Noise and Outliers
  - Often interfere with the operation of the clustering algorithm

Clustering Algorithms

- Partitional clustering
  - K-means and its variants
- Hierarchical clustering
- Density-based clustering

Evaluating Partitional Clusters

- Most common measure is **Sum of Squared Error (SSE)**
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - can show that m_i corresponds to the center (mean) of the cluster
  - Given two sets of clusters, we prefer the one with the smallest error
  - One easy way **to reduce SSE** is to **increase K**, the number of clusters
    - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Partitioning Algorithms: Basic Concept

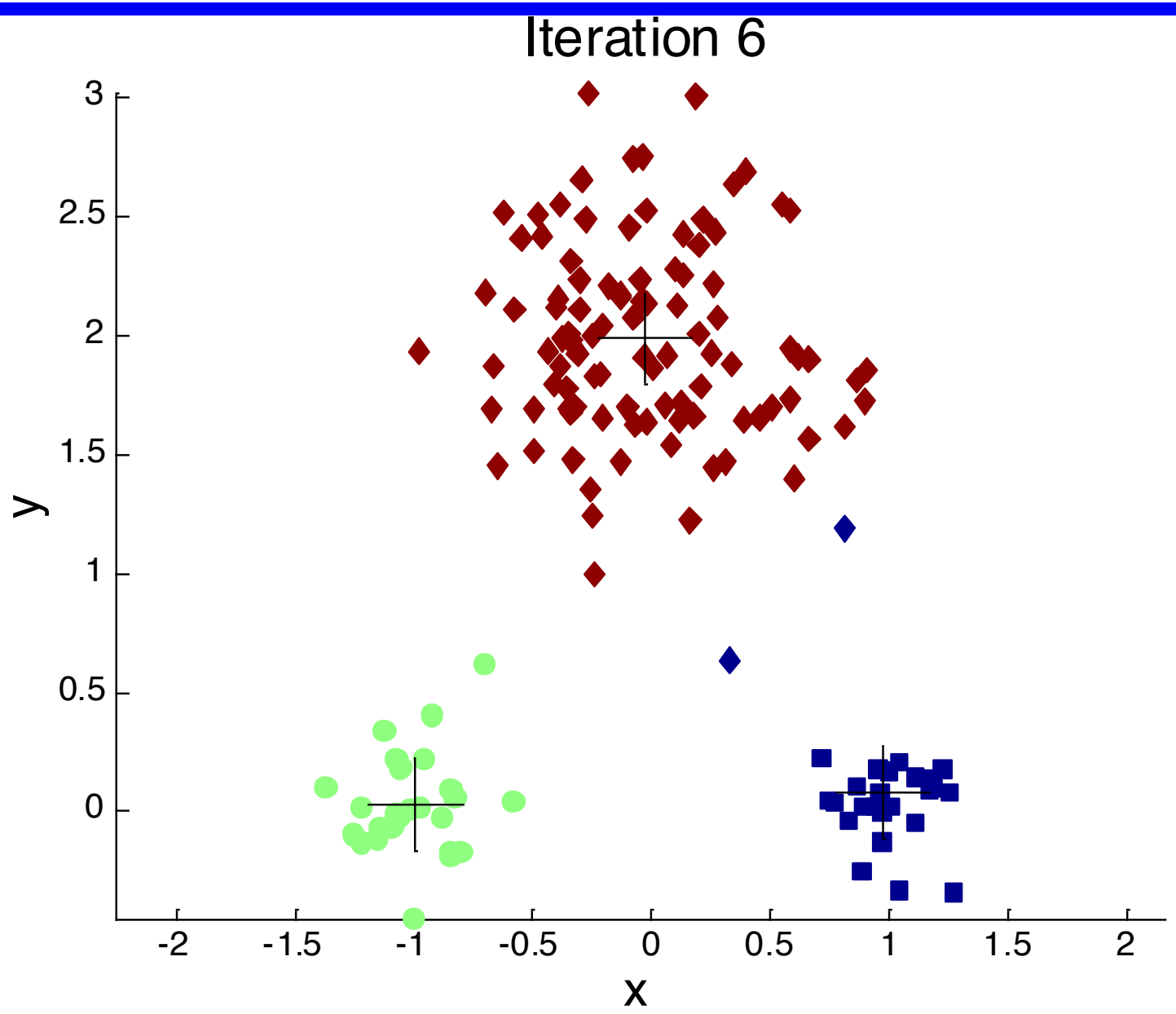
- **Partitioning method**
- Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where m_i is the centroid or medoid of cluster C_i) $SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist^2(m_i, x)$
- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - ***k-means*** (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - ***k-medoids*** or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

K-means Clustering

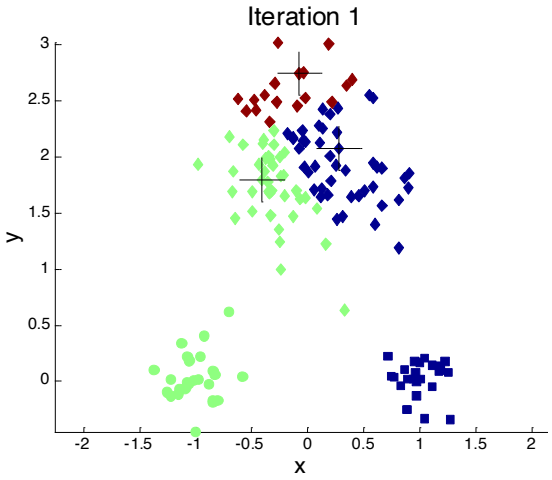
- Partitional clustering approach
- Number of clusters, k, must be specified
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- The basic algorithm is very simple

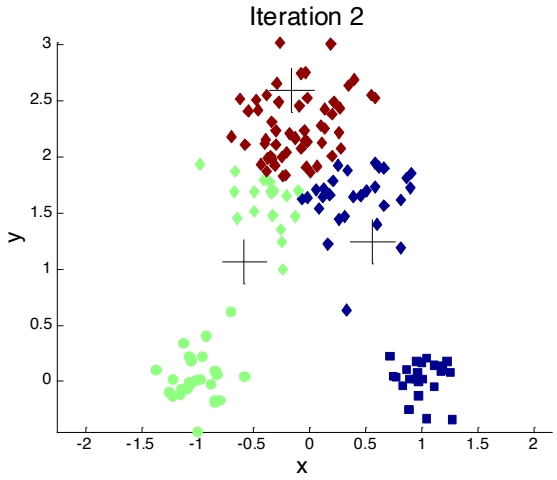
1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

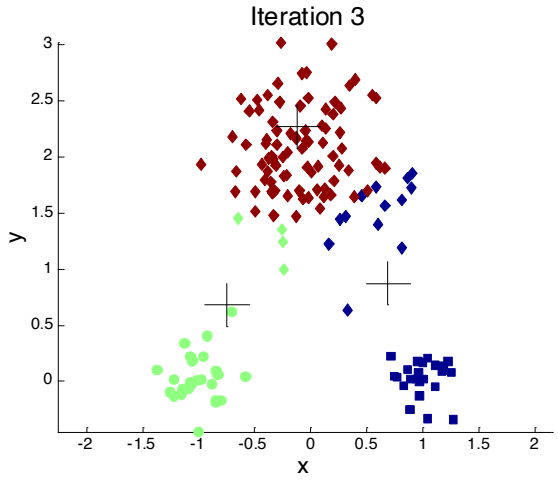
Example of K-means Clustering

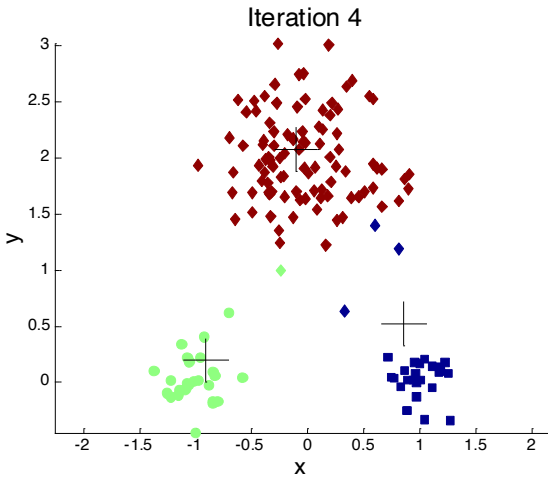


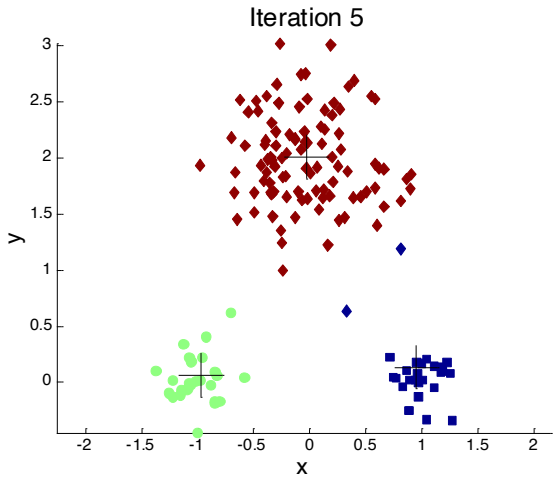
Example of K-means Clustering

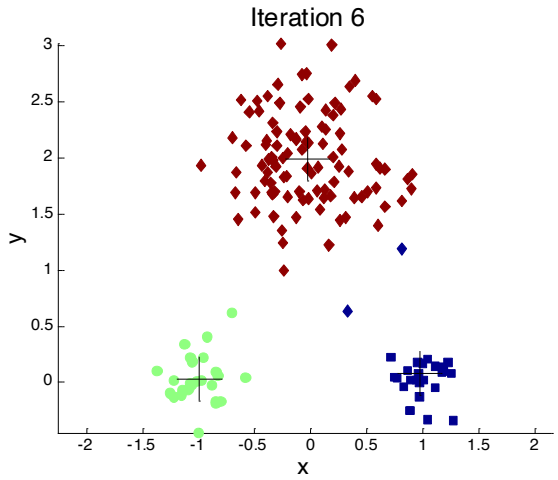








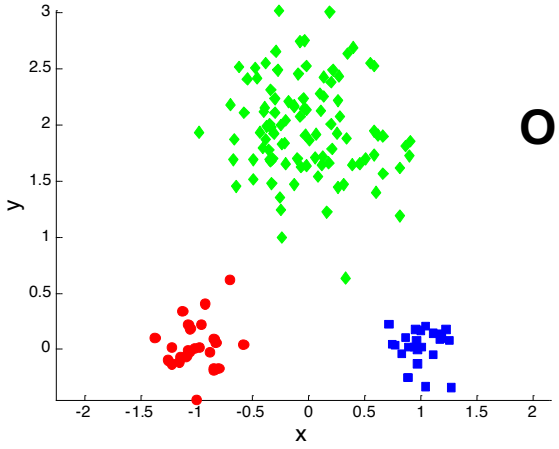




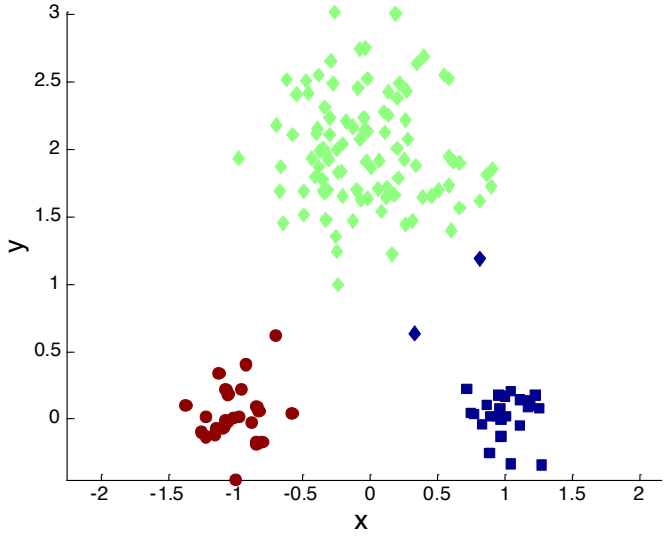
K-means Clustering – Details

- 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 $O(n \times d \times K \times I)$
  - n = number of points, K = number of clusters,
    I = number of iterations, d = number of attributes

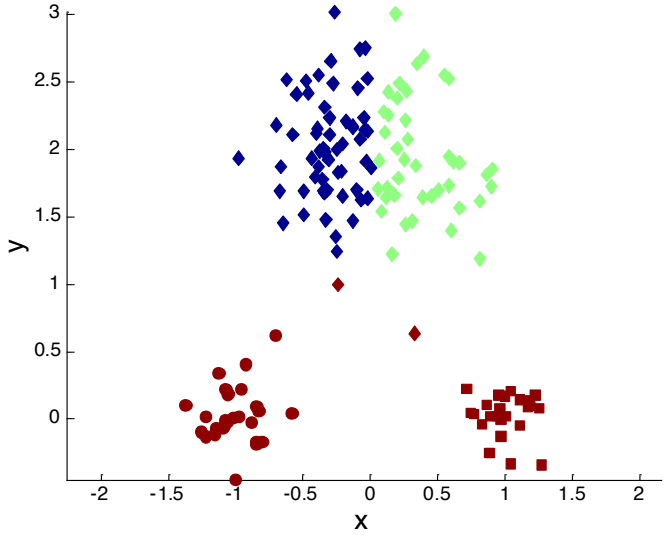
Two different K-means Clusterings



Original Points



Optimal Clustering



Sub-optimal Clustering

Comments on the *K-Means* Method

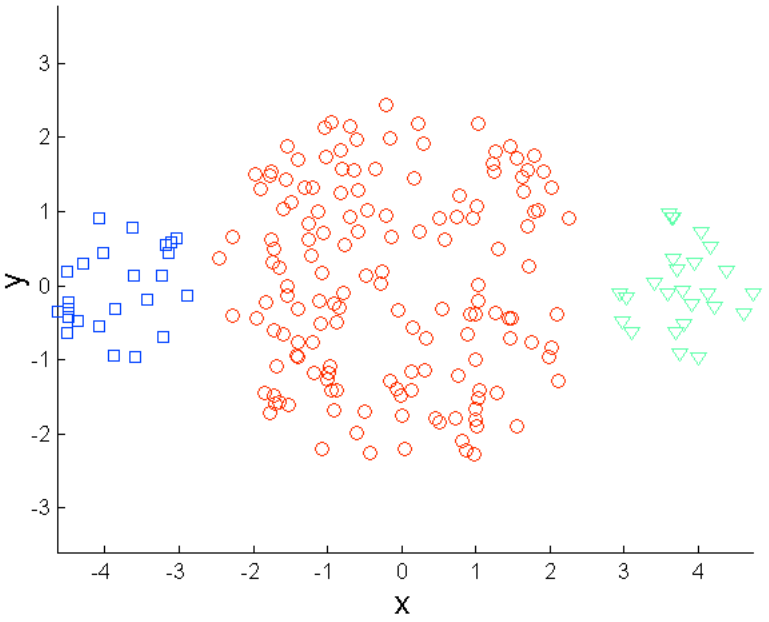
- Strength: *Efficient*: $O(n \times d \times K \times I)$.
- Weakness:
  - Applicable only to objects in a continuous n-dimensional space
    - *k-medoids* can be applied to a wide range of data
    - *k-modes* method can be applied to categorical data
  - Need to specify *k,* the *number* of clusters, in advance, although there are ways to automatically determine the best k (see Hastie et al., 2009)
  - Sensitive to noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

Limitations of K-means

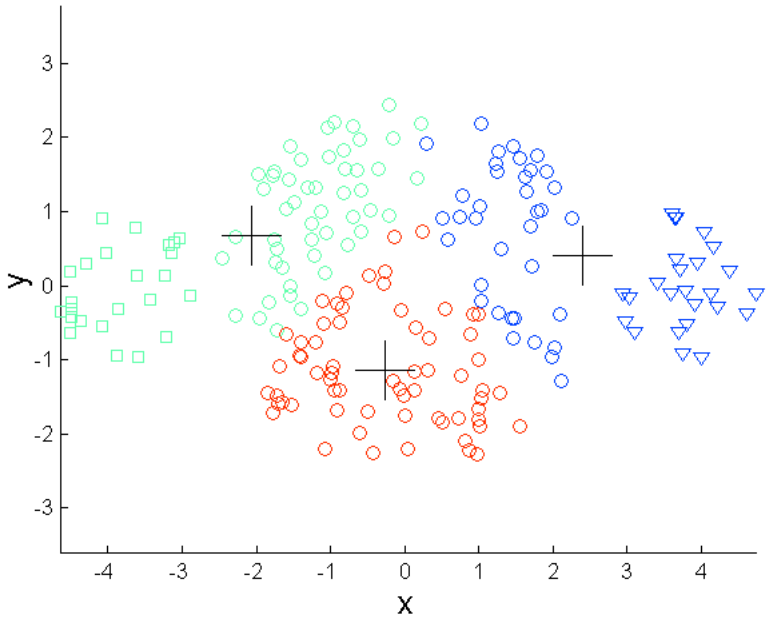
- K-means has problems when clusters are of:
  - Different sizes
  - Different densities
  - Non-globular shapes

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

Limitations of K-means: Differing Sizes

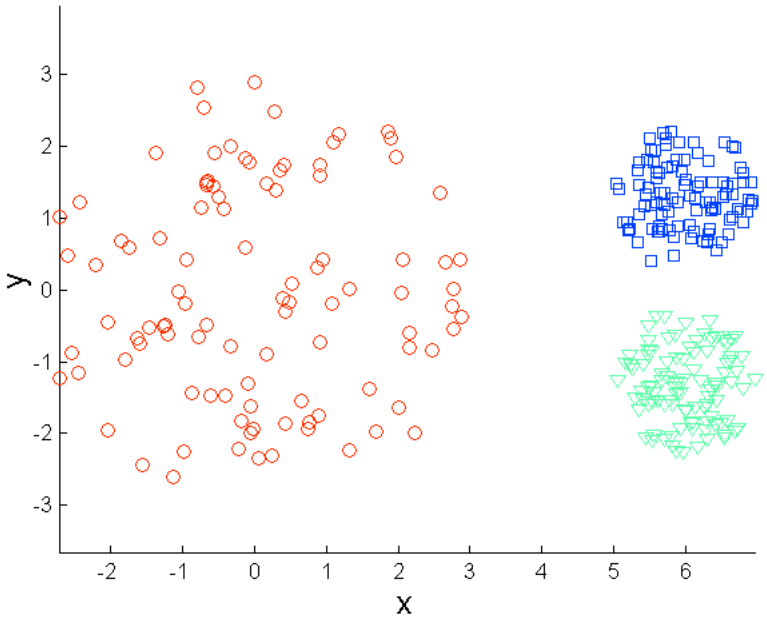


Original Points

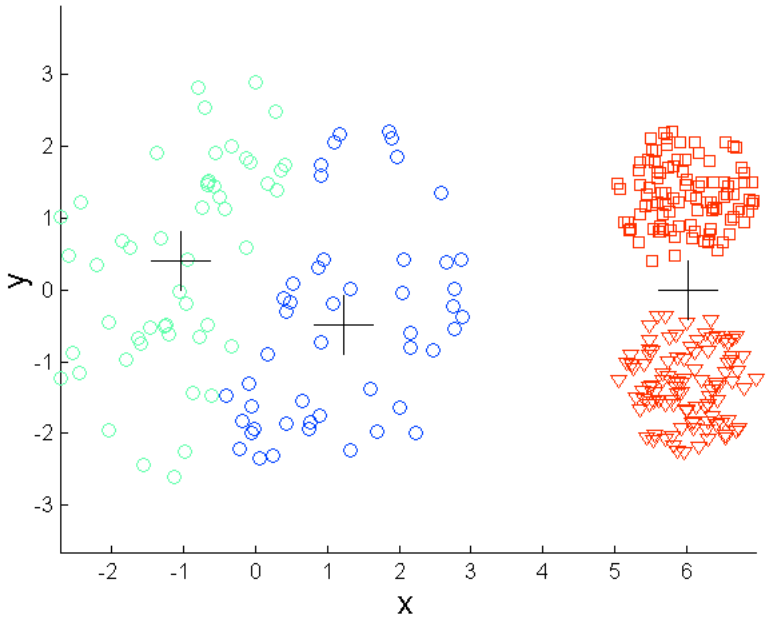


K-means (3 Clusters)

Limitations of K-means: Differing Density

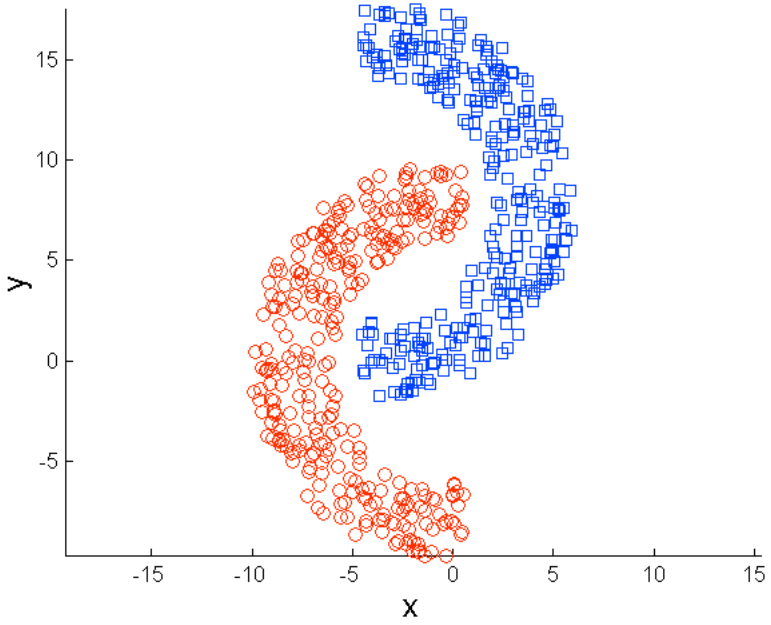


Original Points

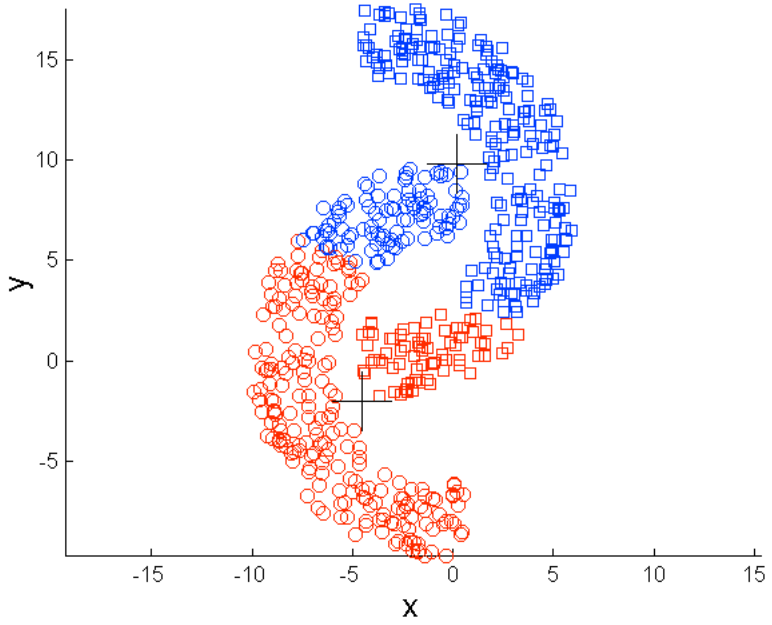


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

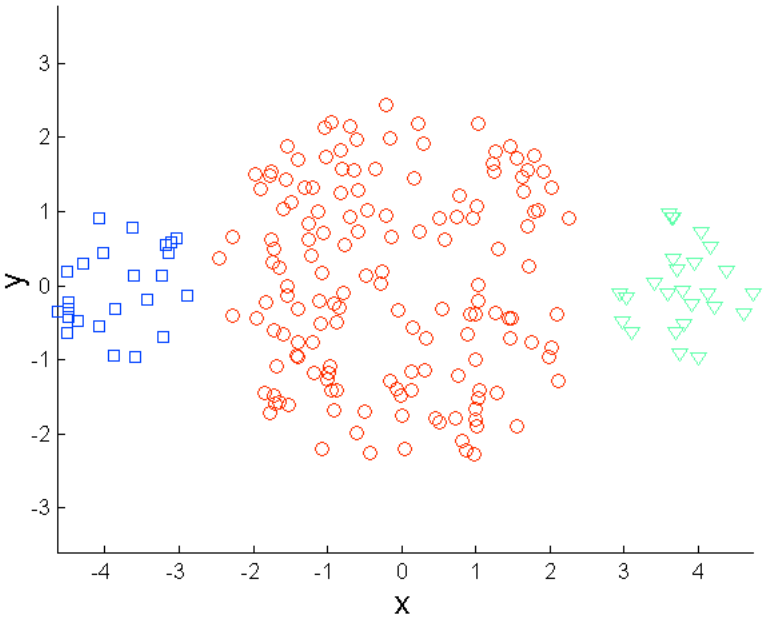


Original Points

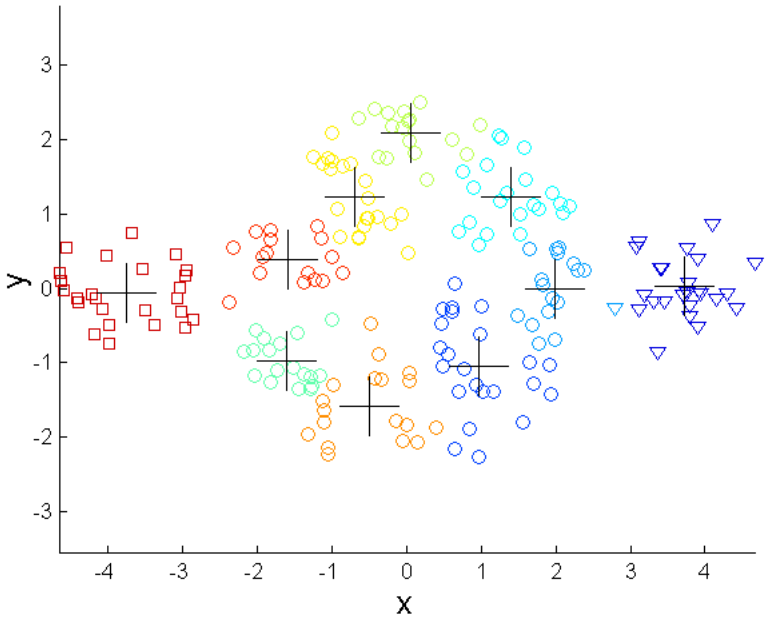


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points

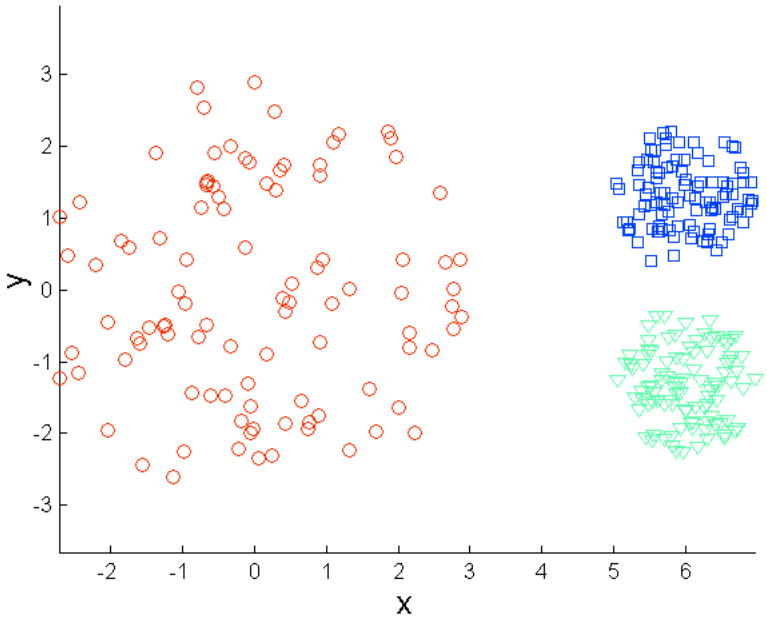


K-means Clusters

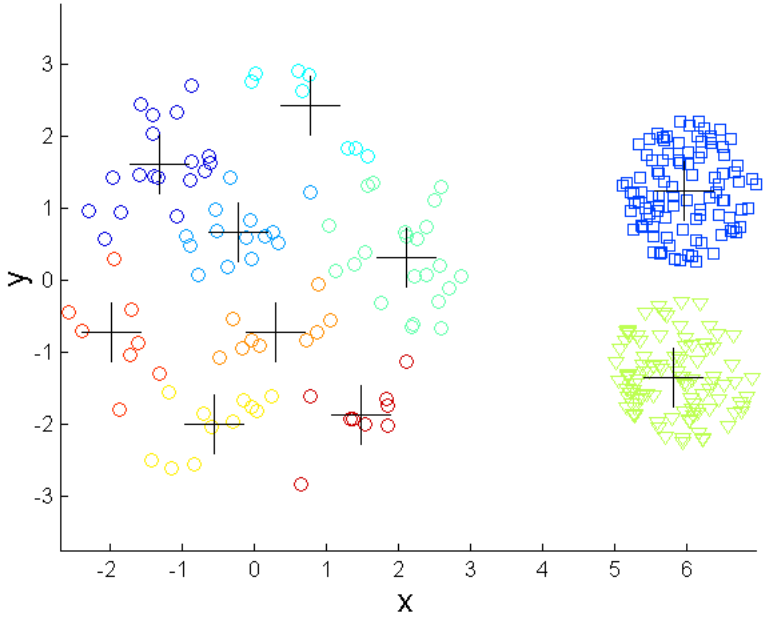
One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming K-means Limitations

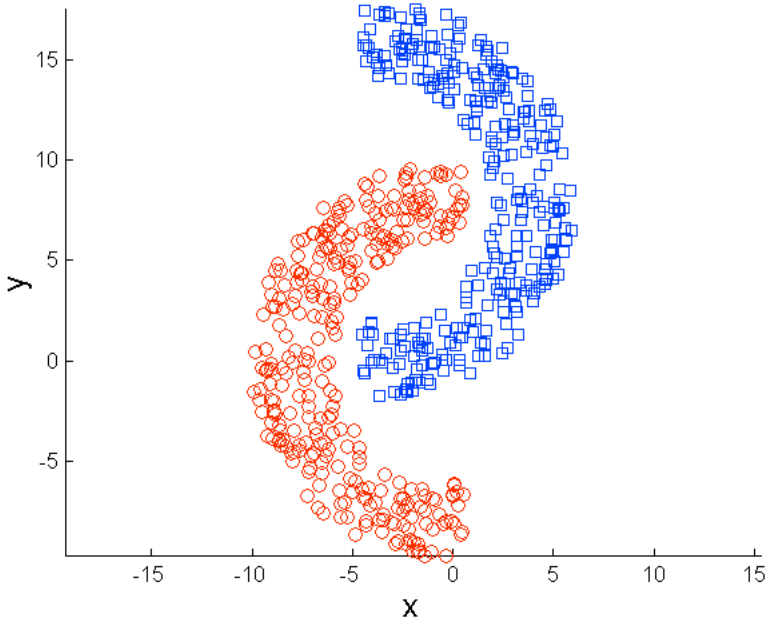


Original Points

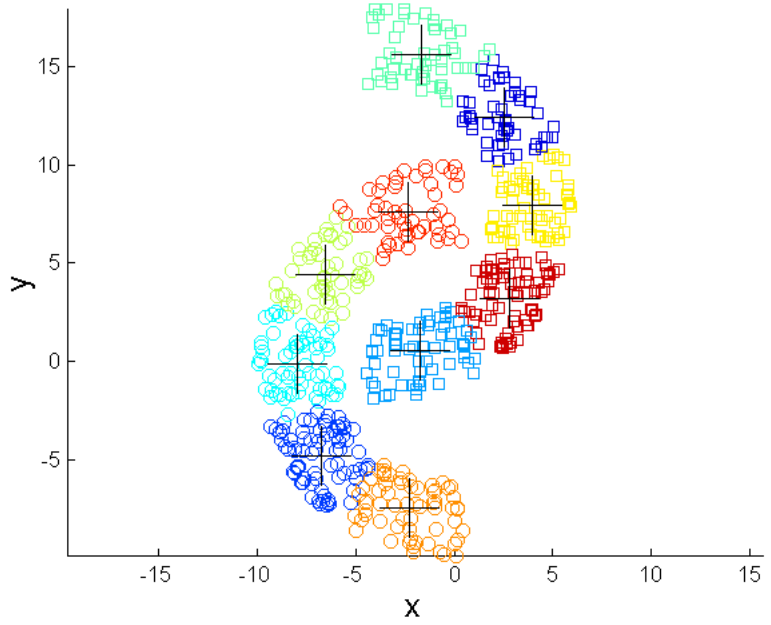


K-means Clusters

Overcoming K-means Limitations

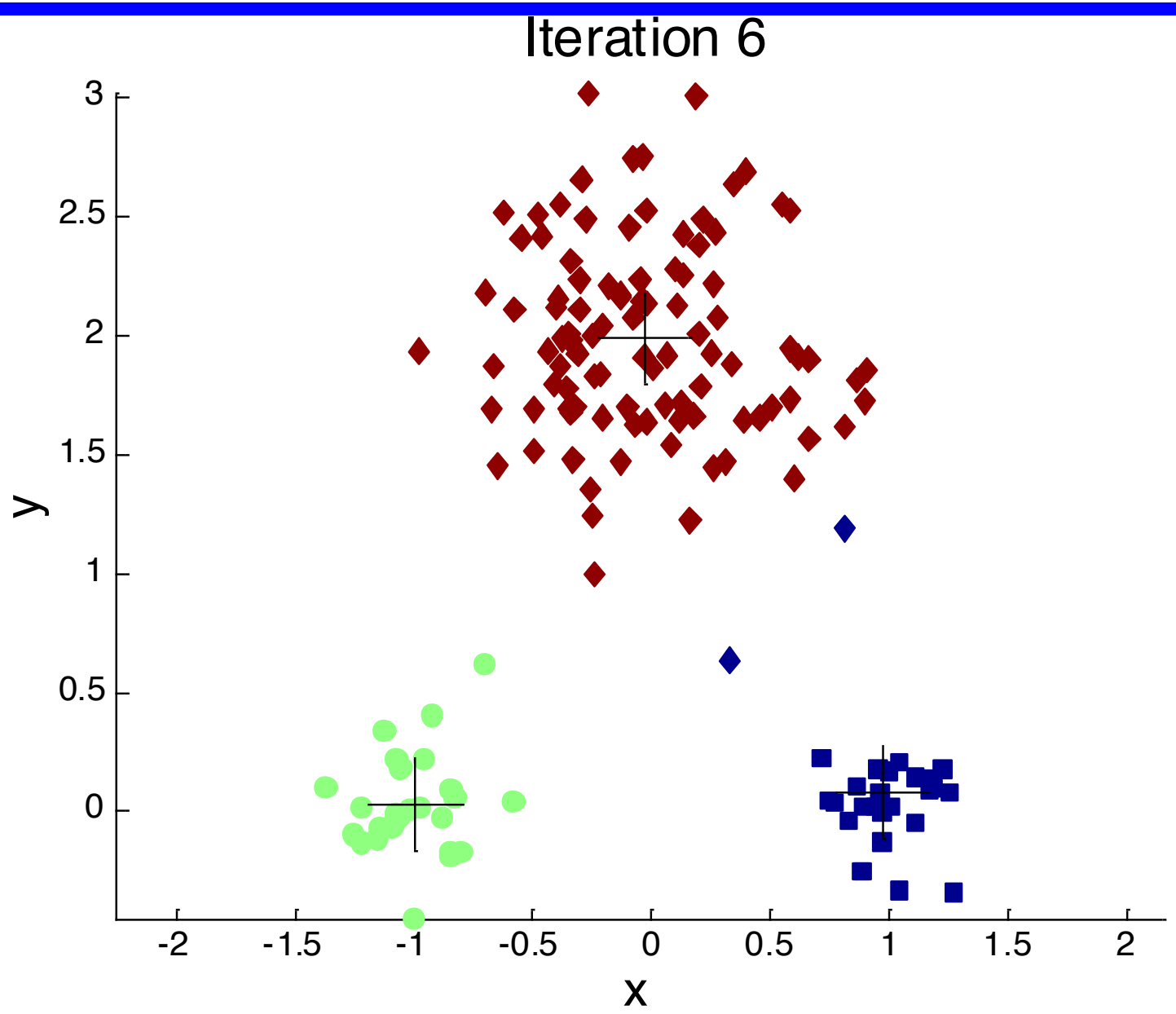


Original Points

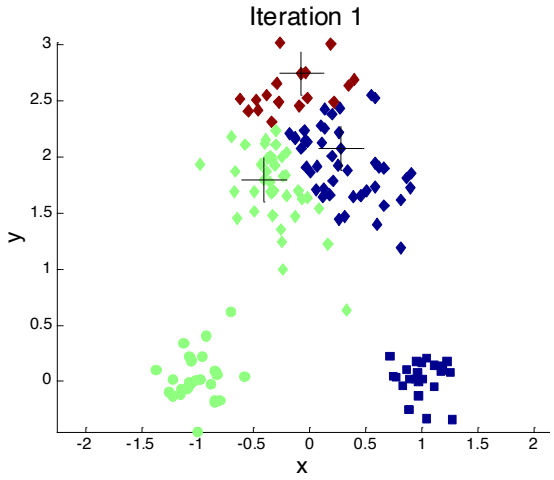


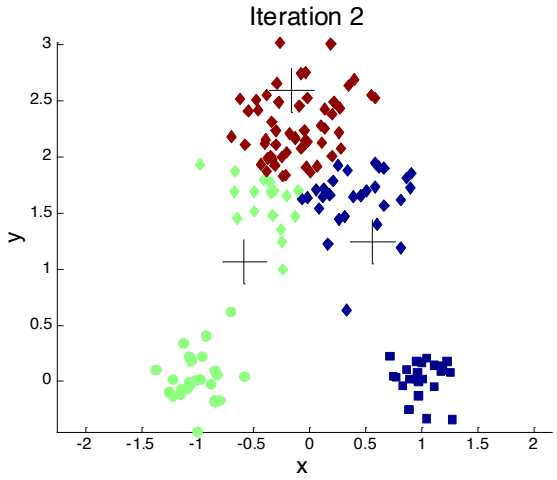
K-means Clusters

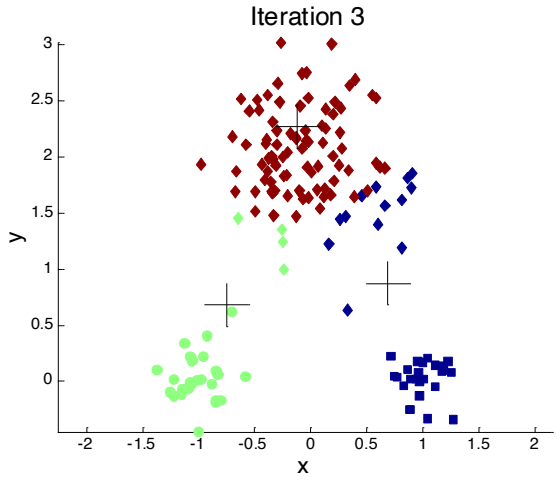
Importance of Choosing Initial Centroids

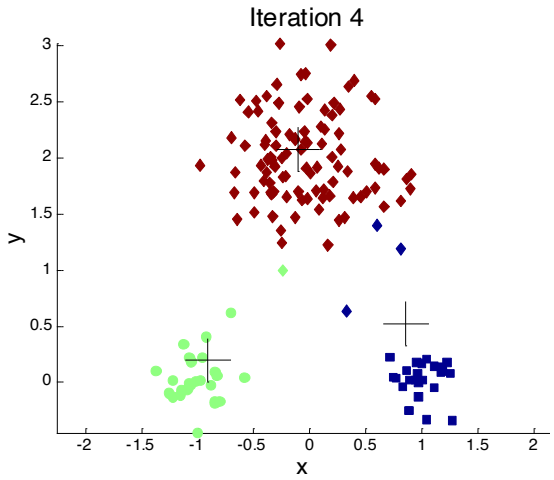


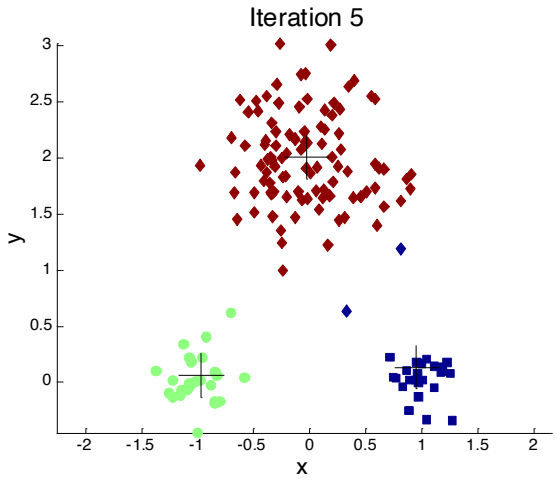
Importance of Choosing Initial Centroids

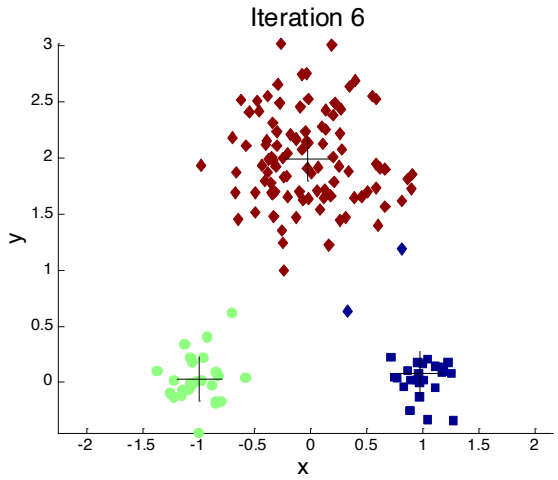




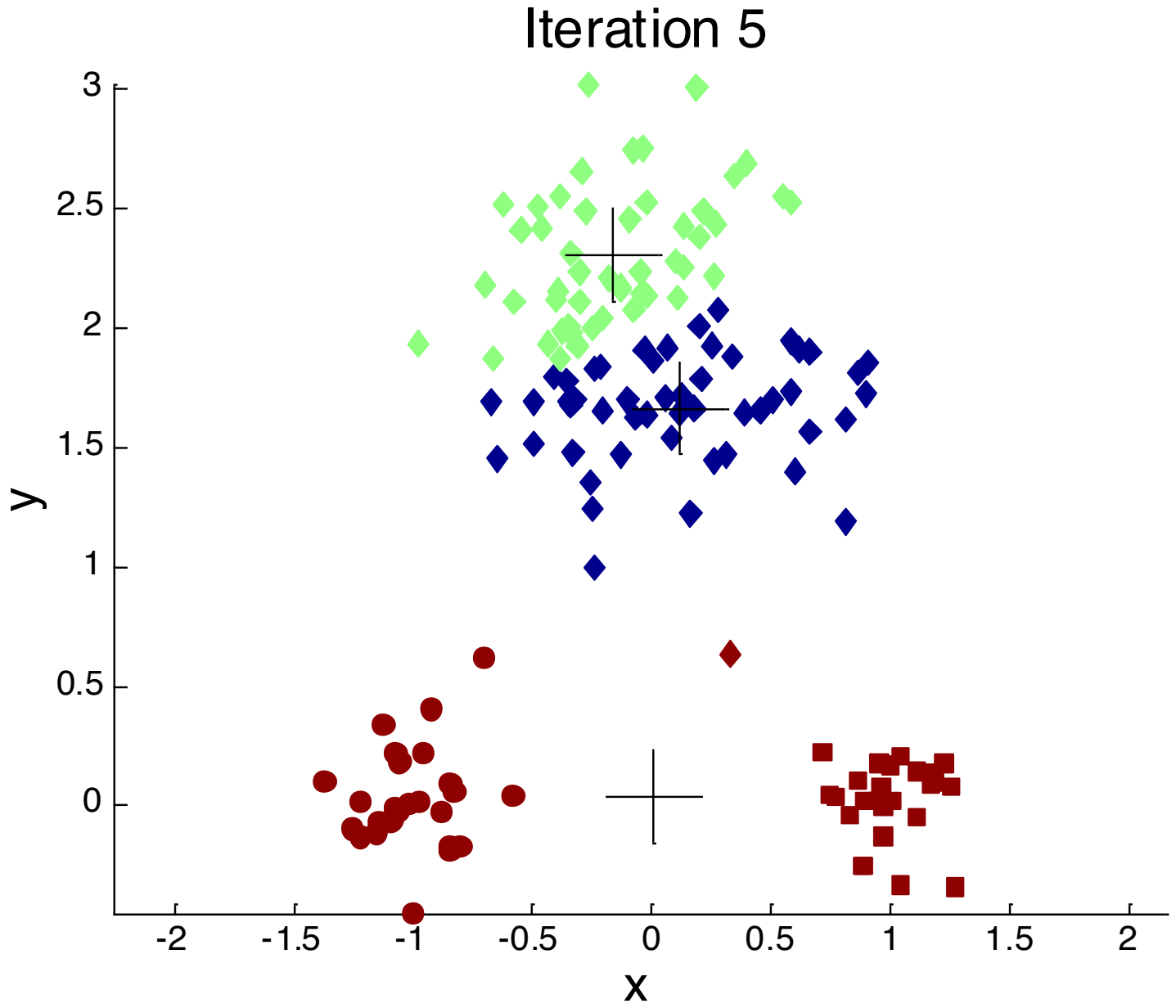




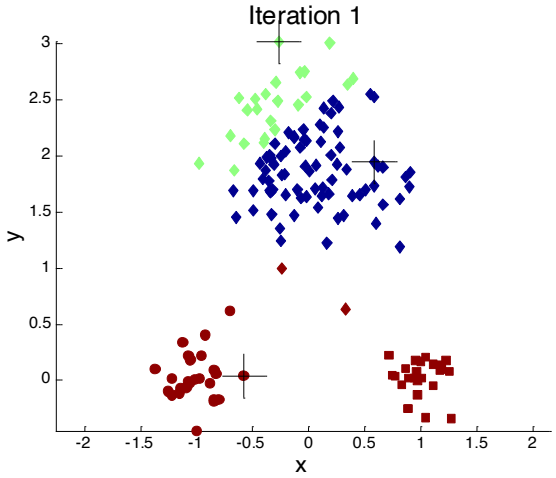


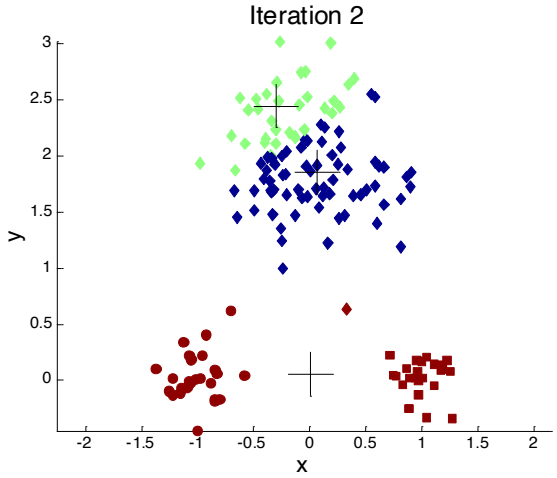


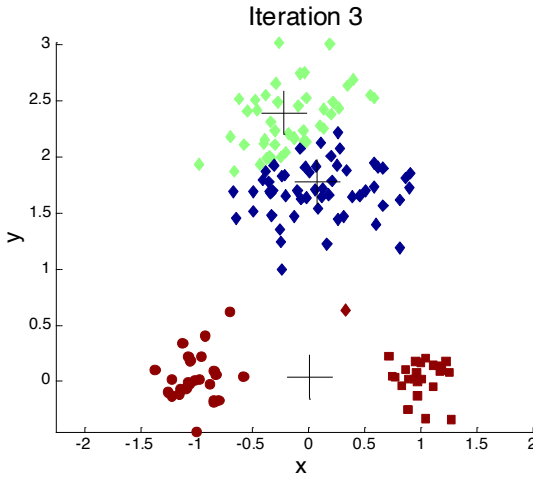
Importance of Choosing Initial Centroids ...

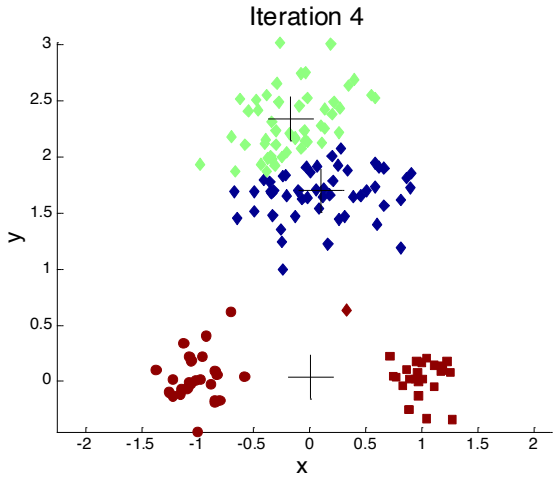


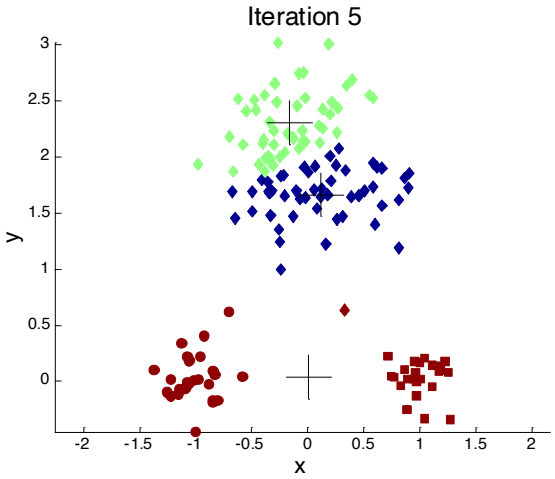
Importance of Choosing Initial Centroids ...







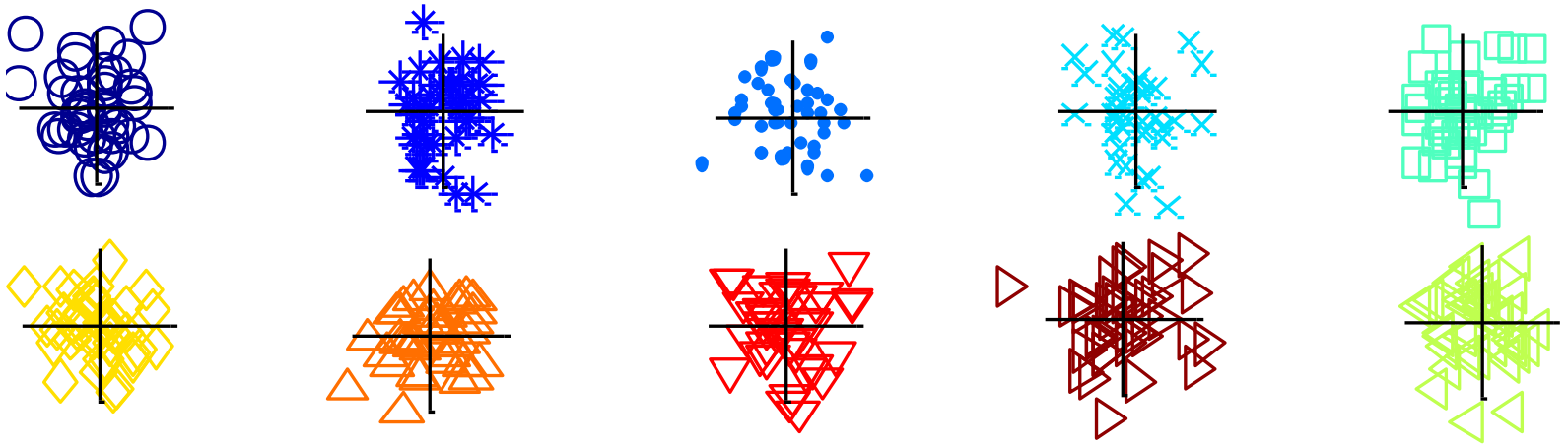




Problems with Selecting Initial Points

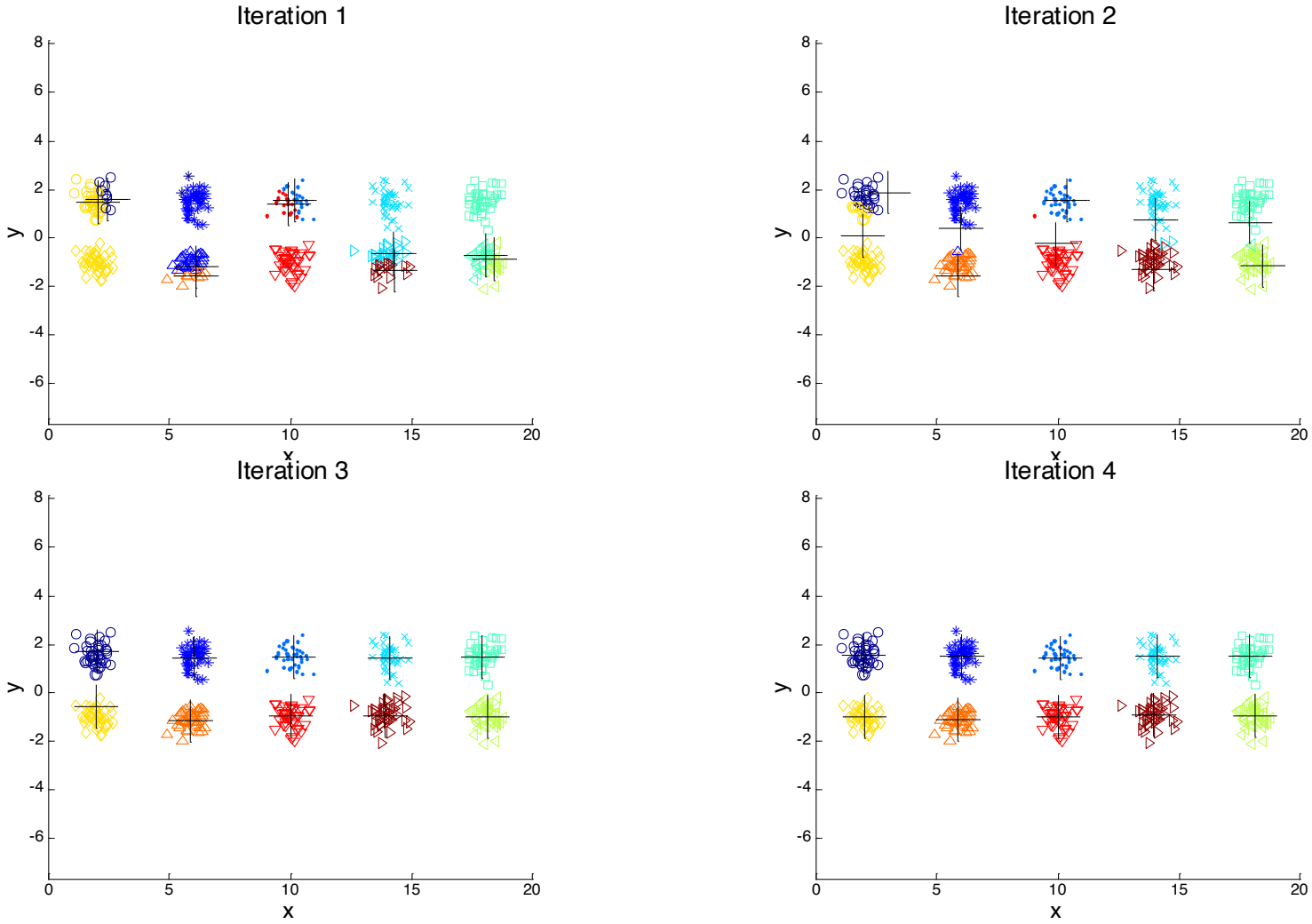
- If there are K ‘real’ clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - Sometimes the initial centroids will readjust themselves in ‘right’ way, and sometimes they don’t

10 Clusters Example



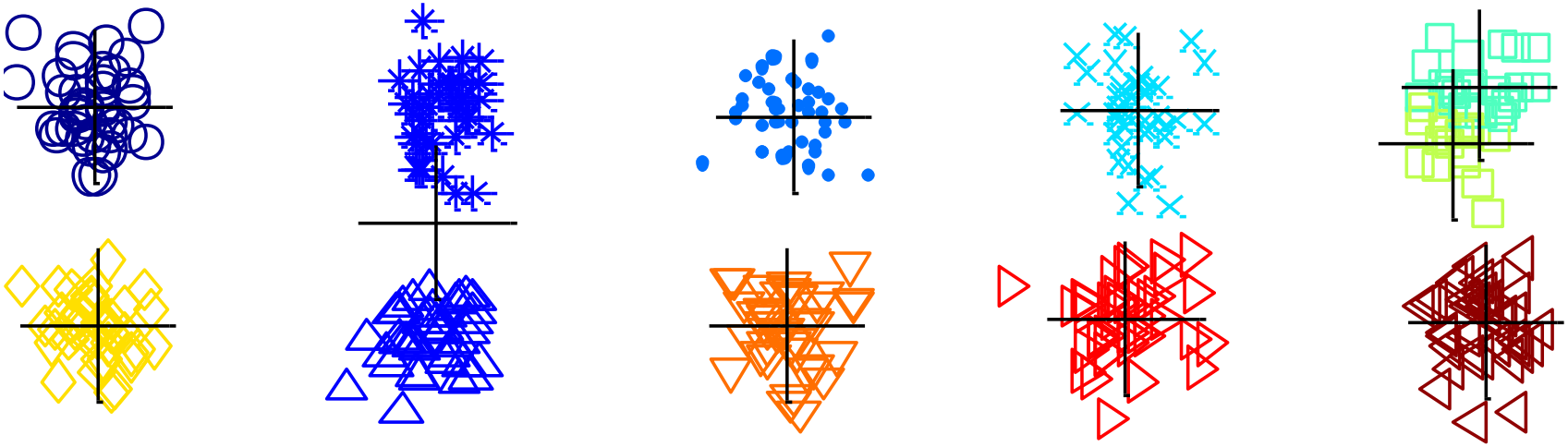
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



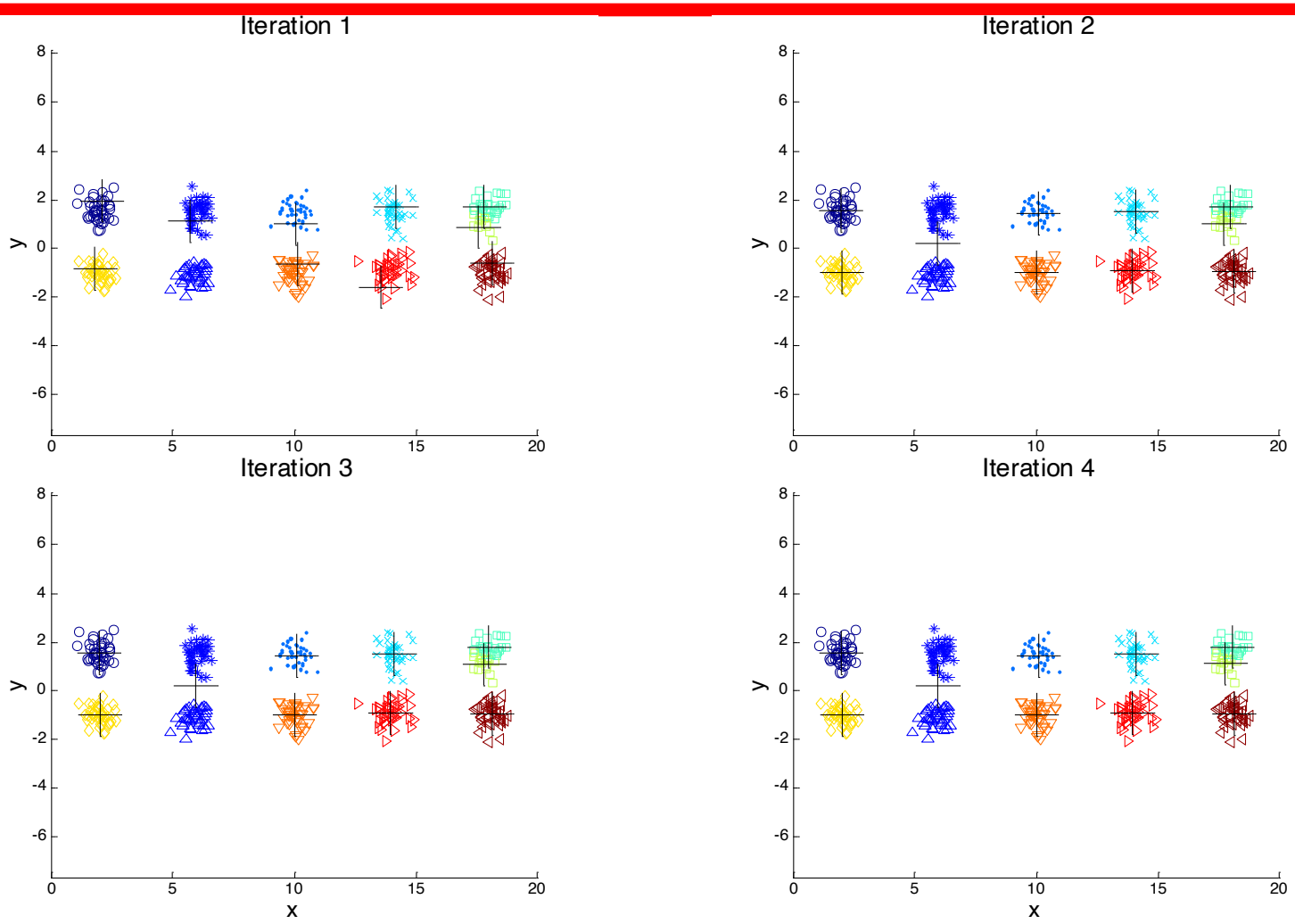
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

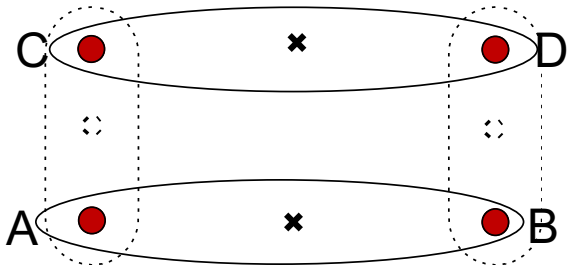
10 Clusters Example



Starting with some pairs of clusters having three initial centroids, while other have only one.

Problems with Selecting Initial Points

- **No error bound**
- **Example**: Consider 4 points A(1,1), B(3,1), C(1,2) and D(3,2) (defining a rectangle whose width is larger than height) and K=2.



By choosing as initial centroid A and C we have that the final centroids are (2,1) and (2,2) with an error equal to 4, whereas the optimal solution, obtained by selecting as initial centroids A and B, has an error equal to 1.

The error can increase indefinitely by moving B and D on the right.

Variations of the *K-Means* Method

- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means,
  - Dissimilarity calculations,
  - Strategies to calculate cluster means.

Seed choice

Results can vary drastically based on random seed selection

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings

Common heuristics

- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- Try out multiple starting points
- Initialize with the results of another clustering method

Furthest centers heuristic

μ_1 = pick random point

for i = 2 to K:

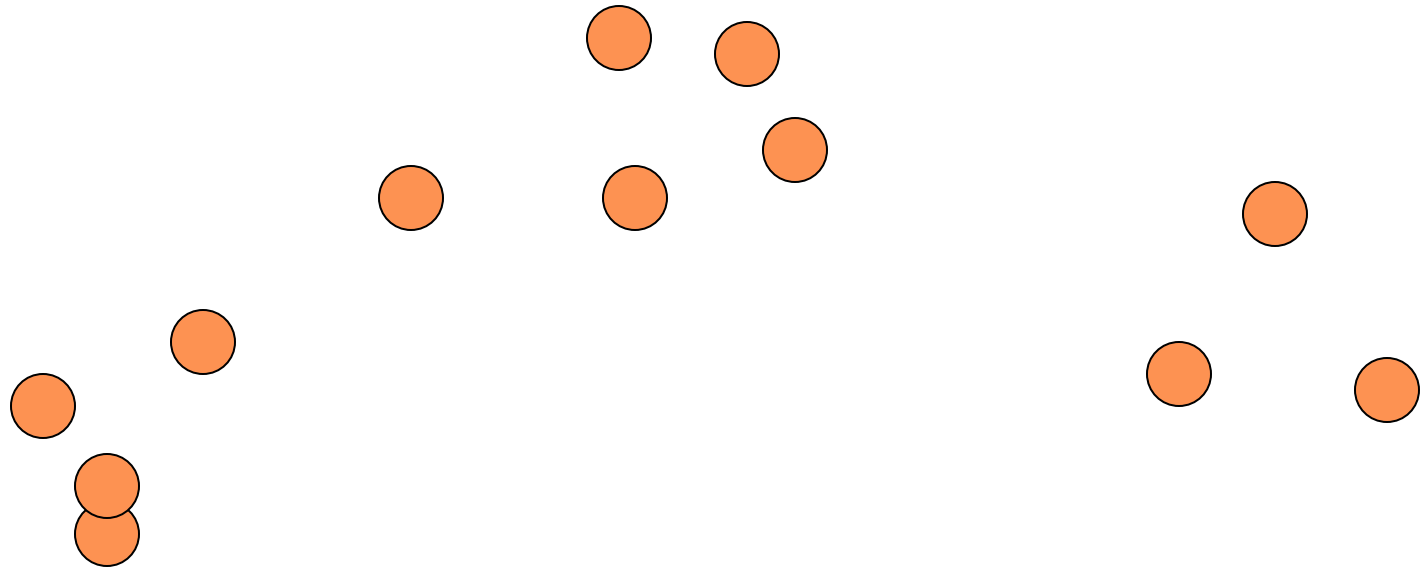
μ_i = point that is furthest from **any** previous centers

$$\mu_i = \underset{x}{\arg\max} \min_{\mu_j : 1 < j < i} d(x, \mu_j)$$

point with the largest distance to any previous center

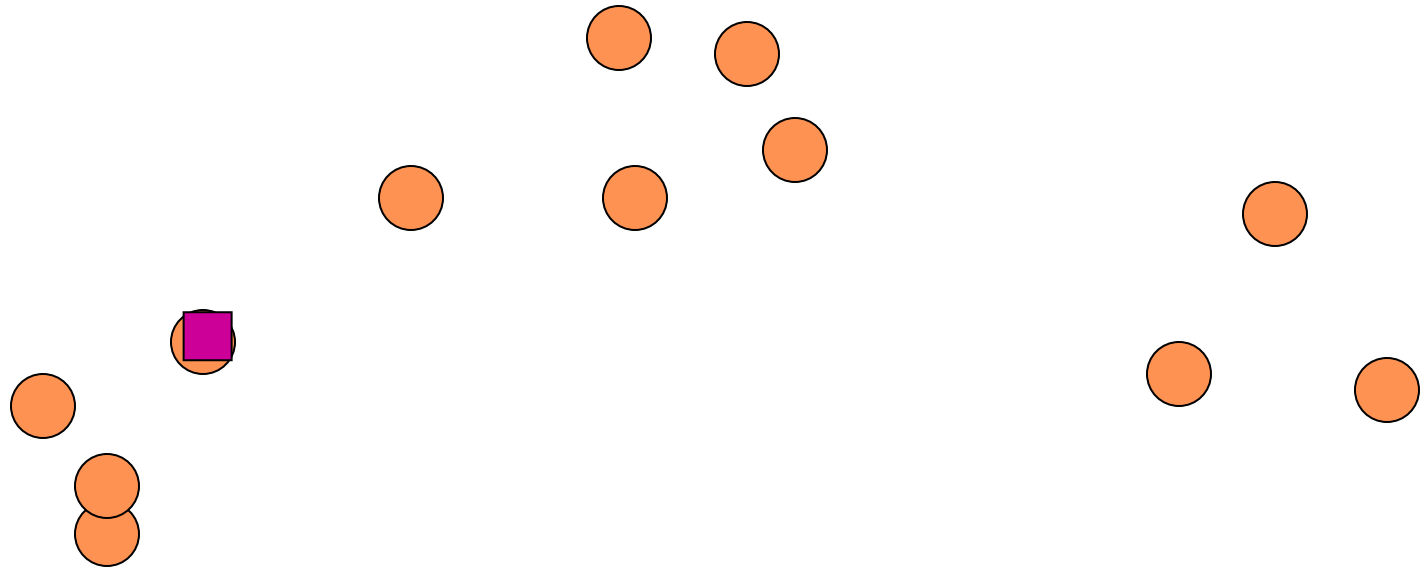
smallest distance from x to any previous center

K-means: Initialize furthest from centers



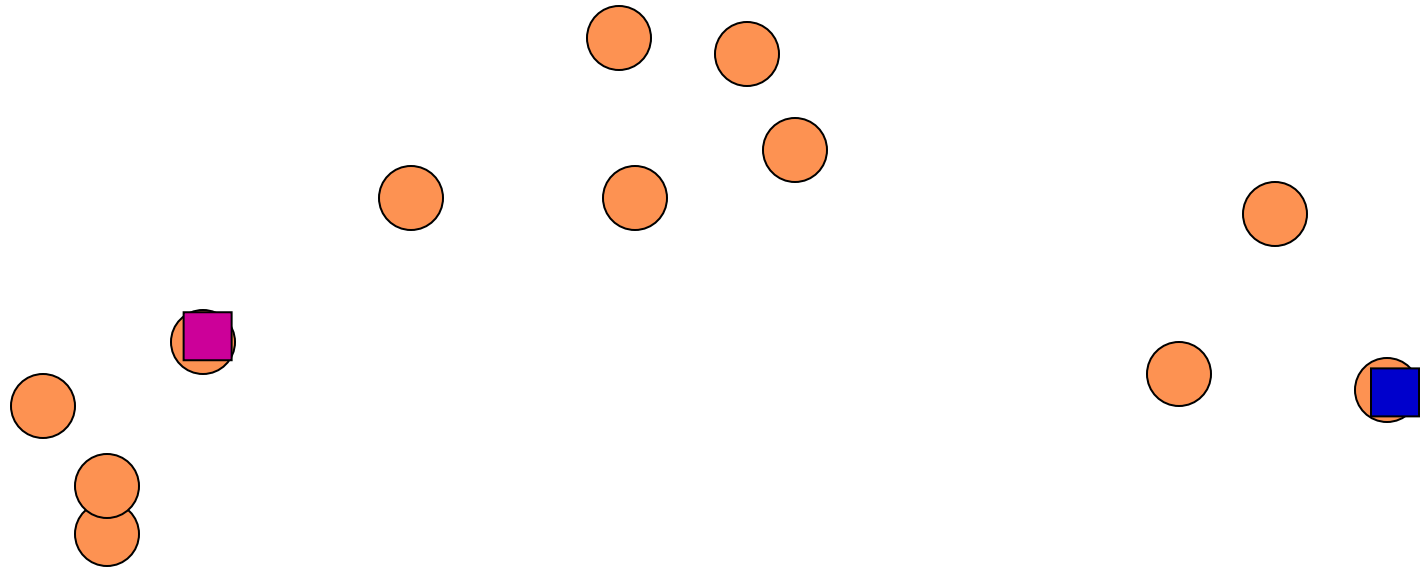
Pick a random point for the first center

K-means: Initialize furthest from centers



What point will be chosen next?

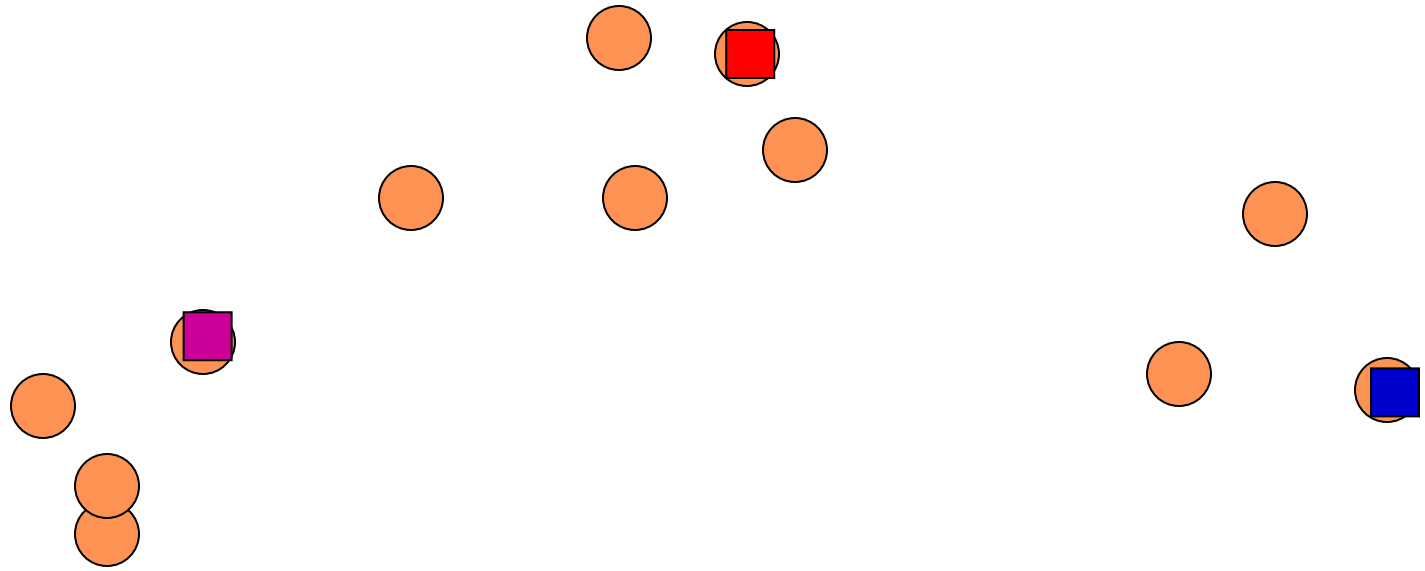
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

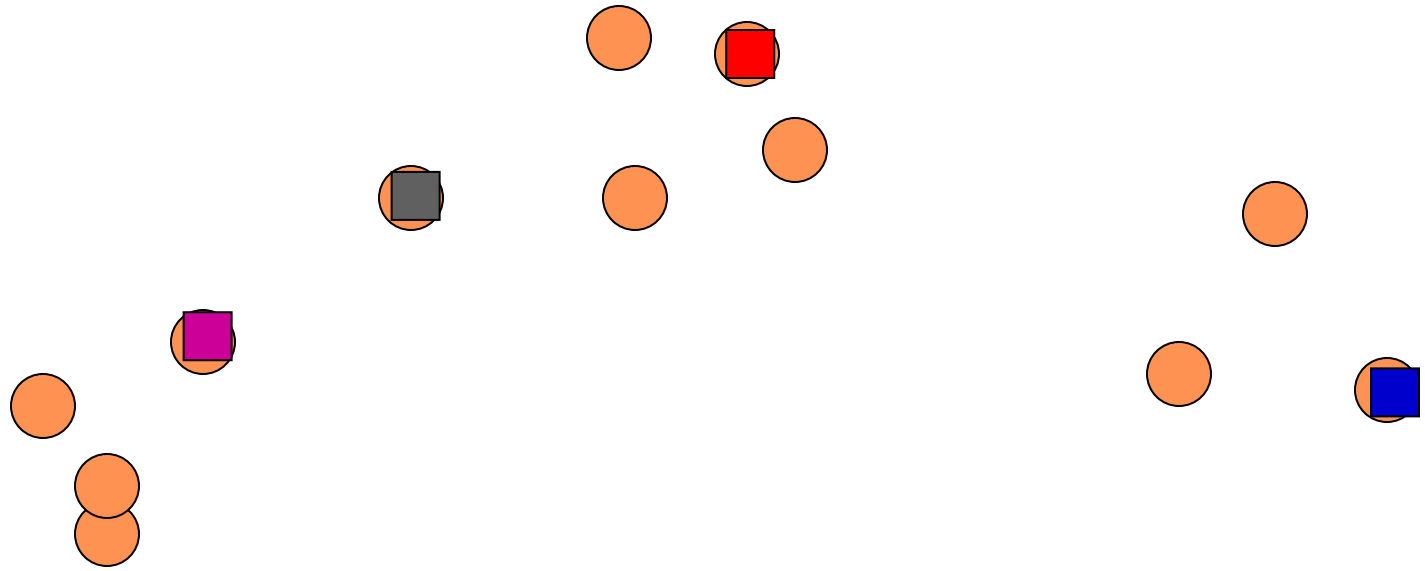
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

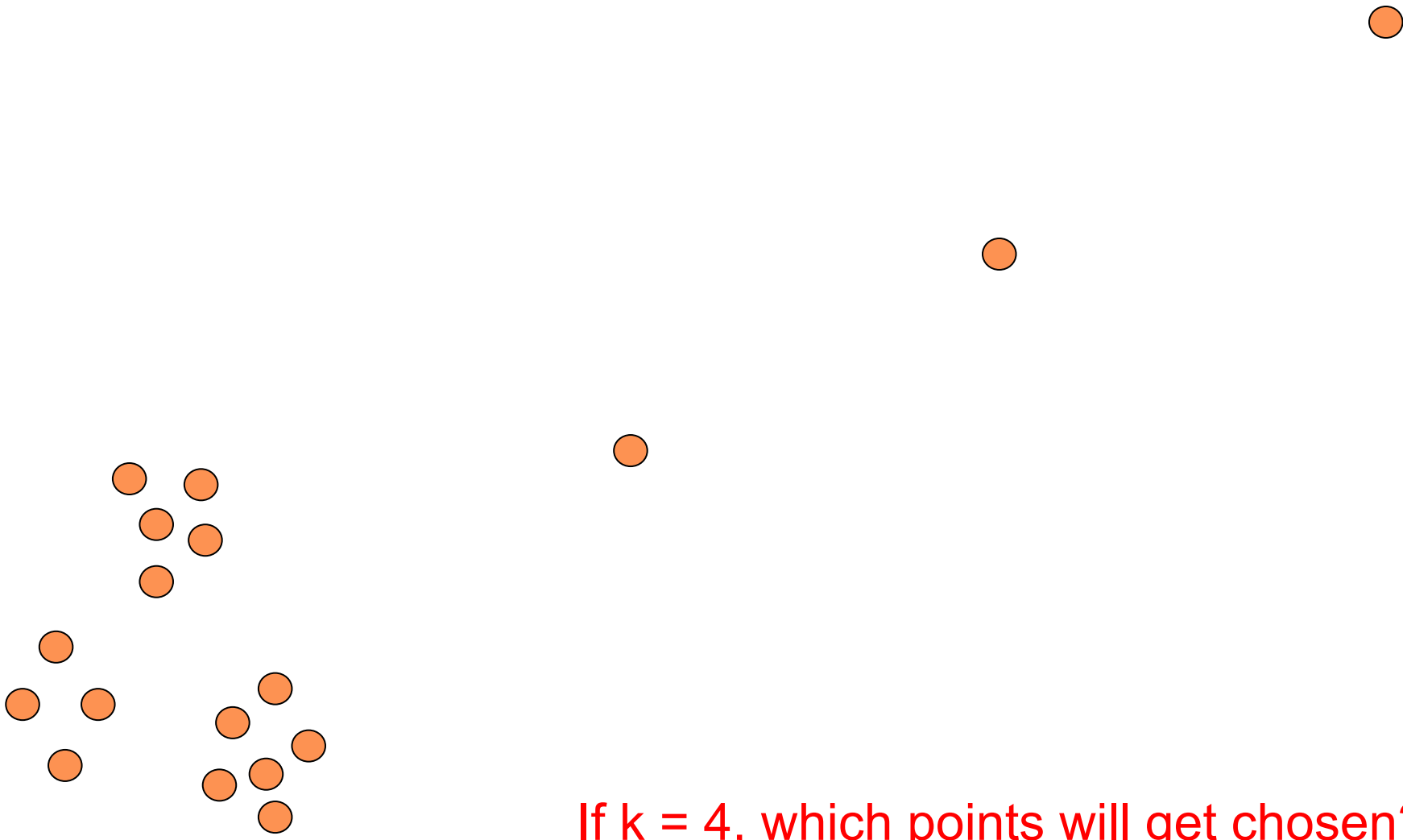
K-means: Initialize furthest from centers



Furthest point from center

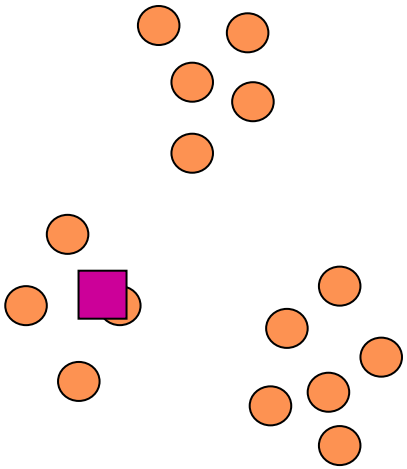
Any issues/concerns with this approach?

Furthest points concerns



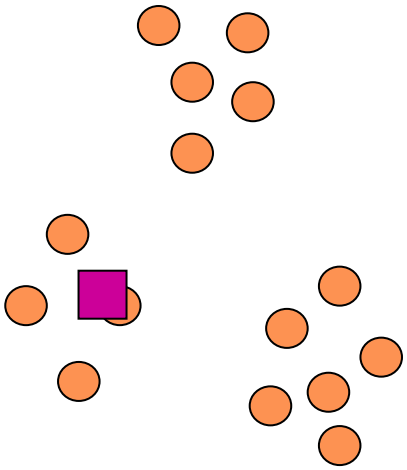
If k = 4, which points will get chosen?

Furthest points concerns



If we do a number of trials, will we get different centers?

Furthest points concerns



Doesn’t deal well with outliers

Further 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

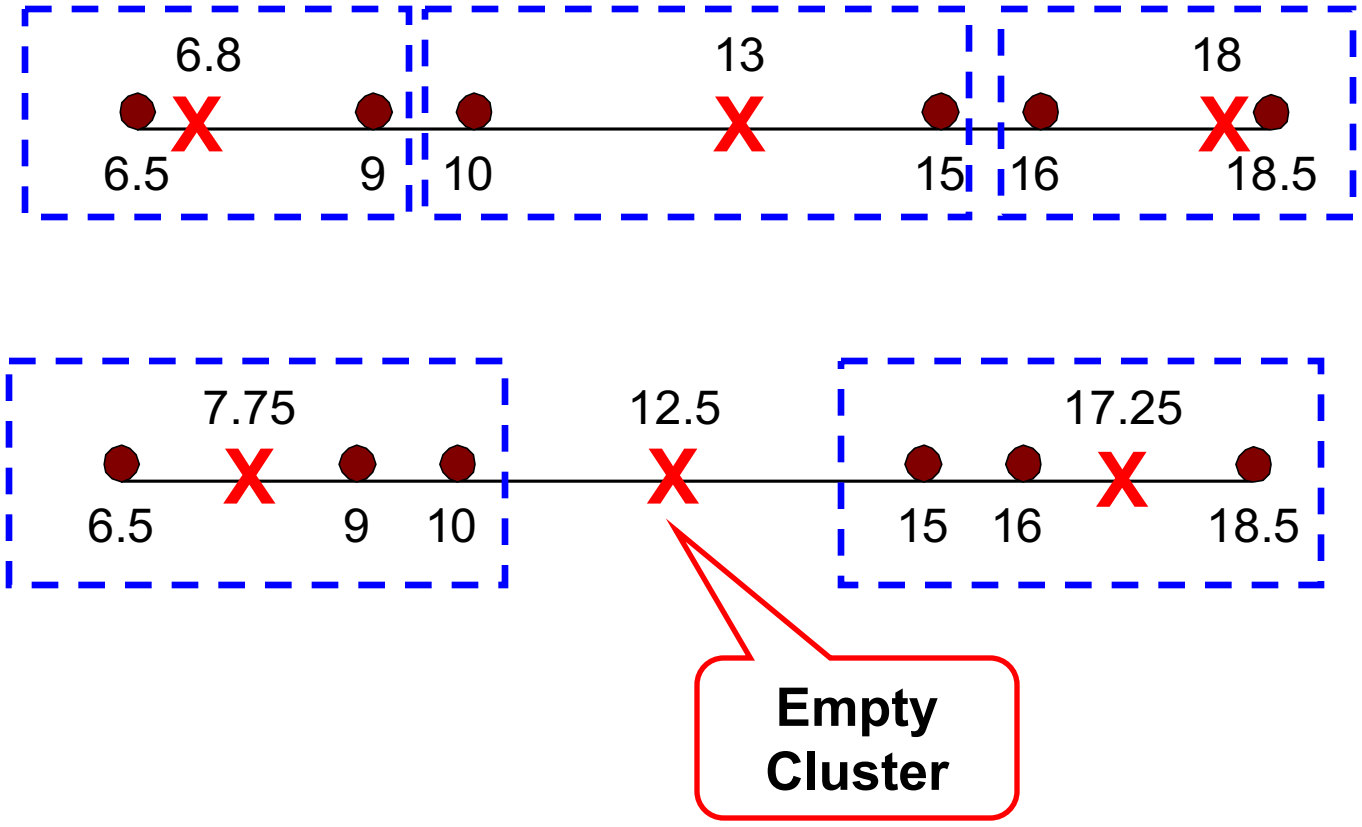
K-means++

- This approach can be slower than random initialization, but very consistently produces better results in terms of SSE
  - The k-means++ algorithm guarantees an approximation ratio O(log k) in expectation, where k is the number of centers
- To select a set of initial centroids, *C*, perform the following

1. Choose one center uniformly at random from among the data points.
2. For each data point *x*, compute D(*x*), the distance between *x* and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point *x* is chosen with probability proportional to $D(x)^2$.
4. Repeat Steps 2 and 3 until *k* centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard k-means clustering.

Empty Clusters

- K-means can yield empty clusters



Handling Empty Clusters

- Basic K-means algorithm can yield empty clusters

- Several strategies (to replace centroid)
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest SSE
  - If there are several empty clusters, the above can be repeated several times.

Updating Centers Incrementally

- In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid

- An alternative is to update the centroids after each assignment (incremental approach)
  - Each assignment updates zero or two centroids
  - More expensive
  - Introduces an order dependency
  - Never get an empty cluster
  - Can use “weights” to change the impact

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 SSE
  - Merge clusters that are ‘close’ and that have relatively low SSE
  - Can use these steps during the clustering process
    - ISODATA algorithm [Jensen’96]

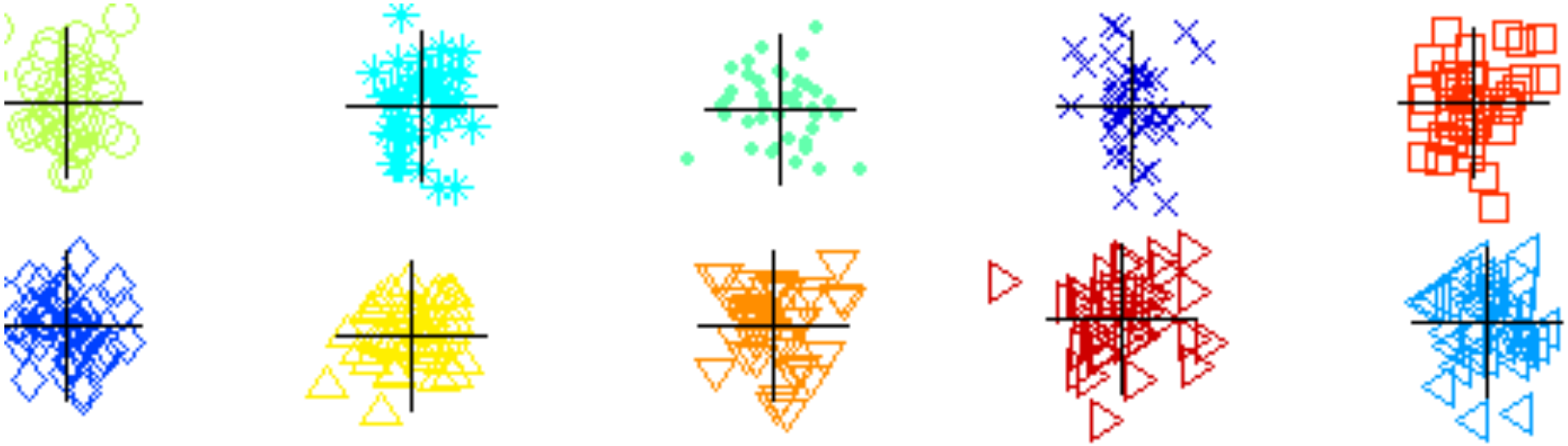
Bisecting K-means

- Bisecting K-means algorithm
  - 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

Bisecting K-means Example



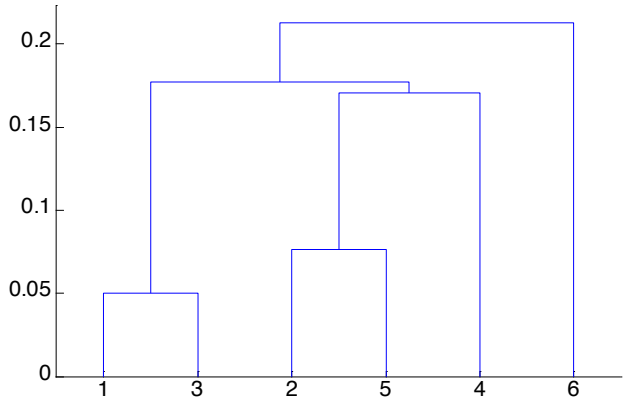
Variations of the *K-Means* Method

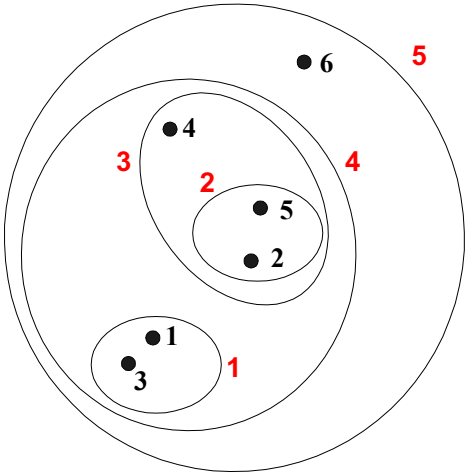
Handling categorical data: *k-modes*

- Replacing means of clusters with modes
- Using new dissimilarity measures to deal with categorical objects
- Using a frequency-based method to update modes of clusters
- A mixture of categorical and numerical data: *k-prototype* method

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





Strengths 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

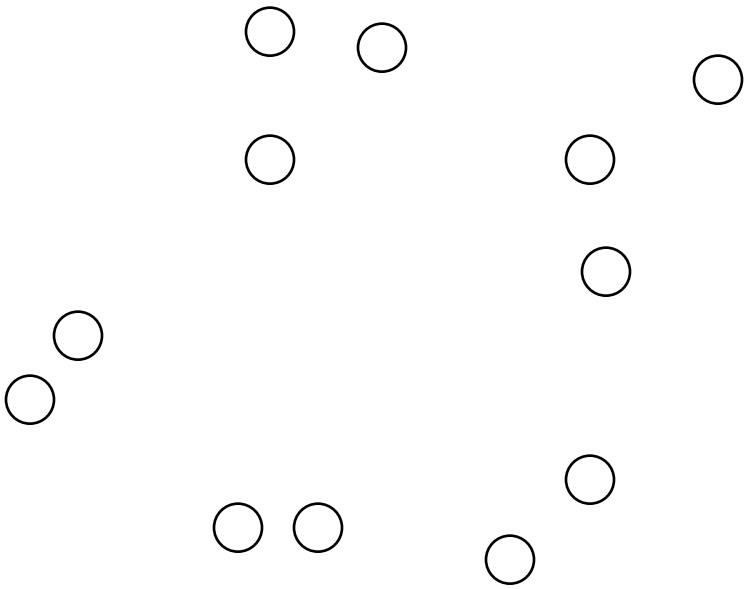
- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains an individual 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

- Most popular hierarchical clustering technique
- Basic algorithm is straightforward
  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
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

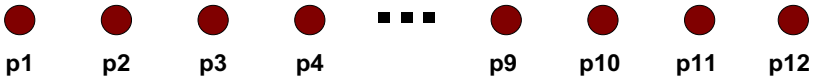
Starting Situation

- Start with clusters of individual points and a proximity matrix



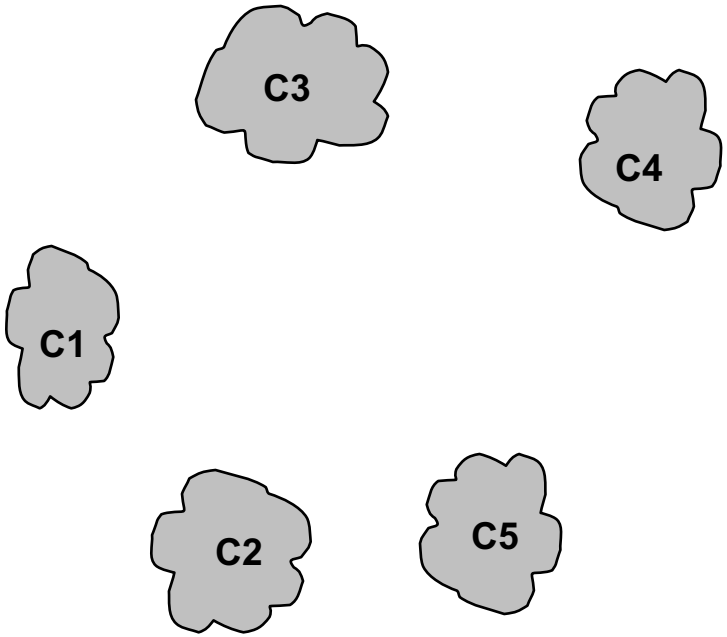
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



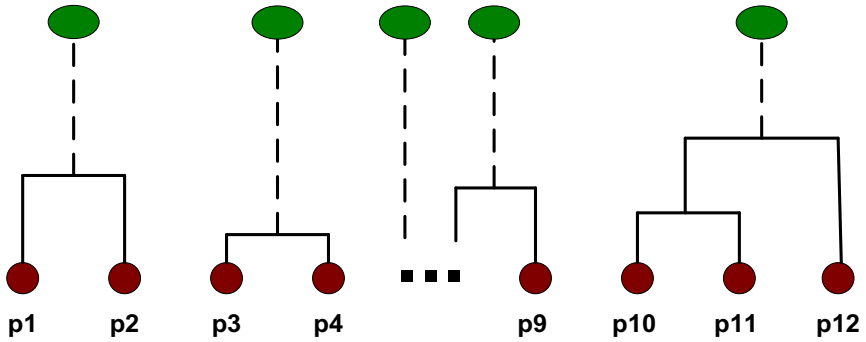
Intermediate Situation

- After some merging steps, we have some clusters



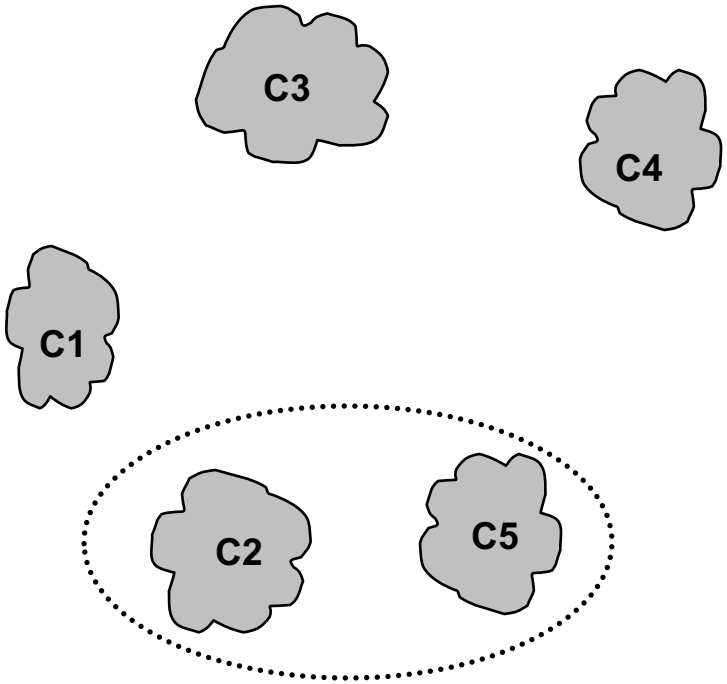
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



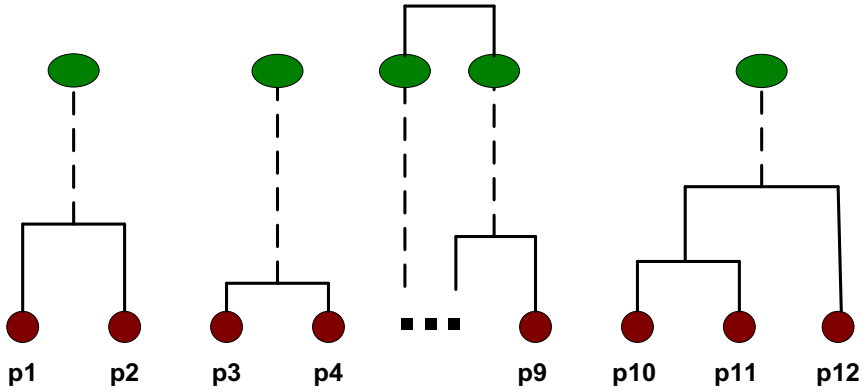
Intermediate Situation

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



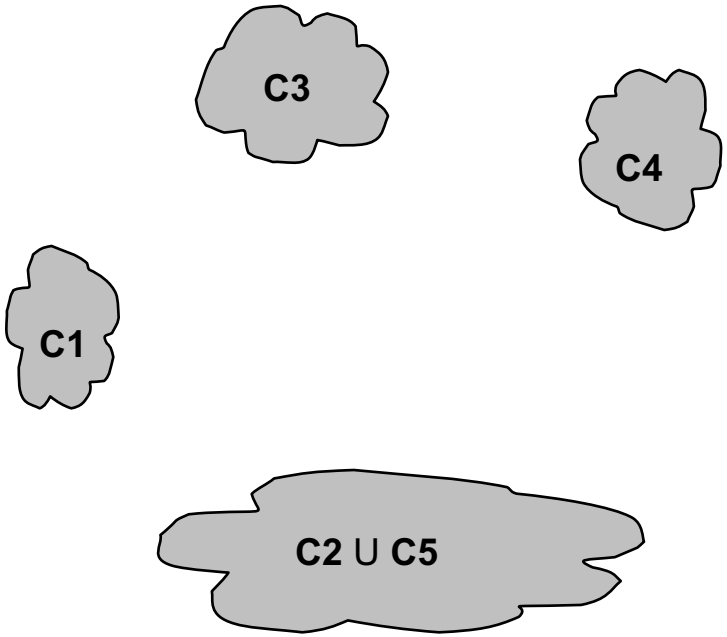
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



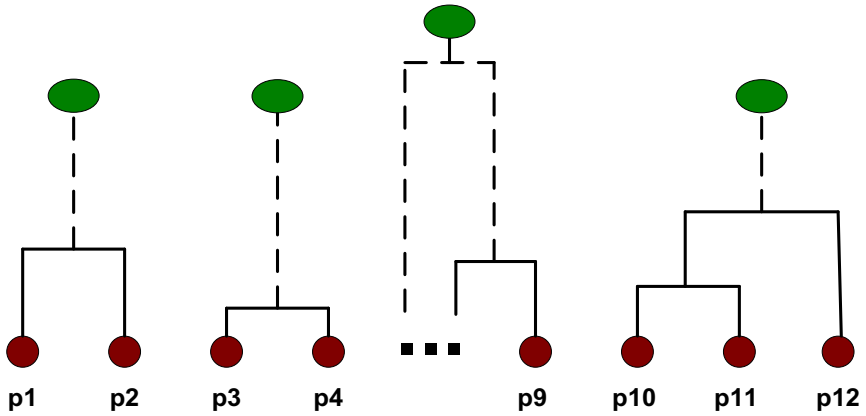
After Merging

- The question is “How do we update the proximity matrix?”

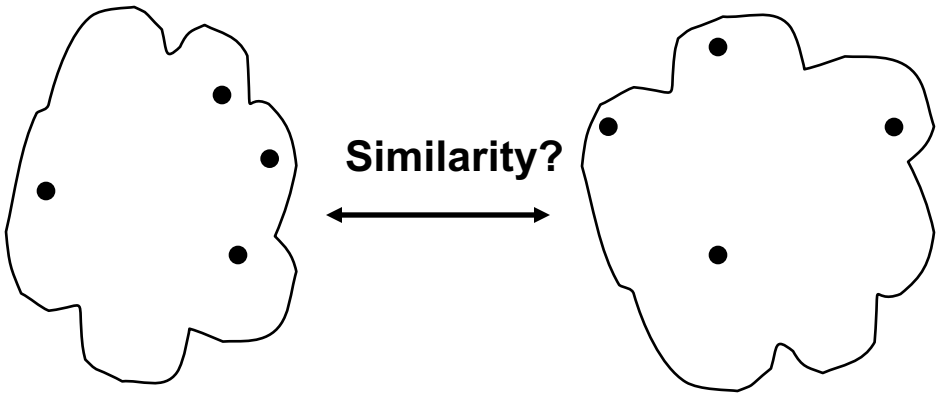


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Distance

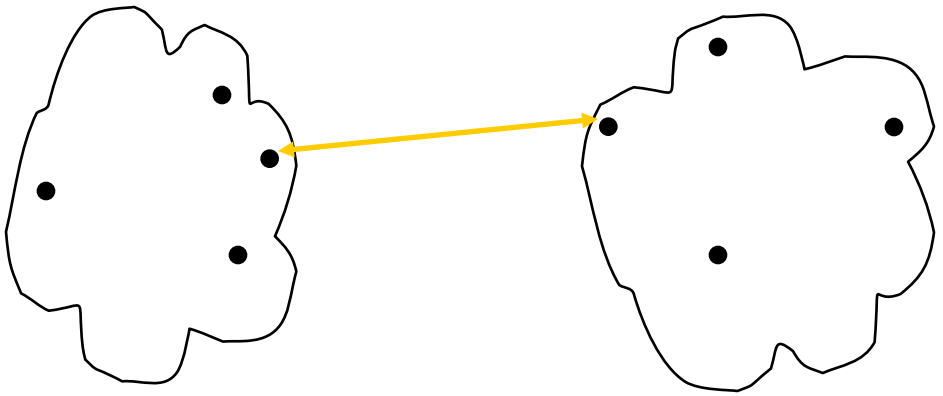


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

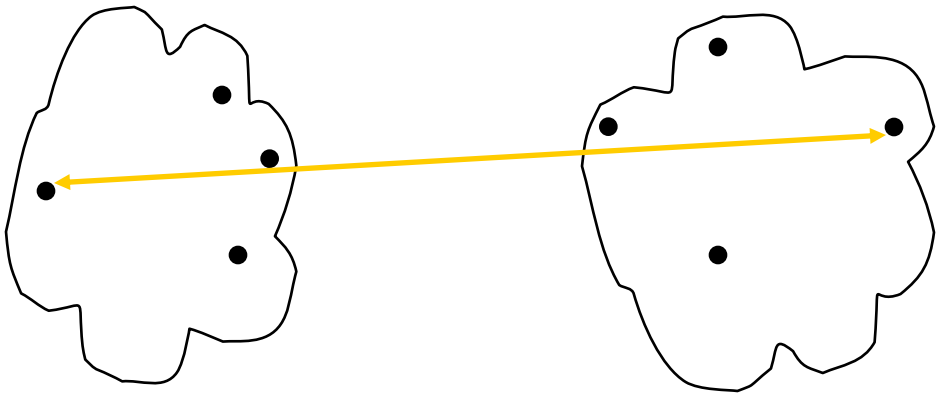


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

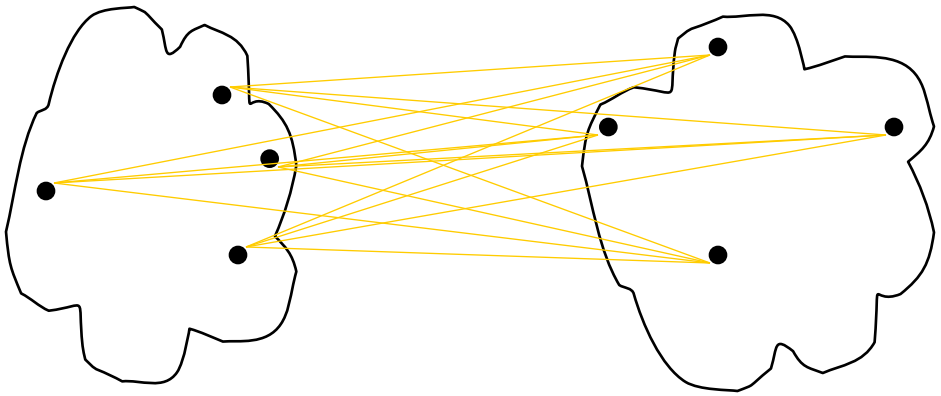


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

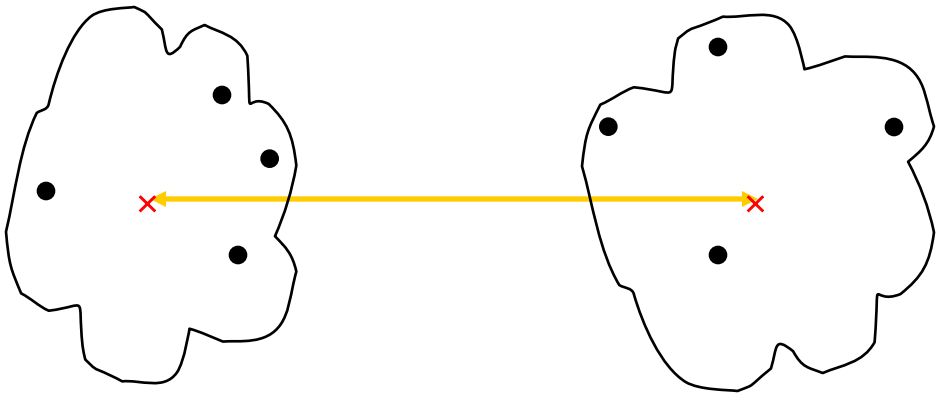


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



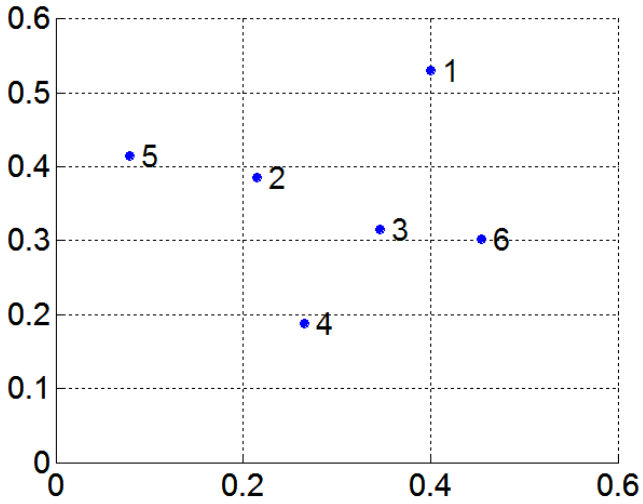
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity

- Example:



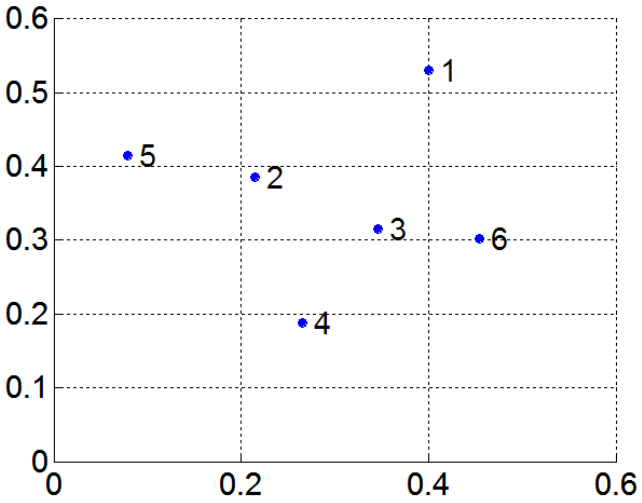
	x	y
p1	0,4005	0,5306
p2	0,2148	0,3854
p3	0,3457	0,3156
p4	0,2652	0,1875
p5	0,0789	0,4139
p6	0,4548	0,3022

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

MIN or Single Link

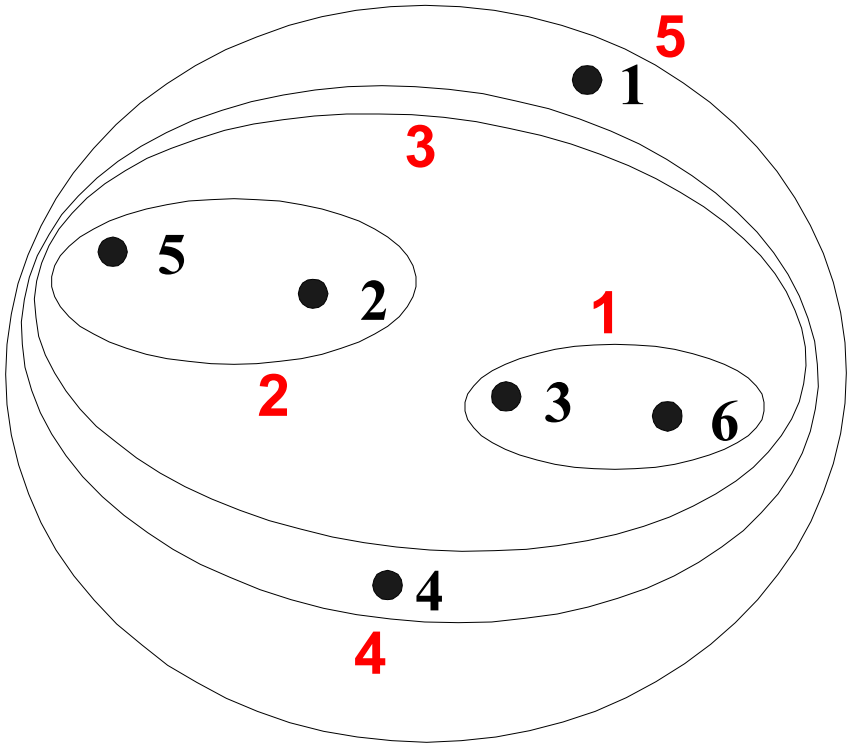
- Proximity of two clusters is based on the two closest points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph
- Example:

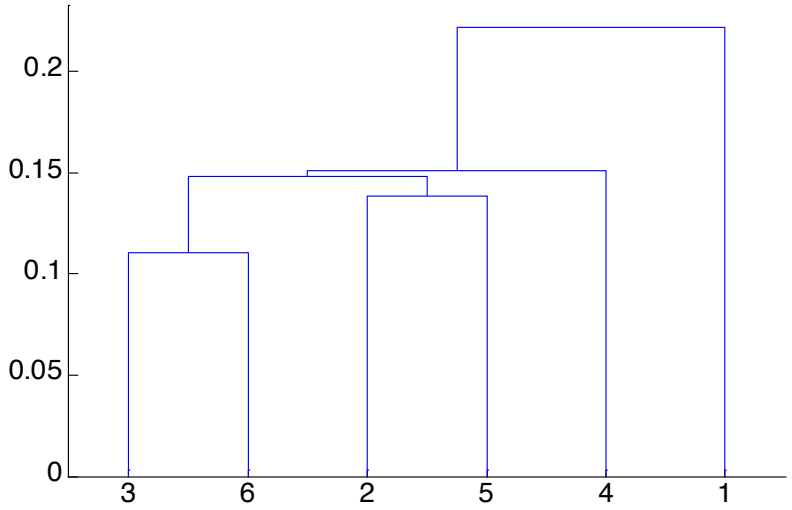


Distance Matrix:

	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

Hierarchical Clustering: MIN

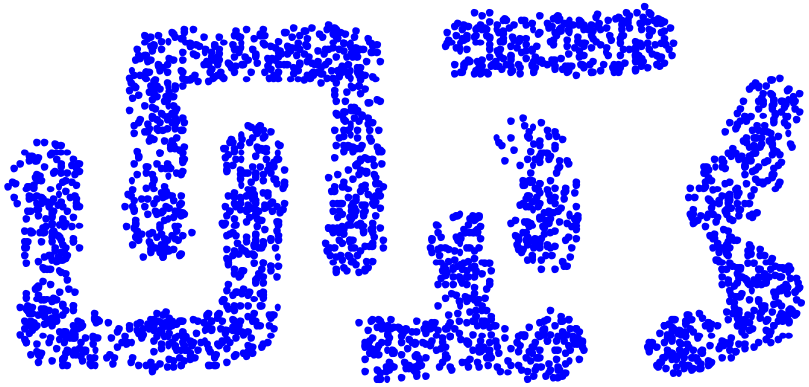




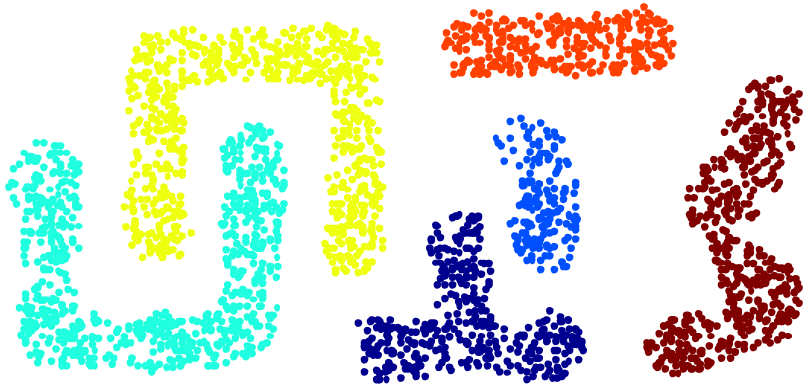
Nested Clusters

Dendrogram

Strength of MIN



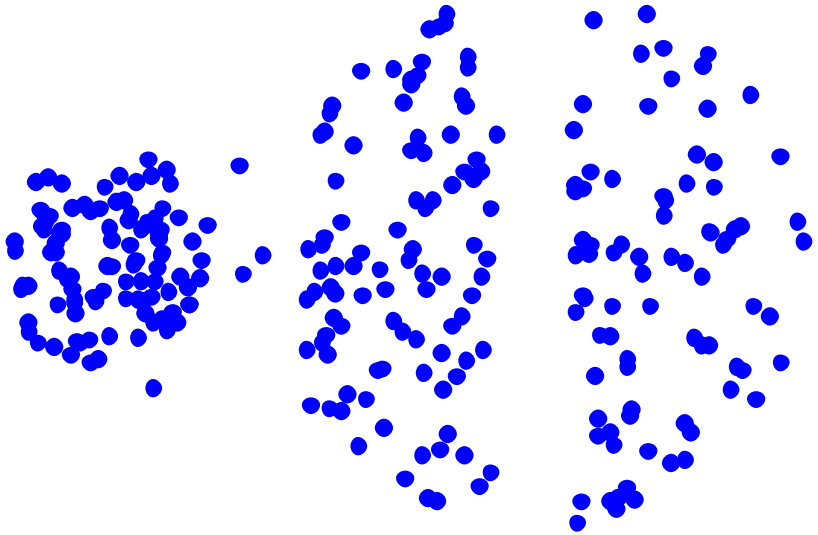
Original Points



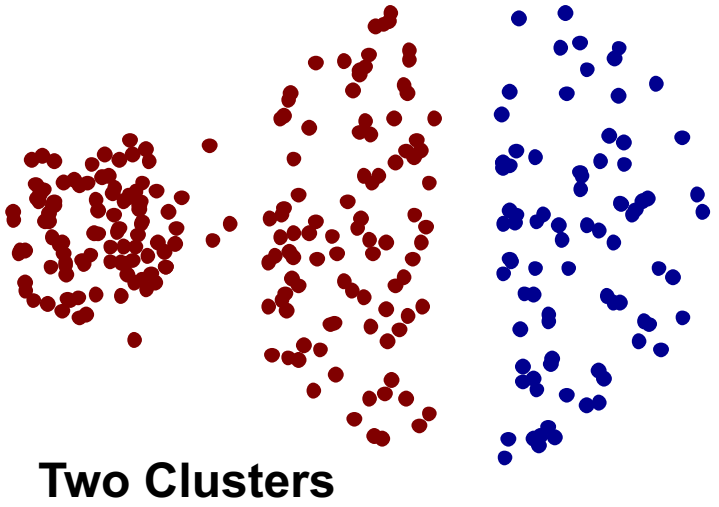
Six Clusters

- Can handle non-elliptical shapes

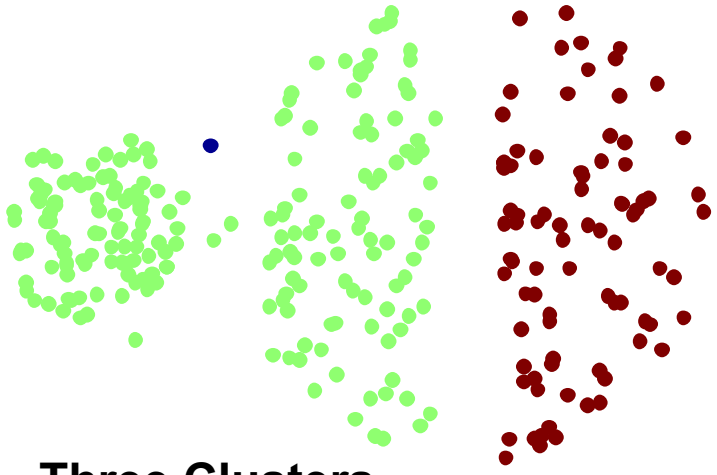
Limitations of MIN



Original Points



Two Clusters

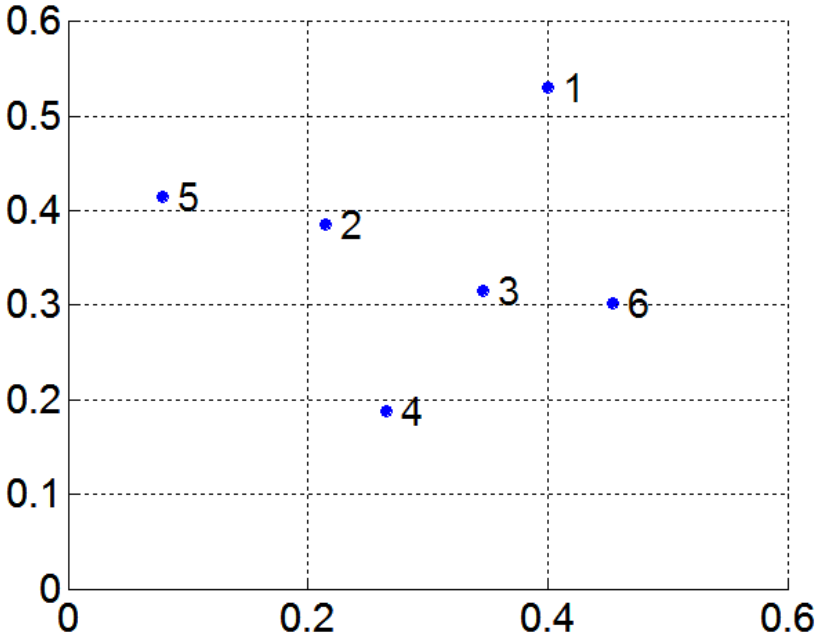


Three Clusters

- Sensitive to noise and outliers

MAX or Complete Linkage

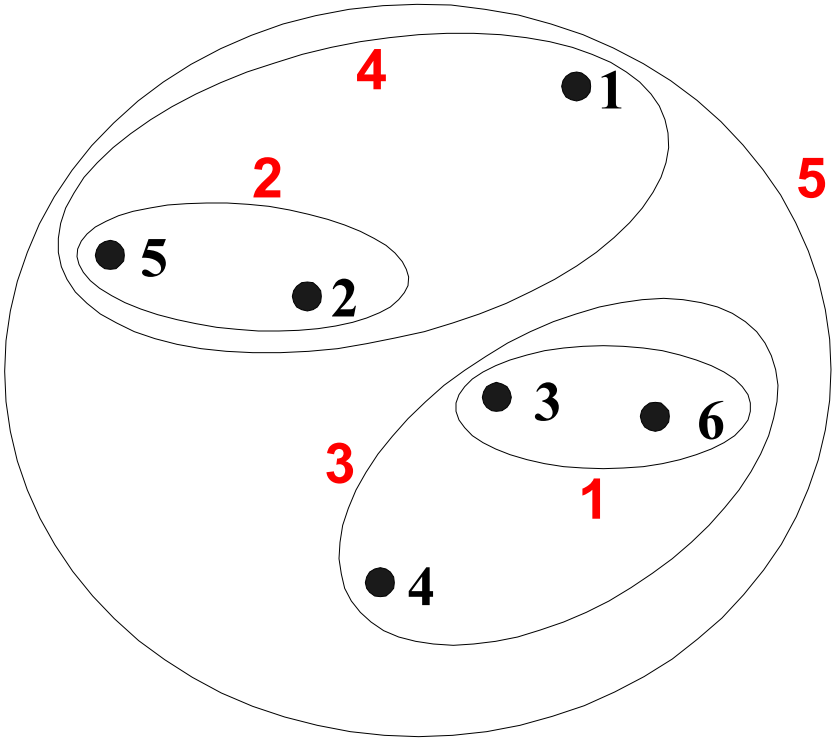
- Proximity of two clusters is based on the two most distant points in the different clusters
  - Determined by all pairs of points in the two clusters



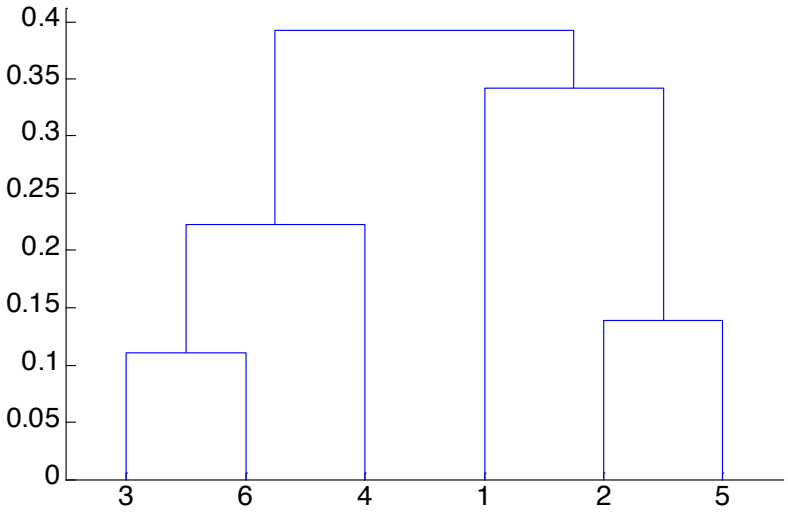
Distance Matrix:

	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

Hierarchical Clustering: MAX

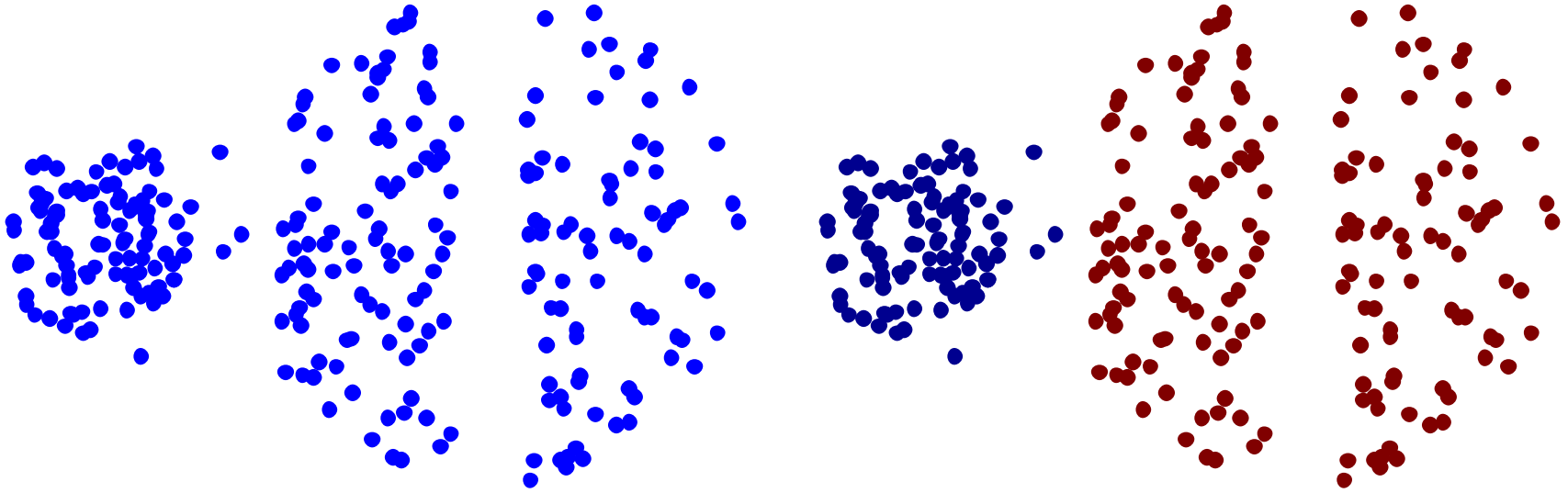


Nested Clusters



Dendrogram

Strength of MAX

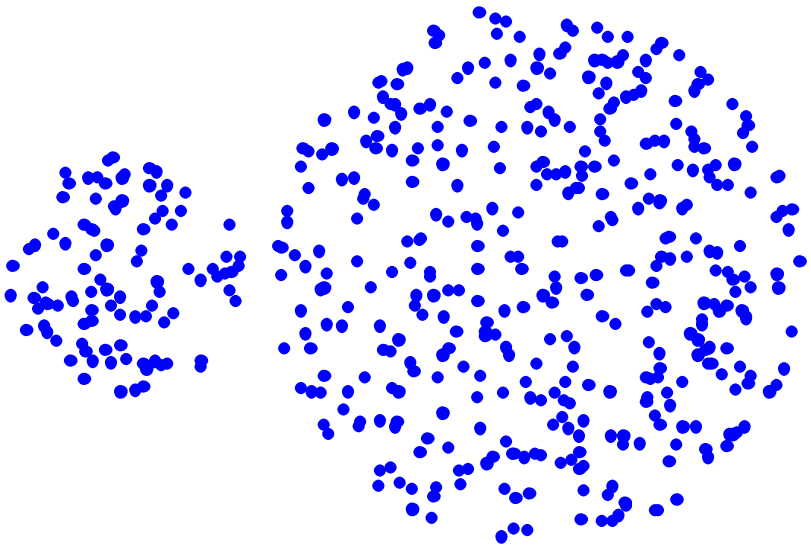


Original Points

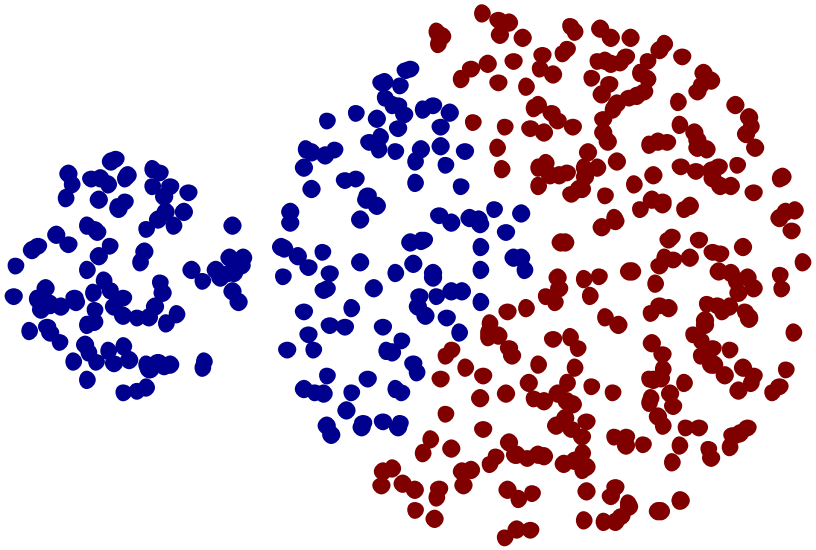
Two Clusters

- Less susceptible to noise and outliers

Limitations of MAX



Original Points



Two Clusters

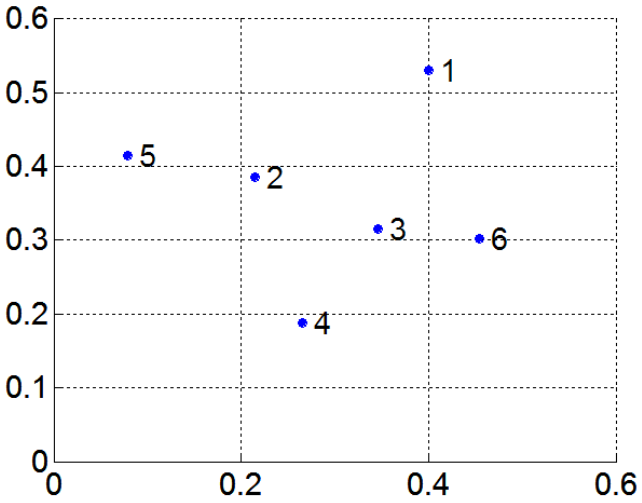
- Tends to break large clusters
- Biased towards globular clusters

Group Average

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

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

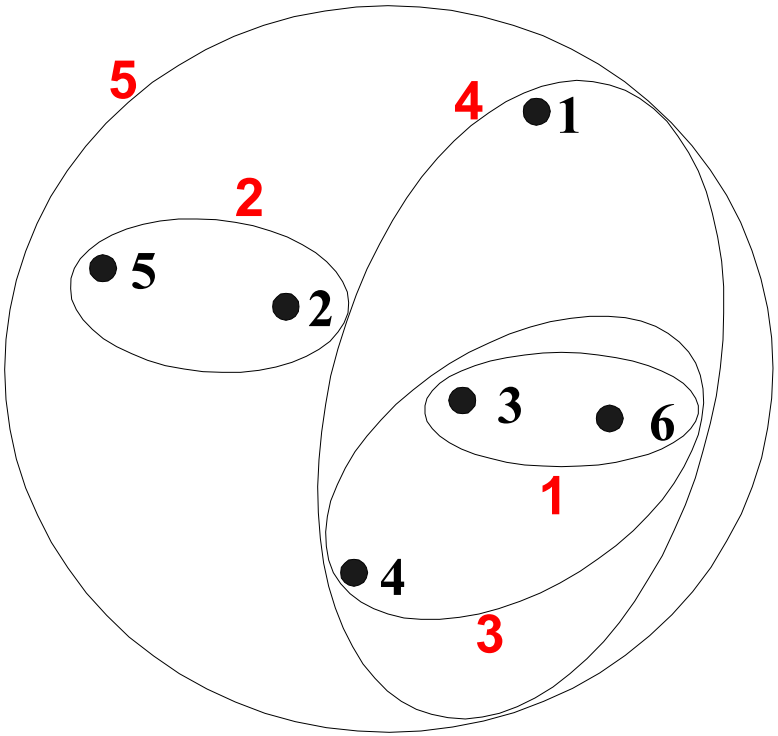
- Need to use average connectivity for scalability since total proximity favors large clusters



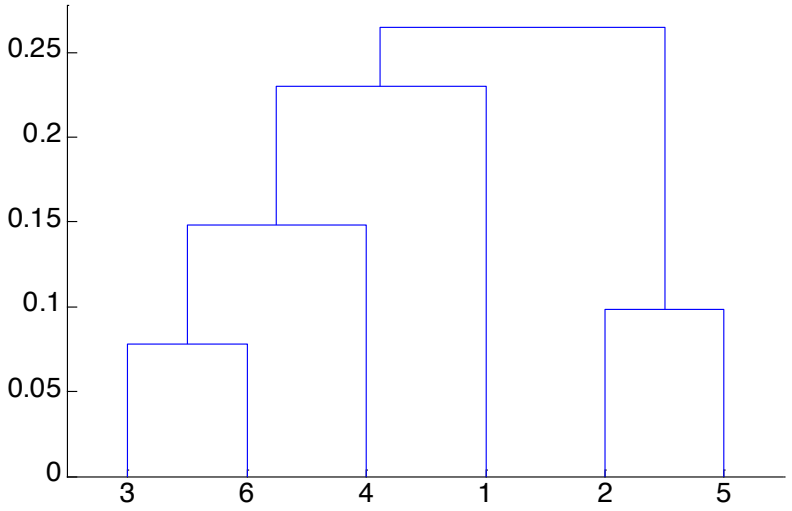
Distance Matrix:

	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

Hierarchical Clustering: Group Average



Nested Clusters



Dendrogram

Hierarchical Clustering: Group Average

- Compromise between Single and Complete Link
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Cluster Similarity: Ward's Method

- Similarity of two clusters is based on the increase in squared error when two clusters are merged

$$\Delta(A,B) = \sum_{x \in A \cup B} ||x - m_{A \cup B}||^2 - \sum_{x \in A} ||x - m_A||^2 \quad - \sum_{x \in B} ||x - m_B||^2$$

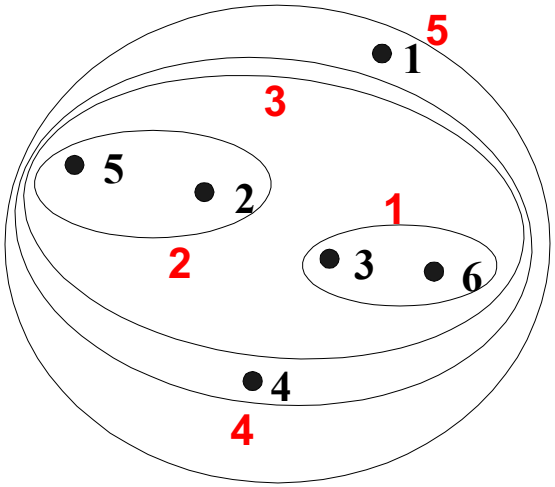
  - Similar to group average if distance between points is distance squared

- Less susceptible to noise and outliers

- Biased towards globular clusters

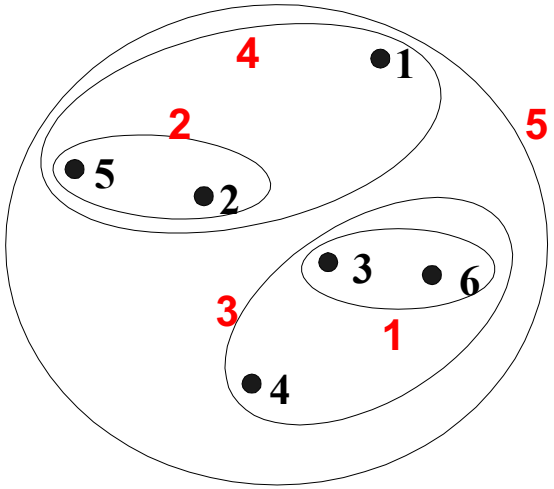
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

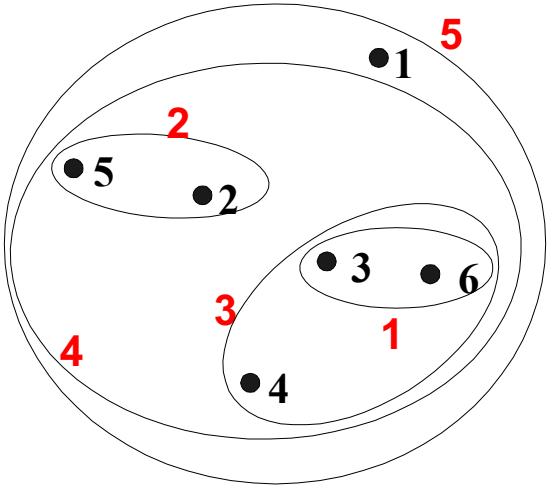
Hierarchical Clustering: Comparison



MIN

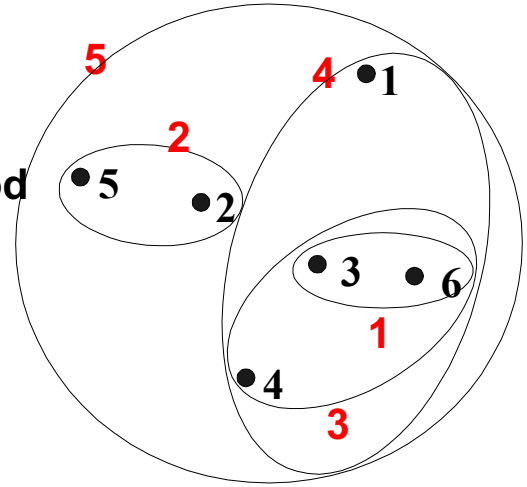
MAX





Group Average

Ward's Method



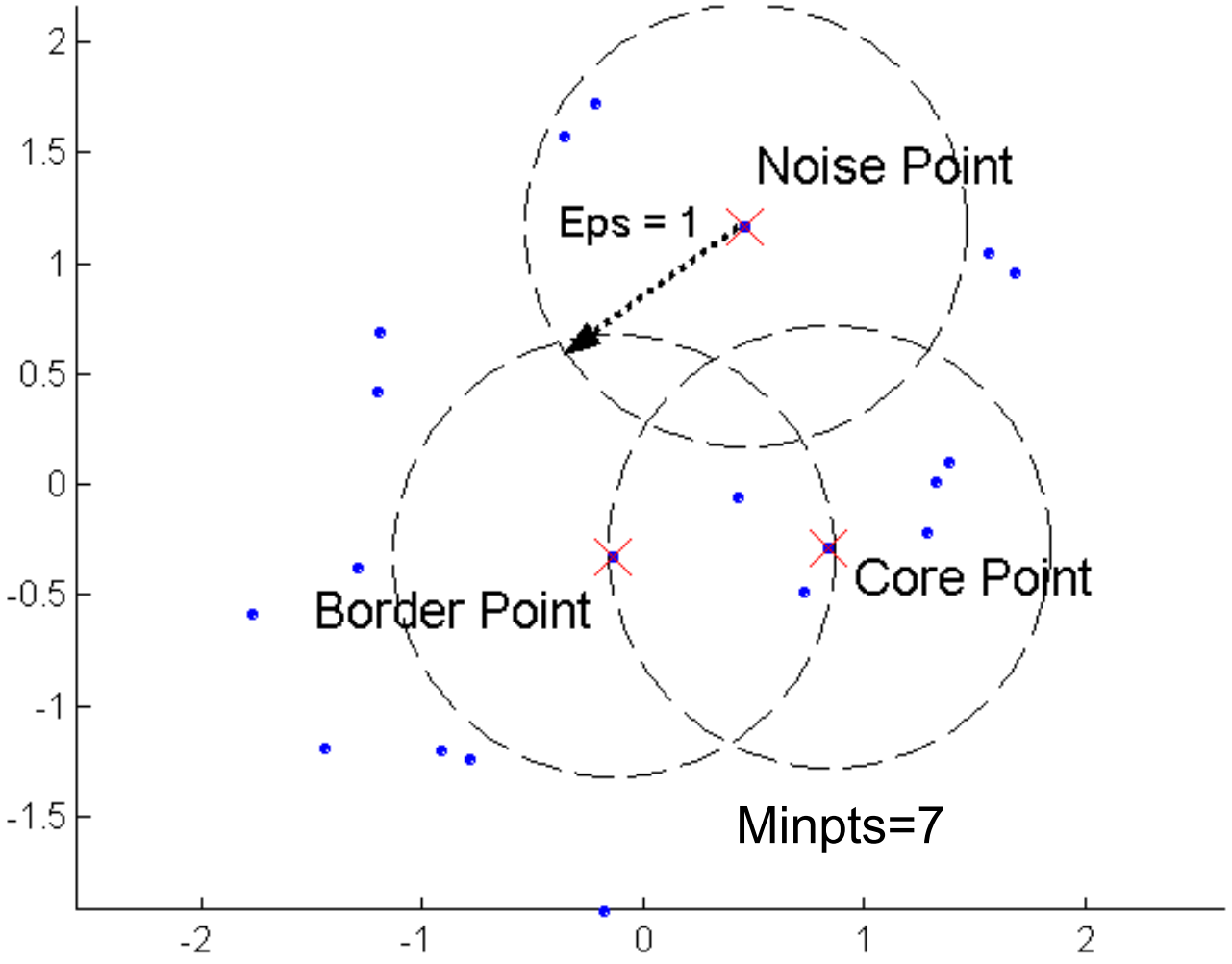
Hierarchical Clustering: Problems and Limitations

- 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 and outliers
  - Difficulty handling clusters of different sizes and non-globular shapes
  - Breaking large clusters

DBSCAN (Density-Based Spatial Clustering Application with Noise)

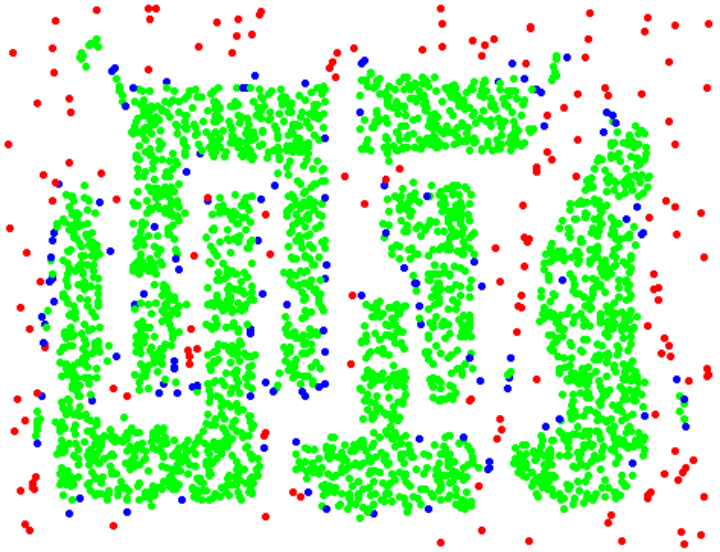
- DBSCAN is a density-based algorithm.
  - **Density** = number of points within a specified radius (Eps)
  - A point is a **core point** if it has at least a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
    - Counts the point itself
  - A **border point** is not a core point, 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. **Noise points are eliminated.**

DBSCAN: Core, Border, and Noise Points



DBSCAN: Core, Border and Noise Points

Original Points

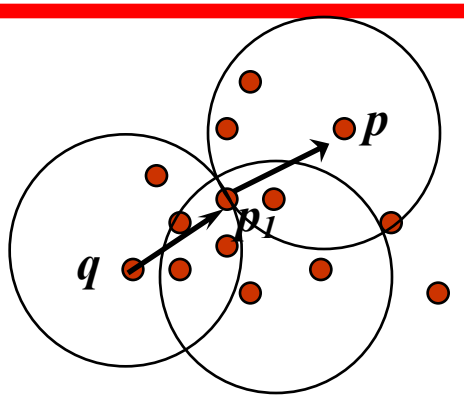


Point types: core, border and noise

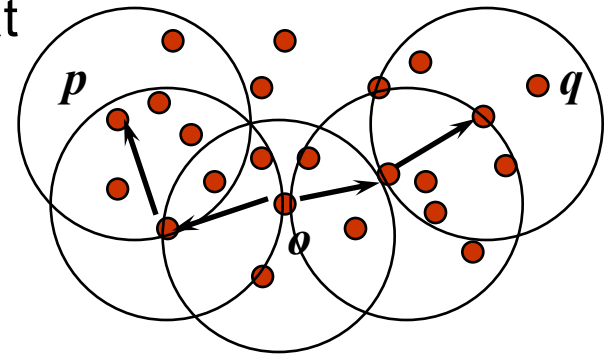
Eps = 10, MinPts = 4

Density-Based Clustering

- Directly density-reachable:
  - A point p is density-reachable from a core point q wrt. (Eps, MinPts) if p is in the neighbourhood of q



- Density-reachable:
  - A point p is density-reachable from a (core) point q wrt. (Eps, MinPts) if there is a chain of points p1, …, pn, p1 = q, pn = p such that pi+1 is directly density-reachable from pi



- Density-connected
  - A point *p* is density-connected to a point *q* wrt. (Eps, *MinPts)* if there is a point *s* such that both, *p* and *q* are density-reachable from *s* wrt. (Eps, *MinPts)*.

DBSCAN Algorithm

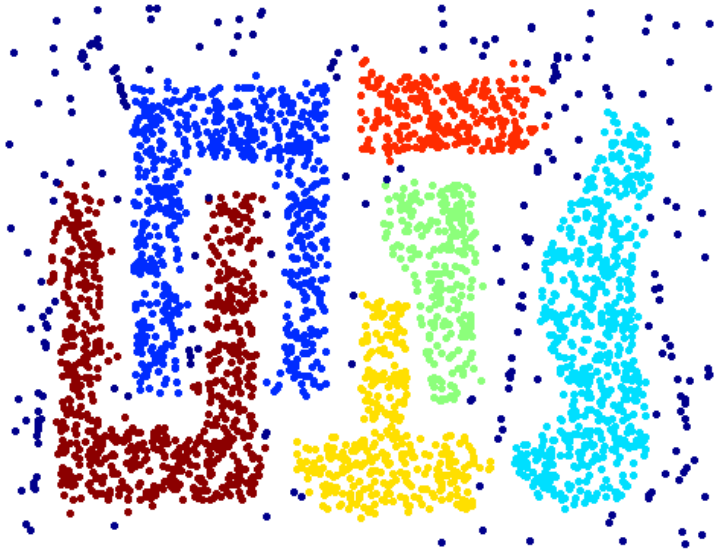
Algorithm 8.4 DBSCAN algorithm.

1. Label all points as core, border, or noise points.
2. Eliminate noise points,
3. Put an edge between all core points that are within 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 complexity: O(n x time to find points in the Eps-neighborhood) = $O(n^2)$, where n = number of points.
- Space complexity: O(n).

When DBSCAN Works Well

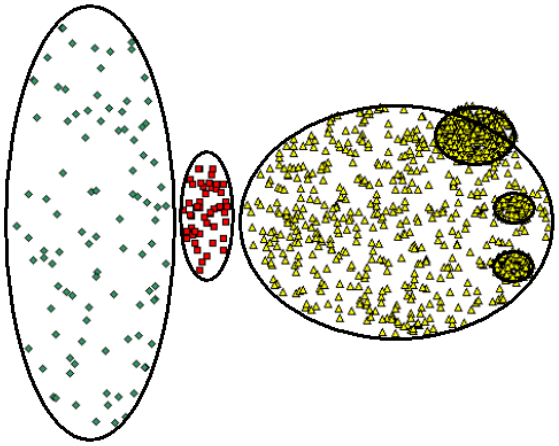
Original Points



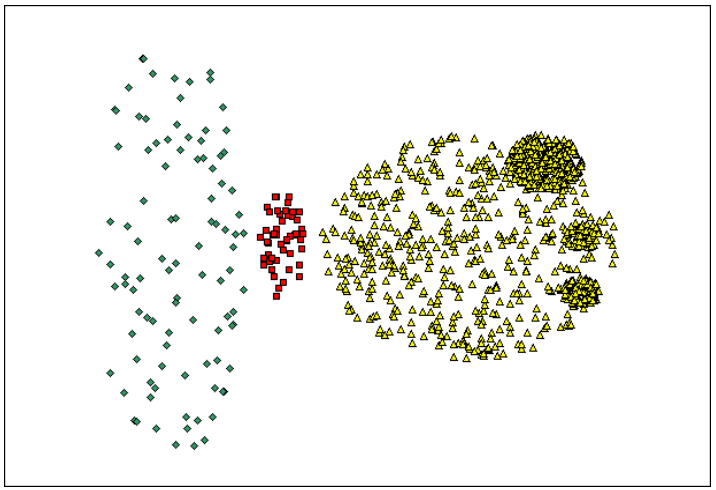
Clusters

- Resistant to Noise,
- Can handle clusters of different shapes and sizes,
- Requires user to supply Minpts and Eps.

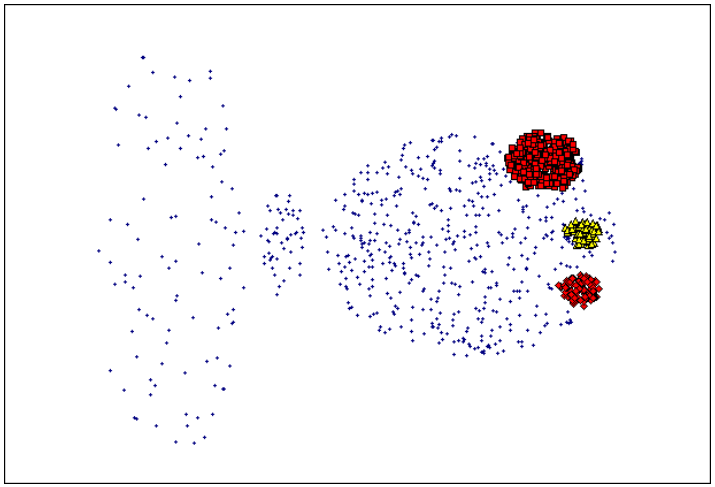
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).

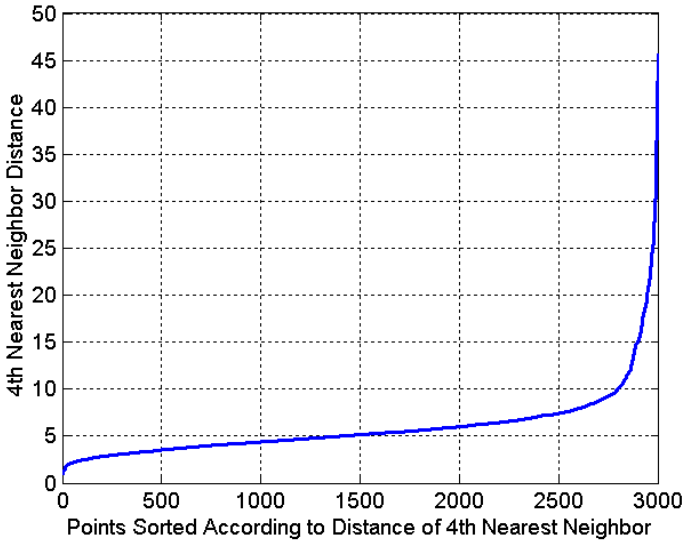


(MinPts=4, Eps=9.92)

- Varying densities
- High-dimensional data

DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor
- The original DBSCAN algorithm used a value of k=4, which appears to be a reasonable value for most 2-dimensional data sets

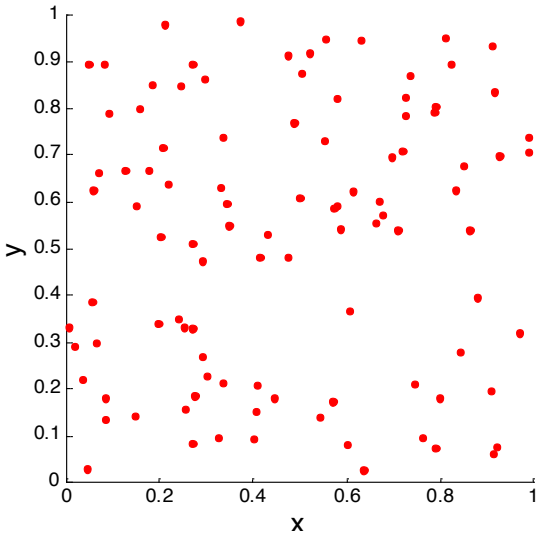


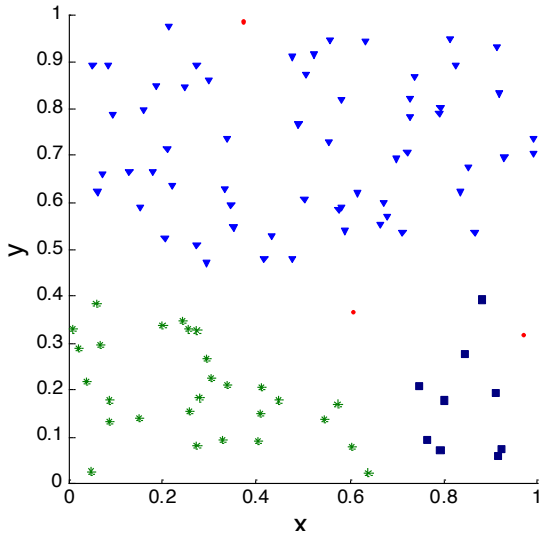
Cluster Validity

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

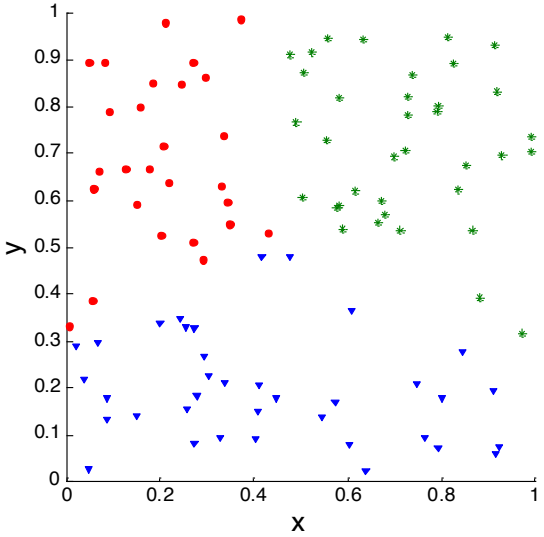
Random Points

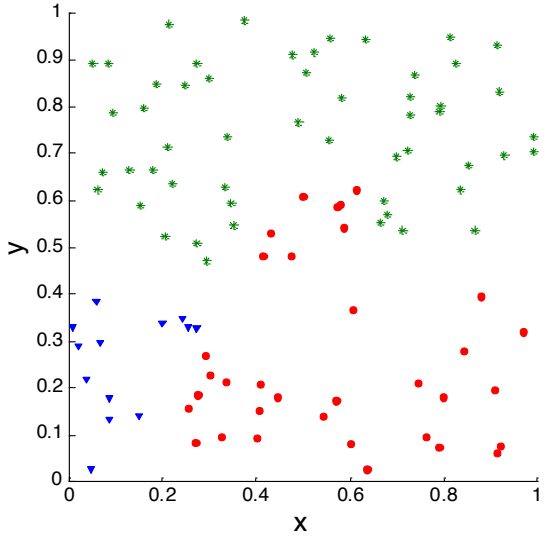




DBSCAN

K-means





Complete Link

Different Aspects of Cluster Validation

1. Determining the **clustering tendency** of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.

   - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the 'correct' number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Measures of Cluster Validity

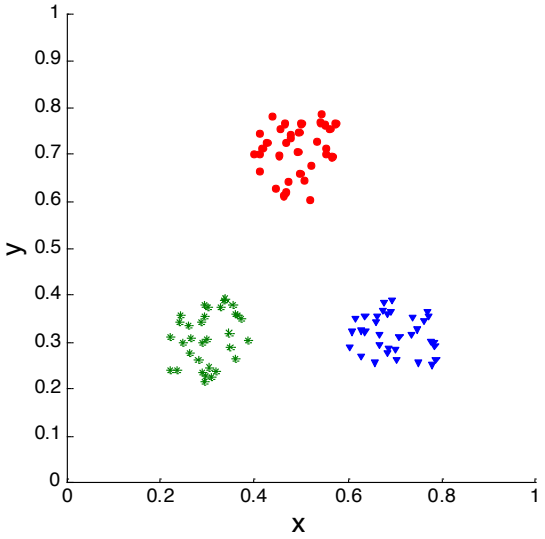
- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - **External Index:** Used to measure the extent to which cluster labels match externally supplied class labels.
    - **Entropy**
  - **Internal Index**: Used to measure the goodness of a clustering structure *without* respect to external information.
    - **Sum of Squared Error (SSE)**
  - **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
- Sometimes these are referred to as **criteria** instead of **indices**
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

Measuring Cluster Validity Via Correlation

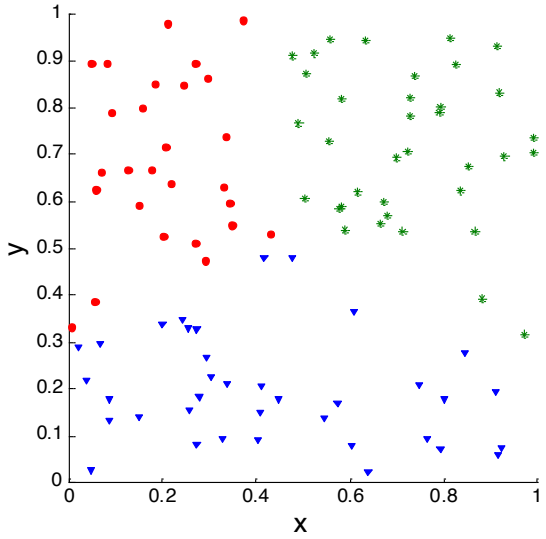
- Two matrices
  - Proximity Matrix
  - Ideal Similarity Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between n(n-1) / 2 entries needs to be calculated.
- High 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.



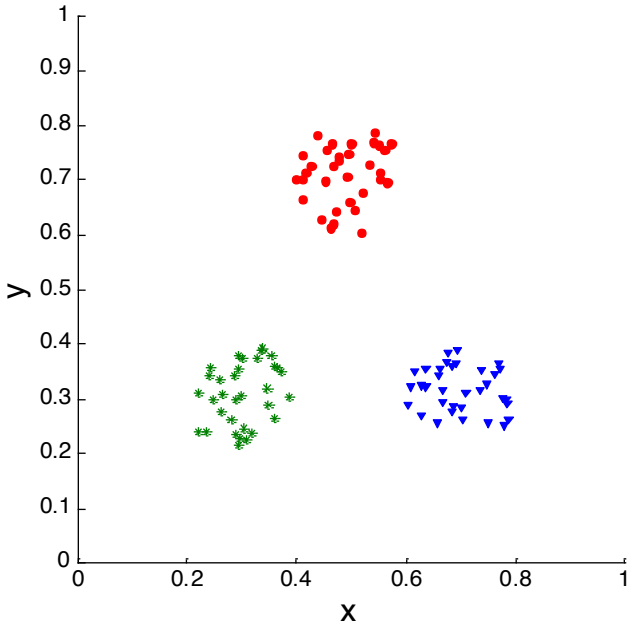
Corr = -0.9235

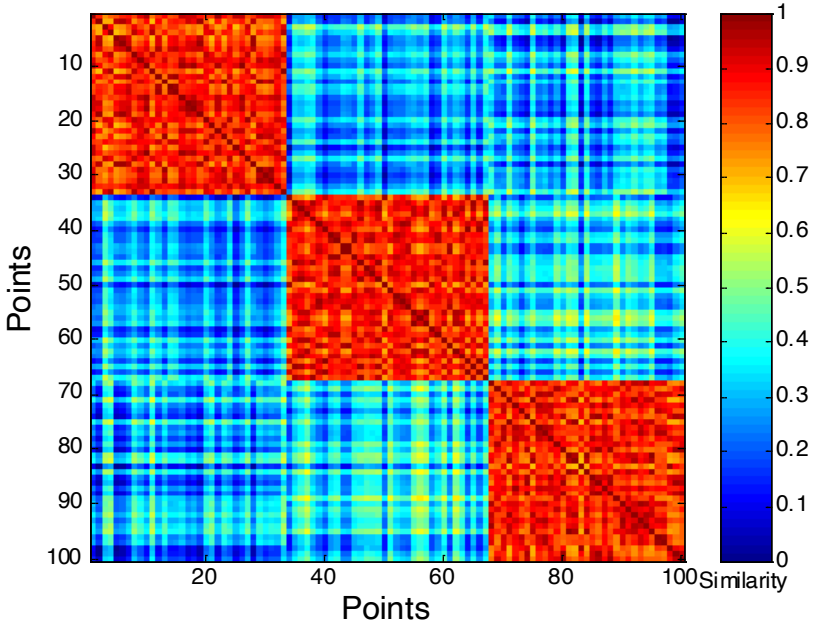


Corr = -0.5810

Using Similarity Matrix for Cluster Validation

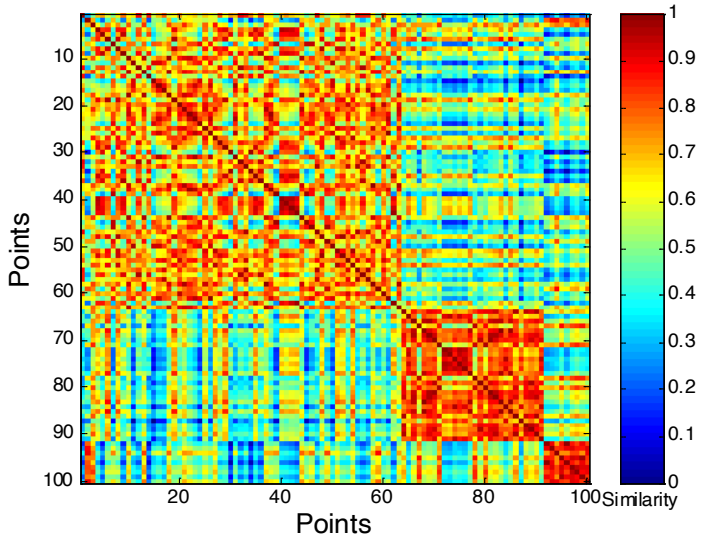
- Order the similarity matrix with respect to cluster labels and inspect visually.

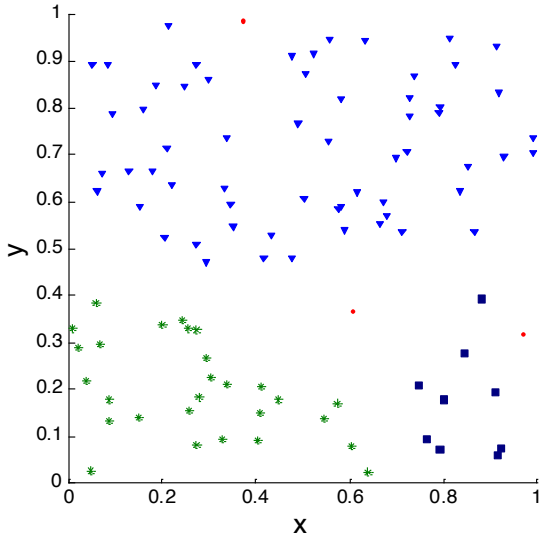




Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

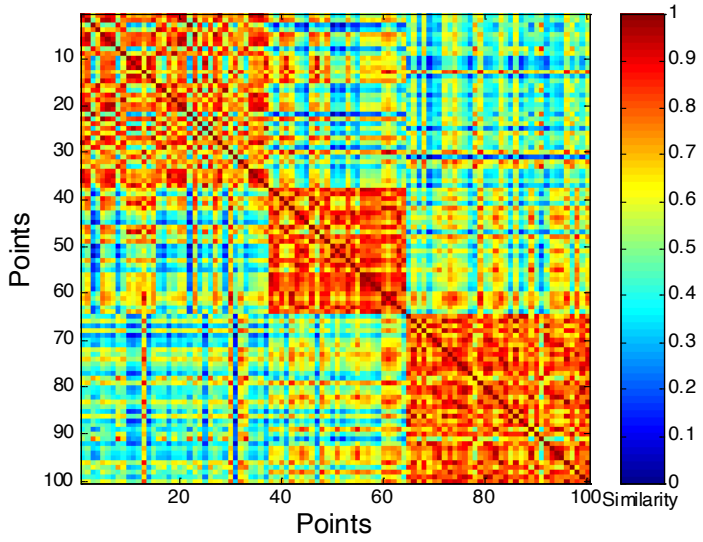


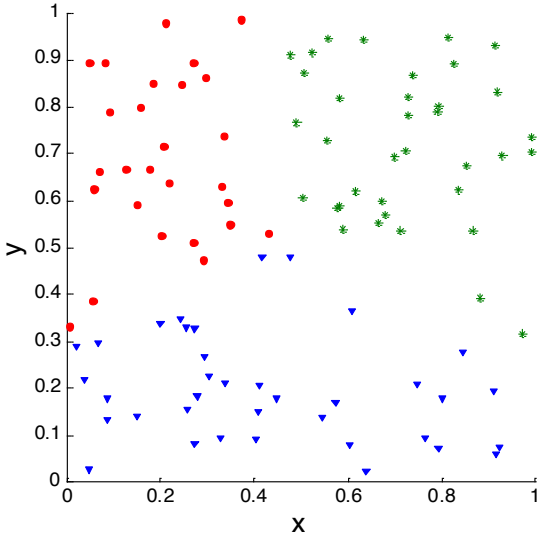


DBSCAN

Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

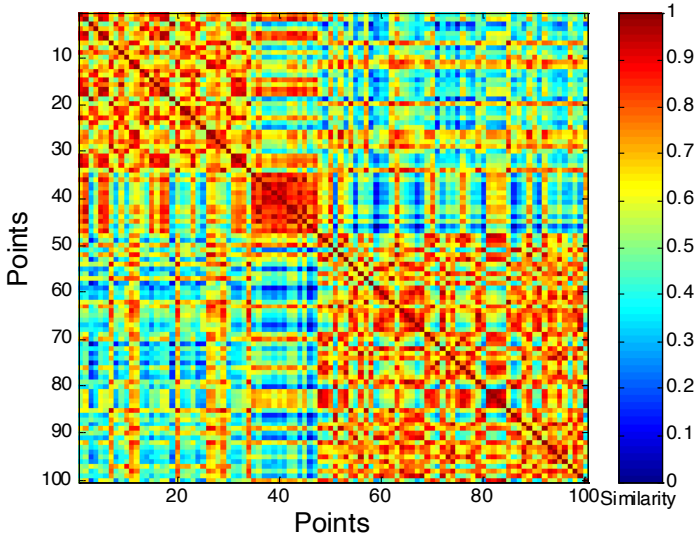


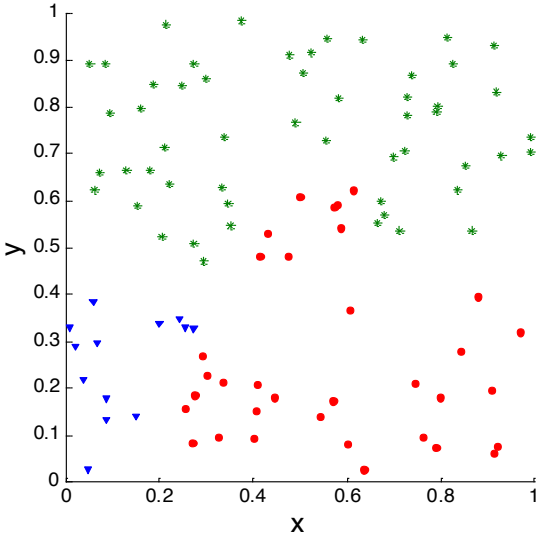


K-means

Using Similarity Matrix for Cluster Validation

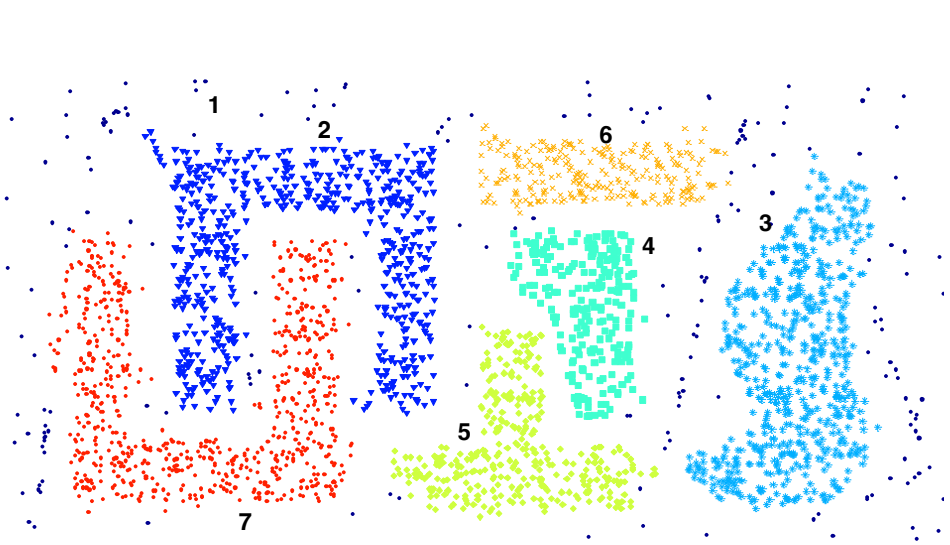
- Clusters in random data are not so crisp

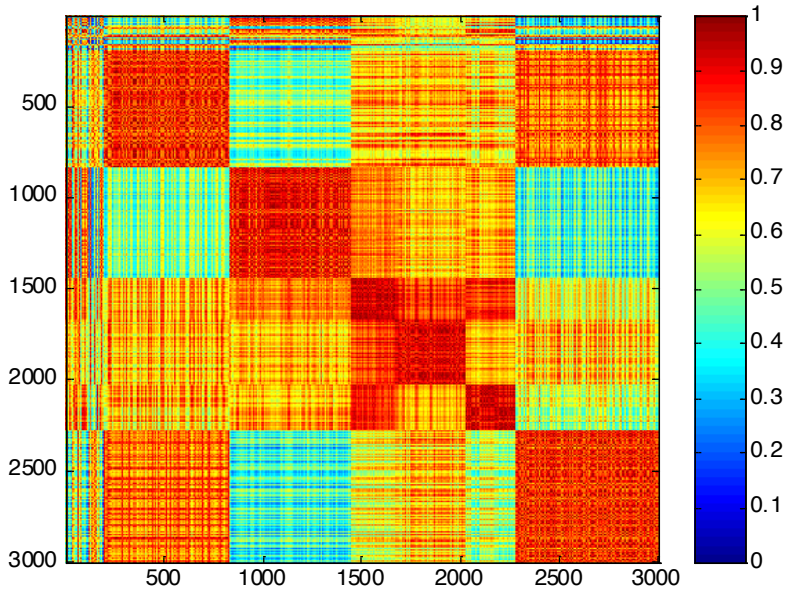




Complete Link

Using Similarity Matrix for Cluster Validation

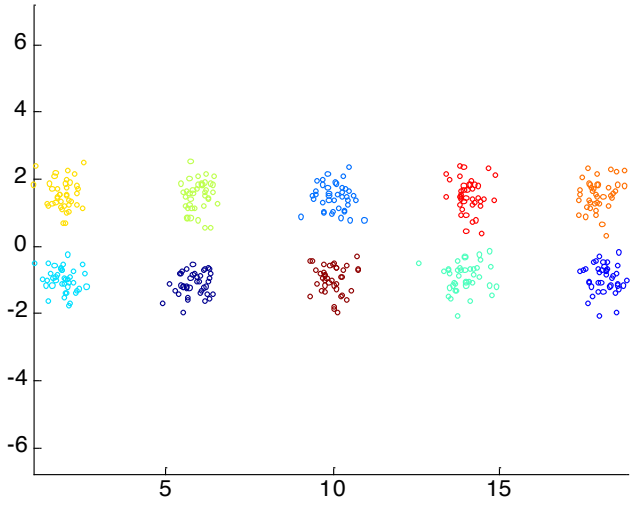


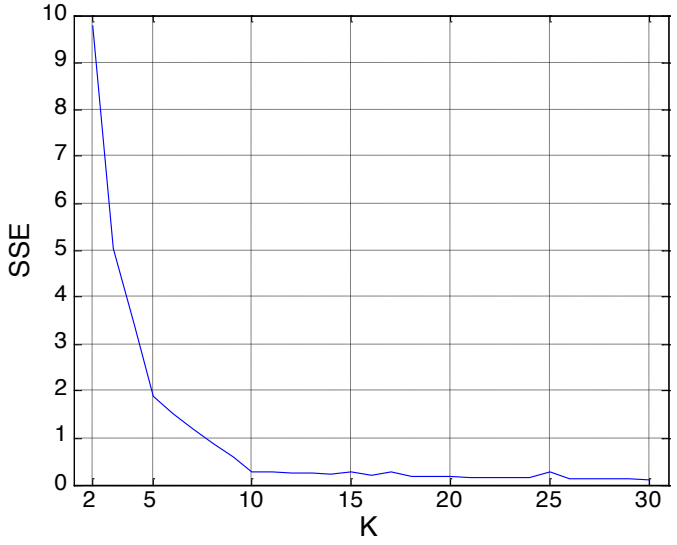


DBSCAN

Internal Measures: SSE

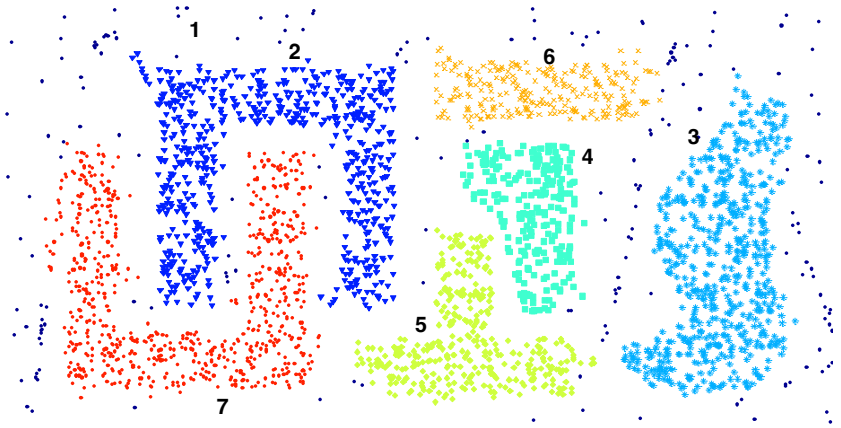
- Clusters in more complicated figures aren't well separated
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information
  - SSE
- SSE is good for comparing two clusterings or two clusters (average SSE).
- Can also be used to estimate the number of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

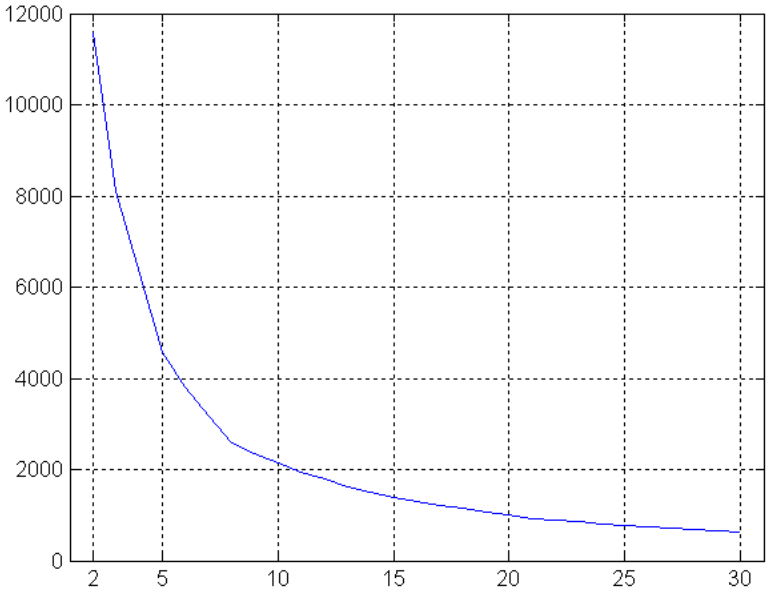




Internal Measures: SSE

- SSE curve for a more complicated data set





SSE of clusters found using K-means

Framework for Cluster Validity

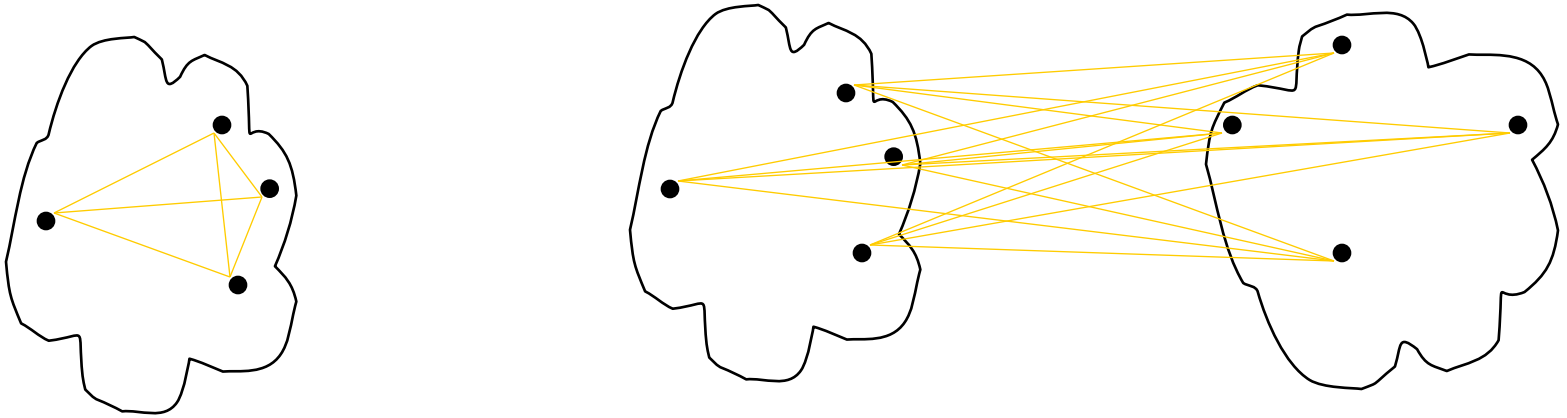
- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more “atypical” a clustering result is, the more likely it represents valid structure in the data
  - 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

Internal Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
  - Example: SSE
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters

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

- $cohesion(C_i) = \sum_{x \in C_i \wedge y \in C_i} proximity(x, y)$
- $separation(C_i, C_j) = \sum_{x \in C_i \wedge y \in C_j} proximity(x, y)$

Internal Measures: Cohesion and Separation

- Prototype view of cohesion and separation
  - $cohesion(C_i) = \sum_{x \in C_i} proximity(x, m_i)$
  - $separation(C_i, C_j) = proximity(m_i, m_j)$
  - $separation(C_i) = proximity(m_i, m)$
- Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

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

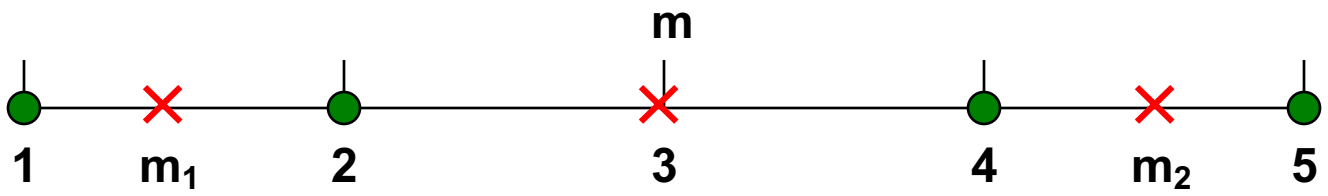
  - Separation is measured by the between cluster sum of squares

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

    - Where $|C_i|$ is the size of cluster i

Internal Measures: Cohesion and Separation

- Example: SSE
  - BSS + WSS = constant



K=1 cluster:

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

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

$$Total = 10 + 0 = 10$$

K=2 clusters:

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

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

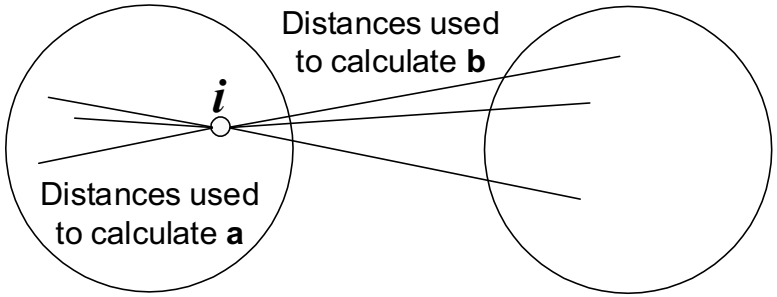
$$Total = 1 + 9 = 10$$

Internal Measures: Silhouette Coefficient

- 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*** = min (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

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

Cluster Analysis: Further Techniques

Hard (Crisp) vs Soft (Fuzzy) Clustering

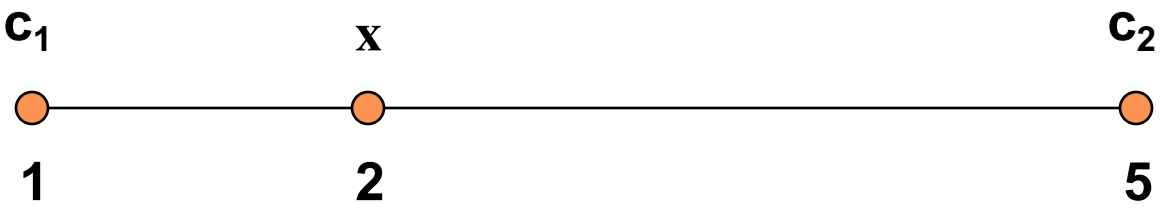
- Hard (Crisp) vs. Soft (Fuzzy) clustering
  - **For soft clustering allow point to belong to more than one cluster**
  - For K-means, generalize objective function

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

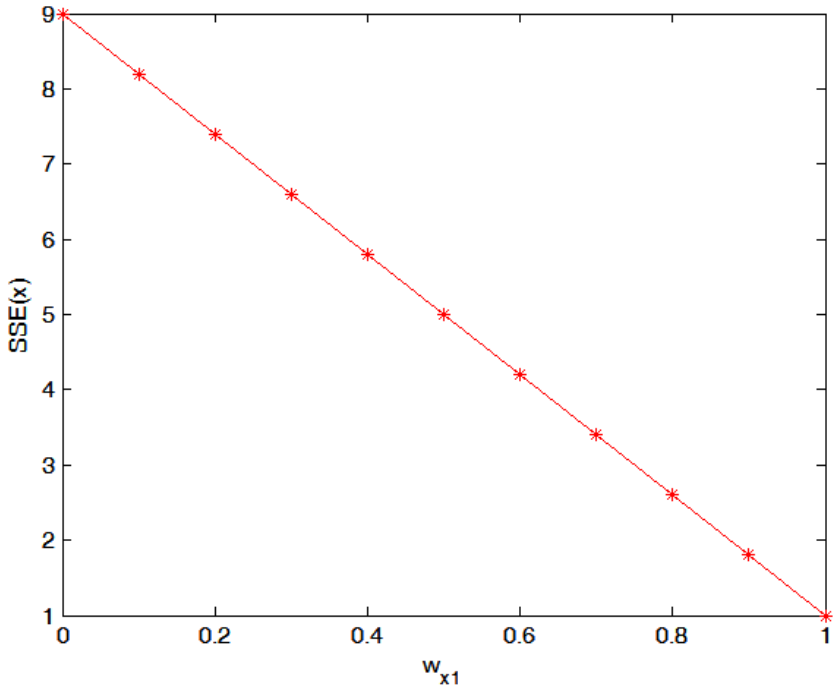
w_{ij} : weight with which object x_i belongs to cluster $\boldsymbol{c_j}$

  - To minimize SSE, repeat the following steps:
    - Fix $\boldsymbol{c_j}$ and determine w_{ij} (cluster assignment)
    - Fix w_{ij} and recompute $\boldsymbol{c_j}$
  - Hard clustering: $w_{ij} \in \{0,1\}$

Soft (Fuzzy) Clustering: Estimating Weights



$$SSE(x) = w_{x1}(2-1)^2 + w_{x2}(5-2)^2$$
$$= w_{x1} + 9w_{x2}$$



SSE(x) is minimized when $w_{x1} = 1$, $w_{x2} = 0$

Fuzzy C-means

- Objective function

p: fuzzifier (p > 1)

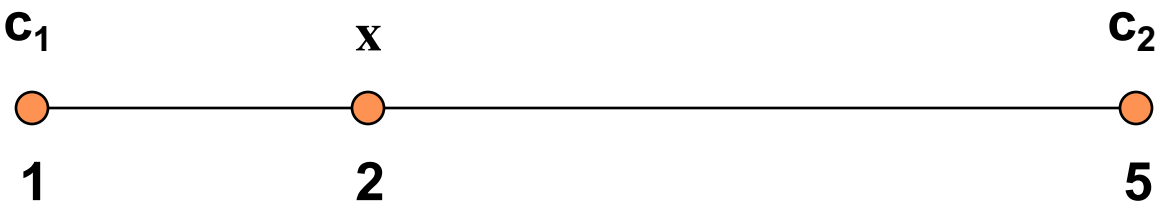
$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{m} w_{ij}^{p} \, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

  - w_{ij}: weight with which object $\boldsymbol{x}_i$ belongs to cluster $\boldsymbol{c_j}$
  - p: a power for the weight not a superscript and controls how "fuzzy" the clustering is

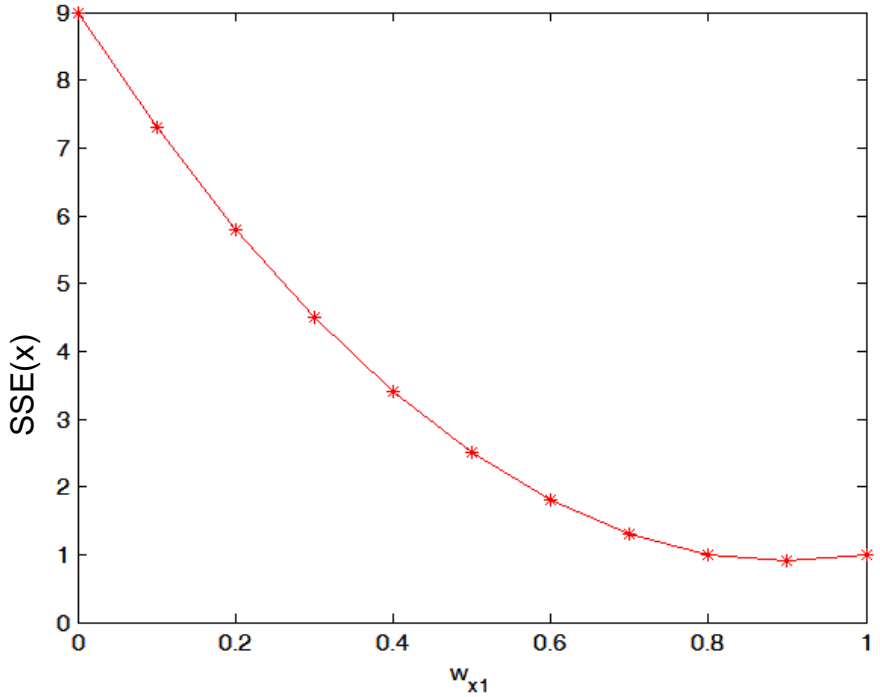
  - To minimize objective function, repeat the following:
    - Fix $\boldsymbol{c_j}$ and determine w_{ij}
    - Fix w_{ij} and recompute $\boldsymbol{c_j}$

  - Fuzzy c-means clustering: $w_{ij} \in [0,1]$

Fuzzy C-means



$$SSE(x) = w_{x1}^2(2-1)^2 + w_{x2}^2(5-2)^2$$
$$= w_{x1}^2 + 9w_{x2}^2$$



SSE(x) is minimized when w_{x1} = 0.9, w_{x2} = 0.1

Fuzzy C-means

- Objective function:

p: fuzzifier (p > 1)

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{m} w_{ij}^{p}\, dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

- Initialization: choose the weights w_{ij} randomly
- Repeat:
  - Update centroids:

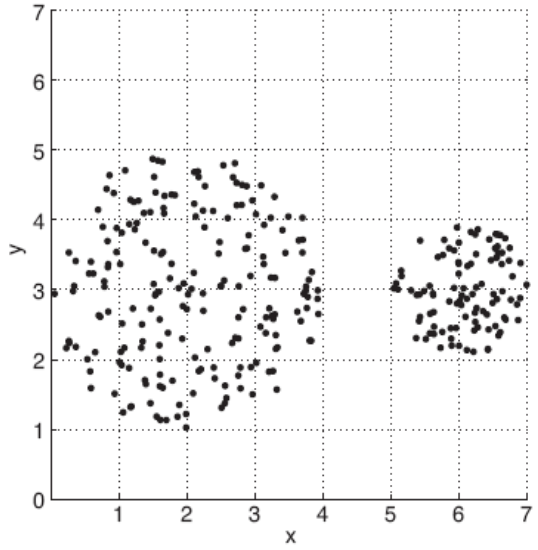
$$\boldsymbol{c_j} = \sum_{i=1}^{m} w_{ij}\boldsymbol{x}_i \,/ \sum_{i=1}^{m} w_{ij}$$

  - Update weights:

$$w_{ij} = (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}} / \sum_{j=1}^{k} (1/dist(\boldsymbol{x}_i, \boldsymbol{c}_j)^2)^{\frac{1}{p-1}}$$

Grid-based Clustering

- A type of density-based clustering



0	0	0	0	0	0	0
0	0	0	0	0	0	0
4	17	18	6	0	0	0
14	14	13	13	0	18	27
11	18	10	21	0	24	31
3	20	14	4	0	0	0
0	0	0	0	0	0	0

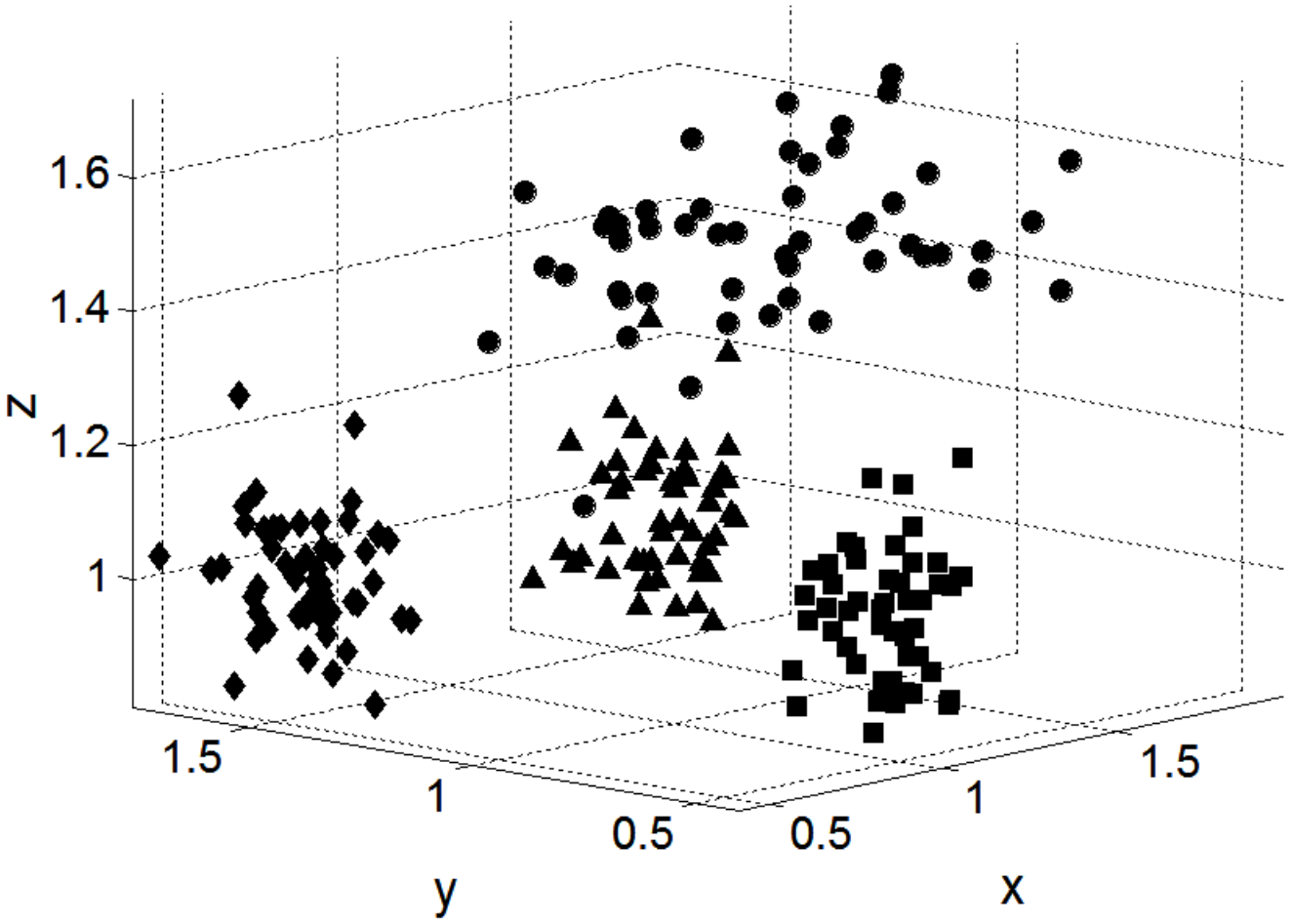
Algorithm 9.4 Basic grid-based clustering algorithm.

1: Define a set of grid cells.
2: Assign objects to the appropriate cells and compute the density of each cell.
3: Eliminate cells having a density below a specified threshold, τ.
4: Form clusters from contiguous (adjacent) groups of dense cells.

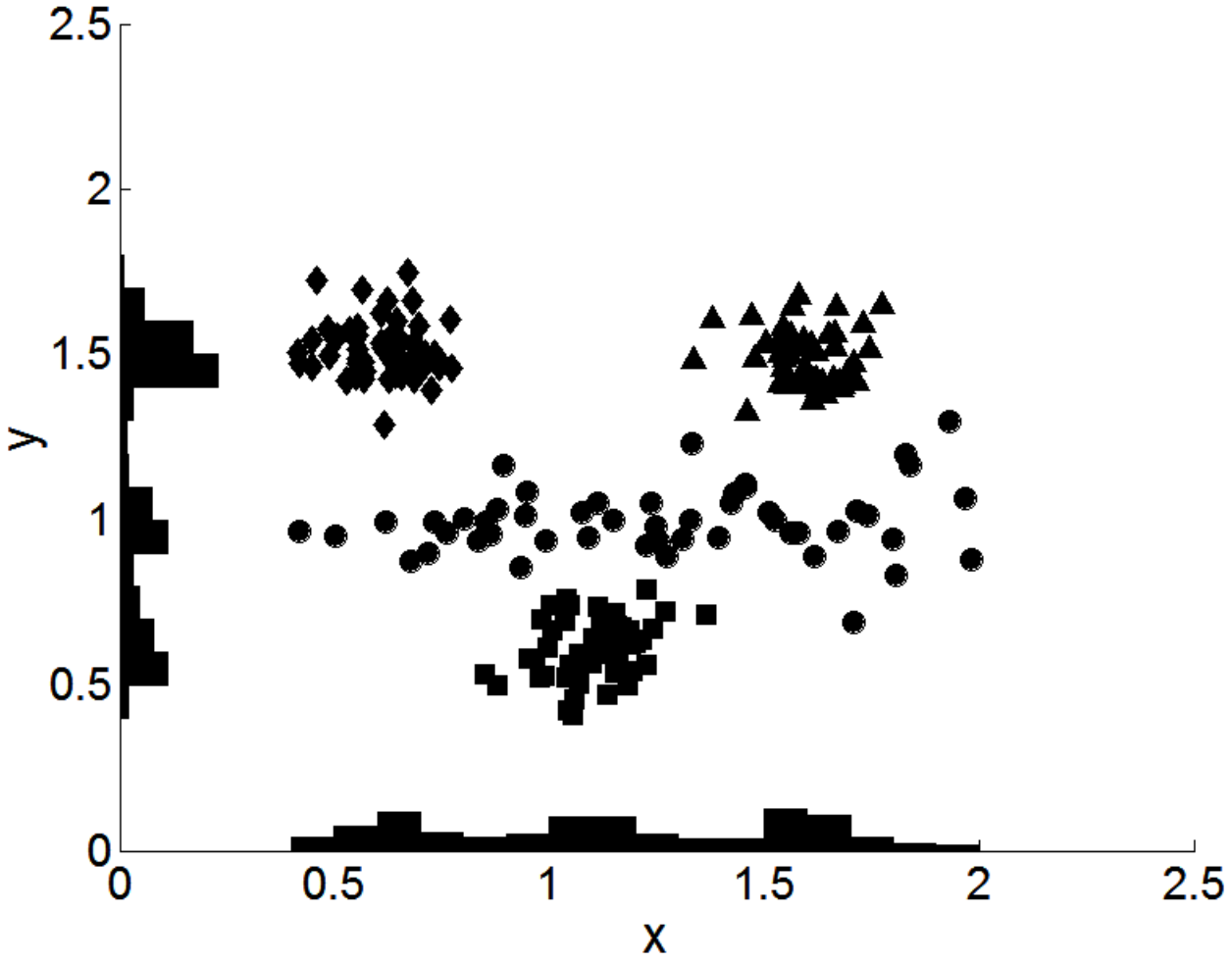
Subspace Clustering

- Until now, we found clusters by considering all of the attributes
- Some clusters may involve only a subset of attributes, i.e., subspaces of the data
  - Example:
    - When k-means is used to find document clusters, the resulting clusters can typically be characterized by 10 or so terms

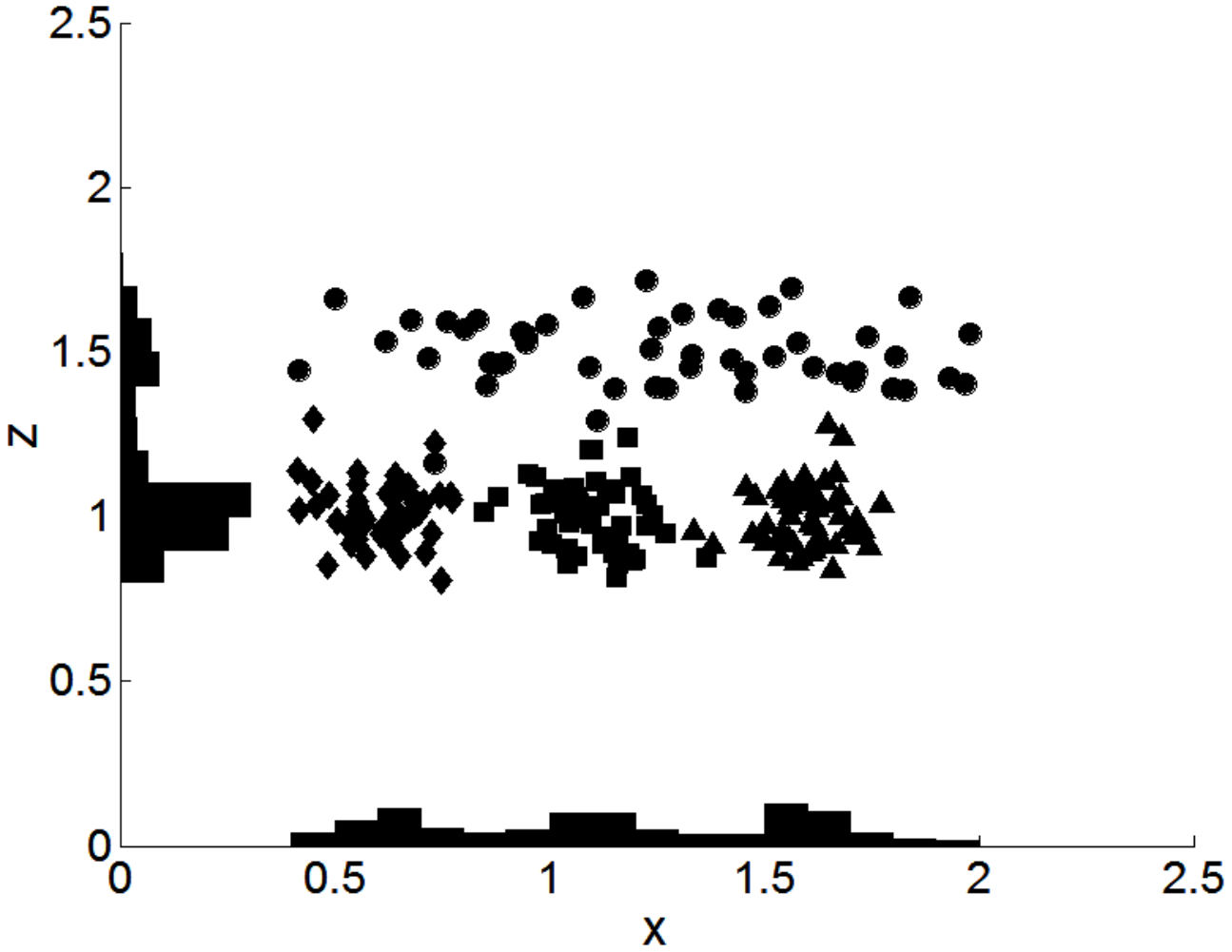
Example



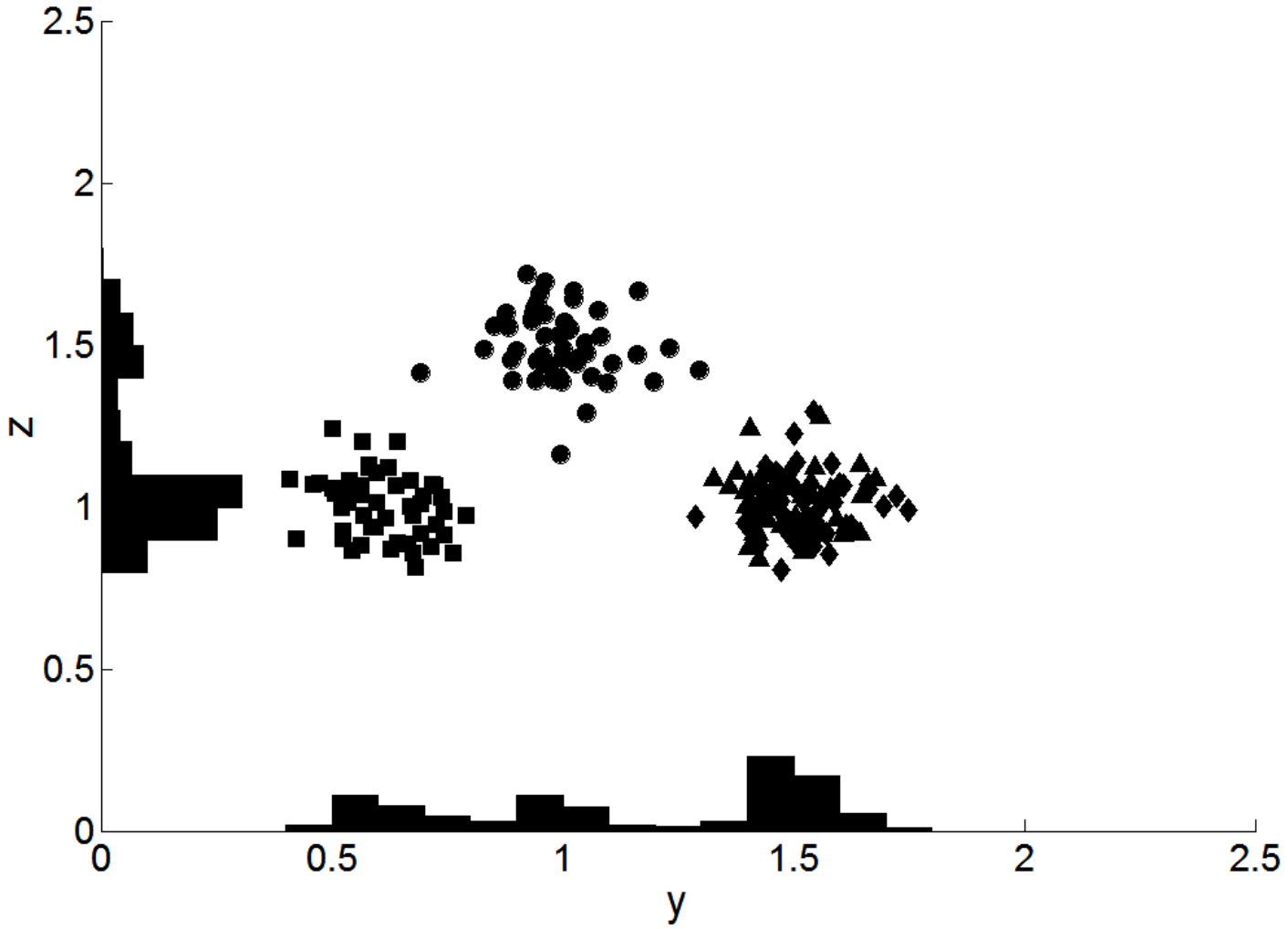
Example



Example



Example



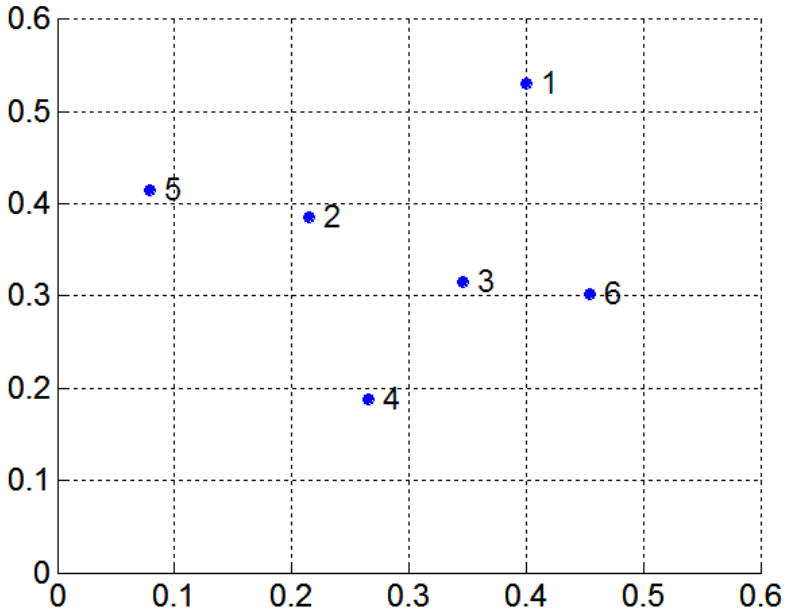
Graph-Based Clustering: General Concepts

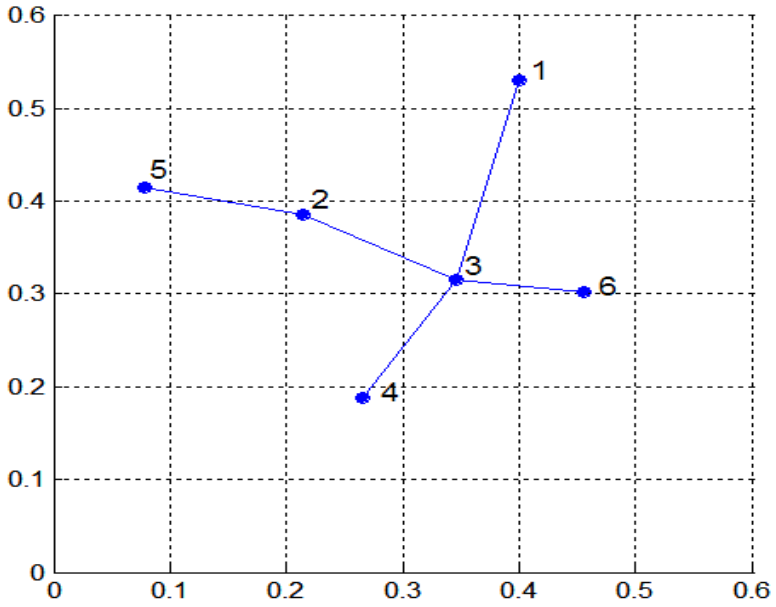
- Graph-Based clustering uses the proximity graph
  - Start with the proximity matrix
  - Consider each point as a node in a graph
  - Each edge between two nodes has a weight which is the proximity between the two points
  - Initially the proximity graph is fully connected
  - MIN (single-link) and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

- In the simplest case, clusters are connected components in the graph.

MST: Divisive 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





MST: Divisive Hierarchical Clustering

- Use MST for constructing hierarchy of clusters

Algorithm 7.5 *MST divisive hierarchical clustering algorithm,*

---

1. *Compute a minimum spanning tree for the dissimilarity graph.*
2. ***repeat***
    3. *Create a new cluster by breaking the link corresponding to the largest distance.*
4. ***until*** *Only singleton clusters remain.*

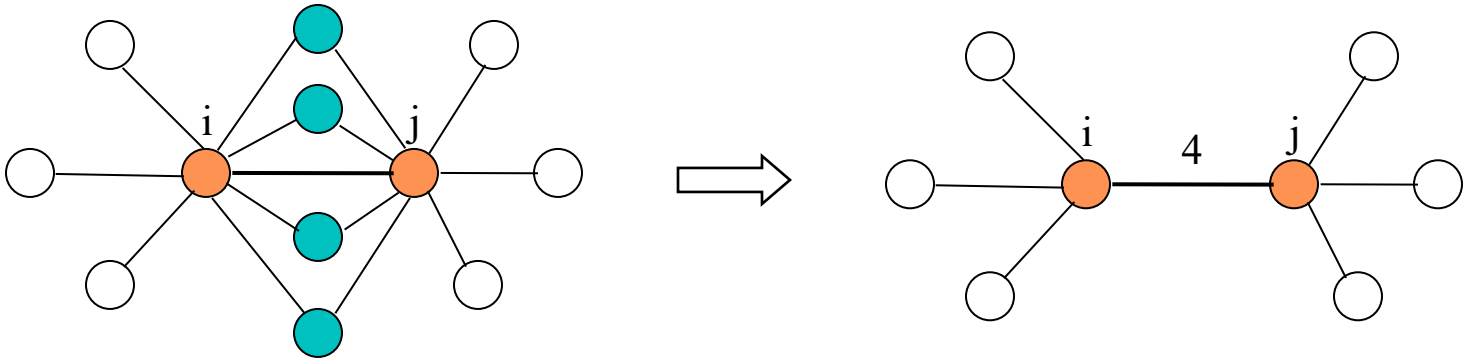
Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ 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

Graph-Based Clustering: SNN Approach

Shared Nearest Neighbor (SNN) graph: the weight of an edge is the number of shared neighbors between vertices given that the vertices are connected



Jarvis-Patrick Clustering

- **Step 1: SNN sparsification:**
  - construct an SSN Graph from data matrix as follows

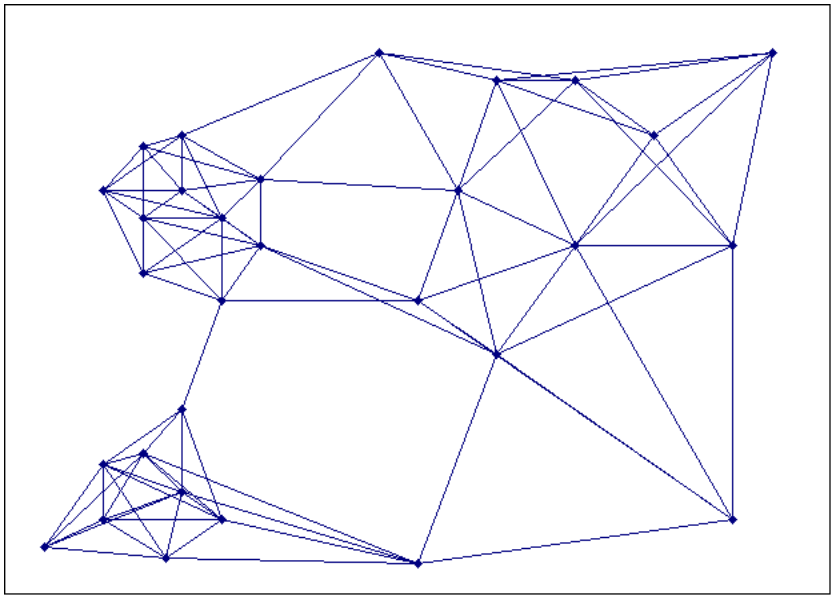
    if p and q have each others in the KNN list then

    create a link between them
- **Step 2: Weighting**
  - weight the links with $sim(p,q) = |NN(p) \cap NN(q)|$

    where NN(p) and NN(q) are k neighbors of p and q resp.
- **Step 3: Filtering**
  - then filter the edges:

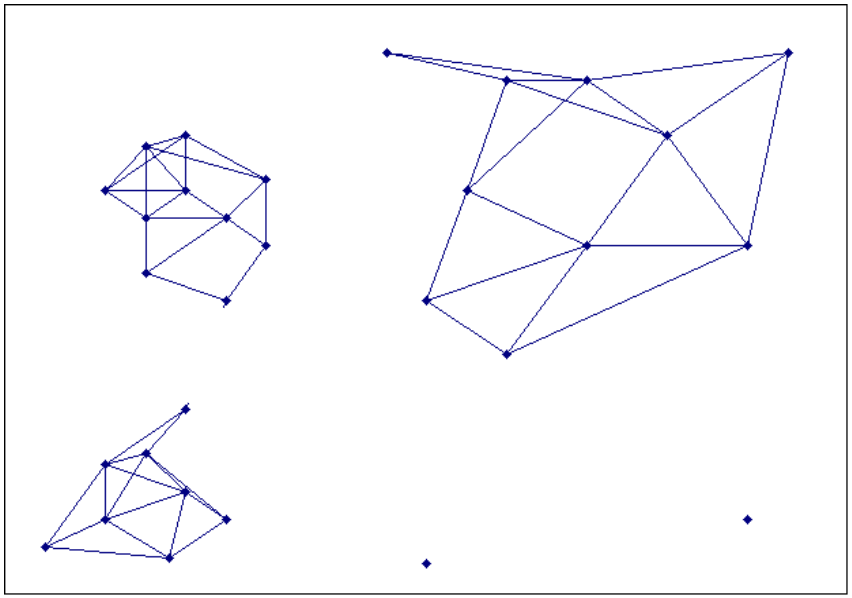
    remove all edges with weight less than some threshold
- **Step 4: Clusters**
  - let all connected compoents be clusters

Creating the SNN Graph



Sparse Graph

Link weights are similarities between neighboring points

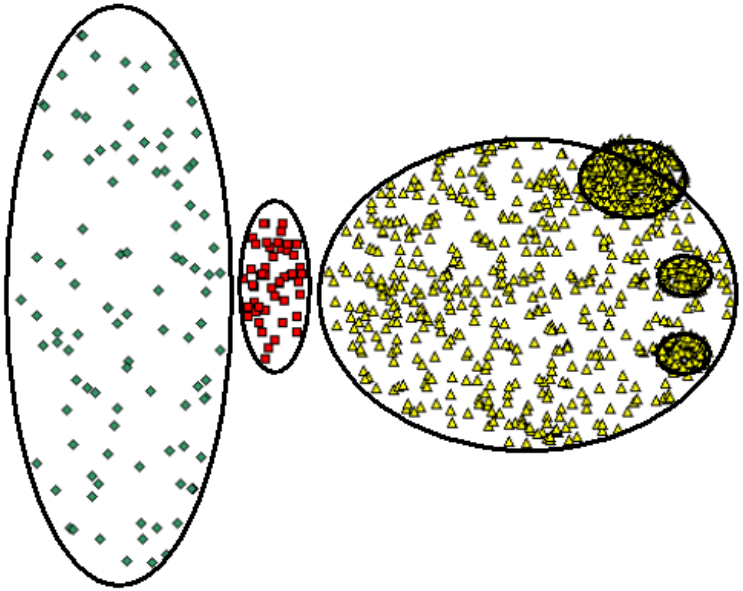


Shared Near Neighbor Graph

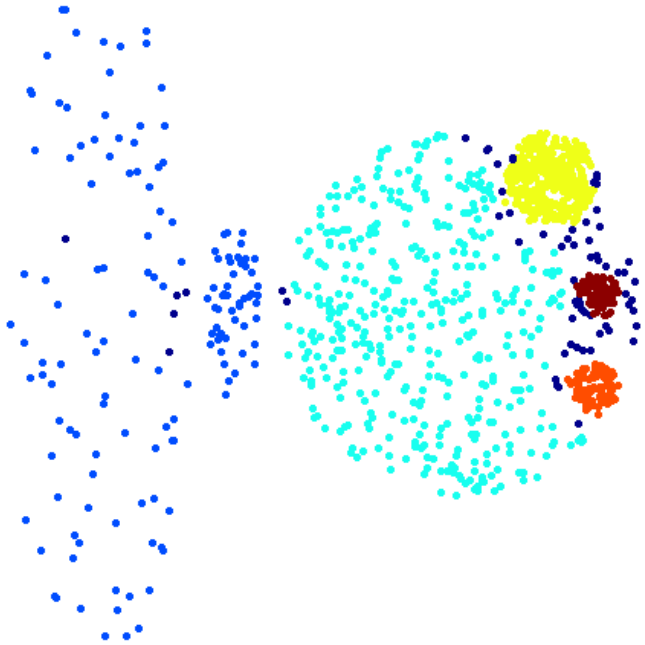
Link weights are number of Shared Nearest Neighbors

- Jarvis-Patrick clustering is too brittle

When Jarvis-Patrick Works Reasonably Well



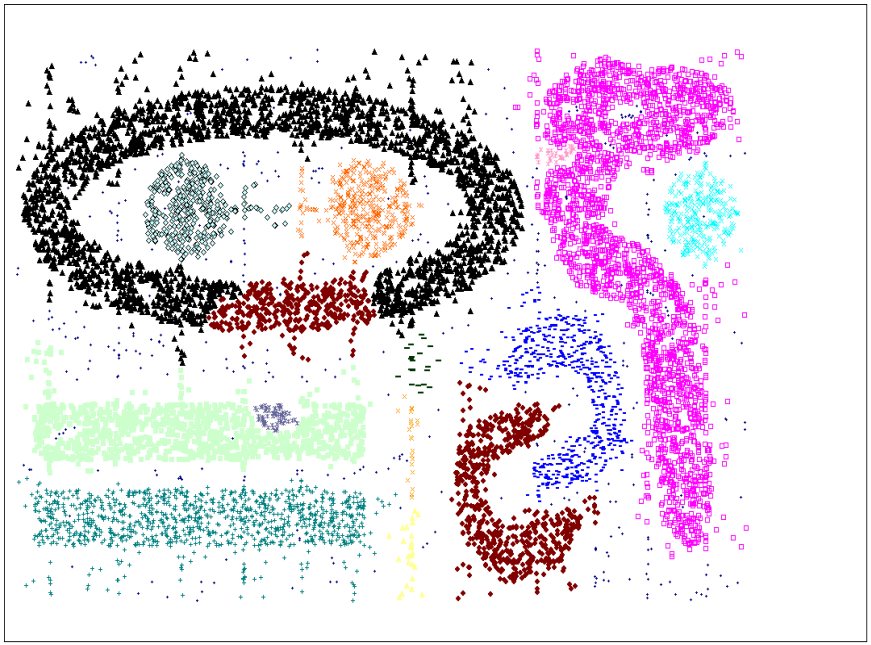
Original Points



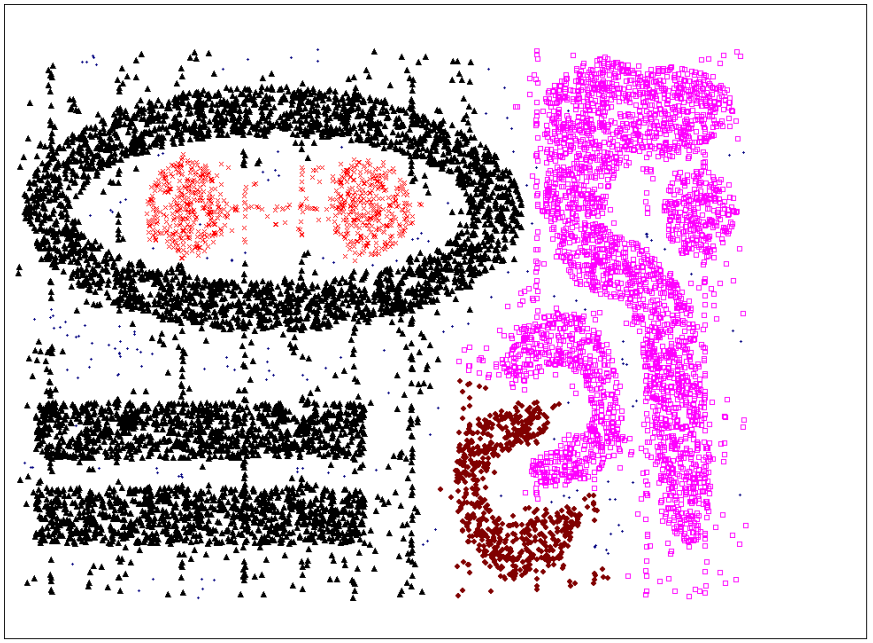
Jarvis Patrick Clustering

6 shared neighbors out of 20

When Jarvis-Patrick Does NOT Work Well



Smallest threshold, T, that does not merge clusters.



Threshold of T - 1

SNN Density-Based Clustering

- Combines:
  - Graph based clustering (similarity definition based on number of shared nearest neighbors)
  - Density based clustering (DBSCAN-like approach)

- SNN density measures whether a point is surrounded by similar points (with respect to its nearest neighbors)

SNN Clustering Algorithm

1. **Compute the similarity matrix**
   This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points

2. **Sparsify the similarity matrix by keeping only the *k* most similar neighbors**
   This corresponds to only keeping the *k* strongest links of the similarity graph

3. **Construct the shared nearest neighbor graph from the sparsified similarity matrix.**
   At this point, we could apply a similarity threshold and find the connected components to obtain the clusters (Jarvis-Patrick algorithm)

4. **Find the SNN density of each Point.**
   Using a user specified parameters, *Eps*, find the number points that have an SNN similarity of *Eps* or greater to each point. This is the SNN density of the point

SNN Clustering Algorithm …

5. **Find the core points**
   Using a user specified parameter, *MinPts*, find the core points, i.e., all points that have an SNN density greater than *MinPts*

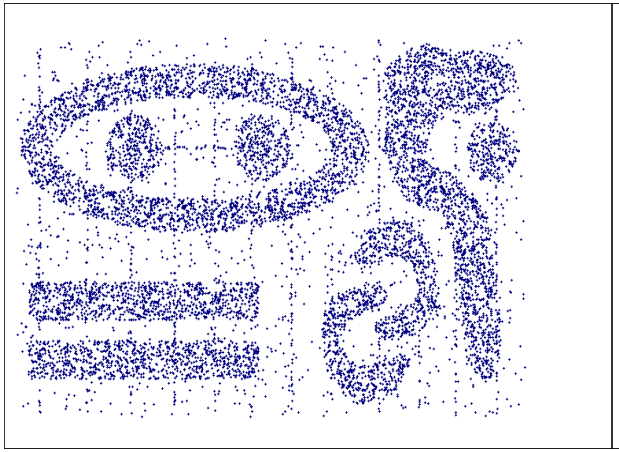
6. **Form clusters from the core points**
   If two core points are within a "radius", *Eps*, of each other they are placed in the same cluster

7. **Discard all noise points**
   All non-core points that are not within a "radius" of *Eps* of a core point are discarded

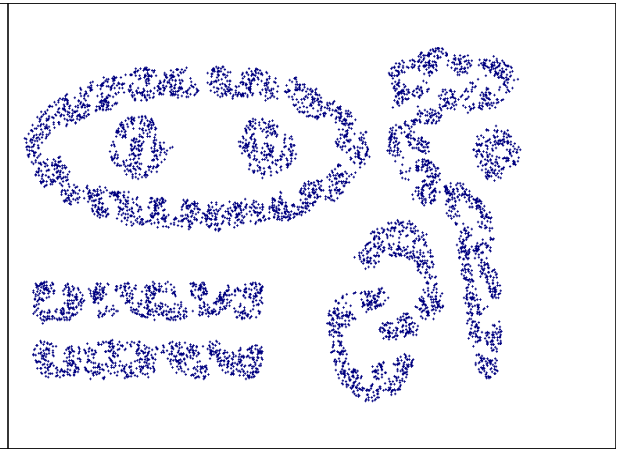
8. **Assign all non-noise, non-core points to clusters**
   This can be done by assigning such points to the nearest core point

   (Note that steps 4-8 are DBSCAN)

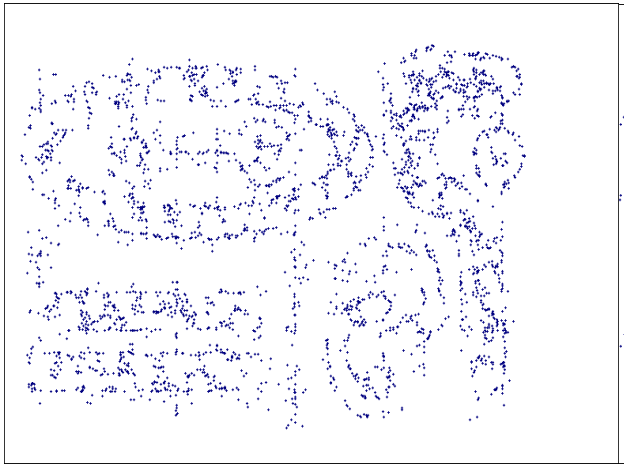
SNN Density



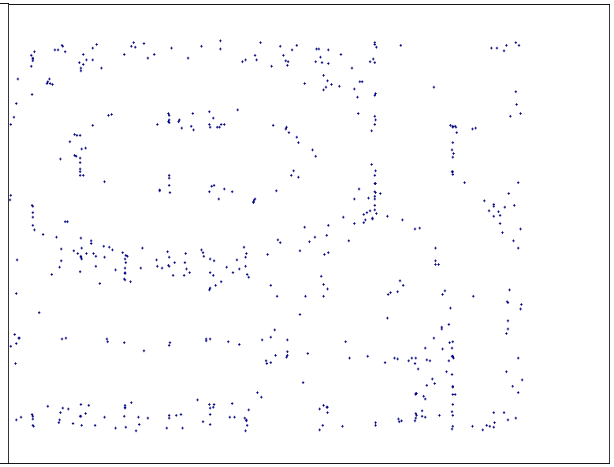
a) All Points



b) High SNN Density (core)

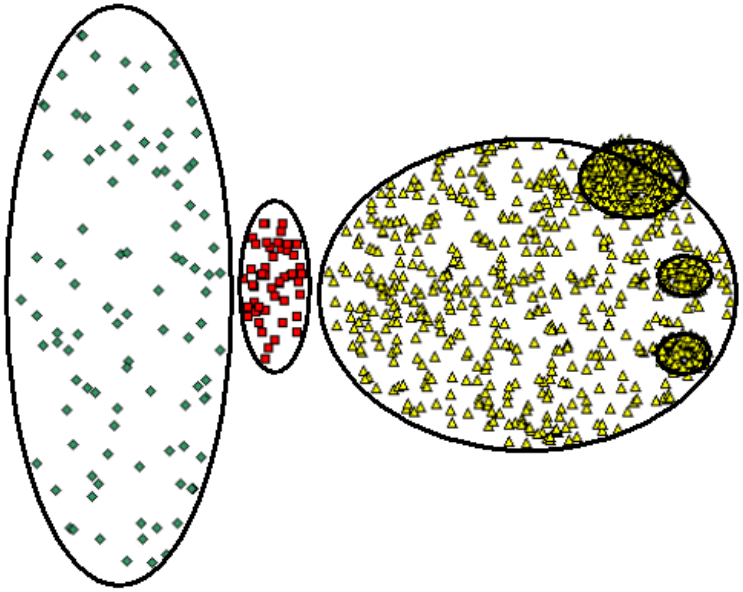


c) Medium SNN Density)border)

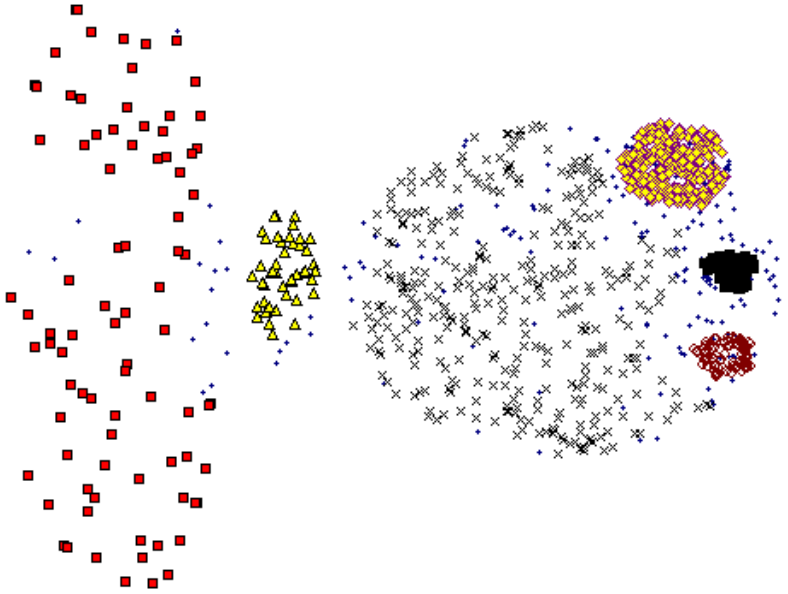


d) Low SNN Density (noise)

SNN Clustering Can Handle Differing Densities

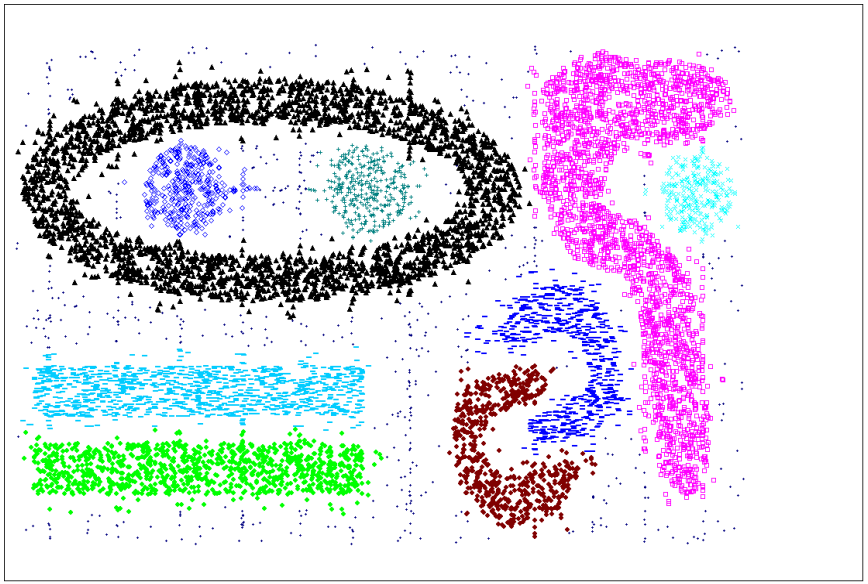


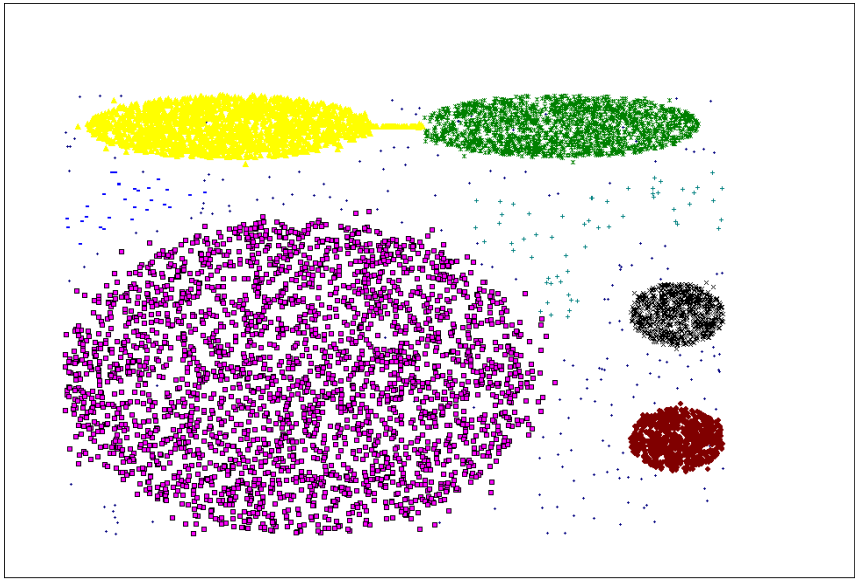
Original Points



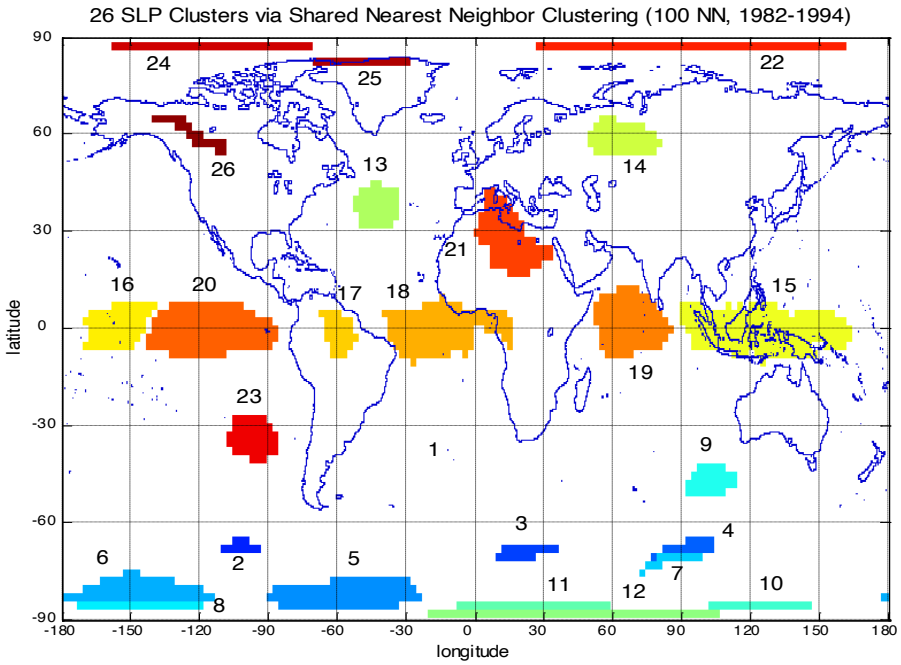
SNN Clustering

SNN Clustering Can Handle Other Difficult Situations

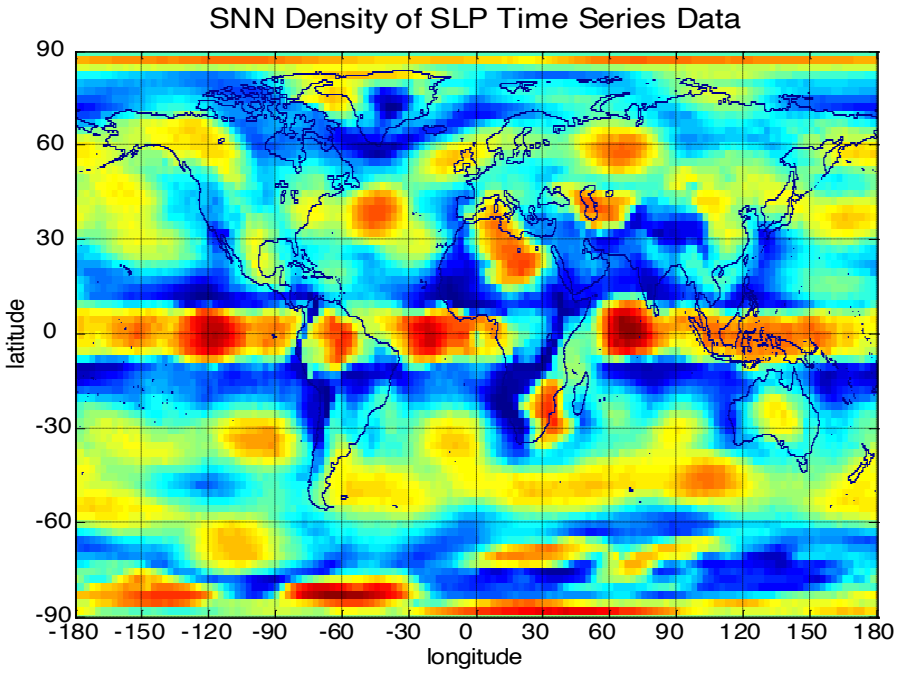




Finding Clusters of Time Series In Spatio-Temporal Data



SNN Clusters of SLP.



SNN Density of Points on the Globe.

Limitations of SNN Clustering

- Does not cluster all the points

- Complexity of SNN Clustering is high
  - O(n * time to find numbers of neighbor within *Eps*)
  - In worst case, this is $O(n^2)$
  - For lower dimensions, there are more efficient ways to find the nearest neighbors
    - R* Tree
    - k-d Trees

Characteristics of Data, Clusters, and Algorithms

- A cluster analysis is affected by characteristics of
  - Data
  - Clusters
  - Clustering algorithms

- Looking at these characteristics gives us a number of dimensions that you can use to describe clustering algorithms and the results that they produce

Characteristics of Data

- High dimensionality
- Size of data set
- Sparsity of attribute values
- Noise and Outliers
- Types of attributes and type of data sets
- Differences in attribute scales
- Properties of the data space
  - Can you define a meaningful centroid

Characteristics of Clusters

- Data distribution
- Shape
- Differing sizes
- Differing densities
- Poor separation
- Relationship of clusters
- Types of clusters
  - Center-based, contiguity-based, density-based

Characteristics of Clustering Algorithms

- Order dependence
- Non-determinism
- Parameter selection
- Scalability
- Underlying model
- Optimization based approach

Comparison of DBSCAN and K-means

- Both are partitional.
- K-means is complete; DBSCAN is not.
- K-means has a prototype-based notion of a cluster; DB uses a density-based notion.
- K-means can find clusters that are not well-separated. DBSCAN will merge clusters that touch.
- DBSCAN handles clusters of different shapes and sizes; K-means prefers globular clusters.

Comparison of DBSCAN and K-means

- DBSCAN can handle noise and outliers; K-means performs poorly in the presence of outliers
- K-means can only be applied to data for which a centroid is meaningful; DBSCAN requires a meaningful definition of density
- DBSCAN works poorly on high-dimensional data; K-means works well for some types of high-dimensional data
- Both techniques were designed for Euclidean data, but extended to other types of data
- DBSCAN makes no distribution assumptions; K-means is really assuming spherical Gaussian distributions

Comparison of DBSCAN and K-means

- K-means has an O(n) time complexity; DBSCAN is O(n^2)
- Because of random initialization, the clusters found by K-means can vary from one run to another; DBSCAN always produces the same clusters
- DBSCAN automatically determines the number of clusters; K-means does not
- K-means has only one parameter, DBSCAN has two.
- K-means clustering can be viewed as an optimization problem and as a special case of EM clustering; DBSCAN is not based on a formal model.