Data Mining
Cluster Analysis: Basic Concepts and Algorithms

Lecture Notes for Chapter 7

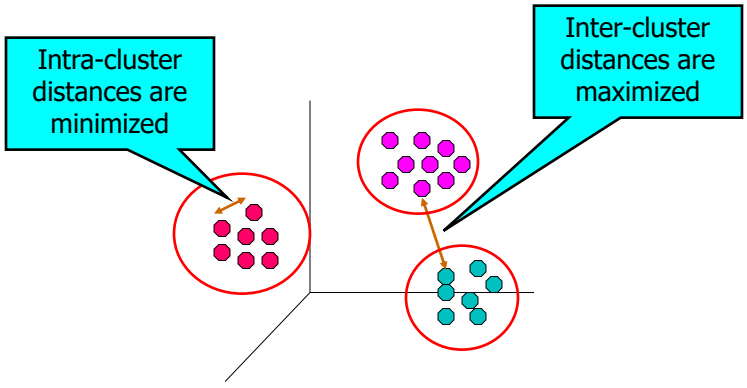
Introduction to Data Mining

by

Tan, Steinbach, Kumar

What is Cluster Analysis?

- Given a set of objects, place them in groups such that the objects in a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups



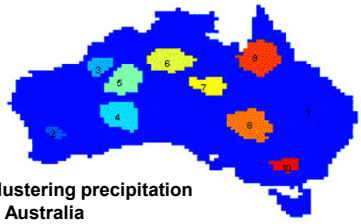
Applications of Cluster Analysis

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

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

- **Summarization**
  - Reduce the size of large data sets

cluster: words/sentences/paragraphs that relate to one another



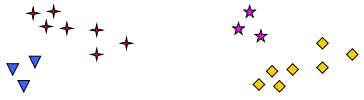
Clustering precipitation in Australia

Notion of a Cluster can be Ambiguous

How many clusters?

Six Clusters

Two Clusters

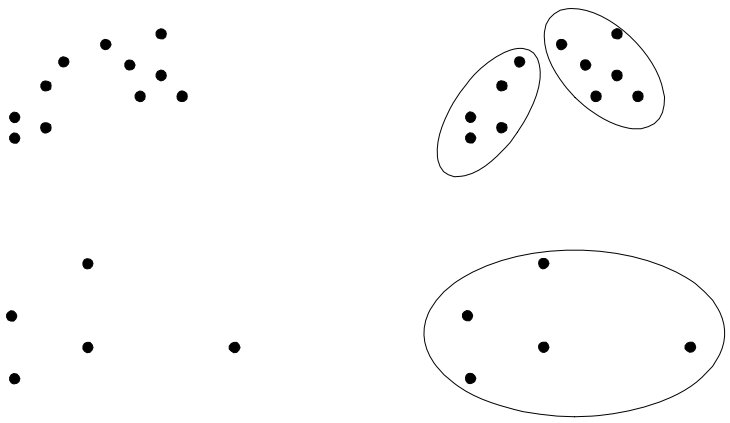


Four Clusters

Types of Clusterings

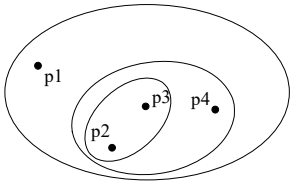
- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
  - Partitional Clustering
    - A division of data objects into non-overlapping subsets (clusters)
  - Hierarchical clustering
    - A set of nested clusters organized as a hierarchical tree

Partitional Clustering

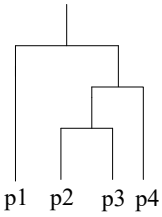


Original Points A Partitional Clustering

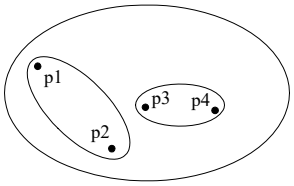
Hierarchical Clustering



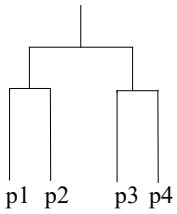
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Other Distinctions Between Sets of Clusters

- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
    - Can belong to multiple classes or could be 'border' points

points

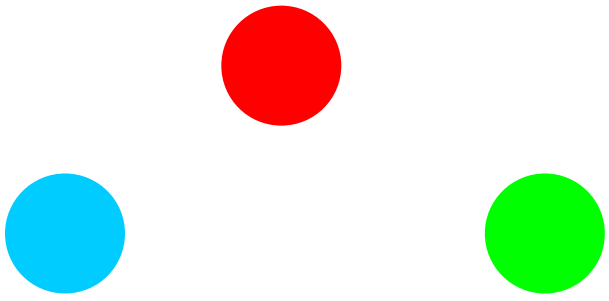
  - Fuzzy clustering (one type of non-exclusive)
    - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
    - Weights must sum to 1 (weights that corresponds to a given point)
    - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data

Types of Clusters

- Well-separated clusters
- Prototype-based clusters
- Contiguity-based clusters
- Density-based clusters
- 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: Prototype-Based

- Prototype-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the prototype or "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

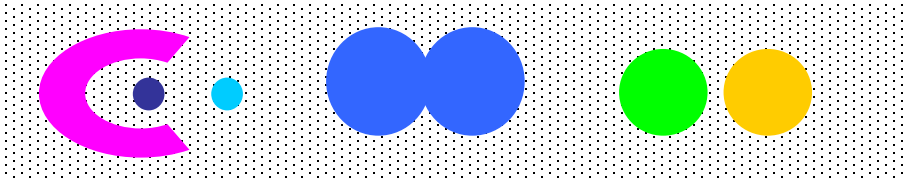
Types of Clusters: Contiguity-Based

- 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: 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.

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

Need more info

- Noise and Outliers
  - Often interfere with the operation of the clustering algorithm
- Clusters of differing sizes, densities, and shapes

Clustering Algorithms

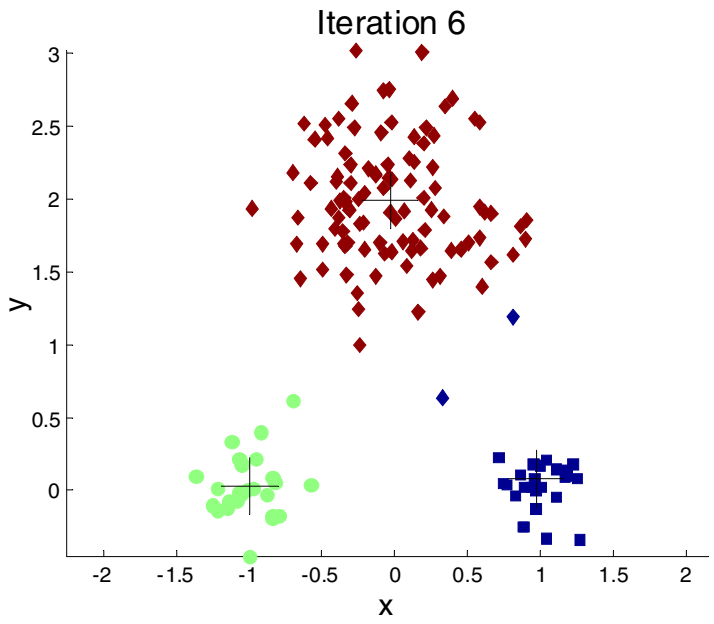
- K-means and its variants
- Hierarchical clustering
- Density-based clustering

K-means Clustering

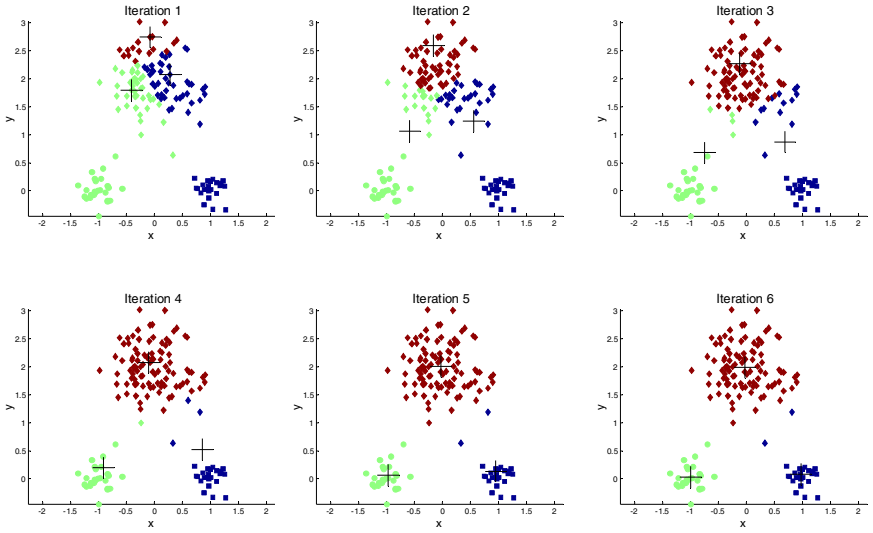
- Partitional clustering approach
- Number of clusters, K, must be specified
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- The basic algorithm is very simple

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

- Simple iterative algorithm.
  - Choose initial centroids;
  - repeat {assign each point to a nearest centroid; re-compute cluster centroids}
  - until centroids stop changing.
- Initial centroids are often chosen randomly.
  - Clusters produced can vary from one run to another
- The centroid is (typically) the mean of the points in the cluster, but other definitions are possible (see Table 7.2).
- K-means will converge for common proximity measures with appropriately defined centroid (see Table 7.2)
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

K-means Objective Function

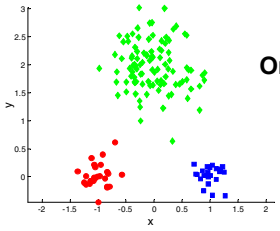
- A common objective function (used with Euclidean distance measure) is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster center
  - To get SSE, we square these errors and sum them.

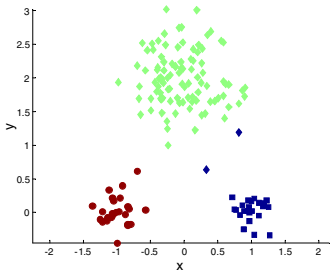
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

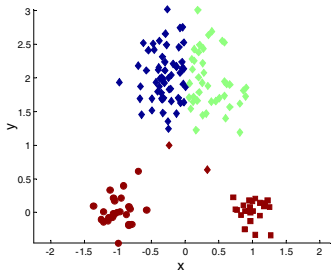
  - x is a data point in cluster C_i and m_i is the centroid (mean) for cluster C_i
  - SSE improves in each iteration of K-means until it reaches a local or global minima.

Two different K-means Clusterings

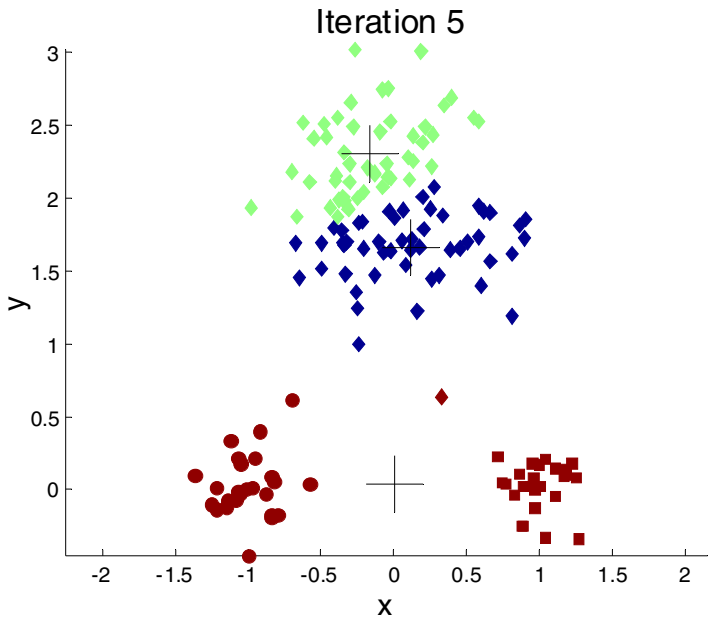
Depends on the initial choice of centroids



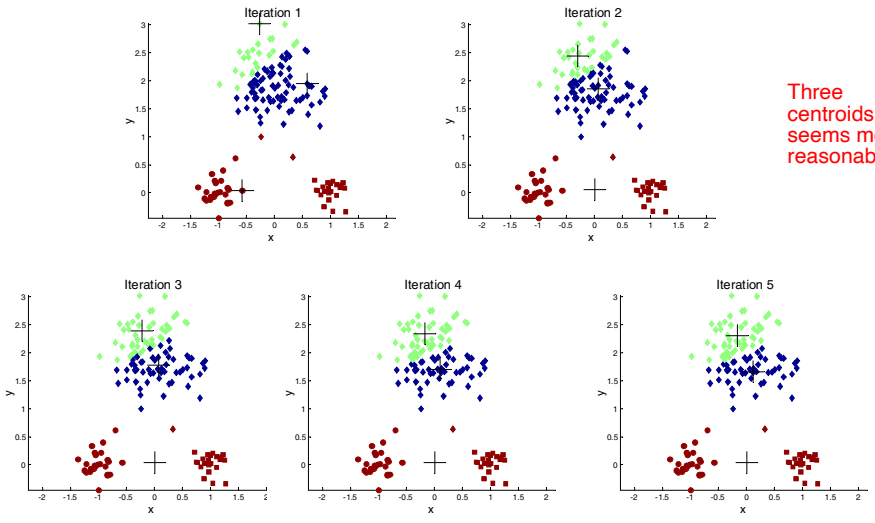




Importance of Choosing Initial Centroids …



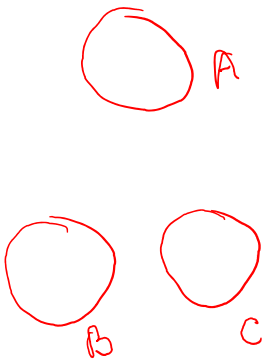
Importance of Choosing Initial Centroids …



Three centroids seems more reasonable.

Importance of Choosing Intial Centroids

- Depending on the choice of initial centroids, B and C may get merged or remain separate



Problems with Selecting Initial Points

K=2 => P = 1/2

K=3 => P = 6/27 = 2/4.5

K=5 => P = 120/5^5 = 0.04

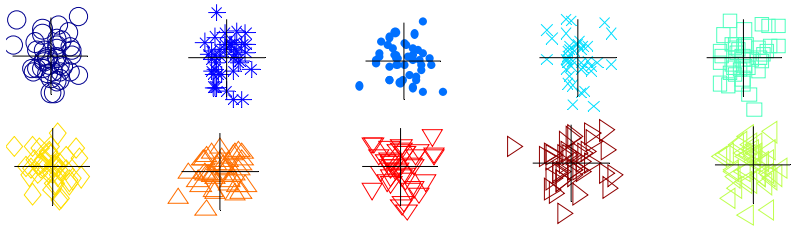
K=7 => P = 0.006

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - If clusters are the same size, n, then

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

  - For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
  - Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
  - Consider an example of five pairs of clusters

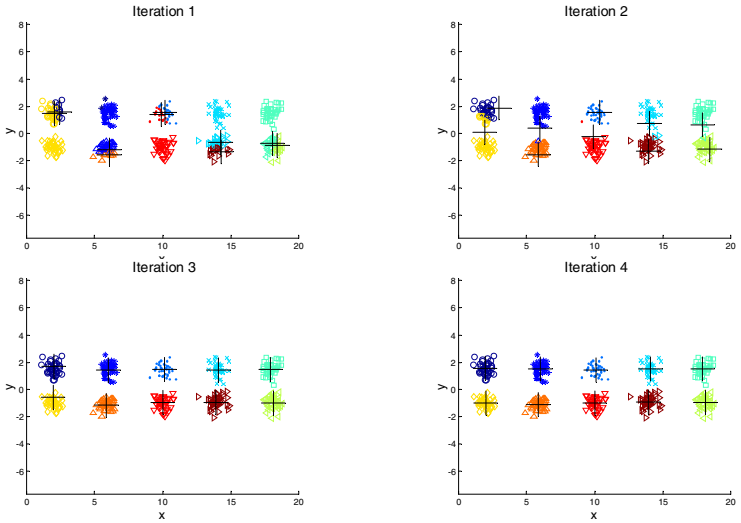
10 Clusters Example



Works every time

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

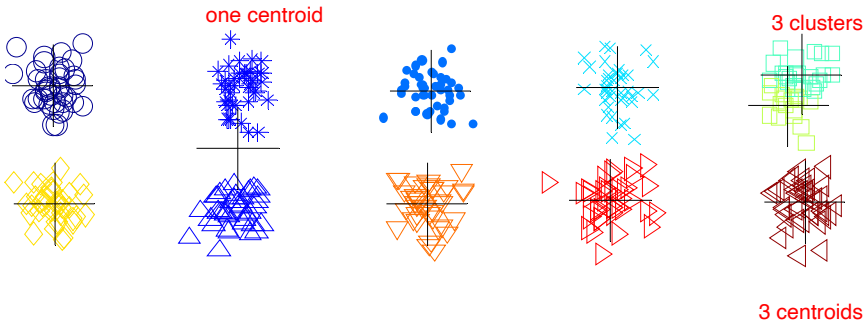
10 Clusters Example



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

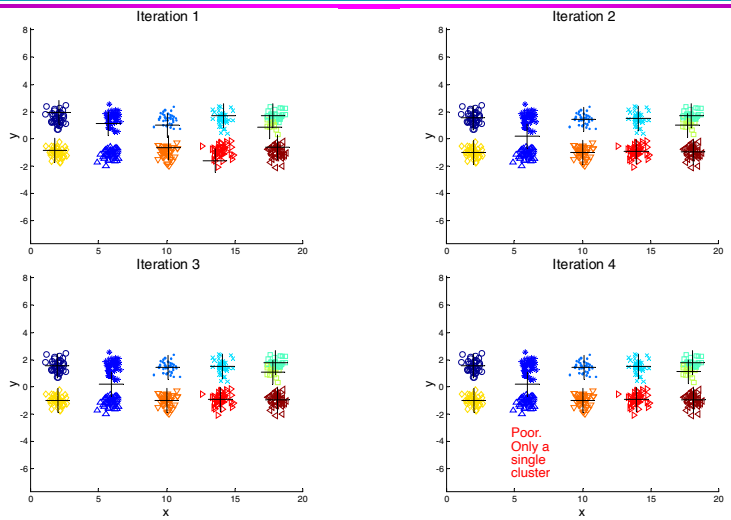
Worked

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.

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Use some strategy to select the k initial centroids and then select among these initial centroids
  - Select most widely separated
    - K-means++ is a robust way of doing this selection
  - Use hierarchical clustering to determine initial centroids
- 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. Select an initial point at random to be the first centroid
2. For k – 1 steps
3. For each of the N points, x_i, $1 \leq i \leq N$, find the minimum squared distance to the currently selected centroids, $C_1, \ldots, C_j$, $1 \leq j < k$, i.e., $\min_j d^2(C_j, x_i)$
4. Randomly select a new centroid by choosing a point with probability proportional to $\frac{\min_j d^2(C_j, x_i)}{\sum_i \min_j d^2(C_j, x_i)}$ is Likely to select a point that is maximally distant from all the centroids.
5. End For

You

Explain the K-means++ algorithm.

Data Mining Guide

The K-means++ algorithm is an improvement over the standard K-means clustering algorithm, designed to choose initial cluster centers in a more intelligent and representative manner. Here's how it works:

1. **Initialization**: Select one center uniformly at random from the data points.
2. **Choose Centers**: For each data point, compute its distance (D(x)) to the nearest, previously chosen center.
3. **Probability Distribution**: Assign a probability to each data point that is proportional to the square of the distance (D(x))^2.
4. **Select New Center**: Randomly choose a new center from the data points using the probability distribution calculated in the previous step.
5. **Repeat**: Repeat steps 2-4 until K centers have been chosen.

The idea behind K-means++ is that by selecting initial centers that are far apart from each other, the algorithm is less likely to converge to a suboptimal solution. This initialization strategy tends to yield better and more stable clustering results compared to random initialization in standard K-means.

Once the initial centers are chosen using K-means++, the standard K-means algorithm proceeds as follows:

1. Assign each data point to the nearest center.
2. Recompute the center of each cluster based on the mean of the data points assigned to it.
3. Repeat steps 1 and 2 until convergence, i.e., until the centers no longer change significantly or the maximum number of iterations is reached.

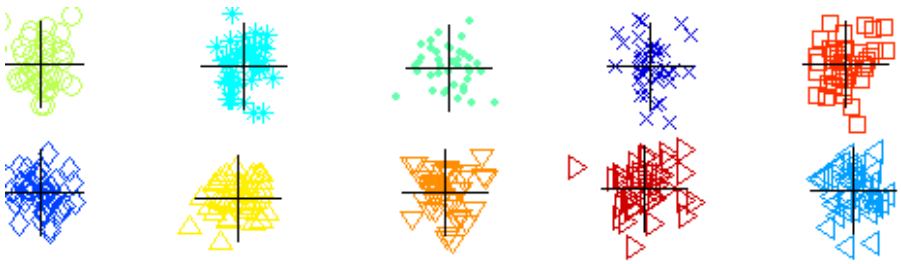
Bisecting K-means

- 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



You

Explain the bissecting K-means algorithm.

‹ 2 / 2 ›

Data Mining Guide

The Bisecting K-means algorithm is a clustering algorithm that recursively partitions a dataset into K clusters by bisecting the cluster with the largest sum of squared distances. Here's how it works:

1. **Initialization**: Start with all data points in a single cluster.
2. **Select Cluster to Split**: Choose the cluster with the largest sum of squared distances (i.e., the cluster that is "most spread out").
3. **Bisect Cluster**: Split the selected cluster into two subclusters using regular K-means clustering. This involves:
   - Running the standard K-means algorithm with K=2 on the selected cluster's data points.
   - Assigning each data point to the nearest center of the two centers obtained from K-means.
4. **Repeat**: Repeat steps 2 and 3 until the desired number of clusters (K) is reached.

The Bisecting K-means algorithm differs from standard K-means in that it doesn't perform multiple initializations with random seeds. Instead, it iteratively bisects clusters until the desired number of clusters is obtained. This approach can sometimes produce better clustering results, especially when dealing with datasets with complex cluster structures or varying cluster densities.

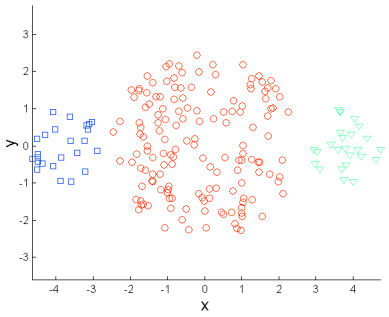
After the algorithm completes, each data point will belong to one of the K clusters, and the centers of these clusters will represent the final centroids.

Limitations of K-means

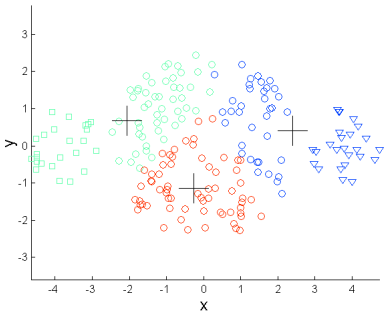
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has problems when the data contains outliers.
  - One possible solution is to remove outliers before clustering

Limitations of K-means: Differing Sizes

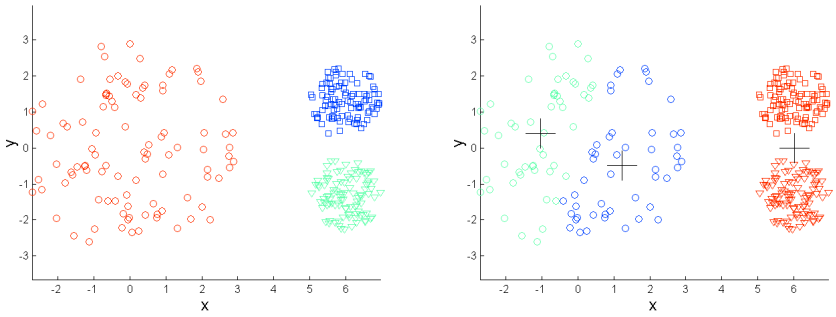


Original Points



K-means (3 Clusters)

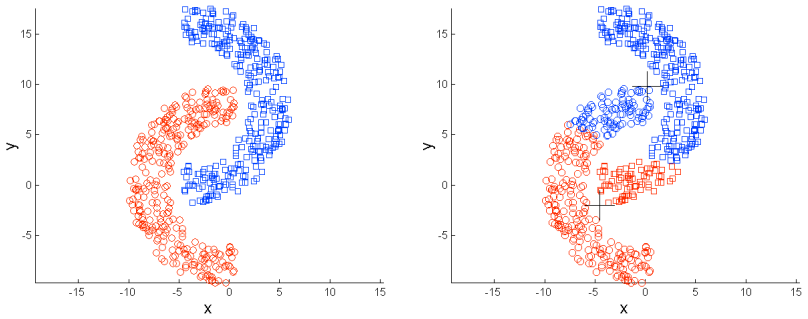
Limitations of K-means: Differing Density



Original Points

K-means (3 Clusters)

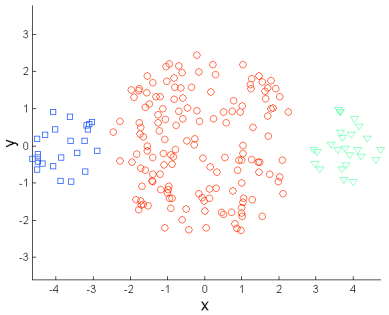
Limitations of K-means: Non-globular Shapes



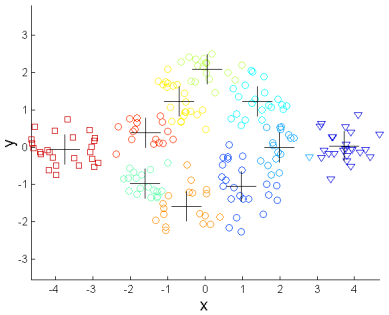
Original Points

K-means (2 Clusters)

Overcoming K-means Limitations



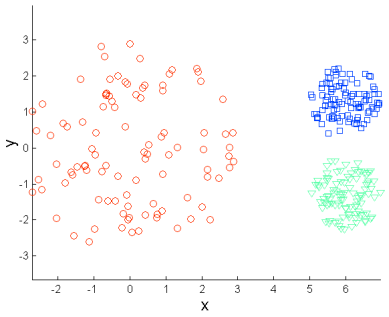
Original Points



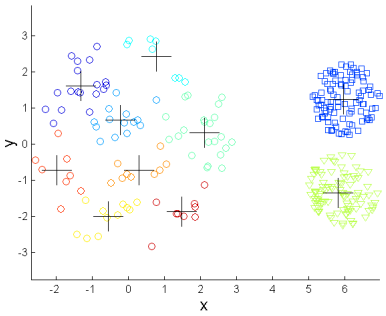
K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

Overcoming K-means Limitations



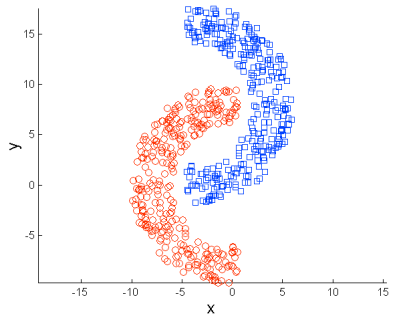
Original Points



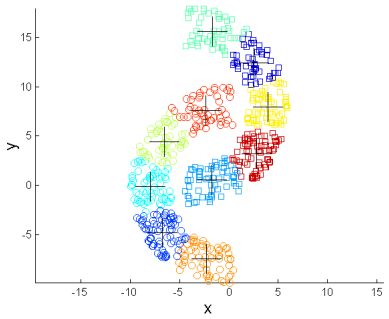
K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

Overcoming K-means Limitations



Original Points

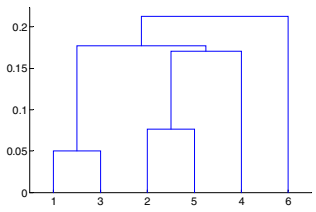


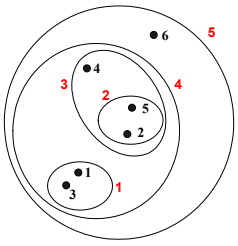
K-means Clusters

One solution is to find a large number of clusters such that each of them represents a part of a natural cluster. But these small clusters need to be put together in a post-processing step.

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

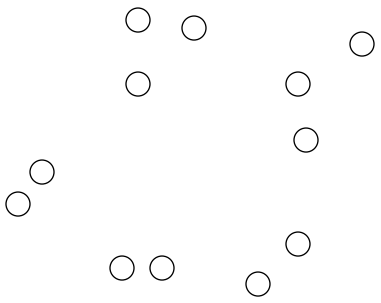
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster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

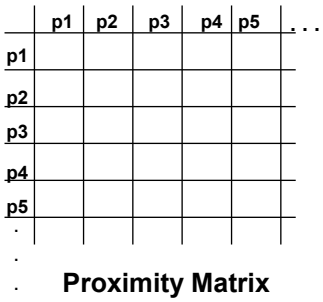
Agglomerative Clustering Algorithm

- **Key Idea: Successively merge closest clusters**
- Basic algorithm
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Steps 1 and 2

- Start with clusters of individual points and a proximity matrix

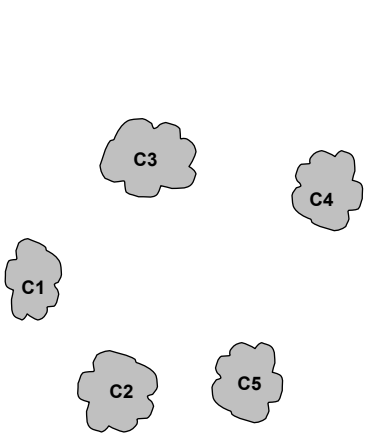


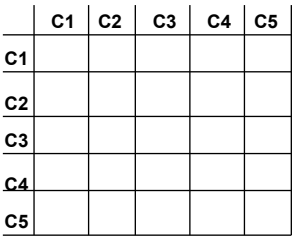


Proximity Matrix

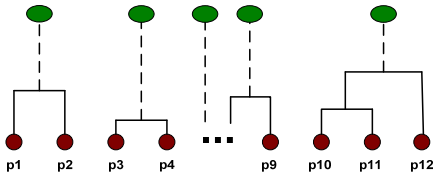
Intermediate Situation

- After some merging steps, we have some clusters



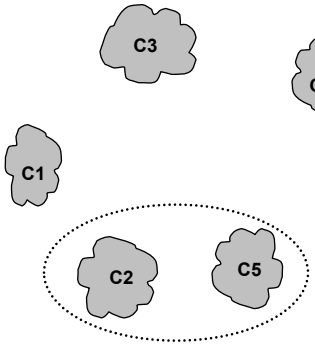


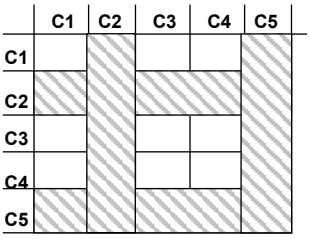
Proximity Matrix



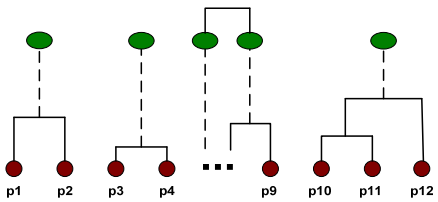
Step 4

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



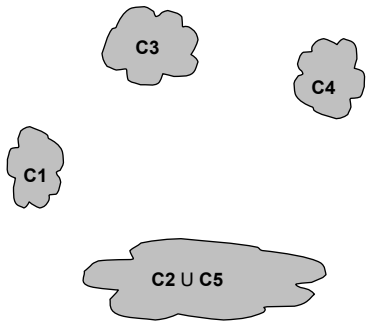


Proximity Matrix



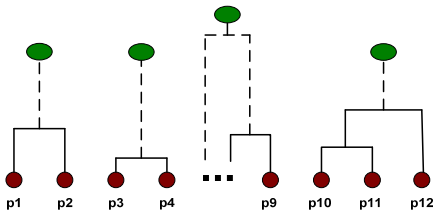
Step 5

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

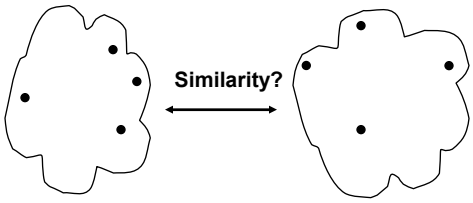


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Distance

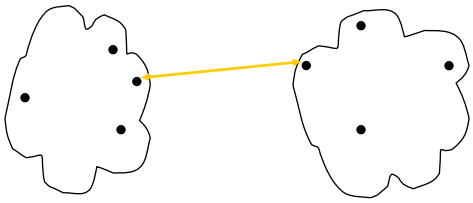


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

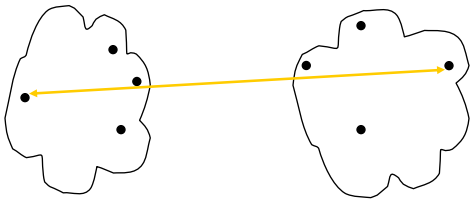


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

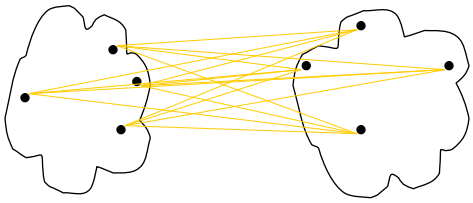


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

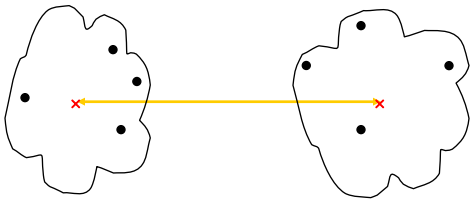


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

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

You

What are the different methods to compute inter-cluster distance, as used in hierarchical clustering?

Data Mining Guide

In hierarchical clustering, inter-cluster distance (or dissimilarity) measures are crucial for deciding which clusters to merge in agglomerative approaches or split in divisive approaches. The choice of distance measure can significantly affect the shape and characteristics of the resulting clusters. Here are the primary methods used to compute inter-cluster distance:

1. **Single Linkage (Nearest Point)**:
   - The distance between two clusters is defined as the shortest distance from any member of one cluster to any member of the other cluster.
   - It can result in "chaining" where clusters may be elongated and not compact.
2. **Complete Linkage (Farthest Point)**:
   - The distance between two clusters is defined as the maximum distance from any member of one cluster to any member of the other cluster.
   - This method tends to produce more compact and well-separated clusters compared to single linkage.
3. **Average Linkage (Average Distance)**:
   - The distance between two clusters is computed as the average distance between all pairs of points, where one point belongs to the first cluster and the other point belongs to the second cluster.
   - This method provides a balance between the single and complete linkage methods, avoiding some of their extremes.

4. **Centroid Method**:
   - The distance between two clusters is calculated based on the distance between their centroids (i.e., the mean point of each cluster).
   - Clusters are treated as single entities, and this method can potentially lead to inversions in the hierarchy (non-monotonic distances).
5. **Ward's Method**:
   - This method is different from the others in that it does not calculate distances between two clusters directly. Instead, it uses an analysis of variance approach to evaluate the distances between clusters. The criterion for selecting two clusters to merge is based on the minimum increase in total within-cluster variance after merging.
   - It tends to produce clusters of similar size and is particularly useful for identifying spherical clusters.

Each of these methods has its strengths and weaknesses, and the choice of method can be influenced by the nature of the data and the specific clustering objectives.

You

What about MIN and MAX?

Data Mining Guide

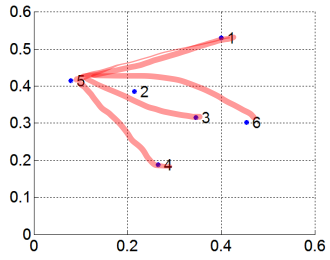
The terms "MIN" and "MAX" in the context of hierarchical clustering are essentially synonymous with the Single Linkage and Complete Linkage methods, respectively. These methods are used to compute inter-cluster distances and determine how clusters are merged in agglomerative hierarchical clustering or split in divisive hierarchical clustering. Here's a brief recap:

- **MIN (Single Linkage, Nearest Point)**:
  - In the Single Linkage method (also known as the MIN method), the distance between two clusters is defined as the minimum distance between any member of one cluster and any member of the other cluster. This means that the inter-cluster distance is determined by the closest pair of points, with one point in each cluster. Single Linkage is sensitive to outliers and can lead to "chaining," where clusters may end up being long and straggly rather than compact.
- **MAX (Complete Linkage, Farthest Point)**:
  - The Complete Linkage method (also referred to as the MAX method) calculates the distance between two clusters as the maximum distance between any member of one cluster to any member of the other cluster. This approach tends to produce more compact and well-separated clusters compared to Single Linkage, as it focuses on the farthest pair of points when considering the distance between clusters.

Both MIN and MAX are straightforward and widely used methods for measuring distances between clusters in hierarchical clustering. The choice between them (and other methods) should be guided by the specific characteristics of the dataset and the goals of the clustering exercise.

MIN or Single Link

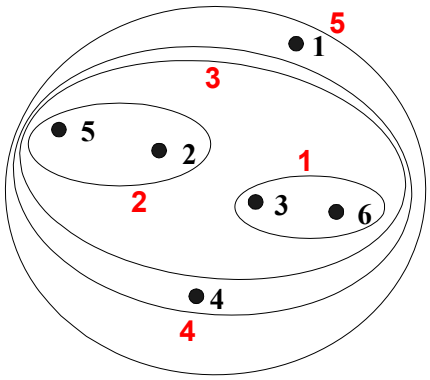
- Proximity of two clusters is based on the two closest points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph
- Example:

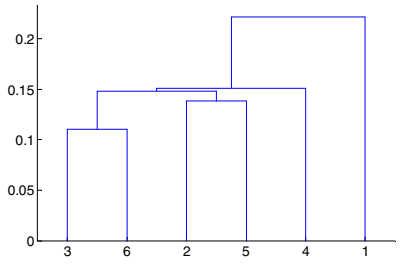


Distance Matrix: symmetric

	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

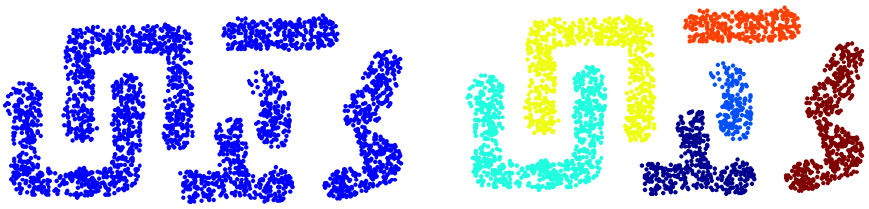
Hierarchical Clustering: MIN





Nested Clusters **Dendrogram**

Strength of MIN

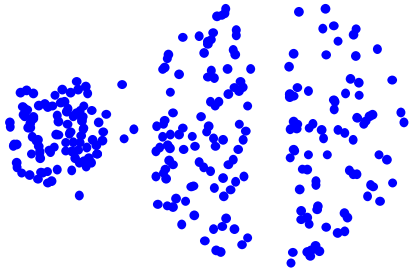


Original Points

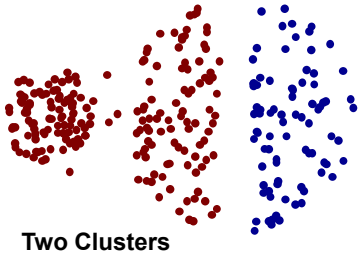
Six Clusters

- Can handle non-elliptical shapes

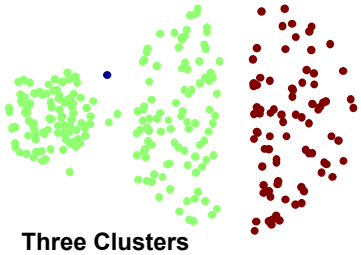
Limitations of MIN



Original Points



Two Clusters



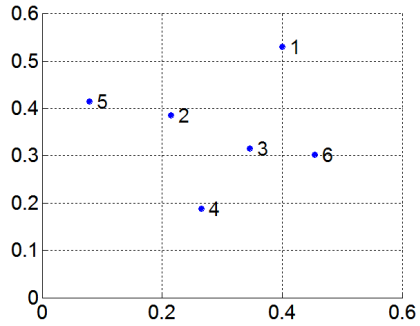
Three Clusters

- Sensitive to noise

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

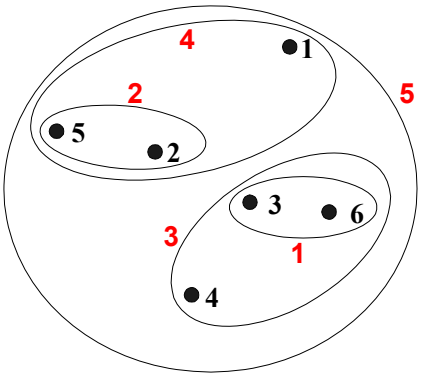
(so is MIN or single linkage)



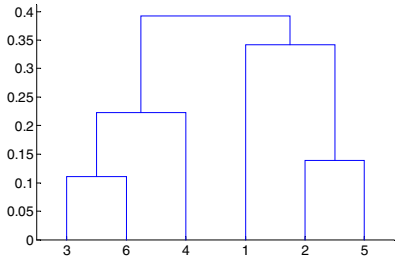
Distance Matrix:

	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

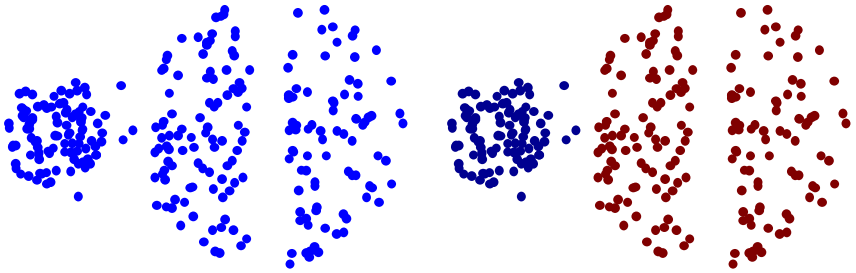


Nested Clusters



Dendrogram

Strength of MAX

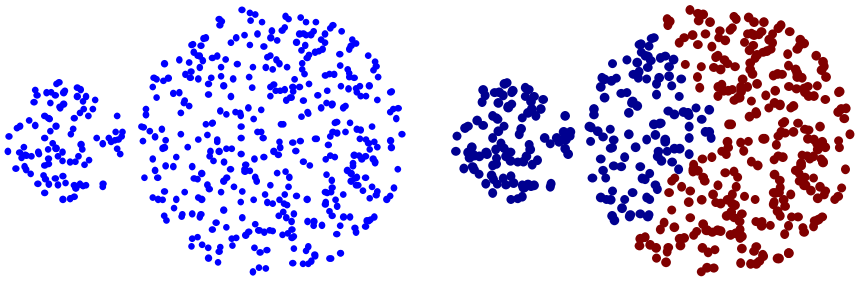


Original Points Two Clusters

- Less susceptible to noise

  Less susceptible to outliers (outliers are less likely to be taken into account when merging clusters)

Limitations of MAX



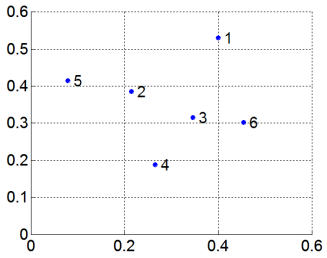
Original Points Two Clusters (when sizes are very different)

- 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.

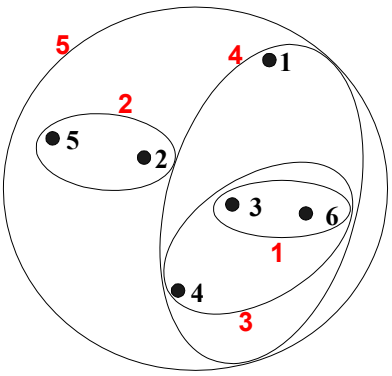
$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| \times |\text{Cluster}_j|}$$



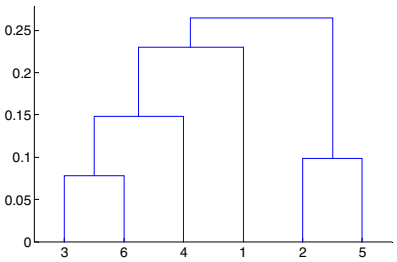
Distance Matrix:

	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
- Limitations
  - Biased towards globular clusters

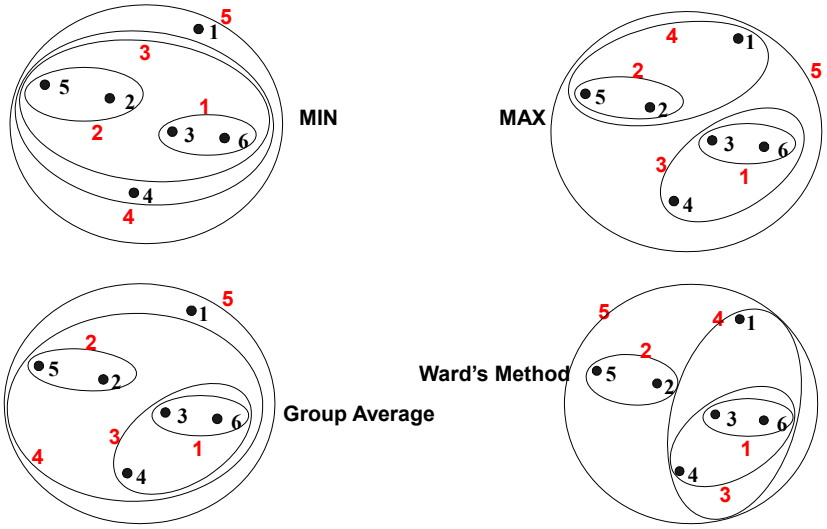
Cluster Similarity: Ward's Method

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

sum_I (x_i - x_c)^2

where x_c is the cluster centroid

Hierarchical Clustering: Comparison



Hierarchical Clustering: Time and Space requirements

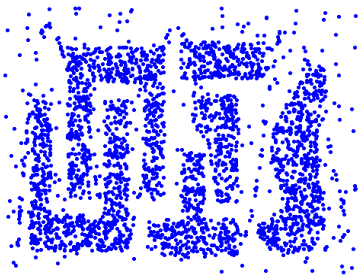
- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time with some cleverness

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

Density Based Clustering

- Clusters are regions of high density that are separated from one another by regions on low density.

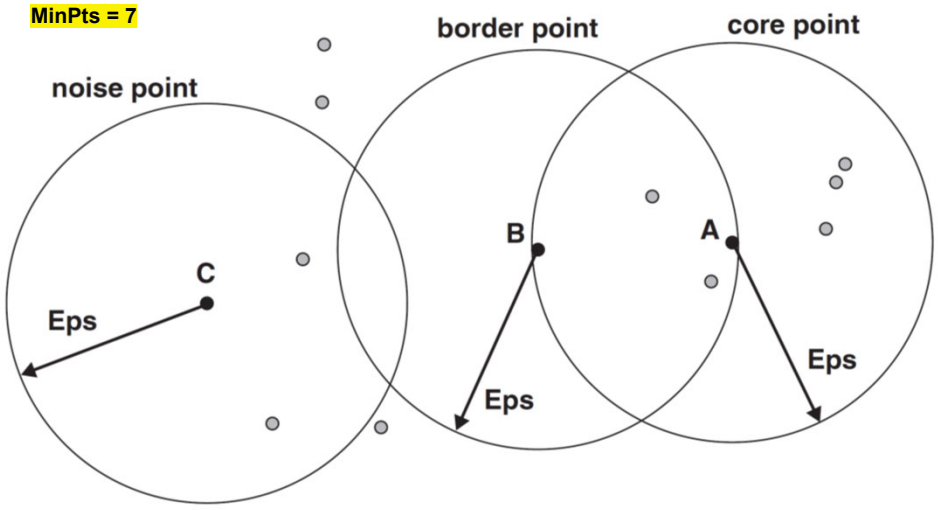


DBSCAN Density-Based Spatial Clustering of Applications with Noise)

- 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

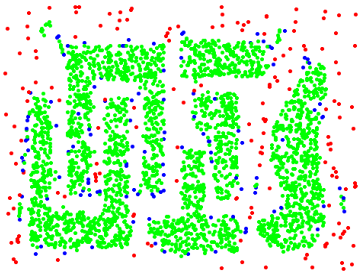
Different from density in physics, which is mass per unit volume.

DBSCAN: Core, Border, and Noise Points



DBSCAN: Core, Border and Noise Points

Original Points



Point types: core, border and noise

Eps = 10, MinPts = 4

DBSCAN Algorithm

- Form clusters using core points, and assign border points to one of its neighboring clusters

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

2: Eliminate noise points.

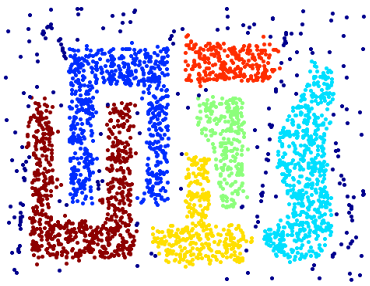
3: Put an edge between all core points within a distance *Eps* of each other.

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

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

When DBSCAN Works Well

Original Points

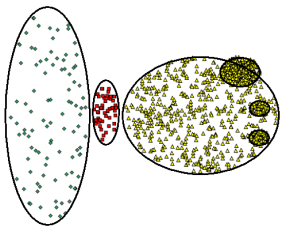


Clusters (dark blue points indicate noise)

- Can handle clusters of different shapes and sizes
- Resistant to noise

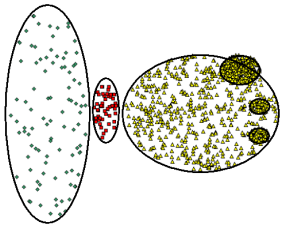
Number of clusters is not specified ahead of time.

When DBSCAN Does NOT Work Well

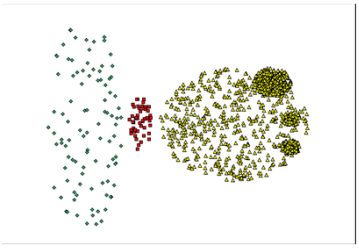


Original Points

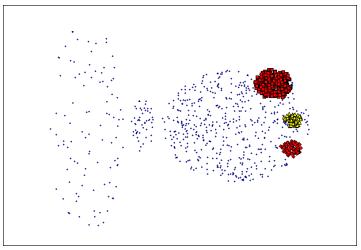
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92).

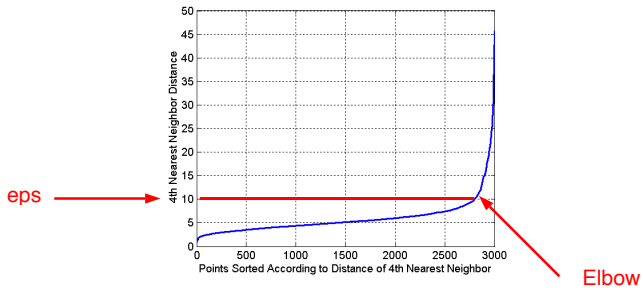


(MinPts=4, Eps=9.75)

- 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 close distance
- Noise points have the kth nearest neighbor at farther distance
- So, plot sorted distance of every point to its kth nearest neighbor



DBSCAN can be a very effective clustering algorithm, but there are certain conditions where it may not perform optimally:

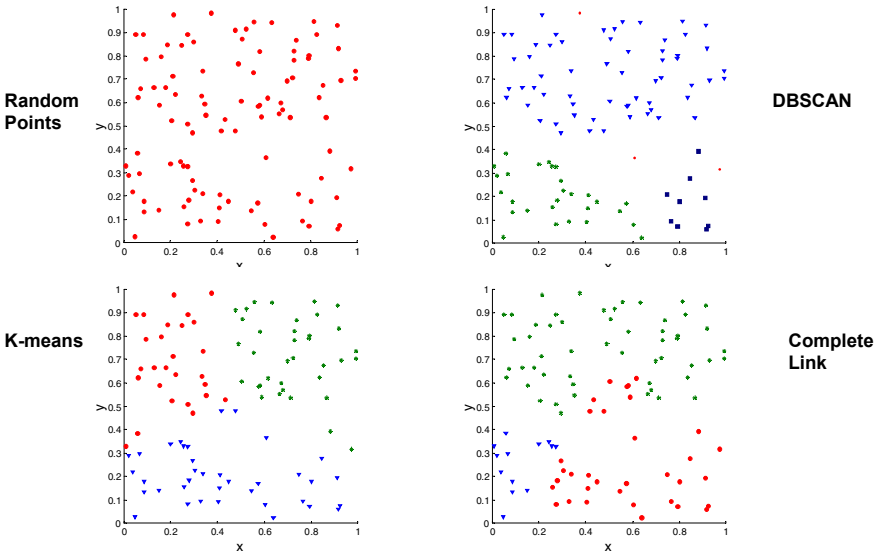
1. **Varying Density Clusters**: DBSCAN assumes that clusters have a similar density, determined by the `eps` and `minPts` parameters. If the dataset has clusters with varying densities, DBSCAN might not be able to identify all the clusters correctly. It may merge multiple low-density clusters or split a high-density cluster into multiple parts.
2. **High-Dimensional Data**: As the number of dimensions increases, the distance between points becomes less meaningful due to the curse of dimensionality. This makes it difficult for DBSCAN to find clusters in high-dimensional spaces because the `eps` parameter becomes harder to estimate and less effective.
3. **Parameter Sensitivity**: Choosing the right values for `eps` and `minPts` can be challenging. If `eps` is too small, many points will be classified as noise. If `eps` is too large, clusters may merge. Similarly, a small `minPts` value may merge distinct clusters, while a large `minPts` value may lead to many points being classified as noise.
4. **Border Points with Multiple Neighboring Clusters**: DBSCAN may have difficulty correctly assigning border points that are equidistant to core points of different clusters. These points could potentially be assigned to multiple clusters, which may not always be desirable.
5. **Clusters with Non-Spherical Shapes**: While DBSCAN can detect clusters of arbitrary shapes, it can struggle with elongated or irregularly shaped clusters where the density is not uniform throughout the cluster.
6. **Data with Noise**: If the data has a lot of noise, DBSCAN might create many small clusters or might not be able to form any clusters at all, as noise points do not contribute to forming a cluster but can prevent nearby points from being considered part of a cluster.
7. **Computational Performance**: Although DBSCAN doesn't require the number of clusters to be specified and can handle large datasets, its computational complexity can be an issue with very large datasets. In the worst-case scenario, the complexity is O(n^2), although with indexing techniques, this can be reduced to O(n log n) for many datasets.
8. **Difficulty with Clusters of Varying Densities**: DBSCAN can struggle when the dataset contains clusters of varying densities. It may not be able to capture the global structure of the data with a single `eps` value.

To mitigate some of these issues, variations of DBSCAN have been developed, such as OPTICS (Ordering Points To Identify the Clustering Structure), which can handle varying densities better by replacing the global `eps` parameter with a reachability distance that adapts to the local density conditions.

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"!
  - In practice the clusters we find are defined by the clustering algorithm
- 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



Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following two types.
  - Supervised: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
    - Often called *external indices* because they use information external to the data
  - Unsupervised: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
    - Often called *internal indices* because they only use information in the data
- You can use supervised or unsupervised measures to compare clusters or clusterings

Unsupervised 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
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

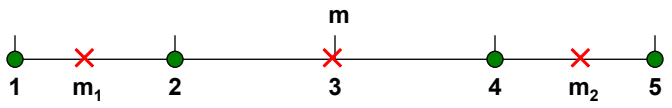
    $$SSE = \sum_i \sum_{x \in C_i} (x - m_i)^2$$
  - Separation is measured by the between cluster sum of squares

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

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

Unsupervised Measures: Cohesion and Separation

- Example: SSE
  - SSB + SSE = constant



K=1 cluster:

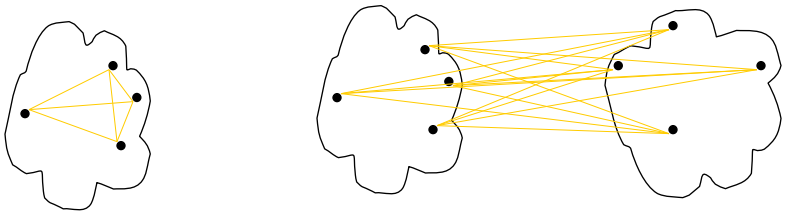
$$SSE = (1-3)^2 + (2-3)^2 + (4-3)^2 + (5-3)^2 = 10$$
$$SSB = 4 \times (3-3)^2 = 0$$
$$Total = 10 + 0 = 10$$

K=2 clusters:

$$SSE = (1-1.5)^2 + (2-1.5)^2 + (4-4.5)^2 + (5-4.5)^2 = 1$$
$$SSB = 2 \times (3-1.5)^2 + 2 \times (4.5-3)^2 = 9$$
$$Total = 1 + 9 = 10$$

Unsupervised 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

You

How are SSE and SSB used together to evaluate the goodness of clusters?

ChatGPT

Sum of Squared Errors (SSE) and Sum of Squared Between (SSB) are two metrics that can be used together to evaluate the goodness of clusters in clustering algorithms like k-means. These metrics offer complementary perspectives on clustering quality by assessing both the compactness of individual clusters (SSE) and the separation between clusters (SSB). Here's how they work and can be used together:

Sum of Squared Errors (SSE)

- **Definition**: SSE measures the compactness of clusters by calculating the sum of the squared distances between each point in a cluster and the centroid of that cluster. A lower SSE value indicates that points within a cluster are closer to the centroid, suggesting tighter clusters.
- **Formula**: For each cluster C_i, with centroid m_i, SSE is calculated as:

$$\text{SSE} = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - m_i||^2$$

where k is the number of clusters, x is a point in cluster C_i, and m_i is the centroid of cluster C_i.

Sum of Squared Between (SSB)

- **Definition**: SSB measures the separation between clusters by calculating the sum of the squared distances between the centroids of each cluster and the overall mean of the data. A higher SSB value indicates that clusters are further apart from each other, suggesting better separation.
- **Formula**: Let M be the overall mean of the dataset, and m_i the centroid of cluster C_i with n_i points. SSB is calculated as:

$$\text{SSB} = \sum_{i=1}^{k} n_i ||m_i - M||^2$$

where k is the number of clusters.

Using SSE and SSB Together

To evaluate the goodness of clusters, both SSE and SSB are considered:

- **Compactness and Separation**: A good clustering result typically has low SSE (indicating compact clusters) and high SSB (indicating well-separated clusters). These criteria reflect the goal of maximizing intra-cluster similarity while minimizing inter-cluster similarity.
- **Trade-offs**: There's often a trade-off between SSE and SSB when choosing the number of clusters. Increasing the number of clusters can decrease SSE (since clusters become smaller and points are closer to their centroids) but may also decrease the distinction between clusters (potentially lowering SSB if new clusters are not well-separated).
- **Evaluation and Decision Making**: By plotting SSE and SSB against the number of clusters, one can visually inspect the trade-offs and choose an optimal number of clusters. The elbow method is commonly used with SSE to find a good number of clusters, while observing SSB ensures that the chosen number also maintains adequate separation between clusters.
- **Balancing Metrics**: Ideally, the choice of the number of clusters and the clustering algorithm parameters should strike a balance where SSE is minimized, and SSB is maximized within practical and domain-specific constraints. This balance indicates that the clusters are both internally cohesive and distinctly separated from each other.

In practice, evaluating clustering quality often involves multiple metrics and domain knowledge to interpret the results meaningfully. SSE and SSB provide a quantitative foundation, but the best clustering solution also depends on how well the clusters capture meaningful patterns in the data as relevant to the specific application or analysis goals.

You

Quantitatively, how are SSB and SSE used together, assuming i know their values? Are they summed? Are there more complex approaches?

ChatGPT

Quantitatively evaluating clustering quality using Sum of Squared Errors (SSE) and Sum of Squared Between (SSB) does not typically involve simply summing these values. Instead, these metrics are used together to provide insights into the clustering structure in terms of compactness (SSE) and separation (SSB). The relationship between these metrics and their combined use can be understood through the Total Sum of Squares (SST), and also by analyzing the ratio or difference between them for a more nuanced understanding of clustering quality.

Relationship with Total Sum of Squares (SST)

The Total Sum of Squares (SST) represents the total variance in the dataset and can be partitioned into the within-cluster sum of squares (SSE) and the between-cluster sum of squares (SSB):

$$\text{SST} = \text{SSE} + \text{SSB}$$

- **SST (Total Sum of Squares)** measures the total variance in the dataset, calculated as the sum of squared distances from each point to the overall mean of the data.
- **SSE (Sum of Squared Errors)** measures the variance within each cluster.
- **SSB (Sum of Squared Between)** measures the variance between clusters.

This relationship highlights that reducing SSE (increasing cluster compactness) while maintaining or increasing SSB (enhancing cluster separation) improves clustering quality.

Using SSB and SSE Together

1. **Ratio or Index**: Rather than summing SSB and SSE, one might look at their ratio or create an index that captures the trade-off between compactness and separation. For example, a high SSB/SSE ratio might indicate good clustering by showing that the separation between clusters is large relative to the compactness within clusters.
2. **Dunn Index**: This is an example of a more complex approach, where the metric considers both the smallest distance between observations of different clusters (minimizing inter-cluster distances) and the largest intra-cluster distance (maximizing intra-cluster compactness). The Dunn Index looks for a high value, indicating well-separated and compact clusters.
3. **Silhouette Score**: Another method involves calculating the silhouette score for each point, which measures how similar a point is to its own cluster compared to other clusters. The silhouette score provides a more nuanced view that inherently considers both cohesion (similar to SSE) and separation (similar to SSB) but on a per-point basis.

Evaluating Clustering Quality

- **Interpretation**: Quantitative measures must be interpreted in the context of the dataset and the specific clustering task. A low SSE value indicates that clusters are compact, but without considering SSB, one cannot determine if the clusters are also well-separated.
- **Visualization**: Plotting SSE and SSB against the number of clusters can help visualize the trade-offs involved in choosing the optimal number of clusters.
- **Domain Knowledge**: Ultimately, the best evaluation comes from considering these metrics alongside domain-specific knowledge and the practical significance of the identified clusters.

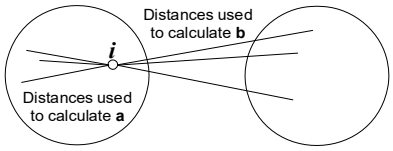
In summary, while SSE and SSB are not summed together for a composite score, examining both metrics provides a comprehensive view of clustering performance. A balanced approach to clustering seeks to minimize SSE while maximizing SSB, and sophisticated methods or indices can be used to quantify and optimize this balance.

Unsupervised 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)

  - Value can vary between -1 and 1
  - Typically ranges between 0 and 1.
  - The closer to 1 the better.



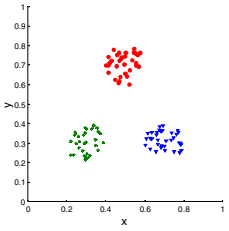
- Can calculate the average silhouette coefficient for a cluster or a clustering

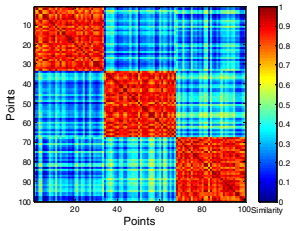
Measuring Cluster Validity Via Correlation

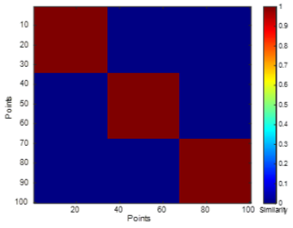
- Two matrices
  - Proximity Matrix
  - Ideal Similarity Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between n(n-1) / 2 entries needs to be calculated.
- High magnitude of correlation indicates that points that belong to the same cluster are close to each other.
  - Correlation may be positive or negative depending on whether the similarity matrix is a similarity or dissimilarity matrix
- 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 well-clustered data set.



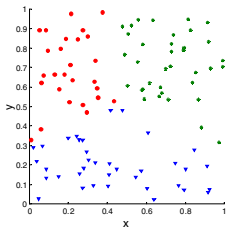


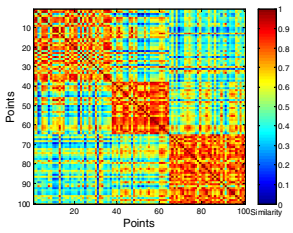


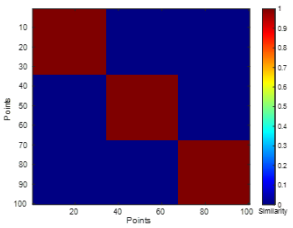
Corr = 0.9235

Measuring Cluster Validity Via Correlation

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





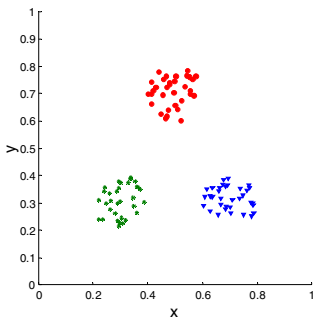


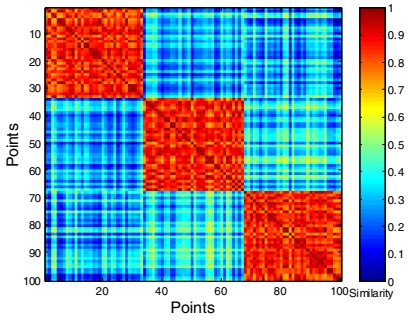
K-means

Corr = 0.5810

Judging a Clustering Visually by its Similarity Matrix

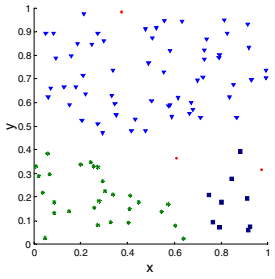
- Order the similarity matrix with respect to cluster labels and inspect visually.

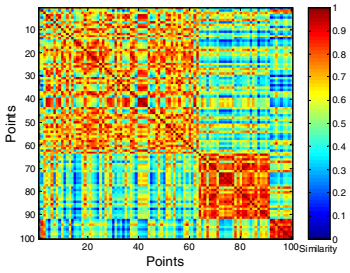




Judging a Clustering Visually by its Similarity Matrix

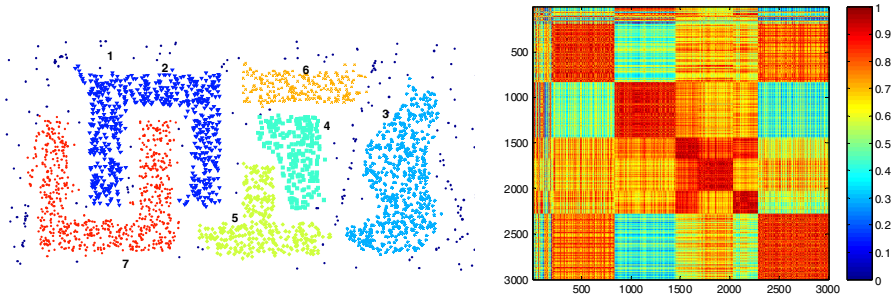
- Clusters in random data are not so crisp





DBSCAN

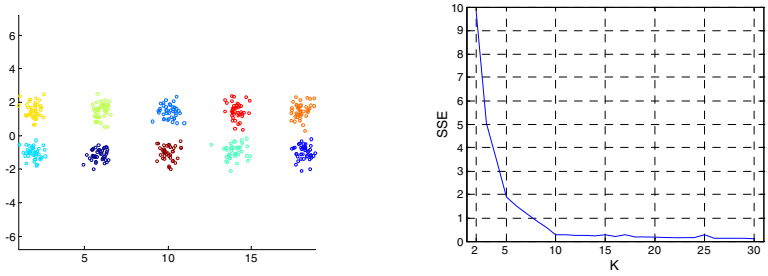
Judging a Clustering Visually by its Similarity Matrix



DBSCAN

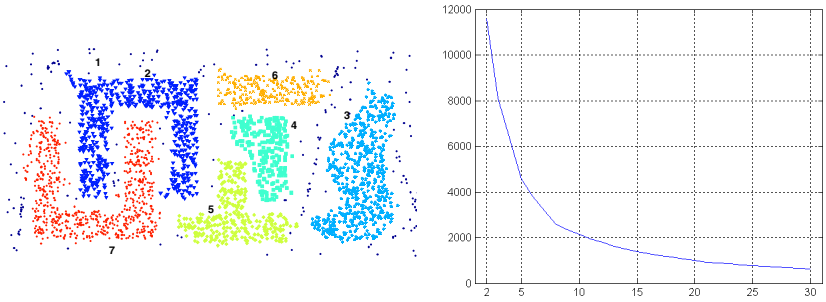
Determining the Correct Number of Clusters

- SSE is good for comparing two clusterings or two clusters
- SSE can also be used to estimate the number of clusters



Determining the Correct Number of Clusters

- SSE curve for a more complicated data set



SSE of clusters found using K-means

Supervised Measures of Cluster Validity: Entropy and Purity

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

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

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

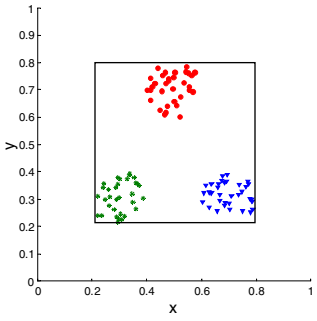
purity Using the terminology derived for entropy, the purity of cluster j, is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$.

Assessing the Significance of Cluster Validity Measures

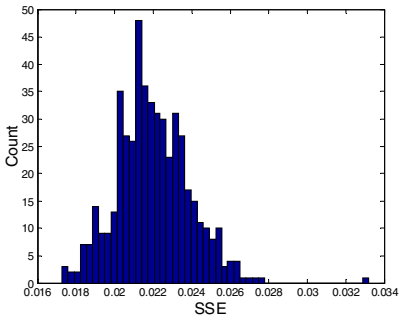
- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more "atypical" a clustering result is, the more likely it represents valid structure in the data
  - Compare the value of an index obtained from the given data with those resulting from random data.
    - If the value of the index is unlikely, then the cluster results are valid

Statistical Framework for SSE

- Example
  - Compare SSE of three cohesive clusters against three clusters in random data



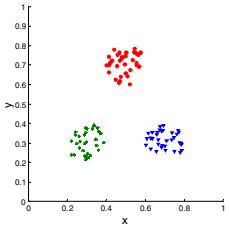
SSE = 0.005

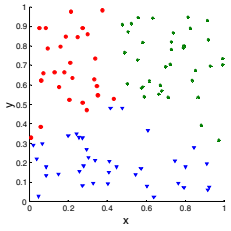


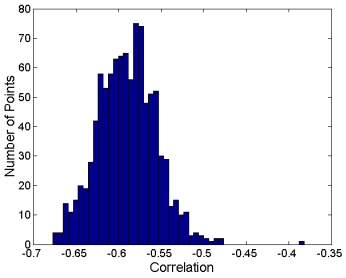
Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values

Statistical Framework for Correlation

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







Corr = -0.9235

Corr = -0.5810

Correlation is negative because it is calculated between a distance matrix and the ideal similarity matrix. Higher magnitude is better.

Histogram of correlation for 500 random data sets of size 100 with *x* and *y* values of points between 0.2 and 0.8.

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**

- H. Xiong and Z. Li. *Clustering Validation Measures*. In C. C. Aggarwal and C. K. Reddy, editors, Data Clustering: Algorithms and Applications, pages 571–605. Chapman & Hall/CRC, 2013.