Unit 3: Machine Learning

Best wishes to readers,
from Dr. A. Saroj

TCS-509

TOPICS COVERED IN UNIT 3

- **Ensemble Learning: Bagging**, Random Forest, Boosting -AdaBoost
- **Clustering:** K-means, Hierarchical Clustering, Fuzzy c-means, DBScan
- **Feature Engineering: Dimensionality** issues, feature selection & extraction, Low Variance Filter, High Correlation Filter, Feature selection -Component Analysis: PCA, IDA, Discriminant analysis

Ensemble Learning

- Ensemble learning is a technique that create multiple models and then combine them to produce improved result.

- Ensemble learning usually produces more accurate solutions than a single model would.

- Ensemble learning methods is applied regression as well as classification.
- Ensemble learning for regression creates multiple repressors i.e. multiple regression models such as linear, Polynomial etc.

- Ensemble learning for classification creates multiple classifiers i.e. multiple classification models such as logistic, Decision trees, SVM, KNN etc.

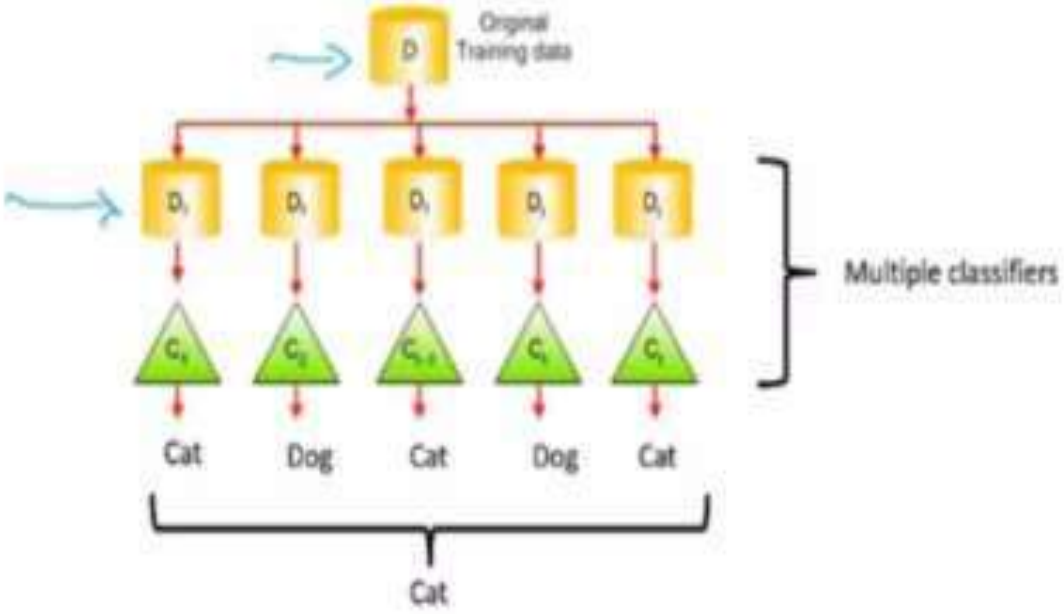
Ensemble Learning Cont….

- There are two steps in Ensemble Learning:
- Multiple machine learning models were generated using same or different machine learning method. These are called base models.
- The prediction perform on the basis of base model.

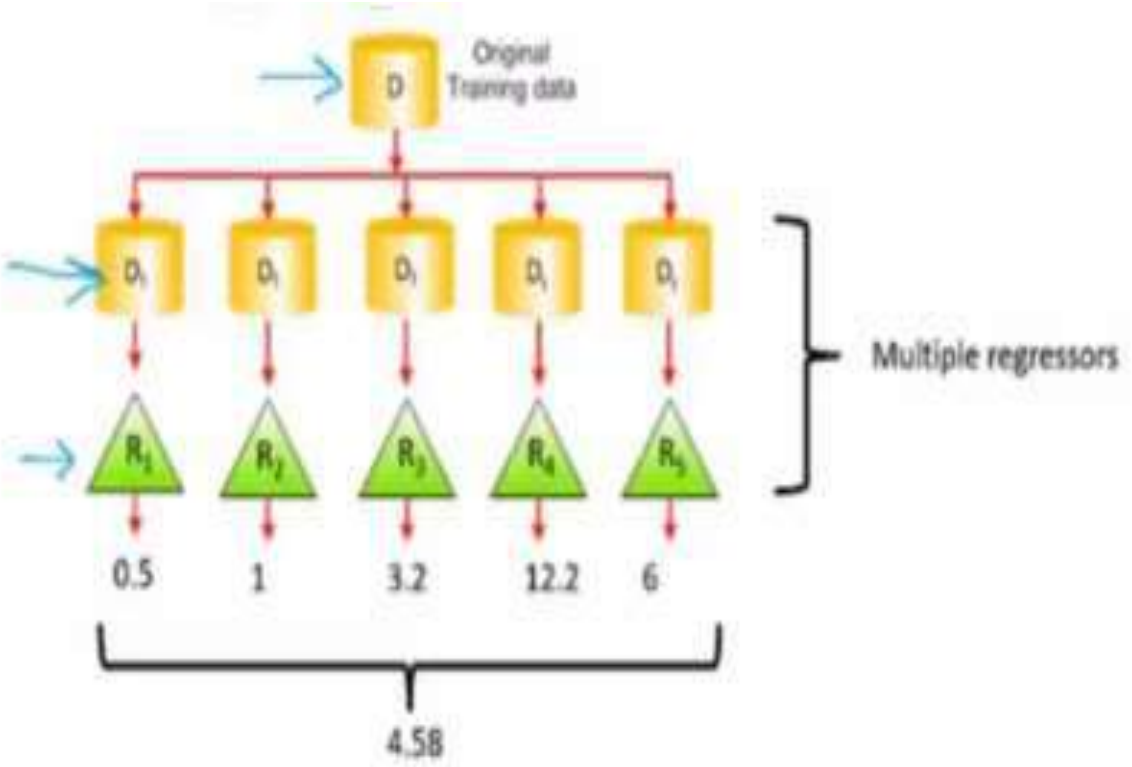
- Techniques/Method
- Voting/Averaging
- Stacking
- Bootstrap Aggregating/Bagging
- Boosting

Ensemble Learning Techniques

- Voting/Averaging
- Voting used in classification
- Averaging used in regression



Averaging used in regression



Ensemble Approaches

- Bagging
  - **B**ootstrap aggregating
- Boosting
- Random Forests
  - Bagging reborn

Bagging

- Main Assumption:
  - Combining many unstable predictors to produce a ensemble (stable) predictor.
  - Unstable Predictor: small changes in training data produce large changes in the model.
    - e.g. Neural Nets, trees
    - Stable: SVM (sometimes), Nearest Neighbor.
- Hypothesis Space
  - Variable size (nonparametric):
    - Can model any function if you use an appropriate predictor (e.g. trees)

The Bagging Algorithm

Given data: $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

For $m = 1 : M$

- Obtain bootstrap sample D_m from the training data D
- Build a model $G_m(\mathbf{x})$ from bootstrap data D_m

The Bagging Model

- Regression

$$\hat{y} = \frac{1}{M}\sum_{m=1}^{M} G_m(\mathbf{x})$$

- Classification:
  - Vote over classifier outputs $G_1(\mathbf{x}),\ldots,G_M(\mathbf{x})$

Bagging Details

- Bootstrap sample of N instances is obtained by drawing N examples at random, with replacement.
- On average each bootstrap sample has 63% of instances
  - Encourages predictors to have uncorrelated errors
    - This is why it works

Bagging Details Cont..

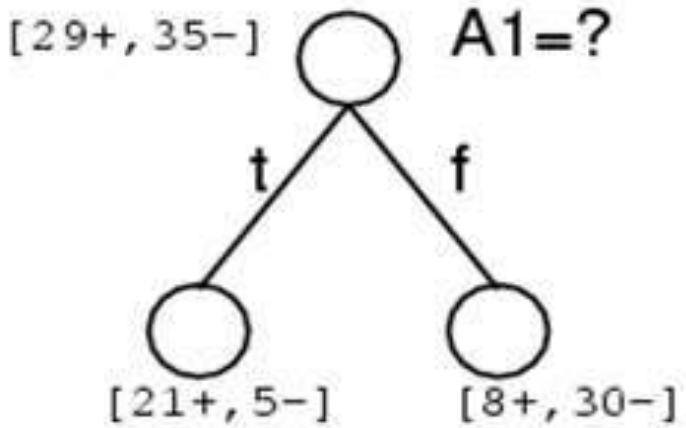
- Usually set $M = \sim 30$
  - Or use validation data to pick M
- The models $G_m(\mathbf{x})$ need to be unstable
  - Usually full length (or slightly pruned) decision trees.

Boosting

- Main Assumption:
  - Combining many weak predictors (e.g. tree stumps or 1-R predictors) to produce an ensemble predictor
  - The weak predictors or classifiers need to be **stable**
- Hypothesis Space
  - Variable size (nonparametric):
    - Can model any function if you use an appropriate predictor (e.g. trees)

Commonly Used Weak Predictor (or classifier)

A Decision Tree Stump (1-R)



Boosting (Continued)

- Each predictor is created by using a biased sample of the training data
  - Instances (training examples) with high error are weighted higher than those with lower error
- Difficult instances get more attention
  - This is the motivation behind boosting

Background Notation

- The $I(s)$ function is defined as:

$$I(s) = \begin{cases} 1 \text{ if } s \text{ is true} \\ 0 \quad \text{otherwise} \end{cases}$$

- The $\log(x)$ function is the natural logarithm

The AdaBoost Algorithm

(Freund and Schapire, 1996)

Given data: $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$

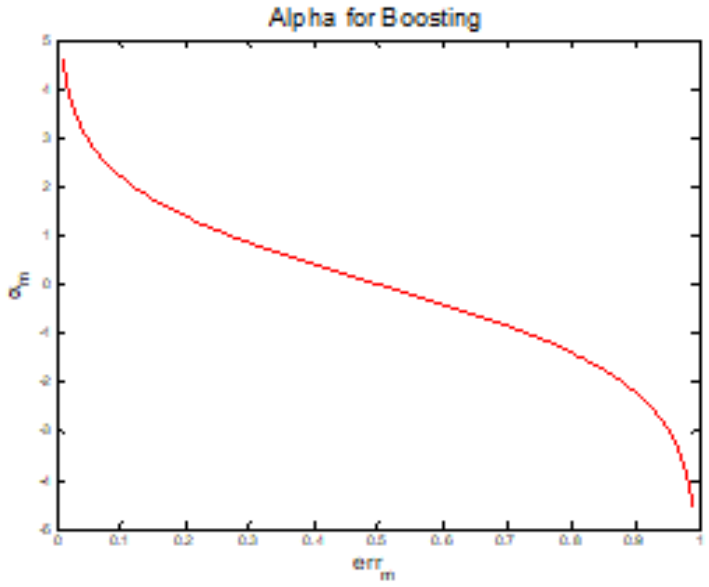
1. Initialize weights $w_i = 1/N, i = 1, \ldots, N$
2. For $m = 1 : M$
   a) Fit classifier $G_m(\mathbf{x}) \in \{-1, 1\}$ to data using weights w_i
   b) Compute $$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i}$$
   c) Compute $\alpha_m = \log((1 - err_m)/err_m)$
   d) Set $w_i \leftarrow w_i \exp[\alpha_m I(y_i \neq G_m(\mathbf{x}_i))], \quad i = 1, \ldots, N$

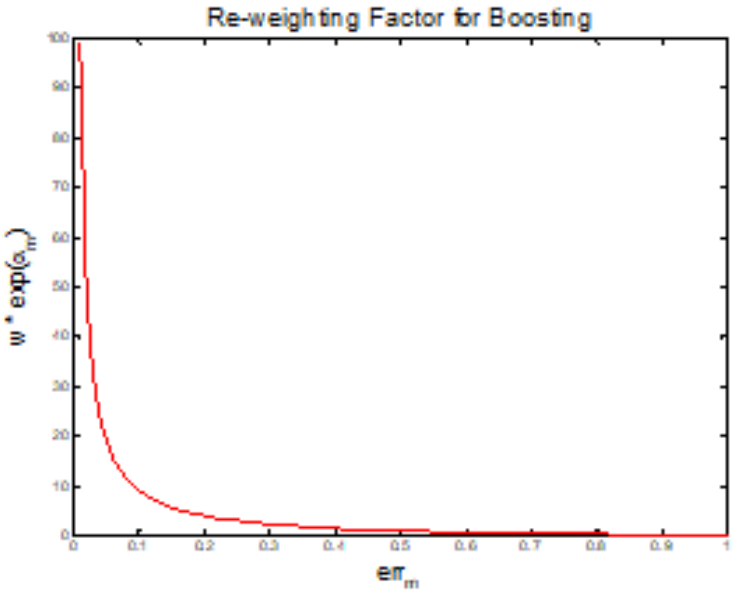
The AdaBoost Model

$$\hat{y} = \text{sgn}\left[\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})\right]$$

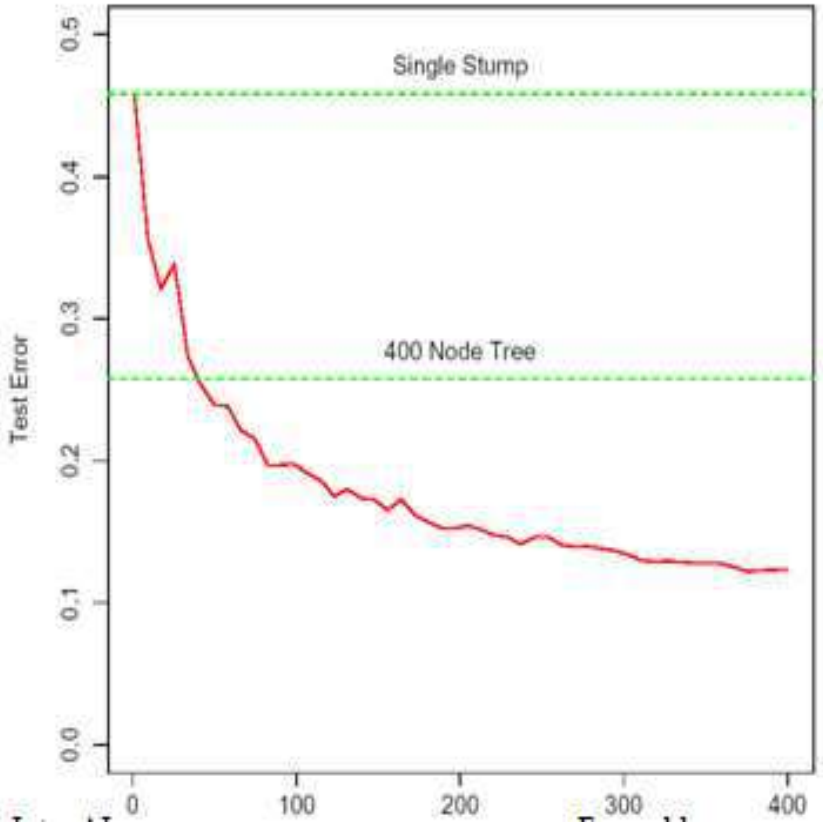
AdaBoost is NOT used for Regression!

The Updates in Boosting



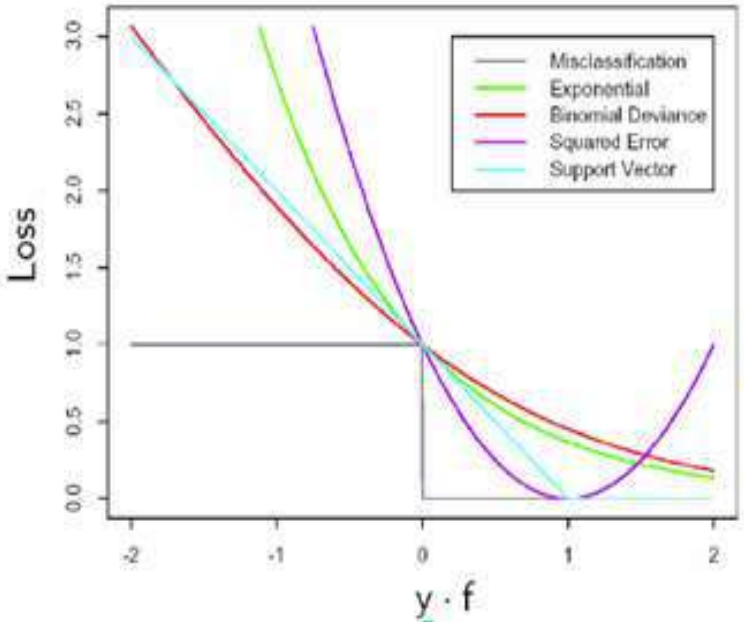


Boosting Characteristics



Simulated data: test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 400 node tree.

Loss Functions for $y \in \{-1, +1\}, f \in \Re$



- Misclassification

$$I(\operatorname{sgn}(f) \neq y)$$

- Exponential (Boosting)

$$\exp(-yf)$$

- Binomial Deviance
(Cross Entropy)

$$\log(1+\exp(-2yf))$$

- Squared Error

$$(y-f)^2$$

- Support Vectors

$$(1-yf) \cdot I(yf > 1)$$

Clustering- Unsupervised learning

- Given a set of unlabeled data points / items
- Find patterns or structure in the data
- Clustering: automatically group the data points / items into groups of ‘similar’ or ‘related’ points
- Main challenges

– How to measure similarity?

– What is the ideal number of clusters? Few larger clusters, or more number of smaller clusters?

Motivations for Clustering

- **Understanding the data better**
  - Grouping Web search results into clusters, each of which captures a particular aspect of the query
  - Segment the market or customers of a service
- **As precursor for some other application**
  - Summarization and data compression
  - Recommendation

Different types of clustering

- **Partitional**
  - Divide set of items into non-overlapping subsets
  - Each item will be member of one subset
- **Overlapping**
  - Divide set of items into potentially overlapping subsets
  - Each item can simultaneously belong to multiple subsets

Different types of clustering Cont..

- **Fuzzy**
  - Every item belongs to every cluster with a membership weight between 0 (absolutely does not belong) and 1 (absolutely belongs)
  - Usual constraint: sum of weights for each individual item should be 1
  - Convert to partitional clustering: assign every item to that cluster for which its membership weight is highest

Different types of clustering Cont..

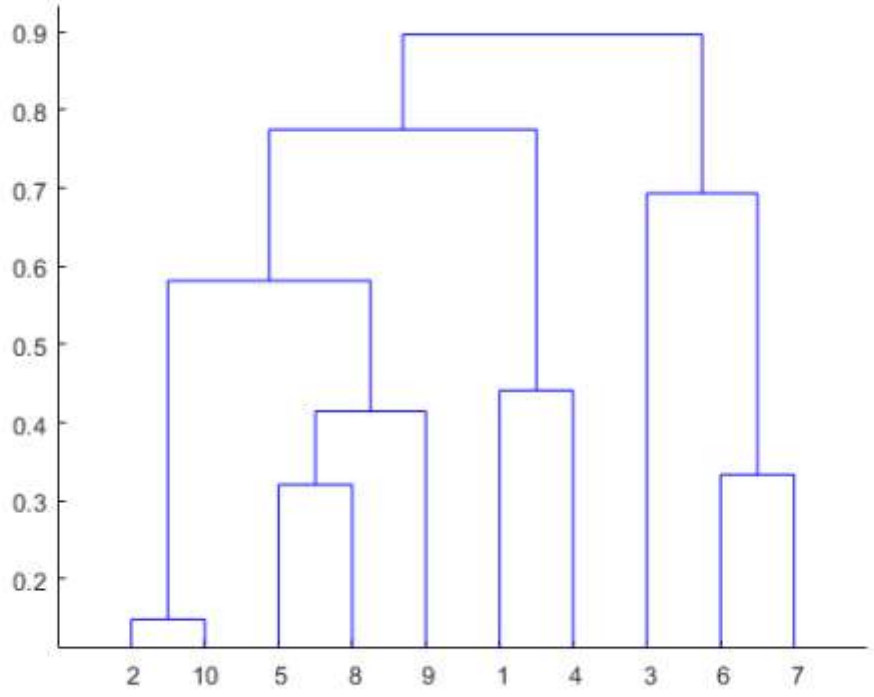
- **Hierarchical**

– Set of nested clusters, where one larger cluster can contain smaller clusters

– Organized as a tree (dendrogram): leaf nodes are singleton clusters containing individual items, each intermediate node is union of its children sub-clusters

– A sequence of partitional clusterings – cut the dendrogram at a certain level to get a partitional clustering

An example dendrogram



Different types of clustering Cont..

- **Complete vs. partial**

– A complete clustering assigns every item to one or more clusters

– A partial clustering may not assign some items to any cluster (e.g., outliers, items that are not sufficiently similar to any other item)

Types of clustering methods

- **Prototype-based**

– Each cluster defined by a prototype (centroid or medoid), i.e., the most representative point in the cluster

– A cluster is the set of items in which each item is closer (more similar) to the prototype of this cluster, than to the prototype of any other cluster

– Example method: K-means

Types of clustering methods

- Density-based
  - Assumes items distributed in a space where ‘similar’ items are placed close to each other (e.g., feature space)
  - A cluster is a dense region of items, that is surrounded by a region of low density
  - **Example method: DBSCAN**

Types of clustering methods

- **Graph-based**

– Assumes items represented as a graph/network where items are nodes, and ‘similar’ items are linked via edges

– A cluster is a group of nodes having more and / or better connections among its members, than between its members and the rest of the network

– Also called ‘community structure’ in networks

– Example method: Algorithm by Girvan and Newman

Common Distance Measures

- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

  They include:

1. The Euclidean distance (also called 2-norm distance) is given by:

$$d(x,y) = \sum_{i=1}^{p} |x_i - y_i|$$

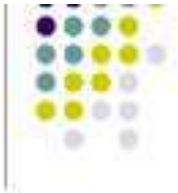
2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d(x,y) = \sqrt[2]{\sum_{i=1}^{p} |x_i - y_i|^2}$$

Common Distance Measures Cont..

3.The maximum norm is given by:

$$d(x,y) = \max_{1 \le i \le p} |x_i - y_i|$$



4. The Mahalanobis distance corrects data for different scales and correlations in the variables.
5. Inner product space: The angle between two vectors can be used as a distance measure when clustering high dimensional data
6. Hamming distance (sometimes edit distance) measures the minimum number of substitutions required to change one member into another.

K-means clustering

- Prototype-based, partitioning technique
- Finds a user-specified number of clusters (K)
- Each cluster represented by its centroid item
- There have been extensions where number of clusters is not needed as input

K-means algorithm

- The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k < n$.
- It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.
- It assumes that the object attributes form a vector space.

K-means algorithm Cont..

- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

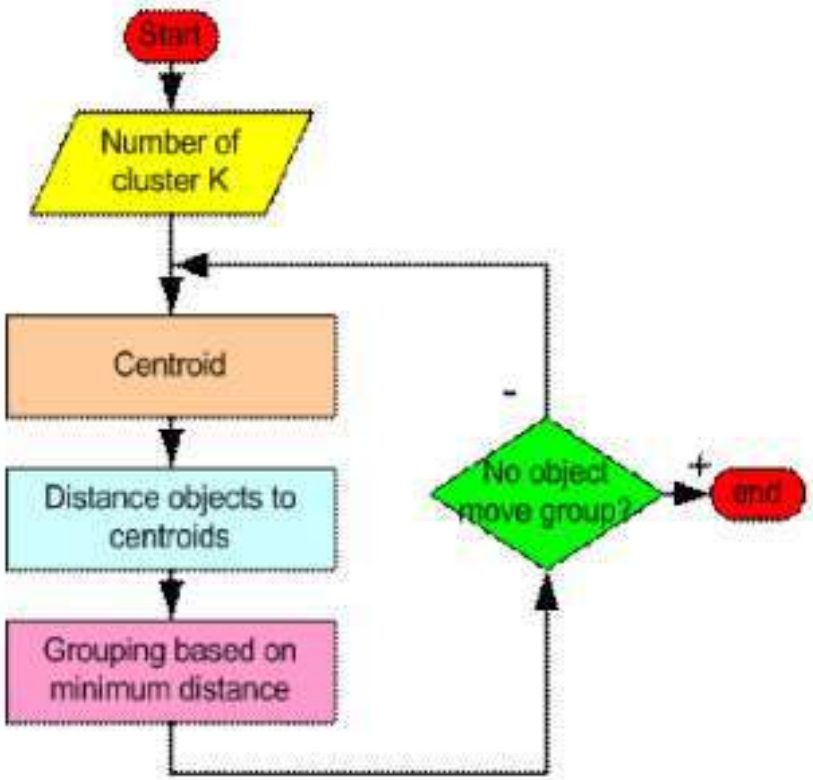
$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where x_n is a vector representing the the n^{th} data point and u_j is the geometric centroid of the data points in S_j.

K-means algorithm Cont..

- Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.
- K is positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

How the K-Means clustering algorithm works?



K-Means clustering algorithm

- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly,or systematically as the following:

1.Take the first k training sample as single-element clusters

2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

K-Means clustering algorithm Cont..

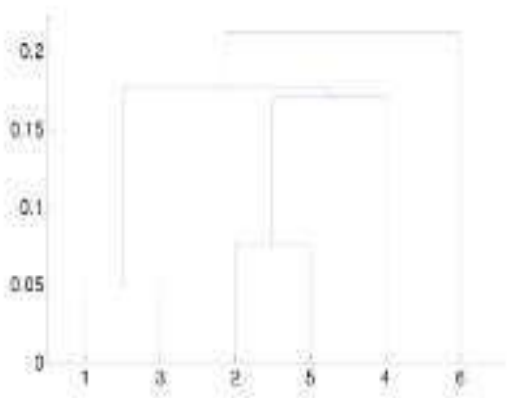
- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

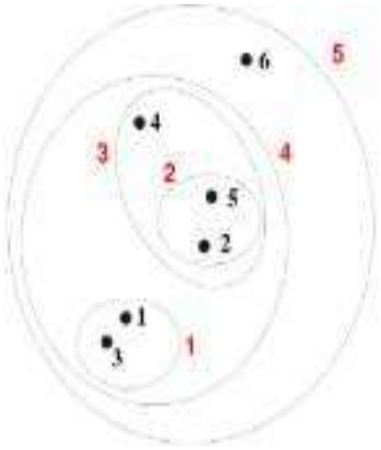
Example of K-Means algorithm (using K=2)

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

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.





Strengths of Hierarchical Clustering

- No assumptions on the number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level

- Hierarchical clusterings may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

Hierarchical Clustering: Problem definition

- Given a set of points $X = \{x_1, x_2, \ldots, x_n\}$ find a sequence of ***nested partitions*** $P_1, P_2, \ldots, P_n$ of X, consisting of $1, 2, \ldots, n$ clusters respectively such that $\sum_{i=1 \ldots n} Cost(P_i)$ is ***minimized.***

- Different definitions of $Cost(P_i)$ lead to different hierarchical clustering algorithms
  - $Cost(P_i)$ can be formalized as the cost of any partition-based clustering

Hierarchical Clustering Algorithms

- 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 a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Complexity of hierarchical clustering

- Distance matrix is used for deciding which clusters to merge/split

- At least quadratic in the number of data points

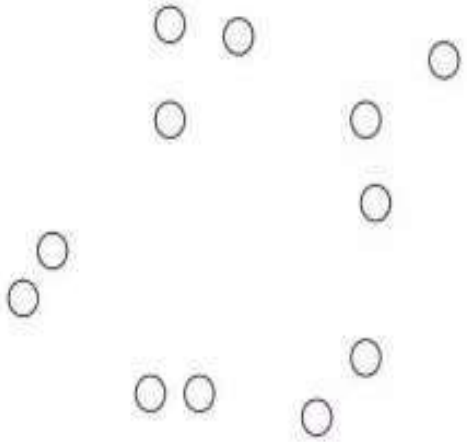
- Not usable for large datasets

Agglomerative clustering algorithm

- Most popular hierarchical clustering technique
- Basic algorithm
  - Compute the distance matrix between the input data points
  - Let each data point be a cluster
  - Repeat
    - Merge the two closest clusters
    - Update the distance matrix
  - Until only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

Starting Situation

Start with clusters of individual points and a distance matrix

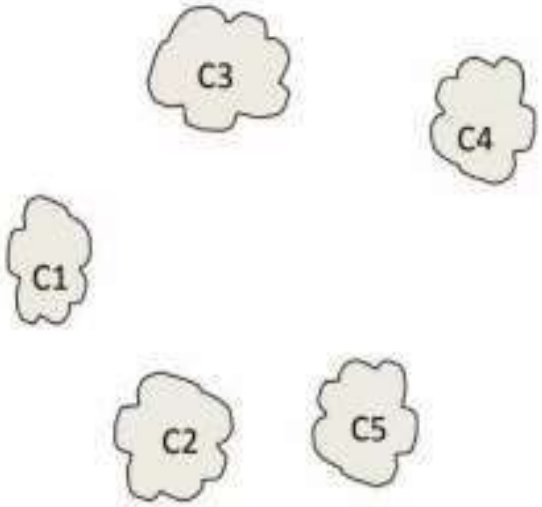


	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

Distance Matrix

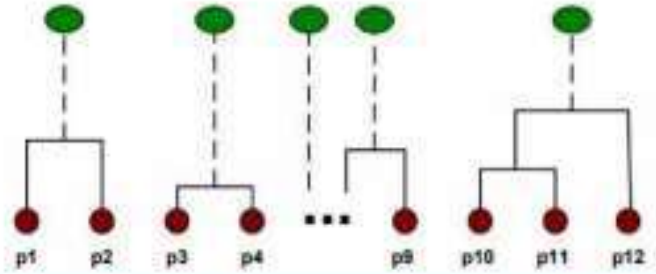
Intermediate Situation

- After some merging steps, we have some clusters
- Choose two clusters that has the smallest distance (largest similarity) to merge



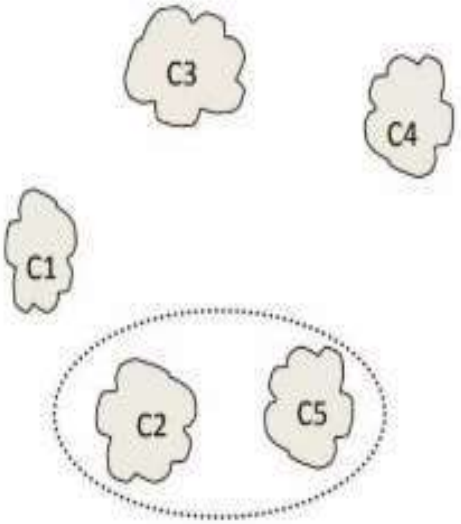
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance Matrix



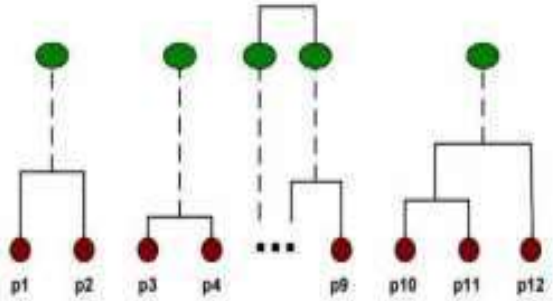
Intermediate Situation

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



	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Distance Matrix

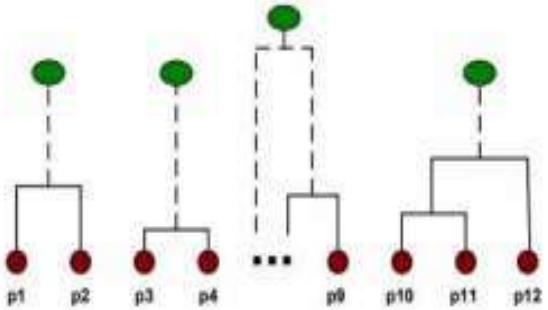


After Merging

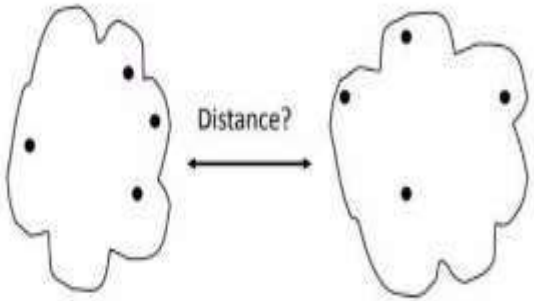
The question is "How do we update the distance matrix?"

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

Distance Matrix



How to Define Inter-Cluster Distance



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

Distance Matrix

- Single link method (Min)
- Complete link method (Max)
- Average link (group Average)
- Centroid method (Distance between centriods)

Distance between two clusters

- **Single-link distance** between clusters C_i and C_j is the **minimum distance** between any object in C_i and any object in C_j

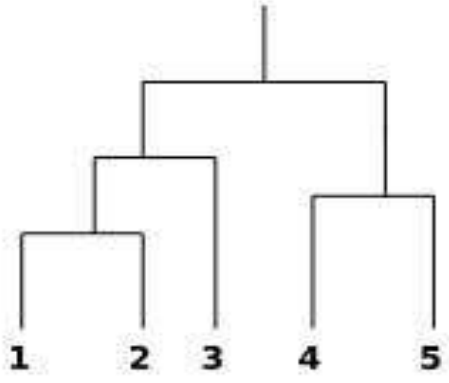
- The distance is **defined by the two most similar objects**

$$D_{\text{single}} = \min_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

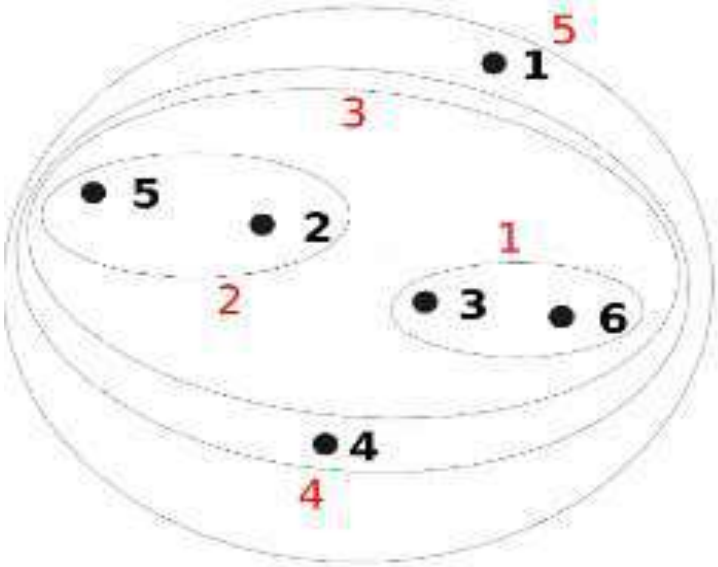
Single-link clustering: example

- Determined by one pair of points, i.e., by one link in the proximity graph.

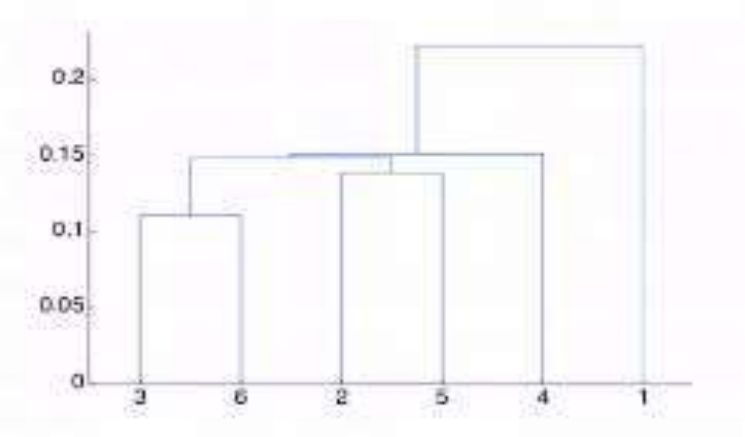
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Single-link clustering: example

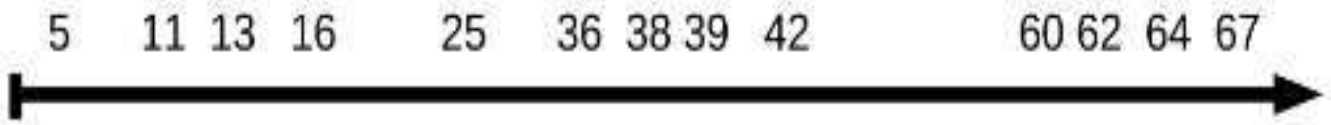


Nested Clusters



Dendrogram

Exercise: 1-dimensional clustering



Exercise:
Create a hierarchical agglomerative clustering for this data.
To make this deterministic, if there are ties, pick the left-most link.

Verify: clustering with 4 clusters has 25 as singleton.

Distance between two clusters

- **Complete-link distance** between clusters C_i and C_j is the **maximum distance** between any object in C_i and any object in C_j

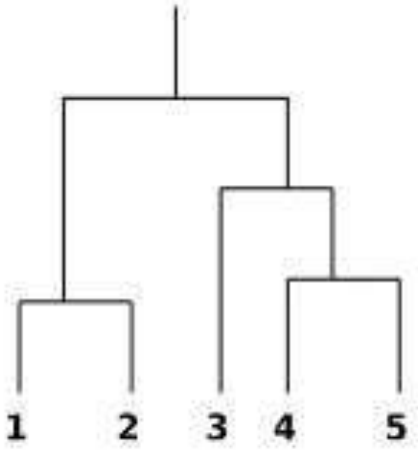
- The distance is **defined by the two most dissimilar objects**

$$D_{\text{complete}} = \max_{x,y}\{d(x,y) \mid x \in C_i, y \in C_j\}$$

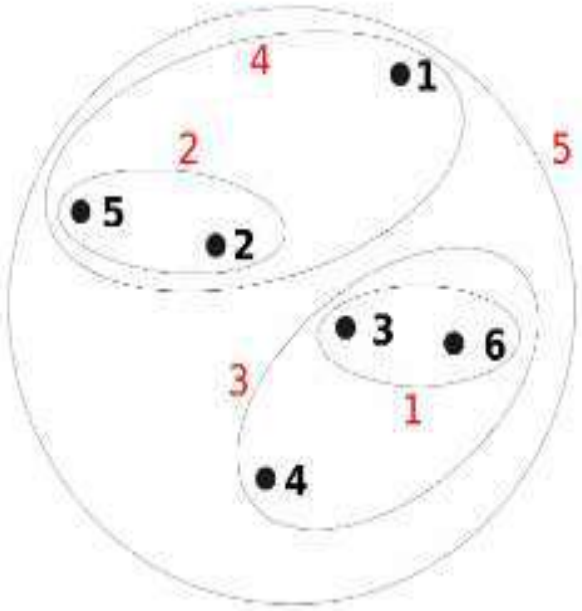
Complete-link clustering: example

- Distance between clusters is determined by the two most distant points in the different clusters

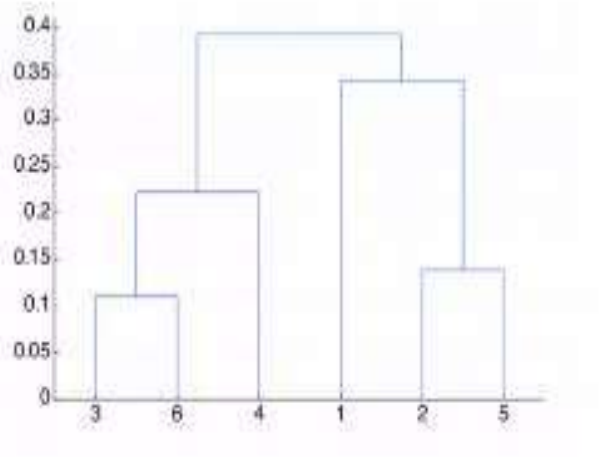
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Complete-link clustering: example



Nested Clusters



Dendrogram

Distance between two clusters

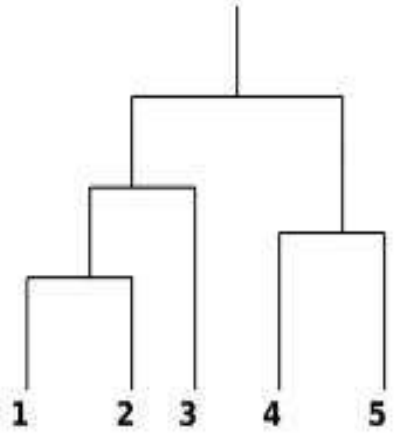
- **Group average distance** between clusters C_i and C_j is the **average distance** between any object in C_i and any object in C_j

$$D_{\text{average}} = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

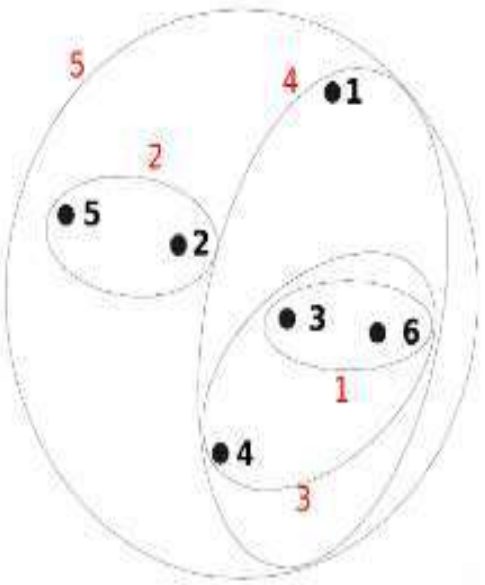
Average-link clustering: example

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

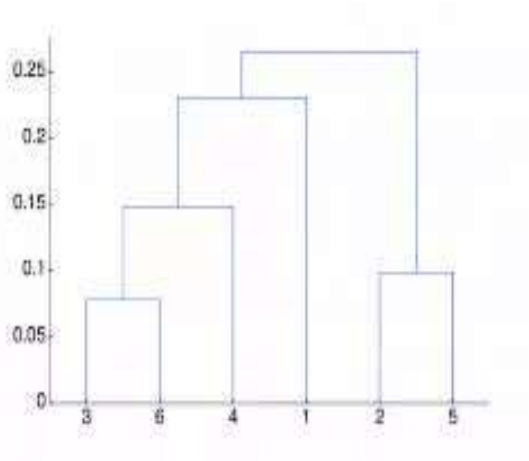
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Average-link clustering: example



Nested Clusters



Dendrogram

Distance between two clusters

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{\text{centroids}}(C_i, C_j) = d(r_i, r_j)$$

Distance between two clusters

- Ward's distance between clusters C_i and C_j is the difference between the total within cluster sum of squares for the two clusters separately, and the within cluster sum of squares resulting from merging the two clusters in cluster C_{ij}

$$D_W(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i: centroid of C_i
- r_j: centroid of C_j
- r_{ij}: centroid of C_{ij}

Ward's distance for clusters

- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

Hierarchical Clustering: Time and Space requirements

- For a dataset **X** consisting of **n** points
- **$O(n^2)$ space**; it requires storing the distance matrix
- **$O(n^3)$ time** in most of the cases
  - There are **n** steps and at each step the size n^2 distance matrix must be updated and searched
  - Complexity can be reduced to **$O(n^2 \log(n))$** time for some approaches by using appropriate data structures

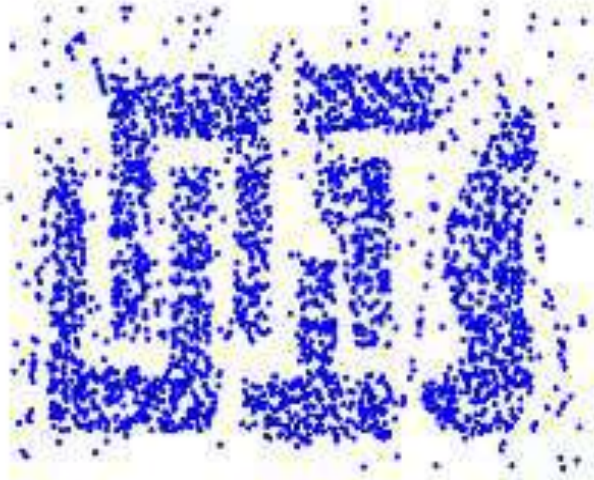
DBScan

Concepts: Preliminary

- **DBSCAN is a density-based algorithm**
- DBScan stands for Density-Based Spatial Clustering of Applications with Noise
- Density-based Clustering locates regions of high density that are separated from one another by regions of low density

  Density = number of points within a specified radius (Eps)

Concepts: Preliminary



Original Points

Point types: core, border and noise

Eps = 10, MinPts = 4

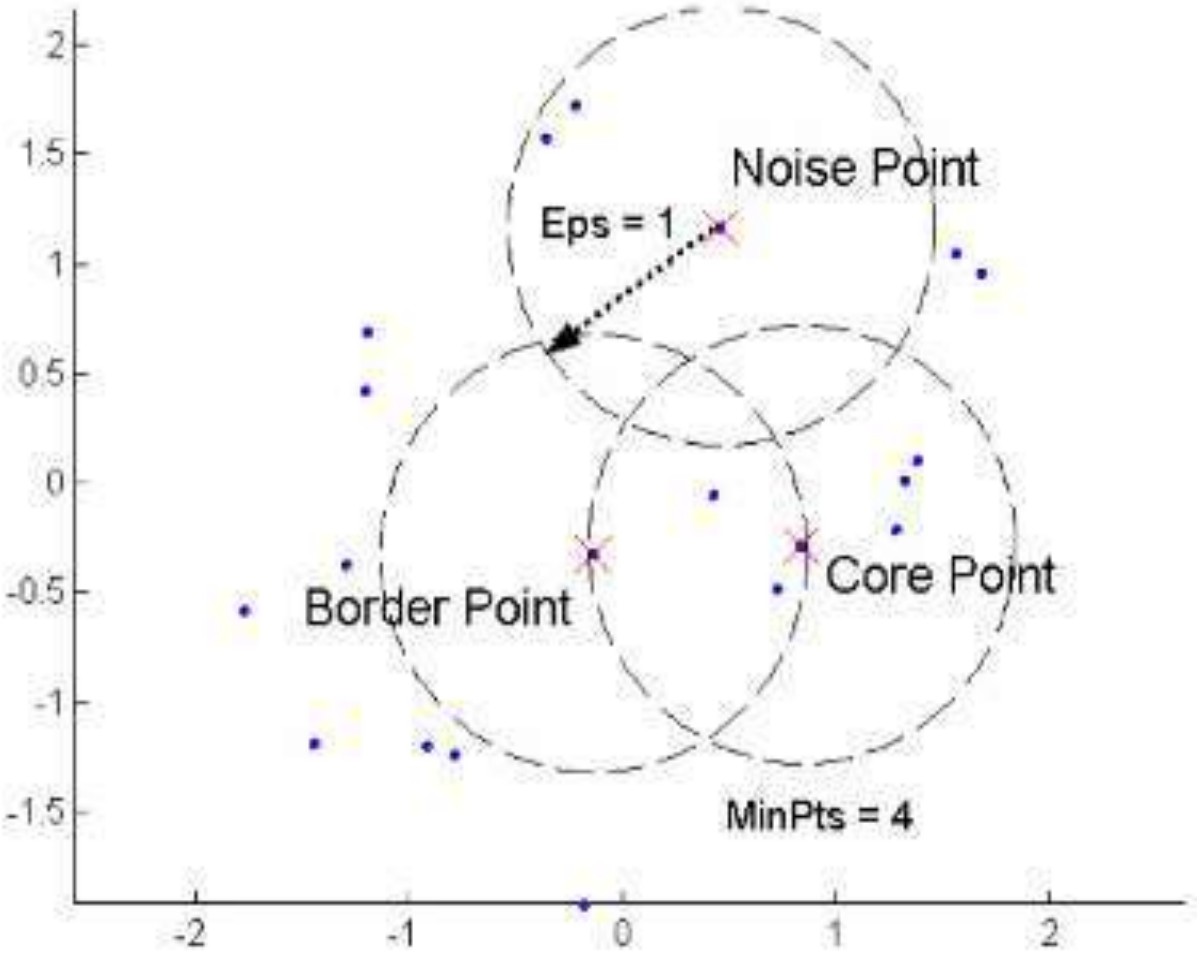
Concepts: Preliminary

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

Concepts: Preliminary

- Any two core points are close enough– within a distance *Eps* of one another – are put in the same cluster
- Any border point that is close enough to a core point is put in the same cluster as the core point
- Noise points are discarded

Concepts: Core, Border, Noise



Parameter Estimation

parameters must be specified by the user.

ε = physical distance(radius),
minPts = desired minimum cluster size

minPts

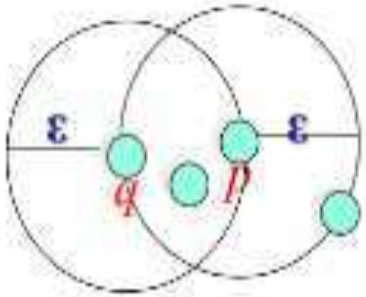
- derived from the number of dimensions *D* in the data set, as $minPts \geq D + 1$
- $minPts = 1$ does not make sense, as then every point on its own will already be a cluster
- *minPts* must be chosen at least 3. larger is better.
- larger the data set, the larger the value of *minPts* should be chosen.

ε

- value can be chosen by using a k-distance graph.
- if ε is chosen much too small, a large part of the data will not be clustered.
- if too high value ,majority of objects will be in the same cluster
- In general, small values of ε are preferable.

Concepts: ε-Neighborhood

- ε-Neighborhood - Objects within a radius of ε from an object. (epsilon-neighborhood)
- Core objects - ε-Neighborhood of an object contains at least MinPts of objects



ε-Neighborhood of *p*

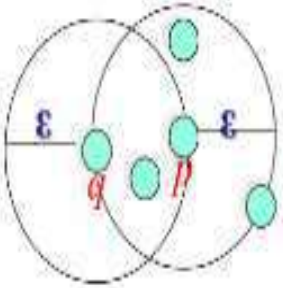
ε-Neighborhood of *q*

p is a core object (MinPts = 4)

q is not a core object

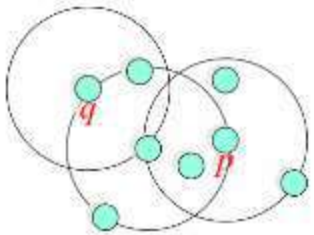
DBScan : Reachability

- **Directly density-reachable**
  - An object q is directly density-reachable from object p if q is within the ε-Neighborhood of p and p is a core object.



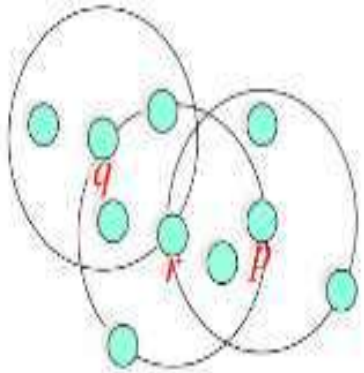
- q is directly density-reachable from p
- p is not directly density-reachable from q.

DBScan : Reachability



DBScan :Connectivity

- **Density-connectivity**
  - Object p is density-connected to object q w.r.t ε and *MinPts* if there is an object o such that both p and q are density-reachable from o w.r.t ε and *MinPts*



- P and q are density-connected to each other by r
- Density-connectivity is symmetric

DBScan Algorithm

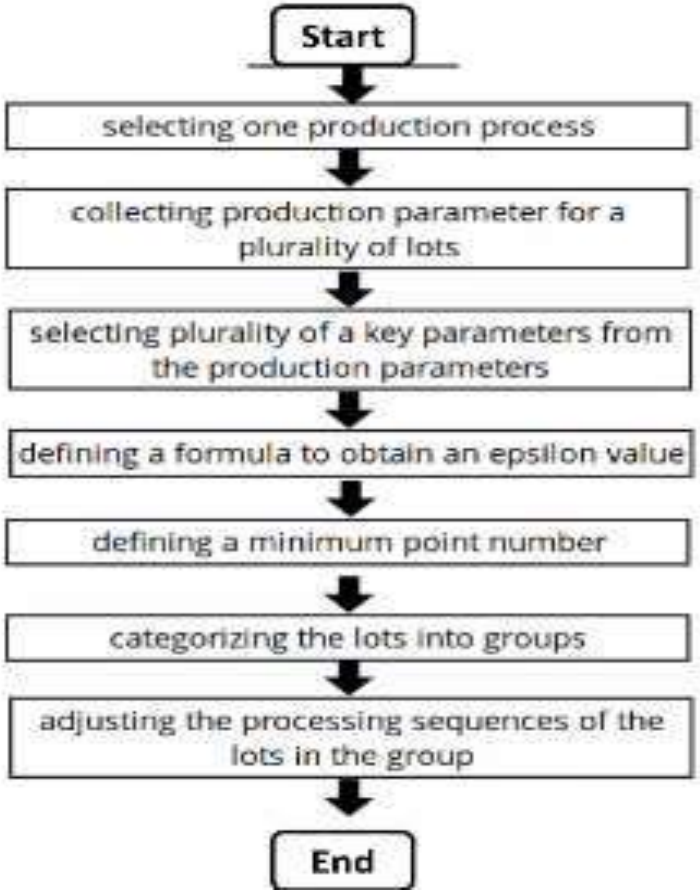
Input: N objects to be clustered and global parameters *Eps*, *MinPts*.

Output: Clusters of objects.

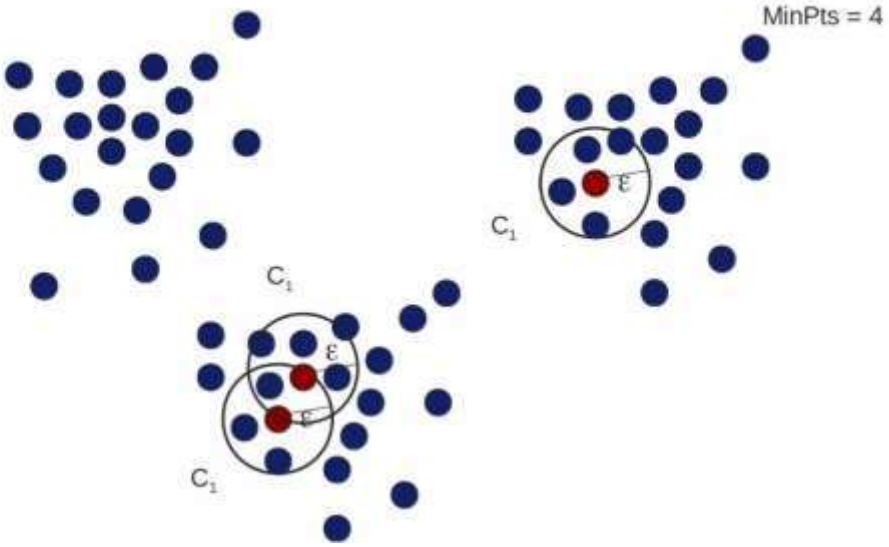
Algorithm:

1) Arbitrary select a point *P*.
2) Retrieve all points density-reachable from *P* wrt ***Eps*** and ***MinPts***.
3) If *P* is a core point, a cluster is formed.
4) If *P* is a border point, no points are density-reachable from *P* and **DBSCAN** visits the next point of the database.
5) Continue the process until all of the points have been processed.

DBScan :Flowchart



DBScan : Example



DBSCAN : Advantages

- Does not require one to specify the number of clusters in the data
- Can find arbitrarily shaped clusters. even find a cluster completely surrounded by a different cluster.
- Has a notion of noise, and is robust to outliers.
- Requires just two parameters and is mostly insensitive to the ordering of the points in the database.
- Designed for accelerate region queries.
- minPts and ε can be set by a domain expert

DBSCAN : Disadvantages

- DBSCAN is not entirely deterministic: Border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.

- The quality of DBSCAN depends on the distance measure used in the function regionQuery. (such as Euclidean distance)

- If the data and scale are not well understood, choosing a meaningful distance threshold ε can be difficult.

DBSCAN : Disadvantages

- DBSCAN is not entirely deterministic: Border points that are reachable from more than one cluster can be part of either cluster, depending on the order the data is processed.

- The quality of DBSCAN depends on the distance measure used in the function regionQuery. (such as Euclidean distance)

- If the data and scale are not well understood, choosing a meaningful distance threshold ε can be difficult.

DBSCAN : Complexity

- **Time Complexity:** $O(n^2)$
  - for each point it has to be determined if it is a core point.
  - can be reduced to O(n*log(n)) in lower dimensional spaces by using efficient data structures (n is the number of objects to be clustered);
- **Space Complexity:** O(n).

Question: what is Outliers? Outliers are often discarded as noise but some applications these noisy data can be more interesting than the more regularly occurring ones. why ?

Solution :

- The points marked as outliers aren't discarded as such, they are just points not in any cluster. You can still inspect the set of non-clustered points and try to interpret them.
- DBSCAN is designed to give clusters without any knowledge of how many clusters there are or what shape they are. It does this by iteratively expanding clusters from starting points in sufficiently dense regions. Outliers are just the points that are in sparsley populated regions (as defined by the eps and minPoints parameters).

- In practice, it takes some care to choose parameters that won't include those outliers. If they are included in clusters they often act as a bridge between clusters and cause them to merge together into an analytically useless blob.