**Types of Data**: Record data, Graphs and Networks, Ordered Data, (Spatial, image and multimedia Data)

**Types of Attributes**: Nominal, Binary, Ordinal, Numeric

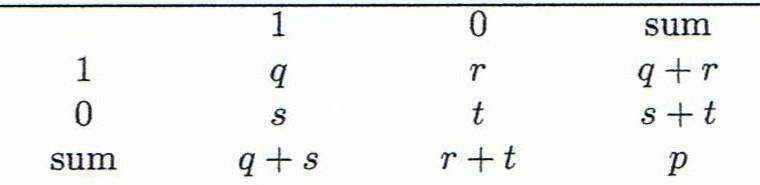
Graphic Displays: Boxplot, Histogram, Quantile plot, Quantile-quantile plot, Scatter plot

**Data Visualization methods**: Pixel-oriented, Geometric projection (Landscapes, Direct visualization, Prosection views, Hyperslice, Parallel coordinates), Icon-based, Hierarchical (Tree-Map, Cone Trees, InfoCube, Dimensional Stacking, Worlds-within-Worlds ), Visualizing complex data and relations

**Similarity, Dissimilarity, and Proximity**:

**Z-score**: 

**Minkowski**: 

**Proximity (for Binary Attributes)**: 

for symmetric: r+s/q+r+s+t

for asymmetric(distance): r+s/q+r+s

**Jaccard**: q/q+r+s

**Proximity (for Categorical Attributes)**:

Simple matching: m: # of matches, p: total # of variables

p-m/p

for Ordinal Variables: eg. freshman: 0; sophomore: 1/3; junior: 2/3; senior 1 Then distance: d(freshman, senior) = 1, d(junior, senior) = 1/3

Cosine: 

**Correlation and distance**:

**Χ2 (chi-square**)(Categorical Data)

**Covariance** (Numerical Data):

eg. (2, 5), (3, 8), (5, 10), (4, 11), (6, 14)

E(X1) = (2 + 3 + 5 + 4 + 6)/ 5 = 20/5 = 4

E(X2) = (5 + 8 + 10 + 11 + 14) /5 = 48/5 = 9.6

σ12 = (2×5 + 3×8 + 5×10 + 4×11 + 6×14)/5 − 4 × 9.6 = 4

**Correlation Coefficient**: measure of the linear dependence between two variables X and Y, Correlation coefficient value range: [–1, 1]; 1 is total positive linear correlation, 0 is no linear correlation, −1 is negative linear correlation;Pearson's correlation: 协方差/标准差的积

**Data Reduction**: Regression and Log-Linear Models; Histograms, clustering, sampling; Data cube aggregation; Data compression

For Non-parametric: histograms, clustering, sampling

For Parametric: Regression

**Data Transformation**: Smoothing; Attribute/feature construction; Aggregation; Normalization (min-max, z-score, decimal scaling); Discretization

Min-max: 

Decimal scaling: 

Discretization Methods: Binning (Top-down split, unsupervised); Histogram (Top-down split, unsupervised); Clustering (Unsupervised, top-down split or bottom-up merge); Decision-tree (Supervised, top-down split); Correlation (Unsupervised, bottom-up merge)

Dimensionality Reduction: Feature selection; extraction

Methods: PCA, Attribute Subset Selection/creation

**Data Warehouse**: Subject-Oriented; Integrated; Time Variant; Nonvolatile;

**Data Cube**: Suppose it contains only 2 base cells: {(a1, a2, a3, …., a100), (a1, a2, b3, …, b100)}

Non-empty aggregate cells if “having count >= 1”:

Each will generate 2^{100} - 1 non-base cells and 4 cells are overlapped and thus minus 4 so we get: 2\*2^{100} - 2 - 4 =  2\*2^{100} – 6.

“having count >= 2”: (a1, a2, \*, ..., \*): 2 (a1, \*, \*, ..., \*): 2 (\*, a2, \*, ..., \*): 2 (\*, \*, \*, ..., \*): 2

Closed Cube: {(a1, a2, a3 . . . , a100):10, (a1, a2, b3, . . . , b100):10} Answer:{(a1, a2, \*, …, \*): 20, (a1, a2, a3 . . . , a100): 10, (a1, a2, b3, . . . , b100): 10}

Multi-Way Array Aggregation: 4x4x4 chunks and A: 40, B: 400, C: 4000

Total memory required: 100×1000 + 40×1000 + 40×400

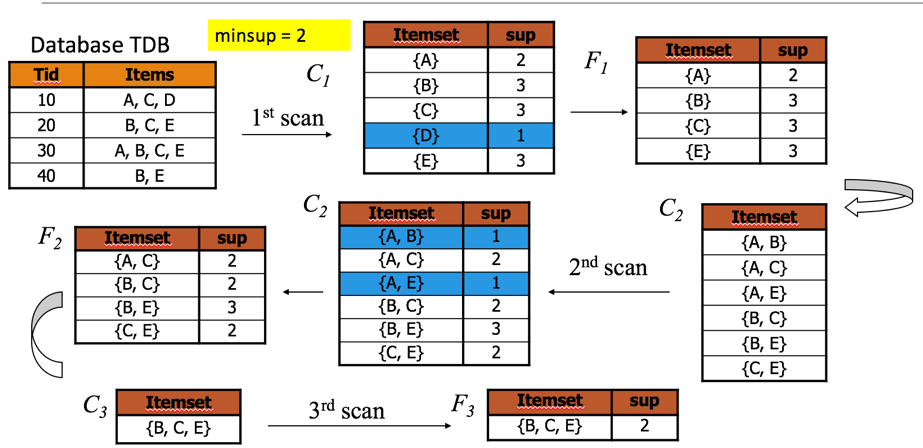
minimum memory occupation: 100 \* 1000, keep the result for one plane in memory

Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes’ space requirement is: 

Transaction DB TDB1: T1: {a1, …, a50}; T2: {a1, …, a100}

Closed patterns (lossless compression): Two: P1: “{a1, …, a50}: 2”; P2: “{a1, …, a100}: 1”

Max patterns (lossy compression): One: P: “{a1, …, a100}: 1”

**Apriori**: Any subset of a frequent itemset must be frequent

Algorithm: Initially, scan DB once to get frequent 1-itemset; Repeat: Generate length-(k+1) candidate itemsets from length-k frequent itemsets; Test the candidates against DB to find frequent (k+1)-itemsets; Set k := k +1; Until no frequent or candidate set can be generated; Return all the frequent itemsets derived

Generate candidates: self-joining Fk then pruning

Pseudo-Code

Ck: Candidate itemset of size k

Fk : Frequent itemset of size k

K := 1;

Fk := {frequent items}; // frequent 1-itemset

While (Fk != ) do { // when Fk is non-empty

Ck+1 := candidates generated from Fk; // candidate generation

Derive Fk+1 by counting candidates in Ck+1 with respect to TDB at minsup;

k := k + 1

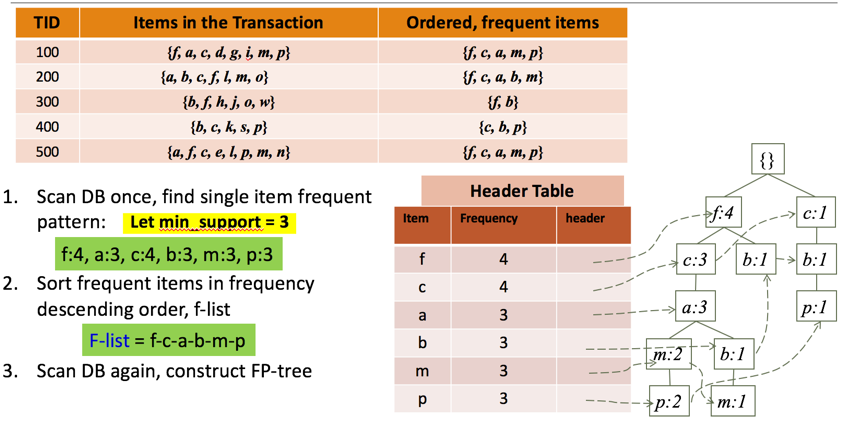
}

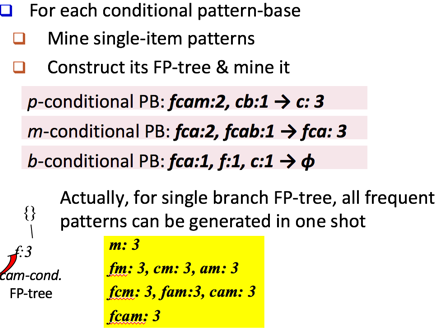
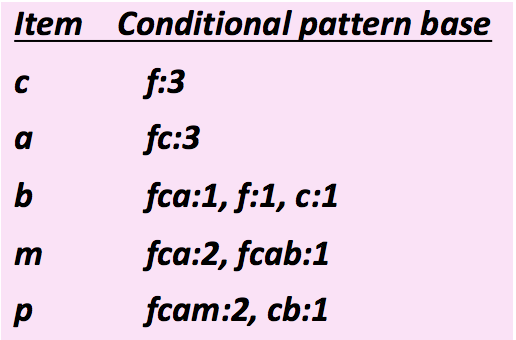
return k Fk // return Fk generated at each level

Partitioning: Scan Database Only Twice---Theorem: Any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB---Scan 1: Partition database and find local frequent patterns; Scan 2: Consolidate global frequent patterns

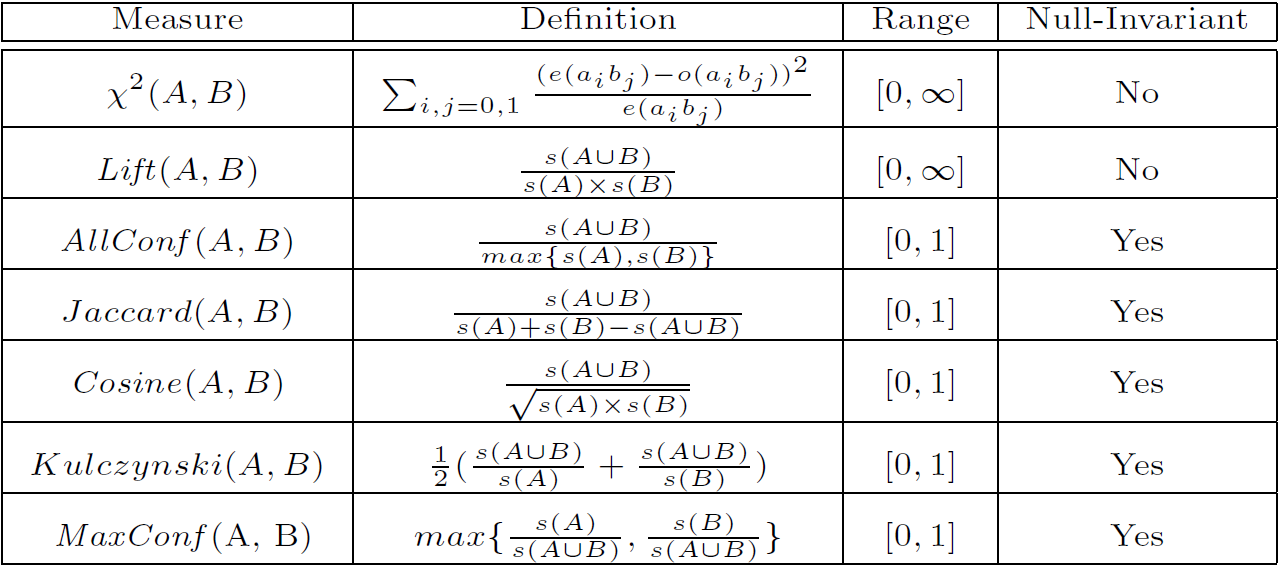
**FPGrowth**: Find frequent single items and partition the database based on each such item; Recursively grow frequent patterns by doing the above for each partitioned database (also called conditional database); To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed.

Mining: Recursively construct and mine (conditional) FP-trees; Until the resulting FP-tree is empty, or until it contains only one path





**Interestingness measures**:





Lift(B, C) = 1: B and C are independent; >1: positively correlated; <1: negatively correlated

Chi-square: 

=0: independent; >0 : correlated, either positive or negative, so it needs additional test

Lift, χ2 and cosine are good measures if null transactions are not predominant; Otherwise, Kulczynski + Imbalance Ratio should be used to judge the interestingness of a pattern

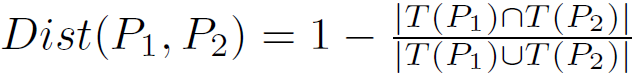
**Mining Diverse Patterns**:

Multiple-Level Associations: Use the lowest min-support to pass down the set of candidates

Multi-Dimensional Associations: Single-dimensional (buys(X, “milk”) buys(X, “bread”)); Inter-dimension (age(X, “18-25”) occupation(X, “student”) buys(X, “coke”)); Hybrid-dimension (age(X, “18-25”) buys(X, “popcorn”) buys(X, “coke”))

Quantitative Associations: Data cube-based aggregation

Negative Correlations: If є = 0.01, we have (P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < є

Compressed and Redundancy-Aware Patterns: 

**Pattern space pruning constraints**:

**Anti-monotonic**: If constraint c is violated, its further mining can be terminated

**Monotonic**: If c is satisfied, no need to check c again **Succinct**: if the constraint c can be enforced by directly manipulating the data

**Convertible**: c can be converted to monotonic or anti-monotonic if items can be properly ordered in processing

**Data space pruning constraints**:

**Data succinct**: Data space can be pruned at the initial pattern mining process

**Data anti-monotonic**: If a transaction t does not satisfy c, then t can be pruned to reduce data processing effort

**Mining Long Patterns**: No matter searching in breadth-first (e.g., Apriori) or depth-first (e.g., FPgrowth), if we still adopt the “small to large” step-by-step growing paradigm, we have to examine so many patterns, which leads to combinatorial explosion!

Pattern fusion strategy: Fuse small patterns together in one step to generate new pattern candidates of significant sizes

**Core patterns** of a colossal pattern α: A set of subpatterns of α that cluster around α by sharing a similar support

**Pattern-Fusion Algorithm**:

Initialization (Creating initial pool): Use an existing algorithm to mine all frequent patterns up to a small size, e.g., 3; Iteration (Iterative Pattern Fusion): At each iteration, K seed patterns are randomly picked from the current pattern pool For each seed pattern thus picked, we find all the patterns within a bounding ball centered at the seed pattern All these patterns found are fused together to generate a set of super-patterns All the super-patterns thus generated form a new pool for the next iteration; Termination: when the current pool contains no more than K patterns at the beginning of an iteration

**Sequential pattern mining**: Given a set of sequences, find the complete set of frequent subsequences (i.e., satisfying the min\_sup threshold)

**Graph Pattern Mining**: Given a labeled graph dataset D = {G1, G2, …, Gn}, the supporting graph set of a subgraph g is Dg = {Gi | g Gi, Gi D}; Apriori (FSG) (new edge) vs. pattern growth ( gSpan) (new vertex)

|  |  |  |
| --- | --- | --- |
| **TP** | **FN** | **P** |
| **FP** | **TN** | **N** |
|  |  | **ALL** |

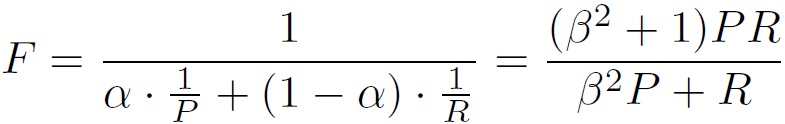
**Classification**

**Confusion Matrix:**

**Accuracy:**(TP+TN)/ALL **ERROR:**1-ACC

**Sensitivity = TP/P Specificity = TN/N**

**Precision = TP/(TP+FP) Recall=TP/(TP+FN)**

***F* measure = **

**Holdout method：**Given data is randomly partitioned into two independent sets

Training set (e.g., 2/3) for model construction, Test set (e.g., 1/3) for accuracy estimation

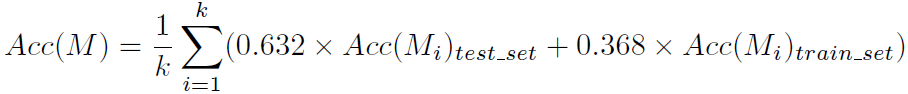
Random sampling: a variation of holdout

Repeat holdout k times, accuracy = avg. of the accuracies obtained

**Cross-validation (k-fold, where k = 10 is most popular):**

Randomly partition the data into k mutually exclusive subsets, each approximately equal size; At i-th iteration, use Di as test set and others as training set; Leave-one-out: k folds where k = # of tuples, for small sized data; \*Stratified cross-validation\*: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

**Bootstrap:** A sampling of the given data set with replacement. Works well with small data sets**;** Samples the given training tuples uniformly with replacement**;** Each time a tuple is selected, it is equally likely to be selected again and re-added to the training set

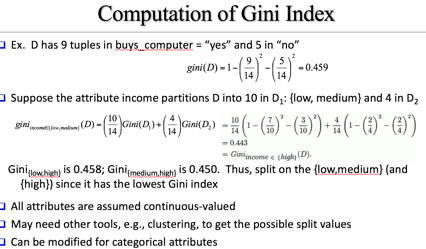
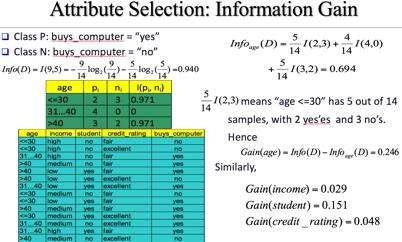
****

**ROC Curves:** Shows the trade-off between the true positive rate and the false positive rate; Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list; The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

**Issues Affecting Model Selection:** Accuracy (classifier accuracy: predicting class label); Speed(training time, classification/prediction time); Robustness**:** handling noise and missing values; Scalability**:** efficiency in disk-resident databases**;** Interpretability: understanding and insight provided by the model

**Entropy:** 

Higher entropy → higher uncertainty; Lower entropy → lower uncertainty



**Comparing Attribute Selection Measures**: Information gain: biased towards multivalued attributes; Gain ratio: tends to prefer unbalanced splits in which one partition is much smaller than the others;

Gini index: biased to multivalued attributes, has difficulty when # of classes is large, tends to favor tests that result in equal-sized partitions and purity in both partitions; CHAID: a popular decision tree algorithm, measure based on χ2 test for independence; C-SEP: performs better than info. gain and gini index in certain cases; G-statistic: has a close approximation to χ2 distribution; MDL: The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree; Multivariate splits (partition based on multiple variable combinations): CART: finds multivariate splits based on a linear comb. of attrs

**Overfitting:** An induced tree may overfit the training data: (1) Too many branches, some may reflect anomalies due to noise or outliers (2) Poor accuracy for unseen samples

**Avoid overfitting:** Prepruning: Halt tree construction early ̵ do not split a node if this would result in the goodness measure falling below a threshold. But Difficult to choose an appropriate threshold; Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees. Use a set of data different from the training data to decide which is the “best pruned tree”

**Ensemble methods:**

**Bagging**: averaging the prediction over a collection of classifiers; taking the majority vote/average

Analogy: Diagnosis based on multiple doctors’ majority vote

Training: Given a set D of d tuples, at each iteration i, a training set Di of d tuples is sampled with replacement from D (i.e., bootstrap); A classifier model Mi is learned for each training set Di

Classification: classify an unknown sample X; Each classifier Mi returns its class prediction; The bagged classifier M\* counts the votes and assigns the class with the most votes to X

Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple

Accuracy: Proved improved accuracy in prediction; Often significantly better than a single classifier derived from D; For noise data: not considerably worse, more robust

**Boosting**: weighted vote with a collection of classifiers; taking a weighted average based on the accuracy

Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy

How boosting works: Weights are assigned to each training tuple; A series of k classifiers is iteratively learned; After a classifier Mi is learned, the weights are updated to allow the subsequent classifier, Mi+1, to pay more attention to the training tuples that were misclassified by Mi; The final M\* combines the votes of each individual classifier, the weight of each classifier's vote is a function of its accuracy

Boosting algorithm can be extended for numeric prediction; Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

**Ensemble**: combining a set of heterogeneous classifiers

**Decision Tree:** 优:数据的准备不必要; 缺:决策树处理缺失数据时的困难, 过度拟合问题的出现,忽略数据集中属性之间的相关性

**RainForest:** 优:高维度,不用做特征选择;缺:过度拟合Comparable in accuracy to Adaboost, but more robust to errors and outliers; Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

**Bayesian Classification:** 优:参数少,对缺失数据不太敏感; Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

**Neural Network:** 优:分类的准确度高,鲁棒性和容错能力; 缺: 需要大量的参数, 不能观察之间的学习过程，输出结果难以解释, 时间长

**Support Vector Machines:** 优:小样本, 解决高维,非线性;缺: 对缺失数据敏感

**K-Nearest Neighbors:** 优:类域的交叉或重叠较多的待分样本,重新训练的代价较低; 缺:慢(lazy learing)

**Sequential Covering Method:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | B | ¬B | ∑row |
| C | 400 | 350 | 750 |
| ¬C | 200 | 50 | 250 |
| ∑col. | 600 | 400 | 1000 |

**Associative (Pattern-Based) Classification:**

**Genetic algorithm:** 优:多个个体的同时比较Robustness

**Distance-based methods**

Partitioning algorithms: K-Means, K-Medians, K-Medoids

Hierarchical algorithms: Agglomerative vs. divisive methods

**Density-based and grid-based methods**

Density-based: Data space is explored at a high-level of granularity and then post-processing to put together dense regions into an arbitrary shape

Grid-based: Individual regions of the data space are formed into a grid-like structure

**convergence criterion**: **Sum of Squared Errors (SSE):** ****

**K-Means**: Select K points as initial centroids; Form K clusters by assigning each point to its closest centroid, Re-compute the centroids (i.e., mean point) of each cluster; Until criterion is satisfied

Efficiency: O(tKn) where n: # of objects, K: # of clusters, and t: # of iterations, an efficient method

K-means clustering often terminates at a local optimal. Initialization is important to high-quality clusters

Need to specify K, the number of clusters, in advance. Runs some values and selected the “best” K value

Sensitive to noisy data and outliers, Variations: Using K-medians, K-medoids, etc

K-means is applicable only to objects in a continuous n-dimensional space, K-modes for categorical data

Not suitable to discover clusters with non-convex shapes; Use density-based clustering, kernel K-means

Cannot handle non-numerical (categorical) data

**Variations of K-Means:** better initial centroid estimates-> K-means++; different representative prototypes-> K-Medoids, K-Medians, K-Modes; feature transformation techniques-> Kernel K-Means

**K-Means++**: first centroid is selected at random, next centroid selected is the one that is farthest from the currently selected (selection is based on a weighted probability score) until K centroids are obtained

**K-Medoids**: Instead of taking the mean value of the object in a cluster as a reference point, medoids can be used, which is the most centrally located object in a cluster

PAM works effectively for small data sets but not for large data sets (due to the computational complexity)

Computational complexity: PAM: O(K(n − K)2) (quite expensive!)

**K-Medians (Handling Outliers problems)**: Medians are used (L1-norm as the distance measure)

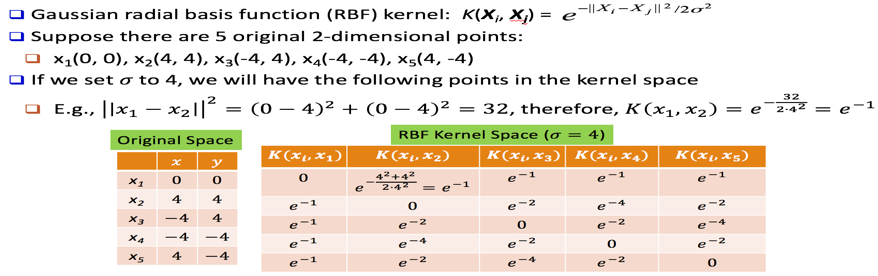
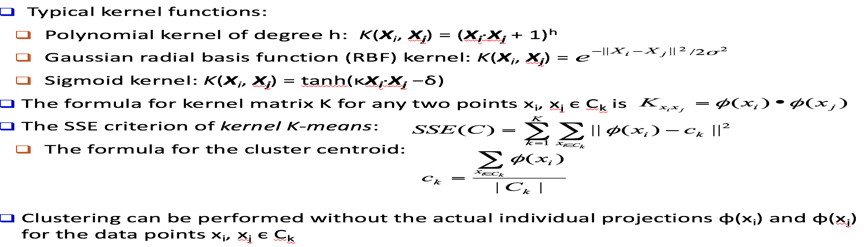
The criterion function for the K-Medians: 

**K-Modes (for Categorical Data)**: An extension to K-Means by replacing means of clusters with modes

This dissimilarity measure (Φ(xj, zj) = 1 – njr/nl when xj = zj ; 1 when xj ǂ zj where zj is the categorical value of attribute j in Zl, nl is the number of objects in cluster l, and njr is the number of objects whose attribute value is r) is frequency-based; A fuzzy K-Modes method is proposed to calculate a fuzzy cluster membership value for each object to each cluster; A mixture of categorical and numerical data: Using a K-Prototype method

**Kernel K-Means**: Project data onto the high-dimensional kernel space, then perform K-Means clustering

Kernel K-Means can be used to detect non-convex clusters; K-Means can only detect clusters that are linearly separable; Computational complexity is higher than K-Means; The widely studied spectral clustering can be considered as a variant of Kernel K-Means clustering



**Hierarchical Clustering**: Generate a clustering hierarchy (drawn as a dendrogram); Not required to specify K, the number of clusters; More deterministic; No iterative refinement

**Dendrogram**: Shows How Clusters are Merged, 类似于对阵图

**Agglomerative(AGNES)**: Start with singleton clusters, continuously merge two clusters at a time to build a bottom-up hierarchy of clusters: Use the single-link method and the dissimilarity matrix; Continuously merge nodes that have the least dissimilarity; Eventually all nodes belong to the same cluster

Agglomerative clustering varies on different similarity measures among clusters: Single link (nearest neighbor); Complete link (diameter); Single link (nearest neighbor); Complete link (diameter)

**Single link (nearest neighbor):** The similarity between two clusters is the similarity between their most similar (nearest neighbor) members; Local similarity-based: Emphasizing more on close regions, ignoring the overall structure of the cluster; Capable of clustering non-elliptical shaped group of objects; Sensitive to noise and outliers

**Complete link (diameter)**: The similarity between two clusters is the similarity between their most dissimilar members; Merge two clusters to form one with the smallest diameter; Nonlocal in behavior, obtaining compact shaped clusters; Sensitive to outliers

**Average link**: The average distance between an element in one cluster and an element in the other (i.e., all pairs in two clusters); Expensive to compute

**Centroid link (Group Averaged Agglomerative Clustering (GAAC))**: The distance between the centroids of two clusters; Let two clusters Ca and Cb be merged into CaUb; The new centroid is:Na is the cardinality of cluster Ca, and ca is the centroid of Ca;The similarity measure for GAAC is the average of their distances

**Ward’s criterion:** The increase in the value of the SSE criterion for the clustering obtained by merging them into Ca U Cb: ****

**Divisive(DIANA)**: Start with a huge cluster, split it into two groups, generating top-down hierarchy of clusters; The process starts at the root with all the points as one cluster; It recursively splits the higher level clusters to build the dendrogram; Can be considered as a global approach; More efficient when compared with agglomerative clustering

Choosing which cluster to split: Check the sums of squared errors of the clusters and choose the one with the largest value; Splitting criterion: Determining how to split: One may use Ward’s criterion to chase for greater reduction in the difference in the SSE criterion as a result of a split; For categorical data, Gini-index can be used; Handling the noise: Use a threshold to determine the termination criterion (do not generate clusters that are too small because they contain mainly noises)

weaknesses of hierarchical: never undo; not scale well: Time complexity of at least O(n2)

Other hierarchical clustering algorithms:

**BIRCH**: Use CF-tree and incrementally adjust the quality of sub-clusters; Birch is a hierarchical clustering algorithm that makes the space of our clustering "smaller" by representing data points as the summary characteristics of those points. Birch has some limitations, namely in that to create clusters it works in terms of circles and, if a cluster isn't spherical, birch has a sad time.

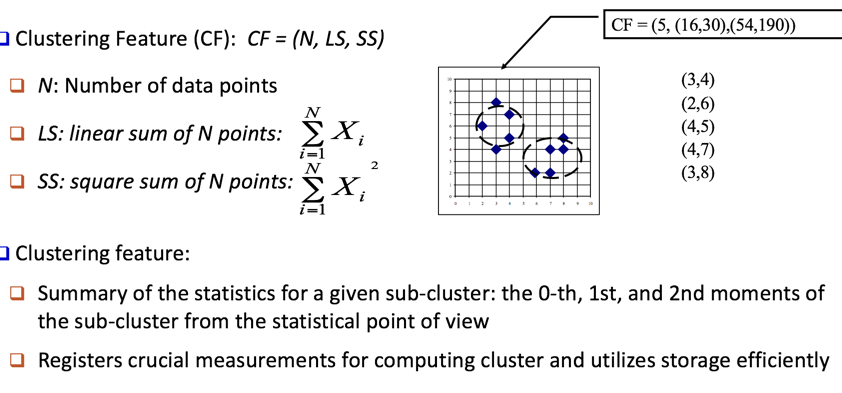
**CURE**: Represent a cluster using a set of well-scattered representative points

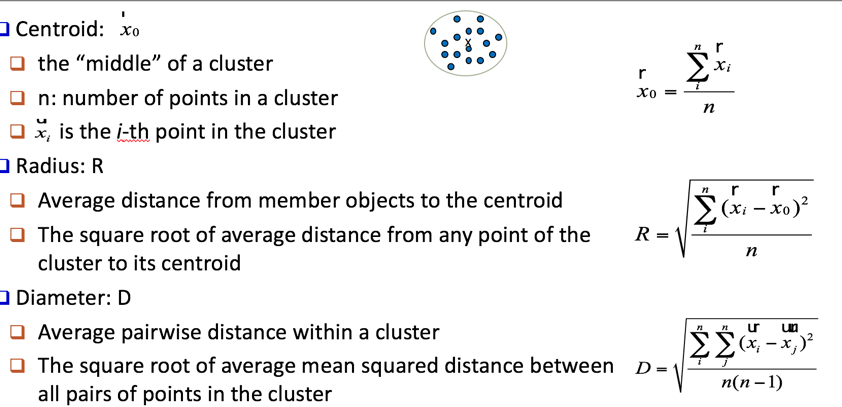
**CHAMELEON**: Use graph partitioning methods on the K-nearest neighbor graph of the data; Chameleon is a good algorithm for finding arbitrarily shaped clusters of high quality, but if you have a large number of data points it is a sad time for computation

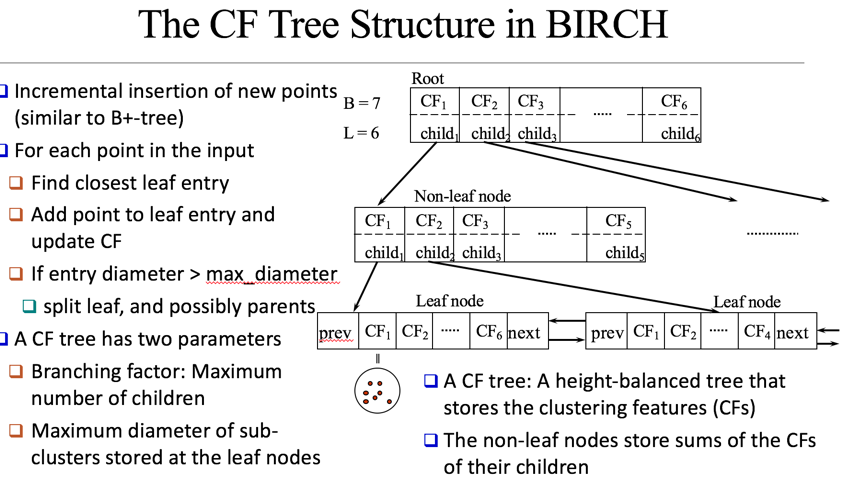
**BIRCH**: Scan DB to build an initial in-memory CF tree (a multi-level compression of the data that tries to preserve the inherent clustering structure of the data); then, Use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree

Key idea: Multi-level clustering; Low-level micro-clustering: Reduce complexity and increase scalability; High-level macro-clustering: Leave enough flexibility for high-level clustering

Scales linearly: Find a good clustering with a single scan, improve the quality with a few additional scans







An integration of agglomerative clustering with other (flexible) clustering methods:

Low-level micro-clustering: Exploring CP-feature and BIRCH tree structure; Preserving the inherent clustering structure of the data

Higher-level macro-clustering: Provide sufficient flexibility for integration with other clustering methods

Concerns:Sensitive to insertion order of data points; Due to the fixed size of leaf nodes, clusters may not be so natural; Clusters tend to be spherical given the radius and diameter measures

**CURE (Clustering Using REpresentatives)**: Represent a cluster using a set of well-scattered representative points

Cluster distance: Minimum distance between the representative points chosen

Shrinking factor α: The points are shrunk towards the centroid by a factor α; Far away points are shrunk more towards the center: More robust to outliers

**CHAMELEON**: Two clusters are merged only if the interconnectivity and closeness (proximity) between two clusters are high relative to the internal interconnectivity of the clusters and closeness of items within the clusters; 1. Use a graph-partitioning algorithm: Cluster objects into a large number of relatively small sub-clusters; 2. Use an agglomerative hierarchical clustering algorithm: Find the genuine clusters by repeatedly combining these sub-clusters; CHAMELEON is capable to generate quality clusters at clustering complex objects

**Algorithmic hierarchical clustering**: Nontrivial to choose a good distance measure; Hard to handle missing attribute values; Optimization goal not clear: heuristic, local search

**Probabilistic hierarchical clustering**: Use probabilistic models to measure distances between clusters; Generative model: Regard the set of data objects to be clustered as a sample of the underlying data generation mechanism to be analyzed; Easy to understand, same efficiency as algorithmic; agglomerative clustering method, can handle partially observed data; a powerful choice if we think there is a prior for the distribution of points.

**Density-Based Clustering Methods:** Discover clusters of arbitrary shape; Handle noise; One scan (only examine the local region to justify density); Need density parameters as termination condition

**DBSCAN**: Sensitive to the Setting of Parameters

**OPTICS:** The cluster-ordering contains information equivalent to the density-based clusterings corresponding to a broad range of parameter settings; Good for both automatic and interactive cluster analysis—finding intrinsic, even hierarchically nested clustering structures; OPTICS or DBSCAN if I'm looking for dense clusters

**Grid-Based Clustering Methods** : Efficiency and scalability: # of cells << # of data points; Uniformity: Uniform, hard to handle highly irregular data distributions; Locality: Limited by predefined cell sizes, borders, and the density threshold; Curse of dimensionality: Hard to cluster high-dimensional data

**STING**: Advantages:Query-independent, easy to parallelize, incremental update; Efficiency: Complexity is O(K); K: # of grid cells at the lowest level, and K << N (i.e., # of data points); Disadvantages: Its probabilistic nature may imply a loss of accuracy in query processing; Sting is a very good clustering tool if our data is primarily discrete and we're looking for interesting clusters of individuals (like a data cube). Sting's biggest problem is that it's weakness is how granular your data is, and that shapes are isothetic, that is, all the cluster boundaries are horizontal or vertical, which may or may not be too useful.

**CLIQUE**: Strengths: Automatically finds subspaces of the highest dimensionality as long as high density clusters exist in those subspaces; Insensitive to the order of records in input and does not presume some canonical data distribution; Scales linearly with the size of input and has good scalability as the number of dimensions in the data increases ------Weaknesses: As in all grid-based clustering approaches, the quality of the results crucially depends on the appropriate choice of the number and width of the partitions and grid cells

|  |  |
| --- | --- |
| **Method** | **General Characteristics** |
| Partitioning Methods | - Find mutually exclusive clusters of **spherical** shape,  - Distance based  - May use mean or mediod (etc) to represent cluster center - Effective for small - to medium-size data sets |
| Hierarchical Methods | -Clustering is a hierarchical decomposition (i.e. multiple levels)  -Cannot correct erroneous merges or splits  -May incorporate other techniques like microclustering or consider object "linkages" |
| Density-based methods | -Can find arbitrarily shaped clusters  -Clusters are dense regions of objects in space that are separated by low-density regions  -Cluster density: Each point must have a minimum number of points within its "neighborhood"  -May filter out outliers |
| Grid-based methods | -Use a multiresolution grid data structure -Fast processing time (typically independent of the number of data objects, yet dependent on grid size) |