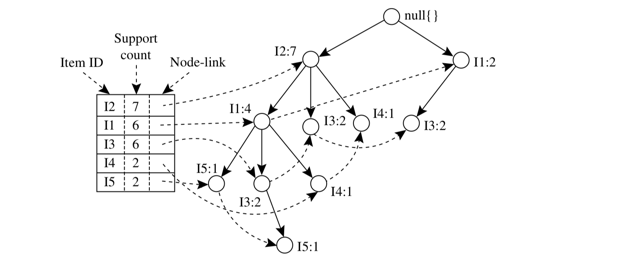
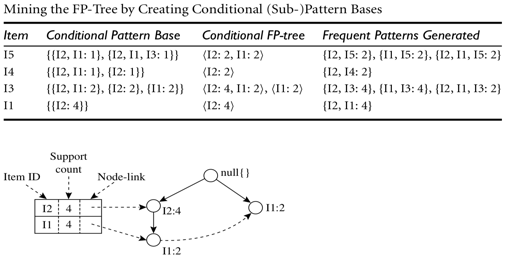
**1**: **DATA MINING**: discovering interesting (certainty, novel, useful) patterns from massive amounts of data, has arisen from need and ability: data being collected on devices, cost to store data, processing speeds. It incorporates techniques from other domains (stats, ML, DB), focuses on scalability and flexibility. **Steps**: cleaning (remove noise), integration (multiple sources), selection (relevant), transformation (consolidating), pattern discovery (methods), evaluation (interesting->knowledge), presentation (visual). **Functionalities** (patterns): *descriptive, supervised* [characterizations (about a class), discrimination (features of class vs. class)], *predictive, supervised* [association and correlation (occurs together), classification (with labels – predict class), regression (predict value)], *descriptive, unsupervised* [cluster (group, no labels)] outliers: detected through cluster **2: DATA:** collection of objects and their **attributes**: nominal (distinctness (=, !=)), binary, ordinal, (+order (<, >)), numeric interval (differences (+, -)), ratio (ratios (\*, /)) **stats**: central (mean, median, mode, midrange), dispersion (range, quartiles, IQR); 5 num sum: min, Q1, Q2, Q3, max, boxplot: same but plots outliers >1.5 IQR, var: ) - **data display**: univariate: Q-plot: Sort in ascending order, f-value = (i-0.5)/N. Plot x = f-value, y = data; Q-Q-plot, plot f vs. f; histogram: distribution of an attribute within bins; bivariate: scatter plot, correlation, provide insight into overall behavior of data, useful identifying noise, cleaning. **Similarity**: multiple attributes: data matrix (n objects x p attributes); dissimilarity matrix (collection of proximities for each pair of n objects); *nominal*: p= attributes, m= matches; *binary*: symmetric: , asymmetric , Jaccard: , good for medical tests, contingency table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Object j | | |
| Obj i |  | 1 | 0 |
| 1 | **q** | **r** |
| 0 | **s** | **t** |

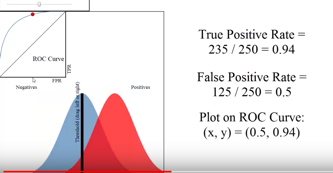
*numeric*: : Manhattan: , Euclidean: , Minkowski: generalized, supreme:/Chebyshev ; *ordinal*: first convert to scale, then use a distance formula; sparse data sets (text analysis, gene feature mapping): use cosine similarity/Tanimoto dist, , [0=dissimilar,1=similar], normalize: divide each component of vector by the distance of the vector, correlation: linear relationship , , entropy: range: [0,], 0= certain, 1= uncertain **3. PREPROCESSING:** quality: accuracy, completeness, consistency, timeliness, believability, interpretability; *1.data cleaning*: missing values (Ignore it, fill it in manually, use a global constant to fill it, use a central tendency for the attribute, use a central tendency for the class, use probable value) 3-6 bias the data, 6 is preferred); noisy data (Smoothing by means: each value replaced by the mean of the bin, by medians, by boundaries: replaced by max or min of the bin, Regression: find best line to fit the data, Outlier analysis: use cluster analysis), tools can help with this process, *2.data integration:* merging various sources (entity identification problem) entity integration, metadata can help clear confusion, redundancy (annual and monthly revenue) can be detected with correlation analysis, chi-sq for nominal data, correlation analysis for numeric, chi square: build a contingency table

|  |  |  |  |
| --- | --- | --- | --- |
|  | m | f | t |
| fiction | 250 (90) | 200 (360) | 450 |
| Non-fic | 50 (210) | 1000 (840) | 1050 |
|  | 300 | 1200 | 1500 |

of actual vs. expected (, sum all cells (act – exp)^2 / exp df = (r-1)\*(c-1)=1, the null hypothesis is that the two are independent, if above the threshold, reject the null and conclude the two are correlated. Correlation: , *3.data reduction:* dimensionality reduction: high d: data becomes sparse and definitions of density become less meaningful Purpose of reducing dimensionality: avoid curse of dimensionality, reduce time and memory to mine, allow data to be visualized. a. wavelet transformations: vector is transformed to wavelets, data is truncated to keep the strongest, more suitable for higher-d data; b. PCA: find a lower-dimensional projection that captures the largest amount of variation in the data. Input data are linear combinations of the principal components, better for sparse data; c. attribute subset selection: Remove redundant features, irrelevant features, especially for classification, trim based on information gain; d. decision tree, numerosity reduction: a. regression and log-linear models: model data, both can be used on sparse data, regressions especially good with skew, l-l good for higher dimensions, b. histograms: partition the data into bins: equal-width: same size bins, equal-freq: same number per bin, good for sparse and dense data, skew or uniform data, up to dimension 5; c. Clustered data, d. sampling: Processing the entire data set is too expensive or time consuming, sample must be representative, simple random sampling w/ or w/out replacement, cluster sample, stratified sampling: ensures a representative sample when data is skewed. Benefit: sublinear time, 4. *Data transformation:* smoothing: binning, regression, clustering/outlier; attribute construction: new attributes are constructed from set, aggregation: from a data cube, normalization: min-max: range: , z-score: range: , mean absolute deviation replace mu with, by decimal scaling: range: (-1, 1), ; discretization: converting a continuous attribute into an ordinal one, automatically generate concept hierarchies, Commonly used in classification, most work best with only a few possible variables, binning, histograms, cluster, decision tree, for nominal data: concept hierarchies from the values per attribute. **6.** **FREQUENT PATTERN** a pattern (set of items, subsequences, substructures) that occur frequently in a data set. , , strong assoc= min\_sup, min\_conf. *closed itemset*: X is frequent and there exists no super-pattern Y ⊃ X with the same support as X. Y must have the same support as X, this is not a lossy compression; *maximal itemset*: X is frequent and there exists no super-pattern Y ⊃X and Y is frequent. Y only has to be frequent, this is a lossy compression, doesn't contain complete information; Downward closure property of frequent patterns: any subset of a frequent itemset must be frequent. Apriori: 1. get all frequent 1-itemsets 2. Generate candidates of length (k+1) from L1 3. Prune the itemsets that are not frequent 4. Generate candidates of length (k+1) if (k-1) = (k-1) 5. Look at all (k-1) subsets of the candidate and prune any that are not frequent 6. Scan for count of itemset length k and repeat until empty. 7. Generate association rules: 1. For each freq-itemset, generate all subsets, 2. For every subset “s ” if Problems: multiple scans of the database, many candidates, support counting, improvements: hash tree, transaction reduction, partitioning, sampling; FP-Growth: Depth first search, Avoids candidate generation, Grow long patterns from short ones using locally frequent items. 1. Find freq 1 itemset 2. order it, 3. scan each transaction and create a tree based on the sorted list. 4. Create conditional pattern bases from the paths up the tree 5. For each pattern base, create a conditional FP Tree 6. From the conditional FP Tree, generate f-patterns. Benefits of the FP Tree: Completeness, compactness, Divide/ conquer:Decompose the mining task and DB according to the FP obtained so far,

Lead to focused search of smaller databases. Other improvements; mine closed patterns, item merging, sub-itemset pruning, item skipping, hashing, linear data. Pattern evaluation: filter out uniteresting rules. , <1= negative correlation, >1 = positive correlation. 4 others: all\_confidence, max\_confidence, Kulczunski, cosine, these are all null-variant so they work well. **CHAPTER 8: CLASSIFICATION** training data with labels to predict category of new data. Model construction: use tuples to train, model usage: classify unknown. *1. Decision tree:* greedy algorithm, top down divide and conquer method. Flowchart like structure every node represents a test of an attribute -> outcome. appropriate for exploration: don't require any domain knowledge or parameter setting, simple and intuitive, frequently used in medicine and astronomy. O(n x |D| x log(|D|)). Use Information gaid to determine split attributes: start with the node of all data, calculate the entropy, calculate weighted average entropy after split of each attribute, split on max information gain. Problem: tends to favor splits with a higher number of values. Gain ratio normalizes with split information Gini measures binary split for each attribute and measures impurity. Trees prone to overfitting: prepruning: halt if below threshold, postpruning: remove after fully grown, scaling: problems with fitting in memory: RainForest: builds (attribute, value, class\_label), matrix: count of value (high/low) /class\_label (play/no) for each attribute (wind)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | PREDICTED | | |
| ACT |  | Y | N | t |
| Y | TP | FN | P |
| N | FP | TN | N |
|  | P’ | N’ | P+N |

BOAT bootstraps smaller trees into a bigger tree, *2. Bayes*: what is the probability someone will buy a computer, given they are middle age, medium income, etc. maximize: ; 1. Calc P(C) for each class (outcome), for each attribute of X, calculate P(X|C1) and P(X|C2), multiply them together to get P(**X**|C1,2), multiply by P(C) to determine most likely class. Assumes class-conditional independence, which simplifies the calculation (why it's called naive), It is a statistical classifier - predicts class membership probabilities, Incremental: each training set can increase/decrease the probability that a hypothesis is correct. Disadvantages: requires initial knowledge of many probabilities, assumes conditional independence. *3. Rules-based*: *Testing model:* use testing data. confusion matrix, Accuracy (balanced class): % correctly classified = ( ; Error rate: misclassification: 1- accuracy = , Imbalanced class: Sensitivity (cancer): , Specificity (spam): , Precision: measure of exactness, what % of tuples that the classifier labeled as positive actually are: Recall: measure of completeness, what % of positive tuples did the classifier label as positive: precision/recall inverse, combine them in F: statistical test can be done to compare if error in model is statistically significant. Null= M1 and M2 are the same. ROC: compare models based on tradeoff of TP/FP for all thresholds. Higher AUC = better model. low threshold = more false positives (maximize true positive), vs. min false positive (spam). Issues affecting model selection: accuracy, speed, robustness, scalability, interpretability, improve classification accuracy: ensemble methods, combining models, bagging, boosting, random forest, improve accuracy less impacted by outliers. **CHAPTER 10: CLUSTER**: data objects similar to each other, dissimilar to other groups, unsupervised learning, can be used as a stand alone method or as a preprocessing technique, good: cohesive within, distinctive, Considerations: single level vs. hierarchical, completely separate or allow for overlap, how to define similarity, subspace clustering, 1. Partitioning methods: distance based, uses iterative relocation techniques to move objects from one partition to another. Works well finding spherical clusters in small-mid sized databases, *K-means*: partition objects into k groups, then repeat: compute the centroid, assign the data points to the nearest centroid. until no change strengths: efficient O(tkn) weaknesses: often terminates at a local optimal point, only applicable to continuous n-dimensional space, need to specify k in efficient, sensitive to outliers, not good with non-convex shapes, sensitive to seed, *K-medoids (PAM)*: more complex, instead of taking mean, take center data point, partition the data, test a different non-medoid point, if it reduces the SSE, then reassign the medoid. PAM works well for small data sets, but it is not efficient for large data sets, CLARA is PAM but using samples; Hierarchical Methods: Divides data into a tree of clusters, does not require K as an input, more efficient, arbitrary shapes, Agglomerative: starts with the individual AGNES: merge nodes that are the nearest neighbor single link (least dissimilarity) vs. Divisive: starts as one cluster DIANA: opposite of AGNES, start as one, Similarity is measured as the closest pair between the two clusters Measures of distance between clusters: *single link*: nearest neighbor, minimum distance *complete link*: maximum distance, largest distance from one element to another *average*: avg distance between an element in one cluster and element in another *centroid*: distance between centroids, *medoid*: distance between medoids; Major weaknesses: cannot be undone and does not scale well, BIRCH overcomes these two weaknesses, only handles numeric data uses clustering feature to summarize the cluster, and CF tree to represent the hierarchy CF = <n,LS,SS> Can find the centroid, radius, diameter; Chameleon: graph based and two-phase: use graph partitioning algo to cluster objects in small sub-clusters use agglomerative hierarchical clustering algo to find genuine clusers by combining sub clusters used for clustering complex objects, however the processing cost is high 0(n^2) Issues with hierarchical clustering methods: choosing a good distance measure is nontrivial, cannot have any missing attribute values, optimization goal not clear: local search Probabilistic clustering: aims to overcome these using probabilistic models to measure distances between cluster use a generative model, assume the data set adopts a common distribution function, find Mu and sigma that maximum likelihood the set of data points is generated same efficiency, but can handle missing data; Density based: Clustering based on density (local clustering) Discover clusters of arbitrary shape, Can handle noise, One scan, but needs a termination condition DBSCAN: discover clusters of arbitrary shape, cluster: max set of density connected-points; EPS: max radius of the neighborhood; MinPts: min points in Eps neighborhood If it is not a core point but can be reached it is a boarder point; Directly density reachable: if it belongs to the neighborhood of a core point; Density reachable: chain regions to reach Computational complexity if spatial index is used, O(n log n) otherwise, O(n^2) sensitive to setting of parameters Algo: select point p, retrieve all points density reachable from p wrt eps and minPts. if core, cluster is formed, else, visit next point OPTICS: extends DBSCAN but much less sensitive to parameter setting Density based clusters are monotonic with respect to the neighborhood threshold The deeper the valley, the denser the cluster, reachability plot core distance: the smallest distance to cover e points reachability distance: min radius that makes p density reachable from q max(core-distance, distance(q,p)) O(n log n) if indexed; Grid based: partition the data space into cells to form a grid structure, when you find a dense region in the cells, Efficient and scalable, uniform but hard to handle irregular distributions, locality, curse of dimensionality STING: Spacial area is divided into rectangular cells at different levels of resolution to form a tree structure Statistical measures for each cell. Calculate the likelihood the cell is relevant at some confidence level, only children at the relevant cells are explored Query independent, complexity is O(K), k<< N disadvantage: probabilistic nature may imply a loss of accuracy for query processing CLIQUE: density based and grid based, connect dense units into a cluster. Starts from a low dimension Start in 1D, Find dense region in each subspace and generate their minimal discriptions, then find promising candidates in 2D, repeat in levelwise manner in higher dimensional space in Apriori manner, then find the connected dense units Strengths: automatically finds subspaces of the highest dimensionality as long as high density clusters exist, insensitive to the order of records, scales linearly Weakness: quality of the data depends on the resolution of the grid *Assessing* if non-random structure exists in the data by measuring probability the data is generated by uniform data distribution Tested with Hopkins Statistic