scatter plot vs. boxplot: scatter plot: each pair of values is a pair of coordinates and plotted as points in the plane **boxplot**: uses min, Q1, median, Q3, max (or whiskers, and outliers) to describe data distribution/dispersion **qqplot:** comparing two data sets by plotting their distributions on two axes of one graph Show distribution shift between two data sets. **covariance vs. Pearson correlation coefficient**:Pearson correlation coefficient (X, Y) = covariance (X, Y) /σXσY (σ: standard deviation) **Principal Component Analysis (PCA) vs. feature selection** **Principal component analysis:** a dimension reduction method, the result is a set of orthogonal transformed features (i.e., principal components)**. feature selection**: select some important (existing) features from the given feature set.
wavelet transform vs. lossless data compression Wavelet transform splits data into average (smooth) and difference, transformation preserves shape. When it is used to compress data, only a fraction of strongest of the wavelet coefficients will be stored. Therefore, it is lossy data compression
icon-based techniques: chernoff faces; stick figures. (optional:) shape coding; color icons; tile bars. **geometric projection techniques**: direct visualization; scatterplot and scatterplot matrices; landscapes; parallel coordinates(optional:) projection pursuit techniques; prosection views; hyperslice
hierarchical techniques: dimensional stacking; worlds-within-worlds; tree-map; cone trees; infocube; tag cloud **histogram:** partitions data with equal width bins, and uses frequency across bins to describe data distributions/dispersion.**boxplot**: uses 5 numbers min, Q1, median, Q3, max (or whiskers, and outliers) to describe data distribution/dispersion. Sampling allows a large data set to be represented by a much smaller random sample (or subset) of the data
correlation analysis **by χ2 test vs. correlation analysis with Pearson correlation co- efficient** .χ2 test is used for judging correlation for categorical/nominal data, where Pearson correlation coefficient for numerical data; some may answer: their ranges is very different: the former: (−∞, +∞), whereas the latter: [−1, +1] .**PCA vs. wavelet transform PCA**: uses orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables(called principal components)**.wavelet transform**: splits data into average (smooth) and difference, transformation preserves shape .**stratified sampling vs. simple sampling** stratified sampling: partition the data into different set, e.g., age-group; and then sample on these sets separately;simple sampling: no partition; random sampling on the whole set **Jaccard coefficient**: comparing patients based on their multiple medical testing results (prin- ciple: asymmetric variables, e.g., in medical tests, usually positive results are much infrequent than negative ones). **Cosine**: similarity between text documents (principle: bag of words forming long vectors for cosine measure to test)**Covariance**: checking positive/negative correlations of numerical data. **Chi-square**:checking positive/negative correlations of categorical data **tag cloud**: visualizing text data, based on the relative word frequency; **worlds-within-worlds**: visualizing multidimensional (> 2D) data, fix several dimen- sion values, study how the measure distributes based on the values of the remaining 2-dimensions.
min-max: scale/normalize data to a specified (min, max) range.**z-score**: scale/normalize data based on $z = x-\mu$, where μ is mean and σ is standard deviation.**data reduction by parametric methods**: reduce data by using parameters to describe the data, e.g., regression model (results are parameters).**data reduction by non-parametric methods**: reduce data by using reduced data sets to describe data, e.g., feature selection, clustering. **(1). Lift**: Measure of dependent/correlated events . Lift(B, C) may tell how B and C are correlated Lift(B, C) = 1: B and C are independent > 1: positively correlated < 1: negatively correlated. **Kulczynski** holds firm and is in balance of both directional implications. **(2)** IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications: Lift, **(3)** χ^2 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 .The larger the χ^2 value, the more likely the variables are related
(5)Covariance: $E(X_1) = (2 + 3 + 5 + 4 + 6)/ 5 = 20/5 = 4$ $E(X_2) = (5 + 8 + 10 + 11 + 14) /5 = 48/5 = 9.6$ $\sigma_{12} = (2\times5 + 3\times8 + 5\times10 + 4\times11 + 6\times14)/5 - 4 \times 9.6 = 4$ **(4)Pearson correlation**: **Correlation** between two variables X_1 and X_2 is the standard covariance, obtained by normalizing the covariance with the standard deviation of each variable

$$lift(B,C) = \frac{c(B \to C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)} \quad (1)$$

$$\Sigma = E[(\mathbf{X} - \mu)(\mathbf{X} - \mu)^T] = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{pmatrix} \quad \text{(covariance matrix)}$$

$$IR(A,B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)} \quad (2) \qquad \sigma_{12} = E[X_1 X_2] - E[X_1]E[X_2] \quad (5) \qquad \rho_{12} = \frac{\sigma_{12}}{\sigma_1 \sigma_2} = \frac{\sigma_{12}}{\sqrt{\sigma_1^2 \sigma_2^2}} \quad (4) \qquad \chi^2 = \sum \frac{(Observed - Expected)^2}{Expected} \quad (3)$$

Closed patterns: A pattern (itemset) X is closed if X is *frequent,* and there exists *no super-pattern* Y ⊃ X, *with the same support* as X(lossless compression)**Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ⊃ X (lossy compression)**Apriori**: (incremental) (bfs)Any subset of a frequent itemset must be frequent. 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 (slow candidate generation is the bottleneck, requires many database scan.)**Downward closure**: any subset of a frequent itemset must be frequent**. FPGrowth**(incremental)(dfs) Find frequent single items and partition the database based on each such single item pattern. Recursively grow frequent patterns by doing the above for each *partitioned database* (also called the pattern's *conditional database*) .To facilitate efficient processing, an efficient data structure, FP-tree, can be constructed. What if FP-tree cannot fit in memory? Project" the database based on frequent single items, Construct & mine FP-tree for each projected DB, Parallel projection: Project the DB on each frequent item Space costly, all partitions can be processed in parallel. Partition projection: Partition the DB in order. Passing the unprocessed parts to subsequent partitions (scan twice). **PrefixSpan Mining**: Prefix Projections Step 1: Find length-1 sequential patterns Step 2: Divide search space and mine each projected DB .No candidate subseqs. to be generated. Projected DBs keep shrinking Cost: constructing DB. Swapping to pseudo-projection when the data fits in memory. **CloSpan**: A closed sequential pattern *s*: There exists no super pattern *s'* such that *s'* ⊃ *s*, and *s'* and *s* have the same support . Reduce # of (redundant) patterns. Attain the same expressive power A constraint *c* is ***anti-monotone*** If an itemset S **violates** constraint *c*, so does any of its superset That is, mining on itemset S can be terminated. Ex. 1: c_1: *sum(S.price)* ≤ *v* is anti-monotone. Ex. 2: c_2: range(S.profit) ≤ 15 is anti-monotone. *support(S)* ≥ *σ* anti-monotone? Yes! Apriori pruning is essentially pruning with an anti-monotonic constraint! A constraint *c* is ***monotone***: If an itemset S **satisfies** the constraint c, so does any of its superset That is, we do not need to check *c* in subsequent mining. Ex. 1: c_1: *sum(S.Price)* ≥ *v* is monotone Ex. 2: c_2: *min(S.Price)* ≤ *v* is monotone. Ex. 3: c_3: range(S.profit) ≥ 15 is monotone.A constraint c is ***data anti-monotone***: In the mining process, if a data entry *t* cannot satisfy a pattern *p* under *c*, *t* cannot satisfy *p*'s superset either. Data space pruning: Data entry *t* can be pruned Ex. 1: c_1: *sum(S.Profit)* ≥ *v* is data anti-monotone Let constraint c_1 be: *sum(S.Profit)* ≥ 25 T_{30}: {b, c, d, f, g} can be removed since none of their combinations can make an S whose sum of the profit is ≥ 25 Ex. 2: c_2: *min(S.Price)* ≤ *v* is data anti-monotone Consider *v* = 5 but every item in a transaction, say T_{10}, has a price higher than 10 Ex. 3: c_3: *range(S.Profit)* > *25* is data anti-monotone
Succinctness: If the constraint *c* can be enforced by directly manipulating the data Ex. 1: To find those patterns without item *i*. Remove *i* from DB and then mine (pattern space pruning). Ex. 2: To find those patterns containing item *i*. Mine only *i*-projected DB (data space pruning). Ex. 3: c_3: *min(S.Price)* ≤ *v* is succinct. Start with only items whose price ≤ v and remove transactions with high-price items only (pattern + data space pruning). Ex. 4: c_4: *sum(S.Price)* ≥ *v* is not succinct. It cannot be determined beforehand since sum of the price of itemset S keeps increasing **Convertible:** *c* can be converted to monotonic or anti-monotonic if items can be properly ordered in processing like avg >= <=. **Modeling data warehouses**: dimensions & measures Star schema: A fact table in the middle connected to a set of dimension tables. Snow flake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake. Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation **Distributive:** if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning E.g., count(), sum(), min(), max() **Algebraic**: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function E.g. avg(x) = sum(x) / count(x) , var, std (using count() and sum(x) and sum(x^2)Is min_N() an algebraic measure? Yes How about standard_deviation()? Yes. **Holistic:** if there is no constant bound on the storage size needed to describe a subaggregate. E.g., median(), mode(), rank() Q1, Q3 can not compute efficiently **Multiway array**: Array-based "bottom-up" algorithm (from ABC to AB,…) 1) Using multi-dimensional chunks 2)Simultaneous aggregation on multiple dimensions 3)Intermediate aggregate values are re-used for computing ancestor cuboids 3)Cannot do *Apriori* pruning: **No iceberg optimization** Efficient for computing the full cube for a small number of dimensions If there are **a large** number of dimensions, "top-down" computation and iceberg cube computation methods (e.g., BUC) should be used. Partition arrays into *chunks* (a small subcube which fits in memory). Compressed *sparse array addressing*: (chunk_id, offset). Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost **BUC**: Top down. Divides dimensions into partitions and facilitates iceberg pruning. **Shell fragment:** High dimension **Decision tree Induction**. Tree is constructed in a **top-down, recursive, divide-and-conquer manner.** At start, all the training examples are at the root. Examples are partitioned recursively based on selected attributes. On each node, attributes are selected based on the training examples on that node, and a heuristic or statistical measure (e.g., **information gain**). Conditions for stopping partitioning. All samples for a given node belong to the same class. There are no remaining attributes for further partitioning . There are no samples left . Prediction: **Majority voting** is employed for classifying the leaf **Entropy (Information Theory).** A measure of uncertainty associated with a random number. Calculation: For a discrete random variable Y taking m distinct values {y_1, y_2, …, y_m} Interpretation:Higher entropy → higher uncertainty. Lower entropy → lower uncertainty. **(1):** Expected information (entropy) needed to classify a tuple in D.**(2):** Information needed (after using A to split D into v partitions) to classify D. **(3):** Information gained by branching on attribute A. Information gain measure is biased towards attributes with a large number of values. **(4) Gain ratio**: Overcomes the problem (as a normalization to information gain). **GainRatio(A) = Gain(A)/SplitInfo(A):** The attribute with the maximum gain ratio is selected as the splitting attribute. **GINI Index:** The attribute provides the smallest $ginisplit(D)$ (or the largest reduction in impurity) is chosen to split the node.**Compare:** The three measures, in general, return good results but **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.(1) (2)

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \quad (3) \qquad Gain(A) = Info(D) - Info_A(D) \quad (4) \qquad SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

$$\text{gini}(D) = 1 - \sum_{i=1}^{n} p_i^2 \text{ , gini}_A(D) = \frac{|D_1|}{|D|}\text{gini}(D_1) + \frac{|D_2|}{|D|}\text{gini}(D_2), \Delta\text{gini}(A) = \text{gini}(D) - \text{gini}_A(D)$$

Overfitting: An induced tree may overfit the training data . Too many branches, some may reflect anomalies due to noise or outliers. Poor accuracy for unseen samples. Two approaches to avoid overfitting . Prepruning: Halt tree construction early- do not split a node if this would result in the goodness measure falling below a threshold. 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". **Decision tree** is Relatively fast learning speed . Convertible to simple and easy to understand classification rules. Easy to be adapted to database system implementations (e.g., using SQL). Comparable classification accuracy with other methods. **Naïve Bayes:** A statistical classifier. Perform probabilistic prediction (i.e., predict class membership probabilities). Foundation—Based on Bayes' Theorem. Performance: A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers. Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct—prior knowledge can be combined with observed data. **Theoretical Standard**: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measure. $\mathbf{p(H|X)} = \frac{\mathbf{p(X|H)\,p(H)}}{\mathbf{p(X)}} \propto \mathbf{p(X|H)\,p(H)}$
Practical difficulty of Naïve Bayes inference: It requires initial knowledge of many probabilities, which may not be available or involving significant computational cost. **Assume attributes are conditionally independent.** Naïve Bayesian prediction requires each conditional probability be non-zero (Use Laplacian correction (or Laplacian estimator)Adding 1 to each case.**Strength**: **Easy to implement**. **Good results obtained in most of the cases**. **Weakness**: **Assumption: attributes conditional independence, therefore loss of accuracy**. **Practically, dependencies exist among variables**. **E.g., Patients: Profile: age, family history, etc. Symptoms: fever, cough etc. Disease: lung cancer, diabetes, etc. Dependencies among these cannot be modeled by Naïve Bayes Classifier. How to deal with these dependencies? A generative classifier models p(Y, X).** It models how the data was "generated"? "what is the likelihood this or that class generated this instance?" and pick the one with higher probability .Naïve Bayes , Bayesian Networks. **A discriminative classifier models p(Y|X).**It uses the data to create a decision boundary: Logistic Regression, Support Vector Machines.Strength of discriminative: Prediction accuracy is generally high .As compared to generative models: Robust, works when training examples contain errors. Fast evaluation of the learned target function.Comparing to (covered in future) Bayesian networks (which are normally slow) Criticism: Long training time, Difficult to understand the learned function (weights), Bayesian networks can be used easily for pattern discovery, Not easy to incorporate domain knowledge, Easy in the form of priors on the data or distribution **Classifier Evalutaion: 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. Repeated random sub-sampling validation: 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 D_i 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 distribution, in each fold is approximately the same as that in the initial data. **Bootstrap**. 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**. Several bootstrap methods, and a common one is .632 bootstrap.** A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$).**Ensemble methods:** Use a combination of models to increase accuracy**,** Combine a series of *k* learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M***Popular ensemble methods**. **Bagging:** Trains each model using a subset of the training set, and models learned in parallel. **Boosting:** Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order. **Bagging:** Bootstrap Aggregation: Analogy: Diagnosis based on multiple doctors' majority vote **Training**: Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap). A classifier model M_i is learned for each training set D_i. **Classification:** classify an unknown sample X . Each classifier M_i returns its class prediction. The bagged classifier M* counts the votes and assigns the class with the most votes to X. **Prediction:** It can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple. **Accuracy**: Improved accuracy in prediction. Often significantly better than a single classifier derived from D. For noise data: Not considerably worse, more robust **Boosting:** 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 M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i. The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.**Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data.** Adaboost: Given a set of *d* class-labeled tuples, (X_1, y_1), …, (X_d, y_d). Initially, all the weights of tuples are set the same (1/d). Generate *k* classifiers in *k* rounds. At round i, Tuples from D are sampled (with replacement) to form a training set D_i of the same size, Each tuple's chance of being selected is based on its weight, A classification model M_i is derived from D_i, Its error rate is calculated using $D_i$as a test set, If a tuple is misclassified, its weight is increased; otherwise, it is decreased. Error rate: err(X_j) is the misclassification error of tuple X_j. Classifier M_i error rate is the sum of the weights of the misclassified tuples:**Bayesian network** (or Bayes network, belief network, Bayesian model or probabilistic directed acyclic graphical model) is a probabilistic **graphical model** .Represented by a set of *random variables* and *their conditional dependencies* via a *directed acyclic graph* (DAG). Allows *class conditional independencies* between *subsets* of variables. **SVM**: linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary"). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. SVM finds this hyperplane(with the largest margin) using support vectors ("essential" training tuples) and margins (defined by the support vectors). Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors.** SVMs can efficiently perform a non-linear classification **using kernel functions**, implicitly mapping their inputs into high-dimensional feature spaces. SVM is effective on high dimensional data. The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data. The **support vectors** are the essential or critical training examples—they lie closest to the decision boundary (MMH). Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high. SVM is not scalable to the # of data objects in terms of training time and memory usage. Scaling SVM by a hierarchical micro-clustering approach. Features: training can be slow(solve comlex equations) but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)Used for: classification and numeric prediction. SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters). **Applications: handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests** . **SVM:** A relatively new classification method for both linear and nonlinear data. It uses a nonlinear mapping to transform the original training data into a higher dimension. With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary"). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. SVM finds this hyperplane(with the largest margin) using support vectors ("essential" training tuples) and margins (defined by the support vectors). Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors.** SVMs can efficiently perform a non-linear classification **using kernel functions**, implicitly mapping their inputs into high-dimensional feature spaces. SVM is effective on high dimensional data. The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data. The **support vectors** are the essential or critical training examples—they lie closest to the decision boundary (MMH). Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high. SVM is not scalable to the # of data objects in terms of training time and memory usage. Scaling SVM by a hierarchical micro-clustering approach. Features: training can be slow(solve comlex equations) but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)Used for: classification and numeric prediction. SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters). **Applications: handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests** .**Neural Network: Feed-Forward Neural Neural Network:** Typical neural network architecture. The output from one layer is used as input to the next layer (no loops). Information is always fed forward, never fed back. From a statistical point of view, networks perform nonlinear regression. Given enough hidden units and enough training samples, they can closely approximate any function. **Recurrent neural network:** Feedback loops are possible (cascade of neurons firing). Some neurons fire for some limited duration of time, before becoming quiescent. That firing can stimulate other neurons, which may fire a little while later, also for a limited duration, which causes still more neurons to fire, and so on. Loops do not cause problems since a neuron's output only affects its input at some later time, not instantaneously **Backpropagation**: Reset weights on the "front" neural units and this is sometimes done in combination with training where the correct result is known. Iteratively process a set of training tuples & compare the network's prediction with the actual known target value. For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value. Modifications are made in the "**backwards**" direction. From the output layer, through each hidden layer back to the first hidden layer, hence "**backpropagation**". Steps: Initialize weights to small random numbers, associated with biases. Propagate the inputs forward (by applying activation function). Backpropagate the error (by updating weights and biases). Terminating condition (when error is very small, etc.) **Rule induction:** Sequential covering algorithm: Extracts rules directly from training data. Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER. **Rules are learned *sequentially***, each for a given class $C_i$will cover many tuples of $C_i$but none (or few) of the tuples of other classes. Steps: **Rules are learned one at a time**. Each time a rule is learned, the tuples covered by the rules are removed. Repeat the process on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold. ***Comp. w. decision-tree induction: learning a set of rules simultaneously.*** **Lazy vs. eager learning**: **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple. **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify. Lazy: less time in training but more time in predicting. Accuracy. Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function. Eager: must commit to a single hypothesis that covers the entire instance space.**KNN Weight the contribution of each** of the *k* neighbors according to their distance to the query x_qGive greater weight to closer neighbors. Robust to noisy data by averaging *k*-nearest neighborsCurse of dimensionality: distance between neighbors could be dominated by irrelevant attributes To overcome it, axes stretch or elimination of the least relevant attributes. **CBR**: Applications: Customer-service (product-related diagnosis), legal ruling. **K-mean:** clustering often *terminates at a local optimal.* Initialization can be important to find high-quality clusters. Need to specify *K*, the *number* of clusters, in advance. There are ways to automatically determine the "*best*" In practice, one often runs a range of 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 Using the K-modes for *categorical data*. Not suitable to discover clusters with *non-convex shapes* Using density-based clustering, kernel *K*-means, etc..**AGNES (AGglomerative NESting)** 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.***Major weaknesses of hierarchical clustering methods***: Can never undo what was done previousl, Do not scale well, Time complexity of at least $O(n^2)$, where *n* is the number of total objects **BIRCH**: Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clusterin. Phase 1: 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) . Phase 2: Use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree. Low-level micro-clustering: Reduce complexity and increase scalability. High-level macro-clustering: Leave enough flexibility for high-level clusterin. *Scales linearly*: Find a good clustering with a single scan and improve the quality with a few additional scans **CURE** (Clustering Using REpresentatives) (S. Guha, R. Rastogi, and K. Shim, 1998). Represent a cluster using a set of well-scattered representative points. Cluster distance: Minimum distance between the representative points chose. This incorporates features of both single link and average link. 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. Choosing scattered points helps CURE capture clusters of arbitrary shapes

Sequential Pattern Mining in Vertical Data Format: The SPADE Algorithm

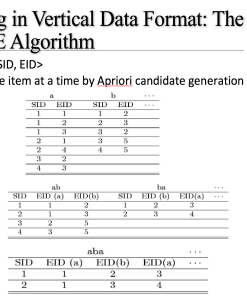
- A sequence database is mapped to: <SID, EID>
- Grow the subsequences (patterns) one item at a time by Apriori candidate generation

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

min_sup = 2

Ref: SPADE (Sequential PAttern Discovery using Equivalent Class) [M. Zaki 2001]

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c



Measure	Definition	Range	Null-Invariant?
$\chi^2(A,B)$	$\sum_{i,j}\frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0,\infty]$	No
$Lift(A,B)$	$\frac{s(A\cup B)}{s(A)\times s(B)}$	$[0,\infty]$	No
$Allconf(A,B)$	$\frac{s(A\cup B)}{max\{s(A),s(B)\}}$	$[0,1]$	Yes
$Jaccard(A,B)$	$\frac{s(A\cup B)}{s(A)+s(B)-s(A\cup B)}$	$[0,1]$	Yes
$Cosine(A,B)$	$\frac{s(A\cup B)}{\sqrt{s(A)\times s(B)}}$	$[0,1]$	Yes
$Kulczynski(A,B)$	$\frac{1}{2}\left(\frac{s(A\cup B)}{s(A)}+\frac{s(A\cup B)}{s(B)}\right)$	$[0,1]$	Yes
$MaxConf(A,B)$	$max\left\{\frac{s(A\cup B)}{s(A)},\frac{s(A\cup B)}{s(B)}\right\}$	$[0,1]$	Yes

χ^2 and lift are not null-invariant

Jaccard, consine, AllConf, MaxConf, and Kulczynski are null-invariant measures

Indexing OLAP Data: **Bitmap Index**

- Index on a particular column
  - Each value in the column has a bit vector: bit-op is fast
  - The length of the bit vector: # of records in the base table
  - The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column
  - not suitable for high cardinality domains
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Base table

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Index on Region

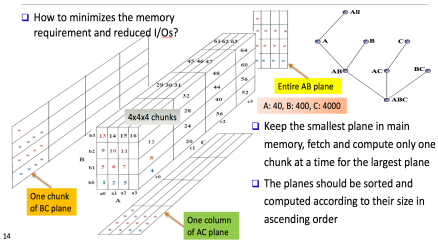
RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

Multi-way Array Aggregation (3-D to 2-D)

- How to minimizes the memory requirement and reduced I/Os?
- Keep the smallest plane in main memory, fetch and compute only one chunk at a time for the largest plane
- The planes should be sorted and computed according to their size in ascending order



Computing a 5-D Cube with 2-Shell Fragments

- Example: Let the cube aggregation function be **count**

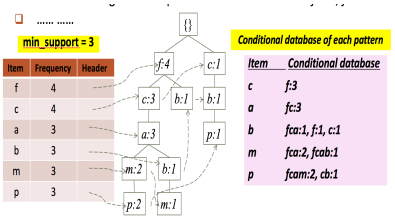
tid	A	B	C	D	E
1	a1	b1	c1	d1	e1
2	a1	b2	c1	d2	e1
3	a1	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

- Divide the 5-D table into 2 shell fragments:
  - (A, B, C) and (D, E)
- Build traditional invert index or RID list

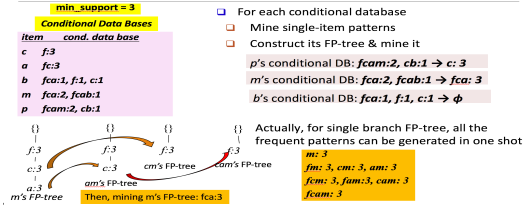
Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Cell	Intersection	TID List	List Size
a1 b1	1 2 3 ∩ 1 4 5	1	1
a1 b2	1 2 3 ∩ 2 3	2 3	2
a2 b1	4 5 ∩ 1 4 5	4 5	2
a2 b2	4 5 ∩ 2 3	ϕ	0

- Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is: $O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)$
- For F < 5, the growth is sub-linear



Mine Each Conditional Database Recursively



- For each conditional database
  - Mine single-item patterns
  - Construct its FP-tree & mine it

p's conditional DB: ***fcam:2, cb:1 → c: 3***

m's conditional DB: ***fca:2, fcab:1 → fca: 3***

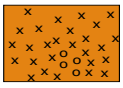
b's conditional DB: ***fca:1, f:1, c:1 → ϕ***

Actually, for single branch FP-tree, all the frequent patterns can be generated in one shot

m: 3
fm: 3, cm: 3, am: 3
fcm: 3, fam:3, cam: 3
fcam: 3

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive examples but numerous negative ones
  - E.g., medical diagnosis, fraud transaction, accident (oil-spill), and product fault
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods on imbalanced data in two-class classification
  - **Oversampling**: Re-sampling of data from positive class
  - **Under-sampling**: Randomly eliminate tuples from negative class
  - **Threshold-moving**: Move the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques**: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks



Constraint	Anti-monotone	Monotone	Succinct
v ∈ S	no	yes	yes
S ⊇ V	no	yes	yes
S ⊆ V	yes	no	yes
min(S) ≤ v	no	yes	yes
min(S) ≥ v	yes	no	yes
max(S) ≤ v	yes	no	yes
max(S) ≥ v	no	yes	yes
count(S) ≤ v	yes	no	weakly
count(S) ≥ v	no	yes	weakly
sum(S) ≤ v (a ∈ S, a ≥ 0)	yes	no	no
sum(S) ≥ v (a ∈ S, a ≥ 0)	no	yes	no
range(S) ≤ v	yes	no	no
range(S) ≥ v	no	yes	no
avg(S) θ v, θ ∈ { =, ≤, ≥ }	convertible	convertible	no
support(S) ≥ ξ	yes	no	no
support(S) ≤ ξ	no	yes	no

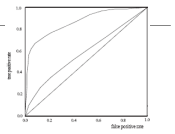
Constraint	Convertible anti-monotone	Convertible monotone	Strongly convertible
avg(S) ≤ , ≥ v	Yes	Yes	Yes
median(S) ≤ , ≥ v	Yes	Yes	Yes
sum(S) ≤ v (items could be of any value, v ≥ 0)	Yes	No	No
sum(S) ≤ v (items could be of any value, v ≤ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≥ 0)	No	Yes	No
sum(S) ≥ v (items could be of any value, v ≤ 0)	Yes	No	No
.....			

17

Computation of Gini Index

- Example: D has 9 tuples in buys_computer = "yes" and 5 in "no"
$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$
- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2
  - $gini_{income \in \{low,medium\}}(D) = \frac{10}{14} gini(D_1) + \frac{4}{14} gini(D_2)$
  $$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = 0.443$$
  $$= Gini_{income \in \{high\}}(D)$$
  - $Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450
  - Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index
- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Model Selection: ROC Curves



- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve (**AUC**: Area Under Curve) is a measure of the accuracy of the model
- 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
- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

55

Example: Attribute Selection with Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

age	p_i	n_i	I(p_i, n_i)
<=30	2	3	0.971
31...40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's.

Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly, we can get

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

12

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

- P(C$_i$): P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357
- Compute P(X|C$_i$) for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

P(X|C$_i$) : P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

P(X|C$_i$)*P(C$_i$) : P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

33

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	¬C	
C	TP	FN	P
¬C	FP	TN	N
	P'	N'	All

- **Classifier accuracy,** or recognition rate
  - Percentage of test set tuples that are correctly classified
  Accuracy = (TP + TN)/All
- **Error rate:** *1 – accuracy*, or
  Error rate = (FP + FN)/All
- **Class imbalance problem**
  - One class may be *rare*
    - E.g., fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - Measures handle the class imbalance problem
    - **Sensitivity** (recall): True positive recognition rate
      - **Sensitivity = TP/P**
    - **Specificity**: True negative recognition rate
      - **Specificity = TN/N**

50

Random Forest

- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- 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 typical bagging or boosting

61

(a) [8] Name four methods that perform effective *dimensionality reduction* and four methods that perform effective *numerosity reduction*.

Answer:

- dimensionality reduction: decision-tree, PCA, wavelets, attribute-reduction/selection.
- numerosity reduction: Any four in {sampling, clustering, discretization, data cube, regression, histogram, data compression}.

(d) [9] Given a 50 GB data set with 40 attributes each containing 100 distinct values, and 512 MB main memory in a laptop, outline an efficient method that constructs decision trees efficiently, and answer the following questions explicitly: (1) how many scans of the database does your algorithm take if the maximal depth of decision tree derived is 5? (2) how do you use your memory space in your tree induction?

Answer:

i. Rainforest alg. AVC set takes much less space $40 \times 100 \times 4 \times C = 16CKB$, where C is the distinct number of class labels.

ii. Thus we only need d scans where d is the depth of the tree so constructed.

iii. Outline an extended SVM algorithm that is scalable in large data sets.

A. construct microcluster (using a CF tree)

B. Train an SVM on the centroids of the microcluster

C. Decluster entries near the boundary

D. Repeat until convergence

(c) [6] Suppose one wants to compute a flight-booking data cube where the condition is that the minimum number of records is 8 and the average fare is over $500. Outline an efficient cube computation method (based on the commonsense about the flight data distribution).

Answer:

i. Transform the above condition into two data cube conditions: min_sup = 8, and $avg(price) \geq 500$.

ii. Use a typical ice-cube computation alg. and relax the condition $avg(price) \geq 500$ into $avg^8(price) \geq 500$. Pruning can be performed if avg of top-8 price is less than 500.

iii. Top-8 can be implemented more efficiently if 8 is a big number.

(a) [4] What are the major differences among the three: (1) information gain, (2) gain ratio, and (3) foil-gain?

Answer:

i. information gain: for decision-tree induction, biased on multi valued attributes.

ii. gain ratio: for decision-tree induction, add dumping factor, resul in more balanced trees.

iii. foil-gain: for rule-induction, more likely to pick the attribut value that has more positive training data associated.

(b) [4] What are the major differences between (1) bagging and (2) boosting?

Answer: Both belong to emsemble approach

i. bagging: simple multiple voting strategy

ii. boosting: add more weights on misclassified samples

(b) [6] Briefly describe one efficient **distributed** pattern growth mining method which can mine enterprise-wide (*i.e.*, global) frequent itemsets for a chain store like Sears, without shipping data to one site.

Answer:

i. Suppose there are k partitions

ii. Pattern-growth on each partition and then merge frequent itemsets as candidate FPs,

iii. One more scan of each partitioned DB to count the merged frequent itemset candidates.

[6] Use one sentence to distinguish each of the following pairs of methods:

(1) *k-means* vs. *KNN*

Answer: k-means: A partitioning clustering algorithm
KNN: A lazy learning algorithm to find its k-nearest neighbor and return labels based on its k-nearest neighbors.

(2) *STING* vs. *CLIQUE*

Answer: A grid-based clustering algorithm based on multi-resolution, hierarchical summarization

(c) [4] What are the major differences among the three (1) piece-wise linear regression, (2) multiple-linear regression, and (3) regression tree?

Answer:

i. piece-wise linear regression: linear regression on each segment of data

ii. multiple-linear regression: linear regression on multiple-dimensional space

iii. regression tree: regression value at the leaf nodes of a tree.

(e) [5] Give each situation that one of the following measure is most appropriate for measuring the quality of classification: (1) *accuracy*, (2) *F-measure*, and (3) *ROC curve*.

Answer:

Accuracy: Suitable for general classification tasks when both positive and negative examples have equal importance

F-measure: harmonic mean of precision and recall, suitable for situations that both precision and recall are important

ROC curve: suitable when true positive rate and false positive rate are both important.

(c) [6] Frequent pattern mining often generates many somewhat "similar" patterns that carry little new information. Give one such example. Then outline one method that may generate less number (*i.e.*, compressed) but interesting patterns.

Answer: Example: If we have two patterns: ABCD: 2403, ABCDE: 2400. Then ABCDE should retain but ABCD is redundant and can be removed.

Method: Mining the **compressed patterns**. The key points are: The pattern compressed problem can be defined as to find a minimal set of representative patterns to cover each frequent pattern. Here "cover" means the distance (of support) of a frequent pattern to a representative pattern is within a threshold. E.g., using Jaccard coefficient as a definition.

(a) [6] Since items have different expected frequencies of sales, it is desirable to use *group-based minimum support thresholds* set up by users. For example, one may set up a small *min-support* for the group of *cameras* but a rather large one for the group of *bread*. Outline an FPgrowth-like algorithm that derive the set of frequent items efficiently in a transaction database.

Answer: Suppose each item is associated with a group ID.

i. Scan the database and find single frequent items, F1, using *group-based minimum support thresholds*. Sort the F1 list based on item frequency descending order.
Note: It is also interesting and could be even smarter (but not required), to sort items based on how far the item from its support threshold: the bigger, the higher (to facilitate pruning).

ii. Construct FP-tree, and for each frequent item, project its conditional databases. Within the conditional database, find single frequent items as step 1, and construct its own FP-tree until either the pattern-base is empty or the FP-tree contains a single branch.

iii. Do the same as steps 1 and 2 recursively, and output the frequent itemsets so obtained.

(b) [4] What is the value range for each of the following normalization methods?

i. min-max normalization,
Answer: [new-min, new-max]

ii. z-score normalization, and
Answer: $(-\infty, +\infty)$

iii. normalization by decimal scaling?
Answer: $(-1, +1]$

(a) [5] What are the major differences among the three: (1) Naïve-Bayesian algorithm, (2) Bayesian Belief Network, and (3) Neural Network?

Answer:

Differences between (1) (2) vs. (3):

(1)(2) are generative classification where the data is assumed to following some distribution. For new samples, the probability belonging

(a) [6] Since items have different values and expected frequencies of sales, it is desirable to use *group-based minimum support thresholds* set up by users. For example, one may set up a small *min-support* for the group of *diamonds* but a rather large one for the group of *shoes*. Outline an Apriori-like algorithm that derive the set of frequent items efficiently in a transaction database.

Answer: Use group min-support to prune itemset generation in each group and use the smallest min-support to prune mixed itemsets. One can generate an efficient alg.

(b) [6] A data cube C has D dimensions, and each dimension has exactly p distinct values in the base cuboid (assuming a dimension contains no levels, i.e., no hierarchies).

i. What could be the **maximum number of cells** of the base cuboid?
Answer: p^D.

ii. What could be the **minimum number of cells** of the base cuboid?
Answer: p.

iii. What could be **maximum number of cells** (including both base cells and aggregate cells) in the data cube C?
Answer: $(p+1)^D$.

i. The profit range for the items in each pattern must be within $50.

Answer: The constraint $range(S.profit) \leq \$50$ is antimonotonic. Grow itemset set S in FP mining, If S violates the constraint, prune it (since S's superset can never satisfy the constraint).

Note: Do not take point off if no perfect constraint formula is written since it is not easy to write precise constraints in formulas. Focus will be on category of constraints and pruning methods. □

(b) [6] Briefly describe one efficient **incremental** frequent-itemset mining method which can incorporate newly added transactions without redoing the mining from scratch.

Answer:

(1) Input: DB (the original transaction database), $freq_{DB}$ (frequent itemset of DB), δDB (the newly added transactions).

(2) Scan δDB once, count frequency for each itemset in $freq_{DB}$ and output those in $freq_{DB}$ whose added support is frequent (i.e., frequent in both ($\delta DB \cup DB$)).

(3) Mine frequent itemsets in δDB. For those frequent itemsets in δDB but not in DB, scan DB once to get their frequency in DB. Output those whose added support is frequent.

(b) [6] Associative classification often involves first mining association rules and then performing classification. Suppose one wants to use an FPtree-like structure to facilitate mining rules for associative classification. Discuss how the original FP-tree data structure and the algorithm should be modified to make the mining efficient.

Answer:

For each tree branch, do not store class label but put each class count into the combined node count. Then mine the rules. If the node combined count is less than min_support, stop mining. The rules generated will still be sorted by confidence and then support. □

i. Give the name of the measure for the distance *between two objects* for each of the **4** *different kinds of data.*

Answer:

typical four but others will be ok.

numerical (interval) data: Euclidean, or Mahattan, or Minkowski

asymmetric binary: Jaccard coefficient

vector (e.g., text document): cosine

nominal: $(p-m)/p$ □

ii. Give the names of **4** measures for the distance *between two clus-*

(a) Based on equation (1.5), prove *all_confidence* is *antimonotonic.* Since

$$sup(A_1, A_2, \ldots, A_k, A_{k+1}) \leq sup(A_1, A_2, \ldots, A_k)$$

and

$$max(sup(A_1), \ldots, sup(A_{k+1})) \geq max(sup(A_1), \ldots, sup(A_k)),$$

Adding any new item will make the quotient monotonically decreasing. Thus if the current k-itemset cannot satisfies the constraint, its corresponding $(k+1)$-itemset cannot satisfy it either. Thus it is antimonotonic.

(b) Given min_σ (*i.e.*, min_sup) and min_α, state how to modify the FP-growth method to mine such strongly correlated patterns by pushing deeply both thresholds into the mining process.

When mining, take both min_sup and all_conf into consideration. That is, when an itemset whose either min_sup or all_conf does not satisfy the thresholds, stop. Otherwise, project the conditional DB and proceed.

(b) [5] Both decision-tree induction and associative classification may generate rules for classification. What are their major differences? Why is it that in many cases an associative induction may lead to better accuracy in prediction?

Answer:

Since decision-tree induction involves only one attribute each time, the rules generated by decision-tree are constrained. Associative classification finds strong associations better frequent patterns and class labels. The rules usually combine multiple attributes. In this context, the latter can represent more complicated boundary, thus leads to better accuracy in prediction. □

(1) Decision-tree induction, (2) piece-wise linear regression, (3) SVM, (4) associative classification, (5) genetic algorithm, and (6) Bayesian Belief Network.

Answer:

(1) Decision-tree induction: Not work, too many attributes.

(2) piece-wise linear regression: Not work, need to huge number of parameters.

(3) SVM: works since it scales to large dimensions.

(4) associative classification: works if the method of frequent pattern mining is further developed as shown in Tung's paper.

(5) genetic algorithm: Not work since it may invoke an exponential number of mutations.

(6) Bayesian Belief Network: Not work since the sample size is too small, the probability distribution cannot be estimated accurately.

(c) [6] What are the major differences among the three: (1) rule-based induction, (2) case-based reasoning, and (3) k-nearest neighbor classification?

Answer:

Differences between (1) and (2)(3):

(1): eager classifier, where a model (represented by a set of rules) is constructed based on training data, then use this model to classify a new example.

(2)(3): lazy classifier, where training data is stored and test example is classified by comparing it with the training data. No model is constructed beforehand.

Differences between (2) and (3):

the k-nearest neighbor approach regards each example is a point in a d-dimensional space, and classifies the test example by common class among its nearest neighbors. Case-based reasoning searches for similar cases of test examples, and classifies test example using the class of its similar case. □

i. clustering stars in the universe: Euclidean distance (spatial and interval based)

ii. clustering text documents: cosine (vector data) similarity

iii. clustering clinical test data: asymmetric data (e.g., Jaccard coefficient)

iv. clustering houses to find delivery centers in a city with rivers and bridges: reachable distance (not direct Euclidean distance), considering obstacles.

(b) [6] Anges, BIRCH, and Chameleon are all hierarchical clustering algorithms. Rank them based on clustering quality. Also, rank them based on scalability to large datasets. Briefly justify your rank.

Answer:

Quality ranking: Chameleon, BIRCH, AGNES

Scalability ranking: BIRCH, AGNES $O(n \log n)$, and CHAMELEON $O(N^2)$ □

(2) clustering houses to find delivery centers in a city with rivers and bridges, and

Answer:

Constraint-based clustering where constraints are obstacles. □

(3) distinguishing snakes hidden in the surrounding grass.

Answer:

Density-based clustering like DBSCAN. □

(1) clustering Microsoft employees based on their working-years and salary,

Answer:

CLARANS, i.e., scalable k-medoids algorithm. □

(e) [4] What are the best distance measure for each of the following applications:

(i) driving distance between two locations in Downtown Chicago,

Answer: Manhattan distance □

(ii) compare similar diseases with a set of medical tests,

Answer: Dissimilarity for asymmetric binary variables or Jaccard coefficient, i.e., $(b+c)/(a+b+c)$ □

(iii) find similar web documents

Answer: Cosine measure of two vectors, i.e., the inner product of two feature vectors, each representing the features (such as keywords or terms) of a document. □

Outline algorithm: RAINFOREST. Build an AVC list (attribute-value-class label). Scan data once, one can populate the AVC group, and then calculate to decide which attribute needed to used for split.

5 scans in most cases since each scan construct one level of the tree.

For the first scan, one needs memory space as follows:

One AVC group: 24 * 30 * 4 * 2 = 5.76 K bytes.

But for the 2nd scan, there will 30 such tables, and in the worst case, there will be 30 such AVC group.

3rd scan, there will be 30^2 such AVC group

4th scan, there will be 30^3 such AVC group.

5th scan, there will be 30^4 such AVC group, each of size 20 * 30 * 4 * 2 = 4800 bytes = 4.8K. Notice 81 * 10^4 * 4.8 K = 4GB in the worst case. But since most of the table will be empty for such values, in most cases, it will still fit into the memory. Thus in practice, it still be ok to do it in 5 scans.

i. The price difference between the most expensive item and the cheapest one in each pattern must be within $100.

Answer:

C : $range(S.price) \leq \$100$ is antimontonic.

Method: Push C into iterative mining, toss S if it cannot satisfy C. □

(i) What is the *null-invariance* property?

Answer: A measure not influenced by the count of $\bar{c}\bar{t}$ (i.e., those containing neither coffer nor tea).

(b) [8] To further improve the Apriori algorithm, several *candidate generation and-test* methods are proposed that *reduce the number of database scans* at mining. Briefly outline two such methods.

Answer: Any of the following algorithms will count:

1. Partitioning: partition DB into k portions, each fit in memory; mine each local partition; merge freq-itemsets; then one more scan DB to consolidate the global patterns.

2. Hashing: First scan, count freq-1 and hash 2-itemsets into buckets. If the bucket count < threshold, all the 2-itemsets in it are infreq.

(c) [5] Suppose a disk-based large relation contains 100 attributes. What is the minimum number of database scans in order to derive a generalized relation by *attribute-oriented induction*?

Answer:

Two scans: one preparing for generalization, one performing generalization (which will derive a small, memory-resident prime-relation)

or

$1+\delta$ scan since the first scan can be replaced by a δ scan or sampling. □

(a) [10] Assume a base cuboid of N dimensions contains only p (where $p > 3$) nonempty base cells, and there is no level associated with any dimension.

i. What is the *maximum number of nonempty cells* (including the cell in the base cuboid) possible in such a materialized datacube?

Answer:

Each cell generates 2^N cells. So p cells will generate in total $p \times 2^N$ cells. However, the p cells at the Apex cuboid are merged into one, i.e., we need to minus $p-1$ cell count. Thus the maximum number of cells generated will be:

$$p \times 2^N - p + 1$$

ii. give one example for each of the three cases that one is the most appropriate measure.

Answer:

χ^2: Use an example like milk vs. bread or play-basketball vs. eat cereal to show it is useful.

Pearson's correlation coefficient: use any number series or plot.

Kulczynski: Use a transaction DB where null is somewhat big, or anything that expresses the meaning.

As long as the idea is correct, it should be fine. □

ii. The sum of the profit of all the items in each pattern is between $10 and $20, and each such item is priced over $10.

Answer:

C_1 : $min(S.price) > \$10$ is succinct, or data anti-monotone.

Method: Push C_1 into iterative mining, select only items satisfying C_1

C_2 : $sum(S.profit) \geq \$10$ is ***convertible*** monotone, C_3 : $sum(S.profit) \leq \$20$ is ***convertible*** antimonotone, if items within a transaction is sorted in profit ascending order.

Method: Push C_3 into iterative mining, toss S if it does not satisfy C_3 and check C_2, and once its satisfies, no more checking needed.

(i) clustering a set of research papers based on their authors and their publication venues

Answer:

RankClus. □

(ii) clustering a set of videos based on their image contents, captions, and where they reside on the web

(b) [6] Explain why BIRCH can handle large amount of data in clustering, and explain how such a methodology can be used to scale up SVM classification in large data sets.

Answer:

BIRCH: balance-tree, CF-tree (using 0, 1, 2 moment to compute the differences and organize data into microclusters and hierarchies. On top of such a hierarchy, one can use any good clustering algorithm. Thus it can handle large data sets.

CB-SVM (Clustering-based SVM)

exploring Birch-like hierarchical structures, construct such a structure for positive and negative data respectively using BIRCH.

Training SVM from top down

subclusters along the support vector can be declustered to refine the support vector by repeatedly train with SVM.

Algebraic. The linear regression line can be computed by the least square method. In a 2-D space, the regression line $y = mx + b$ can be computed as:

Find the four sums: $\sum x$, $\sum x^2$, $\sum y$, and $\sum xy$.

The calculations for the slope m and the y-intercept b are as follows.

$\hat{m} = \frac{n(\sum xy)-(\sum y)(\sum x)}{n(\sum x^2)-(\sum x)^2}$;

$\hat{b} = (\frac{1}{n}\sum y) - \hat{m}(\frac{1}{n}\sum x) = \frac{(\sum y)(\sum x^2)-(\sum x)(\sum xy)}{n(\sum x^2)-(\sum x)^2}$

The students are not expected to provide those details, they can get full points if they answer algebraic with brief explanations.

□

(4) confidence interval in the formula $\bar{x} \pm t_c\hat{\sigma}_x$

Answer:

Algebraic. The mean value $\bar{x}$ and the standard deviation $\hat{\sigma}_x$ are both algebraic. Therefore the formula is also algebraic. □

(b) [6] Explain why both *Apriori* and *FPgrowth* algorithms may encounter difficulties at mining colossal patterns (the patterns of large size, e.g., 100). A new algorithm based on *core pattern fusion* can mine such patterns efficiently. Explain why such an algorithm is efficient and effective at mining most of the colossal patterns.

Answer:

Difficulty of *Apriori* and *FPgrowth*: explosive number of mid-sized patterns, because a frequent large pattern will contain an explosive number of mid-sized patterns.

Method: core-pattern fusion: Based on the idea that a large pattern can be partitioned into midsized patterns in many different ways, and thus many midsized patterns will be able to merge back to get closer back to the large patterns.

It traverses the tree in a bounded breadth way. Only a fixed number of patterns in the current candidate pool will be used as the starting nodes to go down in the pattern tree. Thus it avoids the exponential search space.

Moreover, it identifies shortcuts whenever possible by agglomeration of multiple patterns in the pool. It will quickly towards colossal patterns.

□

(a) [6] Given a fixed *min_support* threshold, σ (e.g., $\sigma = 0.5\%$), present an efficient incremental mining algorithm that can maximally use the previously mined information when a new set of transactions ΔTDB is added to the existing transaction database TDB.

Answer:

Based on the principle: *An itemset is frequent in* $\Delta TDB + TDB$, *it must be frequent in at least one of them.* (easy to prove).

Then assume we have all of the frequent itemset I for TDB. Mine frequent itemset dI in ΔTDB. For each $i \in I$ and $i \in dI$, add their support and output them (they must be frequent). For each i in I but not in dI, add their count in ΔTDB to see if it is frequent. For each i in dI but not in I, add their count in TDB (which needs to scan TDB once) to see if it is frequent.

(3) distinguishing snakes hidden in the surrounding grass, and

Answer:

density-based clustering (DBSCAN, or OPTICS) because snakes can only be identified based on the connected similar colored points. □

(4) clustering shoppers based on their shopping time, the amount of money spent, and the categories of goods they usually buy.

Answer:

numerical data using k-means, categorical data using k-modes, and the combined one using combined distance measure with an extended k-means algorithm (called k-prototype).

(2) Bottom-10 (among all the objects in the corresponding k-dimensional space)

Answer:

Distributive. One can divide the whole dataset to several partitions, and collect the bottom-10 for each partition. The bottom-10 of the whole dataset can be computed by listing all the collections in ascending order and select the top 10 item. The result calculated from the partitions are the same as the one from entire dataset. Thus the bottom-10 measure is distributive.

We can also seen the computation process as a formula, and in this perspective the measure is also algebraic. The students will get full points of the answer of "algebraic" if they give reasonable explanations. □

(c) [6] Why is it that BIRCH encounters difficulties to find clusters of arbitrary shape but OPTICS has no problem to do it? Propose some modifications to BIRCH so that it can help find clusters of arbitrary shape.

Answer:

BIRCH uses Euclidean distance to cluster objects in the hierarchical CF-Tree. OPTICS uses density-based method to find maximal set that are density-connected. Therefore, OPTICS can find objects with arbitrary shape, while BIRCH cannot.

BIRCH needs to embed some distance measure related to density. In this way, data objects that are density-connected can be detected by BIRCH. □

(d) [5] What are the similarities and differences between *semi-supervised classification* and *active learning*?

Answer: Similarity: Both have a small set of labeled data and a large set of unlabeled data.

Differences: Semi-supervised classification does not interact with the expert. It labels the unlabeled data using classifiers training from labeled data (self-training/co-training)

Active learning: selective useful unlabeled examples and ask experts to give labels for them. □

(b) [6] Give each situation that one of the following measures is most appropriate for measuring the quality of classification:

(1) *sensitivity*

Answer: True positive rate. If you want your classifier to measure the true positive cases in a sensitive way. E.g., finding cancer cases. □

(2) *specificity* **Answer:** True negative rate. If you want your classifier to measure the true negative cases in a subtle way. E.g., spam detection, you do not want to filter out useful message. □

(3) *ROC curve*

Answer: For a classifier that returns probability outputs. E.g., in skewed cases (e.g., few positive training data but lots of negative training data. You may like to measure a classifier using ROC curve instead of accuracy. □

(b) [7] Databases are usually used to answer people's queries. When the expected answer set is large, it is often desirable to return only a small set of top-ranked answers. Suppose a user would like to get top-k ranked answers for Thankgiving online shopping, based on his/her own criteria of ranking. But the relevant dimension is pretty high (say over 30 dimensions). Design a data cube that may facilitate efficient processing of such queries.

Answer:

The data cube should be a **ranking cube** with **shell fragments** technology to support top k query in high dimensional space.

The key points of ranking cube are:

1). Partition data on both selection and ranking dimensions;

2). Compute measures for each group/block;

3). Simultaneously push selection and ranking conditions to locate the block with top score.

To support high-dimensional data:

1). Materialize only those atomic cuboids that contain single selection dimensions;

3) Uses the idea similar to high-dimensional OLAP: Achieves low space overhead and high performance in answering ranking queries with a high number of selection dimensions.

(3) *associative classification*: generate rules by mining those association rules whose RHS is class label only. Then sort rules by confidence and then support. Rule application is order-dependent. □

(3) *BIRCH vs. CHAMELEON.*

Answer: BIRCH: A micro-clustering-based clustering algorithm, incrementally build balanced CF-tree (clustering feature).

CHAMELEON: Use graph-partitioning to form small subclusters and then merge them based on inter-connectivity and closeness. □

(ii) clustering a set of videos based on their image contents, captions, and where they reside on the web

Answer:

p-clustering (Frequent pattern based) or any others helpful for high dimensional data □

a) [5] Suppose *incremental update of a data cube* means that new data can be incrementally inserted into the base cuboid without recomputing the whole cube from scratch. Can you do this for an *iceberg cube*? If you can, state how; but if you cannot, state why not.

Answer: No. Because iceberg cube drops the count of the cells if it is below min_sup, it cannot get the correct count for cells in an iceberg cube upon incremental update. □

(b) [10] A research publication database like DBLP contains millions of papers authored by a set of researchers.

(i) [5] Using this dataset, explain why *null-invariance* property is important in the study which authors are "correlated".

Answer: Since most authors are not coauthoring papers, null value is very high. Null-invariance is critical otherwise the "correlation value could be greatly influenced by null values. □

(ii) [5] To mine potential advisors and advisees relationships, what measures would you like to use? Explain your answer.

Answer: Kulczakii value and imbalance factor (or something like

(d) [6] *Cross-validation* can be useful in both classification and clustering. What are the differences in these two cases?

Answer:

Cross-validation in classification: supervised, use it to select the best model: Partition data sets into m sets and use $(m-1)$ sets as training and the remaining one as testing. And such training and testing take m turns and cross validate each other. Select the model that has the highest accuracy.

Cross-validation in clustering: unsupervised, use it to select the best number of clusters, k. Partition the data into m sets, and user $(m-1)$ sets of data for clustering, and then assign the points in the remaining

(i) finding oil spills along a coast line

Answer: Density-based clustering, such as DBSCAN and OPTICS, since it needs to detect arbitrary shaped clusters. □

(ii) clustering employees in a company based on their salaries and years of working experience

Answer: Partitioning-based clustering, better using k-medoids than using k-means (although k-means is basically OK (*i.e.*, give **1** point off) since k-medoids is less sensitive to outliers: a small number of employees/managers' salaries could be substantially higher than others. □

Name four methods that perform effective dimensionality reduction and four methods that perform effective numerosity reduction. (i)Dimensionality reduction: SVD, PCA, decision tree, feature subset selection, feature creation, one may also count: wavelet/Fourier transformation(ii) Numerosity reduction: data compression, regression, clustering, sampling, binning, dis- cretization, histogram, data cube aggregation, and also wavelet/Fourier transformation

computing a dense full cube of low dimensionality (e.g., less than 6 dimensions),Best: multiway array cubing. Worst: shell-fragment (since most part of cube is not precomputed) **performing OLAP operations in a high-dimensional database (e.g., over 50 dimensions)** Best: shell-fragment **computing a large iceberg cube of around 10 dimensions** BUC **computing a sparse iceberg cube of high dimensionality** Shell

(b) [8] It is desirable to construct an AlbumCube to facilitate multidimensional search through digital photo collections, such as by date, photographer, location, theme, content, color, etc.

i. [2] What should be the dimensions and measures for such a data cube?
Answer: Dimension: date (hierarchy could be: date, day of week, month, year, etc.), photographer (hierarchy could be photographer, group, etc.), location (hierarchy could be loc, city, state, country, etc.), theme (hierarchy could be theme keyword, theme category).
Note: One can search by content or color, but more difficulty to set content/color as hierarchies or dimensions, special content-based image search is needed. You may want to take **0.5 point off** if they put these as dimension—many did it.
Measure: usually count is the most natural one. Space (sum of disk space in bytes) may also useful, then max, min, std-deviation in size may also make sense. □

ii. [3] What analytical functions can you provide?
Answer: Besides typical OLAP functions, one may consider clustering, classification, content summarization, those supporting similarity search.
Note: It is OK to give complete scores if most useful OLAP functions, such as drilling, slicing/dicing, etc. are answered □

iii. [3] What are the major challenges on implementing AlbumCube, and how would you propose to handle them?
Answer: High-dimensionality, especially when search and OLAP on contents, colors, etc. Solution: using high-D cubing and multi-dimensional indexing.
Content and color: hard to set as dimensions. Solution: developing similarity/approximate search on those aspects.
Noise: Solution: data cleaning, preprocessing.
Note: Major challenge: high-dimensionality (especially on contents and colors) that makes cube hard to build if putting these as dimensions). You may like to take **0.5-1 point off** if this is missing. □

Kernel Functions for Nonlinear Classification

- Instead of computing the dot product on the transformed data, it is mathematically equivalent to applying a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e.,
  - $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)\, \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(\boldsymbol{X}_i, \boldsymbol{X}_j) = (\boldsymbol{X}_i \cdot \boldsymbol{X}_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(\boldsymbol{X}_i, \boldsymbol{X}_j) = e^{-\|\boldsymbol{X}_i - \boldsymbol{X}_j\|^2/2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(\boldsymbol{X}_i, \boldsymbol{X}_j) = \tanh(\kappa \boldsymbol{X}_i \cdot \boldsymbol{X}_j - \delta)$$

- SVMs can efficiently perform a non-linear classification using kernel functions, implicitly mapping their inputs into high-dimensional feature spaces

CBA: Classification Based on Associations

- CBA [Liu, Hsu and Ma, KDD'98]
- Method
  - Mine high-confidence, high-support class association rules
  - LHS: conjunctions of attribute-value pairs); RHS: class labels
    $p_1 \wedge p_2 \ldots \wedge p_l \rightarrow$ "$A_{\text{class-label}} = C$" (confidence, support)
  - Rank rules in descending order of confidence and support
  - Classification: Apply the first rule that matches a test case; o.w. apply the default rule
  - Effectiveness: Often found more accurate than some traditional classification methods, such as C4.5
  - Why? — Exploring high confident associations among multiple attributes may overcome some constraints introduced by some classifiers that consider only one attribute at a time