ass P: buys_computer = "yes"
ass N: buys_computer = "no"

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

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

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

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

income	student	credit_rating	buys_computer
high	no	fair	no
high	no	excellent	no
high	no	fair	yes
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
low	yes	excellent	yes
medium	no	fair	no
low	yes	fair	yes
medium	yes	fair	yes
medium	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

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

Similarly,

$Gain(income) = 0.029$

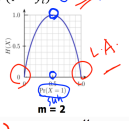
$Gain(student) = 0.151$

$Gain(credit\_rating) = 0.048$

15

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$
  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Computation of Gini Index

- Ex. 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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443 = Gini_{income \in \{high\}}(D).$$

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; Can be modified for categorical attributes (like ex above)
- May need other tools, e.g., clustering, to get the possible split values

erfitting and Tree Pruning

rfitting: An induced tree may overfit the training data
oo many branches, actually fit anomalies due to noise or utliers
oor accuracy for unseen samples: cannot generalize
approaches to avoid overfitting
reprunning: *Halt tree construction early*-do not split a node if his would result in the goodness measure falling below a reshold
- Difficult to choose an appropriate threshold

ostprunning: *Remove branches* from a "fully grown" tree—get sequence of progressively pruned trees
- Use a set of data different from the training data to decide which is the "best pruned tree"

Naïve Bayes Classifier: An Example

- 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(C_i | X): 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")

37

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*
    Prob(income = low) = 1/1003
    Prob(income = medium) = 991/1003
    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts
  - "1" can be viewed as the **pseudo-count** to form priors in Bayesian analysis

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- **Recall**: completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F* measure (*F*$_1$ or *F*-score)**: harmonic mean of precision and recall,

$$\frac{1}{F} = \frac{\frac{1}{precision} + \frac{1}{recall}}{2}, \quad F = \frac{2 \times precision \times recall}{precision + recall}$$

- ***F*$_\beta$**: weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

ssifier Evaluation Metrics: Accuracy, Error te, Sensitivity and Specificity

A\P	C	¬C	
C	TP	FN	P
¬C	FP	TN	N
	P'	N'	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
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity = TP/P**
- **Specificity**: True Negative recognition rate
  - **Specificity = TN/N**

sifier **Accuracy**, or recognition : percentage of test set tuples t are correctly classified

ccuracy = (TP + TN)/All

or rate: *1 – accuracy*, or

ror rate = (FP + FN)/All

37

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use 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 dist. in each fold is approx. the same as that in the initial data

Model Selection: ROC Curves

- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- x-axis: false-positive rate (FPR), y-axis: true-positive rate (TPR): ROC shows the trade-off between FPR and TPR
- The area under the ROC curve (a.k.a AUC) is a measure of the accuracy of the model
- Plot ROC: for classifiers that can 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. Then calculate FPR/TPR for top-*m* tuples for each *m* in the ranked list
  - The plot also shows a diagonal line
  - A model with perfect accuracy will have an area of 1.0
  - The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

Bagging: Boostrap 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 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: can be applied to the prediction of continuous values by taking average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: more robust
  - Proved improved accuracy in prediction

Adaboost (Freund and Schapire, 1997)

en a set of *d* training tuples $(x_1, y_1), \ldots, (x_d, y_d)$, where label $y_i \in \{-1, 1\}$.

ally, the weight for tuple *j* is $w_1(j) = 1/d$

erate *k* classifiers in *k* rounds. At round *i*,
Tuples from *D* are sampled (with replacement) to form a training set D_i of size d. The chance of tuple *j* being selected is based on its weight $w_i(j)$.
A classification model M_i is derived from D_i

3. Compute the weighted error rate ϵ_i of M_i on *D*:

$$\varepsilon_i = \sum_{j=1}^{d} w_j \times err(M_i(x_j) \neq y_j)$$

1 if misclassified
0 otherwise

4. If tuple *j* is **correctly** classified, then adjust its weight:

$$w_{i+1}(j) = w_i(j) * \varepsilon_i / (1 - \varepsilon_i)$$

normalize w_{i+1} such that $\sum_{j=1}^{d} w_{i+1}(j) = 1$

Adaboost

- Finally combine *k* classifiers: to classify a new point *x*:
  1. Compute the weight for classifier *i*:

$$\alpha_i = \log\left(\frac{1 - \epsilon_i}{\epsilon_i}\right)$$

  2. Get the labels from each classifier and combine them using the above weight:

Weighted majority voting

If $\sum_{i=1}^{k} \alpha_i M_i(x) > 0$, classify *x* as 1; otherwise, classify *x* as -1;

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

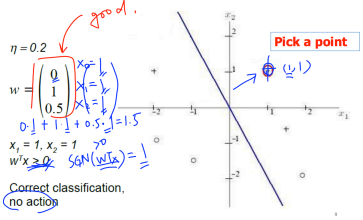
ceptron Algorithm

idomly initialize *w* and set η (learning rate) to be a small positive ue

idomly pick a pair (x, y) from the training set:
If $y \neq sign(w^T x)$, update $w = w + \eta xy$
Otherwise, no action

eat above procedure until the entire training set is classified rectly

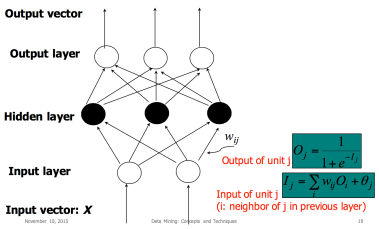
Learning Example



Pick a point

Correct classification, no action

How A Multi-Layer Neural Network Works

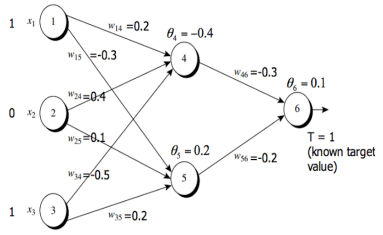


Output vector
Output layer
Hidden layer
Input layer
Input vector: X

w_{ij}

Output of unit j: $O_j = \frac{1}{1 + e^{-I_j}}$

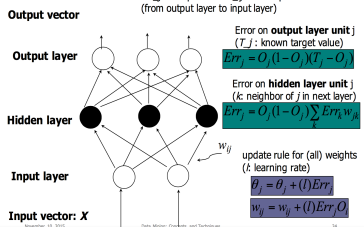
Input of unit j (i: neighbor of j in previous layer): $I_j = \sum_i w_{ij} O_i + \theta_j$



Input and Output Calculations

j	Net Input, I_j	Output, O_j
	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

Network Training: Backpropagation



Output vector
Output layer
Hidden layer
Input layer
Input vector: X

w_{ij}

Err_j and update on *w_ij* are computed **backwards** (from output layer to input layer)

Error on **output layer unit** j (*T_j* : known target value)

$Err_j = O_j(1 - O_j)(T_j - O_j)$

Error on **hidden layer unit** j (k: neighbor of j in next layer)

$Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk}$

update rule for (all) weights (*l*: learning rate)

$\theta_j = \theta_j + (l) Err_j$

$w_{ij} = w_{ij} + (l) Err_j O_i$

Example: backpropagation

Calculations for Weight and Bias Updating

Weight or Bias	New Value
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$

Neural Network as a Classifier

- **Efficiency** of backpropagation: Each epoch (one pass through the training se takes O(|D| * *w*), with |D| tuples and *w* weights.
- **Weakness**
  - Long training time
  - Often need to determine some parameters empirically, e.g., the network topology or "structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind t learned weights and of "hidden units" in the network
- **Strength**
  - High tolerance to noisy data
  - Ability to classify untrained patterns (good generalization?)
  - Well-suited for continuous-valued inputs *and outputs*
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel

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)

If all other training examples are removed and the training is repeated, the same separating hyperplane would be found

The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality

Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

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

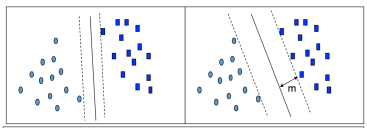
Discussion on the *k*-NN Algorithm

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the **mean value** of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - **Weight** the contribution of each of the *k* neighbors according to their distance to the query x_q
    - Give greater weight to closer neighbors

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by **averaging** *k*-nearest neighbors
- Curse of dimensionality: in high-dimensional space, nearest neighbors becomes far away– less representative of x_q

SVM—When Data Is Linearly Separable



Let data D be $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$, where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

ments on K-Means algorithm

ency: O(tKn) where n: # of objects, K: # of clusters, and t: # of ions

rmally, K, t << n; thus, an efficient method

to specify K, the number of clusters, in advance

ere are ways to automatically determine the "best" K

practice, one often runs a range of values and selected the "best" K value

icted usage:

plicable only to objects in a continuous n-dimensional space

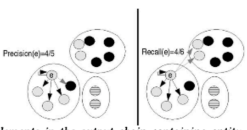
Using K-modes for categorical data

t suitable to discover clusters with non-convex shapes

Using density-based clustering, kernel K-means, etc. instead

r than that: still have at least two issues

B-Cubed Precision & Recall



$$\text{Precision}_i = \frac{\text{number of correct elements in the output chain containing entity}_i}{\text{number of elements in the output chain containing entity}_i}$$

$$\text{Recall}_i = \frac{\text{number of correct elements in the output chain containing entity}_i}{\text{number of elements in the truth chain containing entity}_i}$$

$$\text{Final Precision} = \sum_{i=1}^{N} w_i * \text{Precision}_i$$

$$\text{Final Recall} = \sum_{i=1}^{N} w_i * \text{Recall}_i$$

Note: The B-Cubed formula in the textbook is imprecise! Use this one (from [Bagga & Baldwin 1999])! 77

Constraint-Based Frequent Pattern Mining

Pattern space pruning constraints

- Anti-monotonic: If constraint c is violated, its further mining can be terminated
- Monotonic: If c is satisfied, no need to check c again
- Succinct: c must be satisfied, so one can start with the data sets satisfying c
- Convertible: c is not monotonic nor anti-monotonic, but it can be converted into it if items in the transaction can be properly ordered

Constraint-Based Mining — A General Picture

Constraint	Anti-monotone	Monotone	Succinct
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$\min(S) \le v$	no	yes	yes
$\min(S) \ge v$	yes	no	yes
$\max(S) \le v$	yes	no	yes
$\max(S) \ge v$	no	yes	yes
$\text{count}(S) \le v$	yes	no	weakly
$\text{count}(S) \ge v$	no	yes	weakly
$\text{sum}(S) \le v\ (a \in S, a \ge 0)$	yes	no	no
$\text{sum}(S) \ge v\ (a \in S, a \ge 0)$	no	yes	no
$\text{range}(S) \le v$	yes	no	no
$\text{range}(S) \ge v$	no	yes	no
$\text{avg}(S) \theta v, \theta \in \{<, \le, >\}$	convertible	convertible	no
$\text{support}(S) \ge \xi$	yes	no	no
$\text{support}(S) \le \xi$	no	yes	no

ido-Projection vs. Physical Projection

eudo-projection avoids physically copying postfixes

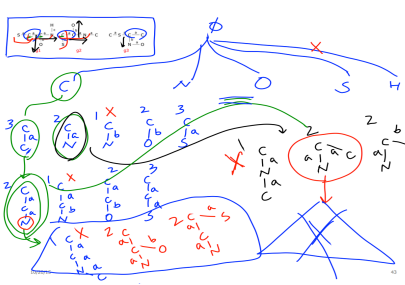
- Efficient in running time and space when database can be held in main memory

owever, it is not efficient when database cannot fit in main emory

- Disk-based random accessing is very costly

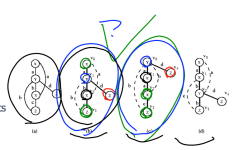
ggested Approach:

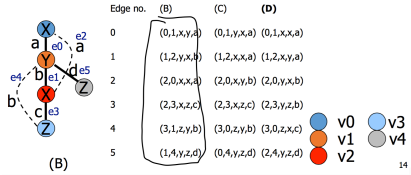
- Integration of physical and pseudo-projection
- Swapping to pseudo-projection when the data set fits in memory



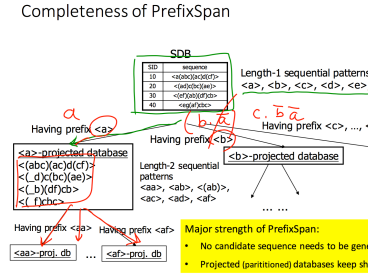
Minimum DFS code

Each Graph may have lots of DFS code (why?):
one smallest lexicographic one is its Minimum DFS Code





Completeness of PrefixSpan



oximity Measure for Binary Attributes

Object *j* -- Mary

Object *i* — Jack	1	0	sum
1	*q*	*r*	*q+r*
0	*s*	*t*	*s+t*
sum	*q+s*	*r+t*	*p*

ngency table for binary data

e measure for symmetric variables:

$$d(i,j) = \frac{r+s}{q+r+s+t} = (1+0)/(2+1+3+0)$$

e measure for asymmetric variables:

$$d(i,j) = \frac{r+s}{q+r+s}$$

coefficient (*similarity* e for *asymmetric* binary s):

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

accard coefficient is the same as "coherence":

$$coh(i,j) = \frac{sup(i,j)}{sup(i) + sup(j) - sup(i,j)} = \frac{q}{(q+r)+(q+s)-q}$$

Special Cases of Minkowski Distance

h = 1: Manhattan (city block, L_1 norm) distance

- E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

h = 2: (L_2 norm) Euclidean distance

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

h → ∞. "supremum" (L_{max} norm, L_∞ norm) distance.

- This is the maximum difference between any component (attribute) of the vectors

$$d(i,j) = \lim_{h\to\infty} \left(\sum_{f=1}^{p} |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f^p |x_{if} - x_{jf}|$$

Chi-Square Calculation: An Example

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

Numbers in parenthesis are expected counts calculated based on the data distribution in the two categories

- Example: Expected count of people playing chest and liking science fiction: 450 * 300 / 1500 = 90

X^2 (chi-square) calculation

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

Degree of freedom for the 2x2 table: (2-1)*(2-1) = 1 --> By looking up the Chi-Square table, we can reject the hypothesis like_science_fiction and play_chess are independent with high confidence→ they are correlated!

Correlation Analysis (for Numeric Data

Correlation coefficient (also called Pearson's product mon coefficient)

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\bar{A}\bar{B}}{n\sigma_A\sigma_B}$$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respe means of A and B, σ_A and σ_B are the respective standar deviations of A and B, and $\Sigma(a_i b_i)$ is the sum of the AB c product.

If $r_{A,B} > 0$, A and B are positively correlated (A's values incr as B's). The higher $r_{A,B}$, the stronger correlation.

$r_{A,B} = 0$: independent.

$r_{AB} < 0$: negatively correlated.

e random sampling

ere is an equal probability of selecting any particular item

ing without replacement

ce an object is selected, it is removed from the population

ing with replacement

elected object is not removed from the population

fied sampling:

rtition the data set, and draw samples from each partition oportionally, i.e., approximately the same percentage of e data)

ed in conjunction with skewed data

Min-max normalization: to [new_min$_A$, new_max$_A$]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

- Ex. Let income range $12,000 to $98,000 normalized to [0.0, 1.0]. Then $73,000 is mapped to $\frac{73{,}600-12{,}000}{98{,}000-12{,}000}(1.0-0)+0 = 0.716$

Z-score normalization (μ: mean, σ: standard deviation):

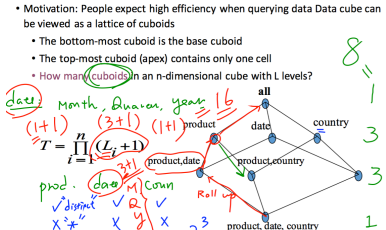
$$v' = \frac{v - \mu_A}{\sigma_A}$$

- Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600-54{,}000}{16{,}000} = 1.225$

Efficient Data Cube Computation

- Motivation: People expect high efficiency when querying data Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels?

$$T = \prod_{i=1}^{n}(L_i + 1)$$



- Close cube:
  - Closed cell c: if there exists no cell d, s.t. d is a descendant of c, and d the same measure value as c.
  - Closed cube: a cube consisting of only closed cells
  - Given 2 base cells: {(a_1, a_2, a_3 . . . , a_{100}):10, (a_1, a_2, b_3, . . , b_{100}):10}, wh the closed cube of the base cuboid? Hint: only 3 cells

losed Patterns and Max Patterns

temset X is closed if X is *frequent* and there exists *no super-ern* X' ⊃ X that *has the same support as* X.

temset X is a max pattern if X is frequent and there exists no er-pattern X' ⊃ X that *is also frequent* (i.e., >= minsup).

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

Find all rules X → Y with minimum support and confidence

- support, s, probability that a transaction contains X ∪ Y
  - s=#T(X∪Y)/#T(Total)
- confidence, c, conditional probability that a transaction having X also contains Y
  - c=#T(X∪Y)/#T(X)

Let minsup = 50%, minconf = 50%

Association rules: (many more!)
- Beer → Diaper (60%, 100%)
- Diaper → Beer (60%, 75%)

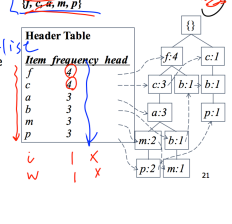
(A rule X→Y is an association rule if
- s(X→Y) is no less than minsup and
- c(X→Y) is no less than minconf

Construct FP-tree from a Transaction Database

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

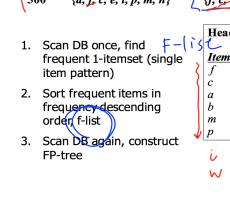


Construct FP-tree from a Transaction Database

TID	Items bought	(ordered) frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

min_support = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree



erestingness Measure: Correlations (Lift)

y basketball ⇒ *eat cereal* [40%, 66.7%] is misleading

The overall % of students eating cereal is 75% > 66.7%.

y basketball ⇒ *not eat cereal* [20%, 33.3%] is more accurate, although with ver support and confidence

asure of dependent/correlated events: lift

$$lift = \frac{P(A \cup B)}{P(A)P(B)}$$

$$\frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

Conditions for stopping partitioning (decision tree induction) • All samples for a given node belong to the same class • There are no remaining attributes for further partitioning –majority voting is employed for classifying the leaf • There are no samples left
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
Decision tree: • relatively faster learning speed (than other classificationmethods) • convertible to simple and easy to understand classification rules • can use SQL queries for accessing databases • comparable classification accuracy with other methods
Naive Bayes Classifier: • **Advantages** Easy to implement; Good results obtained in most of the cases • **Disadvantages** Assumption: class conditional independence, therefore loss of accuracy; Practically, dependencies exist among variables; Dependencies among these cannot be modeled by Naive Bayes Classifier; Deal with these dependencies: Bayesian Belief Networks **Methods for estimating a classifier's accuracy:** Holdout method, random subsampling; Cross-validation; Bootstrap
Comparing classifiers: Confidence intervals; Cost-benefit analysis and ROC Curves
Popular ensemble methods • Bagging: averaging the prediction over a collection of classifiers • Boosting: weighted vote with a collection of classifiers (in the same family, e.g, Naïve Bayes) • Ensemble: combining a set of heterogeneous classifiers
Perceptron algorithm • Advantages: Simple and computationally efficient; Guaranteed to learn a linearly separable problem (convergence, global optimum) • Limitations: Only linear separations;Only converges for linearly separable data;Not really "efficient w many features" **SVM** Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
K-means Issues 1.K-means often terminates at a local optimal: initialization can be important to find high-quality clusters 2. K-Means is sensitive to noisy data and outliers
Efficiency issue of K-Medoids • PAM(Partitioning Around Medoids) does not scale well for large dataset • Efficiency: PAM: O(k(n-k)2) VS. K-Means O(….normally, k, t << n
DBSCAN issue: Sensitive to Parameters **Attribute**: nominal, binary, ordinal, numeric:quantitative(interval-scaled, ratio-scaled) **Positively skewed**: mode<median<mean **Negatively**: mean<median<mode **Graphic Display of basic statistical descriptions**
Boxplot: min,Q1,median,Q3,max IQR: Q3-Q1; **Histogram**: frequency V.S. **Bar Chart**: area (Histograms often tell more than boxplots, the two histograms shown in the left may have the same boxplot representation, but they have rather different data distributions); **Quantile-plot (x-quantile information); Quantile-Quantile Plot**; **Scatterplot** **Categorization of visualization techniques:** Pixel-oriented, Geometric projection, Icon-based, Hierachical, Visualizing complex data and relations.
Redundant attributes may be able to be detected by correlation analysis and covariance analysis **Modeling data warehouse:** Star schema; snowflake schema; fact constellations
Data Cube Measures: **Distributive**(count, min,max,sum); **Algebraic**(agv, MaxN,MinN,standard deviation,center of mass); **Holistic**(median, mode,rank) **OLAP Operations:** Roll up(summarize); Drill down(-->detailed); Slice and dice(project and select); pivot(rotat