CS6220: DATA MINING TECHNIQUES

Set Data: Frequent Pattern Mining

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

November 1, 2015

Reminder

- Midterm
  - Next Monday (Nov. 9), 2-hour (6-8pm) in class
  - Closed-book exam, and one A4 size reference sheet is allowed
  - Bring a calculator (NO cell phone)
  - Cover to today's lecture
- Homework #3 is out tomorrow

Methods to Learn

	Matrix Data	Text Data	Set Data	Sequence Data	Time Series	Graph & Network	Images
Classification	Decision Tree; Naïve Bayes; Logistic Regression SVM; kNN			HMM		Label Propagation*	Neural Network
Clustering	K-means; hierarchical clustering; DBSCAN; Mixture Models; kernel k-means*	PLSA				SCAN*; Spectral Clustering*	
Frequent Pattern Mining			**Apriori; FP-growth**	GSP; PrefixSpan			
Prediction	Linear Regression				Autoregression		
Similarity Search					DTW	P-PageRank	
Ranking						PageRank	

Mining Frequent Patterns, Association and Correlations

- Basic Concepts
- Frequent Itemset Mining Methods
- Pattern Evaluation Methods
- Summary

Set Data

- A data point corresponds to a set of items

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

What Is Frequent Pattern Analysis?

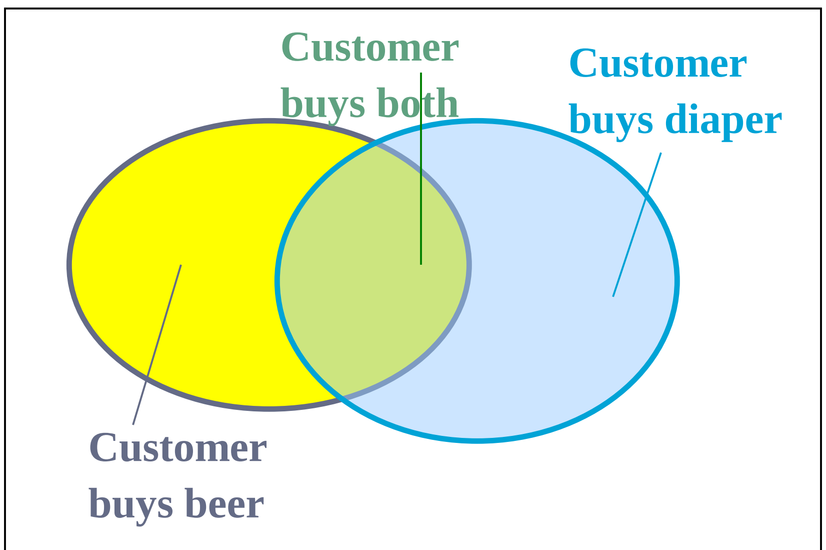
- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
  - First proposed by Agrawal, Imielinski, and Swami [AIS93] in the context of frequent itemsets and association rule mining
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?

Why Is Freq. Pattern Mining Important?

- Freq. pattern: An intrinsic and important property of datasets
- Foundation for many essential data mining tasks
  - Association, correlation, and causality analysis
  - Sequential, structural (e.g., sub-graph) patterns
  - Pattern analysis in spatiotemporal, multimedia, time-series, and stream data
  - Classification: discriminative, frequent pattern analysis
  - Cluster analysis: frequent pattern-based clustering
  - Broad applications

Basic Concepts: Frequent Patterns

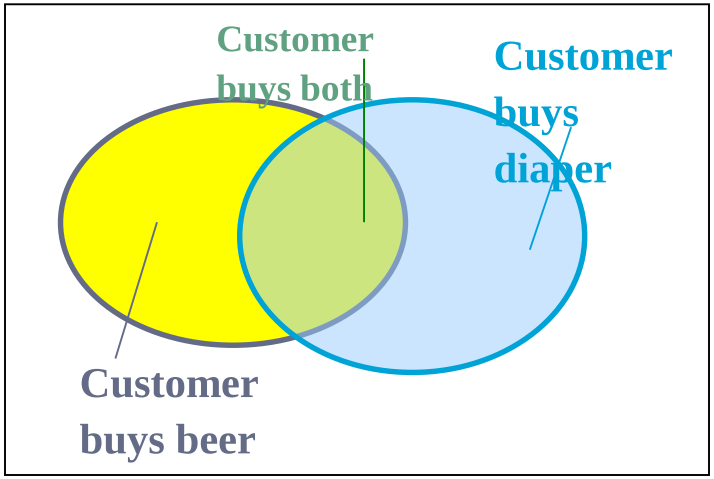
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold

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 the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Strong Association rules
  - *Beer* → *Diaper* (60%, 100%)
  - *Diaper* → *Beer* (60%, 75%)

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{a_1, \ldots, a_{100}\}$ contains $2^{100} - 1 = 1.27*10^{30}$ sub-patterns!
- Solution: *Mine closed patterns and max-patterns instead*
- An itemset X is closed if X is *frequent* and there exists *no super-pattern* Y ↄ X, *with the same support* as X (proposed by Pasquier, et al. @ ICDT'99)
- An itemset X is a max-pattern if X is frequent and there exists no frequent super-pattern Y ↄ X (proposed by Bayardo @ SIGMOD'98)
- Closed pattern is a lossless compression of freq. patterns
  - Reducing the # of patterns and rules

Closed Patterns and Max-Patterns

- Exercise. DB = {$<a_1, \ldots, a_{100}>$, $< a_1, \ldots, a_{50}>$}
  - Min_sup = 1.
- What is the set of closed pattern(s)?
  - $<a_1, \ldots, a_{100}>$: 1
  - $< a_1, \ldots, a_{50}>$: 2
- What is the set of max-pattern(s)?
  - $<a_1, \ldots, a_{100}>$: 1
- What is the set of all patterns?
  - !!

Computational Complexity of Frequent Itemset Mining

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is sensitive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions

Mining Frequent Patterns, Association and Correlations

- Basic Concepts
- Frequent Itemset Mining Methods
- Pattern Evaluation Methods
- Summary

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
  - Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Generating Association Rules

The Apriori Property and Scalable Mining Methods

- The Apriori property of frequent patterns
  - Any nonempty subsets of a frequent itemset must be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}
- Scalable mining methods: Three major approaches
  - Apriori (Agrawal & Srikant@VLDB'94)
  - Freq. pattern growth (FPgrowth—Han, Pei & Yin @SIGMOD'00)
  - Vertical data format approach (Eclat)

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

From Frequent k-1 Itemset To Frequent k-Itemset

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

- From L_{k-1} to C_k (Candidates Generation)
  - The join step
  - The prune step
- From C_k to L_k
  - Test candidates by scanning database

Candidates Generation

Assume a pre-specified order for items, e.g., alphabetical order

- How to generate candidates C_k?
  - Step 1: self-joining L_{k-1}
    - Two length k-1 itemsets l_1 and l_2 can join, only if the first k-2 items are the same, and for the last term, $l_1[k-1] < l_2[k-1]$ (why?)
  - Step 2: pruning
    - Why we need pruning for candidates?
    - How?
      - Again, use Apriori property
      - A candidate itemset can be safely pruned, if it contains infrequent subset

- Example of Candidate-generation from L_3 *to* C_4
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: L_3*L_3
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4= {*abcd*}

The Apriori Algorithm—Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 2; L_{k-1} !=∅; k++) **do begin**
  C_k = candidates generated from L_{k-1};
  for each transaction t in database do
    increment the count of all candidates in C_{k+1} that are contained in t
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

Questions

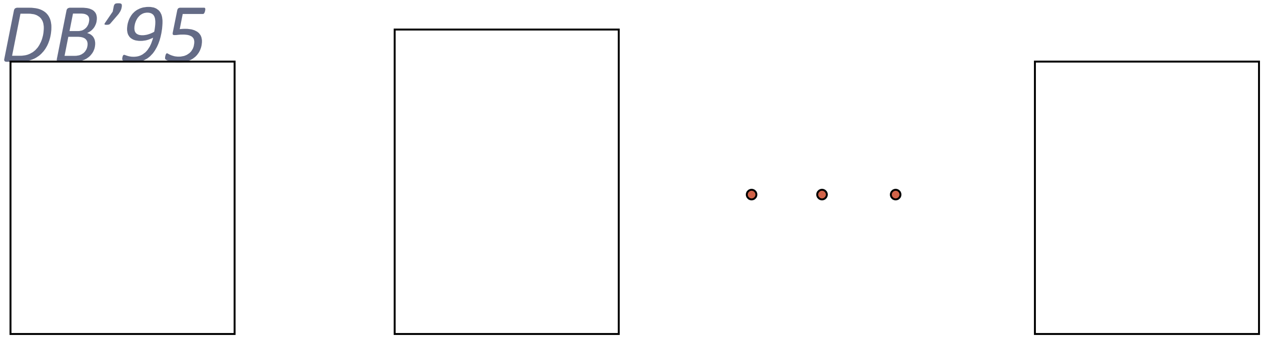
- How many scans on DB are needed for Apriori algorithm?
- When (k = ?) does Apriori algorithm generate most candidate itemsets?
  - Is support counting for candidates expensive?

Further Improvement of the Apriori Method

- Major computational challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce passes of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates

*Partition: Scan Database Only Twice

- Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
  - Scan 1: partition database and find local frequent patterns
  - Scan 2: consolidate global frequent patterns
- A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*



$DB_1 + DB_2 + \cdots + DB_k = DB$

$sup_1(i) < \sigma DB_1$ $\quad sup_2(i) < \sigma DB_2$ $\quad sup_k(i) < \sigma DB_k$ $\quad sup(i) < \sigma DB$

*Hash-based Technique: Reduce the Number of Candidates

- A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
  - Candidates: a, b, c, d, e
  - Hash entries
    - {ab, ad, ae}
    - {bd, be, de}
    - …
  - Frequent 1-itemset: a, b, d, e
  - ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold

count	itemsets
35	{ab, ad, ae}
88	{bd, be, de}
. . .	. . .
102	{yz, qs, wt}

Hash Table

- J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD'95*

*Sampling for Frequent Patterns

- Select a sample of original database, mine frequent patterns within sample using Apriori
- Scan database once to verify frequent itemsets found in sample, only *borders* of closure of frequent patterns are checked
  - Example: check *abcd* instead of *ab, ac, ..., etc.*
- Scan database again to find missed frequent patterns
- H. Toivonen. Sampling large databases for association rules. In *VLDB'96*

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
  - Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Generating Association Rules

Pattern-Growth Approach: Mining Frequent Patterns Without Candidate Generation

- Bottlenecks of the Apriori approach
  - Breadth-first (i.e., level-wise) search
    - Scan DB multiple times
  - Candidate generation and test
    - Often generates a huge number of candidates
- The FPGrowth Approach (J. Han, J. Pei, and Y. Yin, SIGMOD' 00)
  - Depth-first search
  - Avoid explicit candidate generation

Major philosophy

- Grow long patterns from short ones using local frequent items only
  - "abc" is a frequent pattern
  - Get all transactions having "abc", i.e., project DB on abc: DB|abc
  - "d" is a local frequent item in DB|abc → abcd is a frequent pattern

FP-Growth Algorithm Sketch

- Construct FP-tree (frequent pattern-tree)
  - Compress the DB into a tree
- Recursively mine FP-tree by FP-Growth
  - Construct conditional pattern base from FP-tree
  - Construct conditional FP-tree from conditional pattern base
  - Until the tree has a single path or empty

Construct FP-tree from a Transaction Database

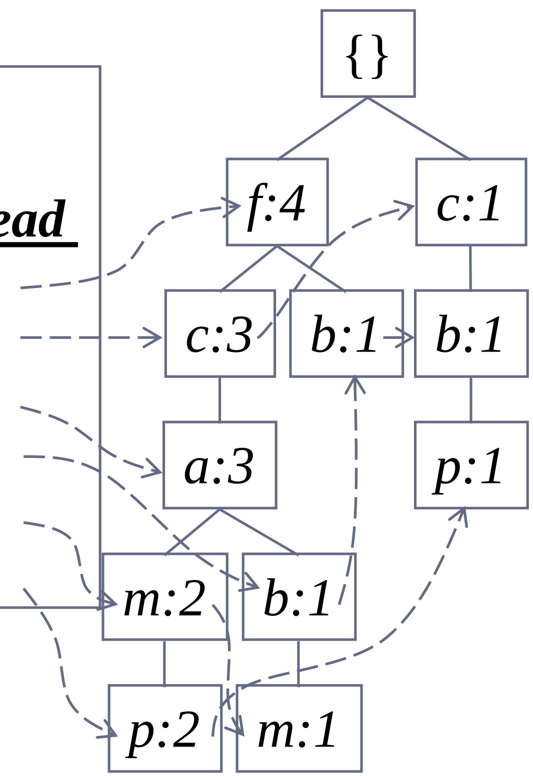
TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

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

Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	



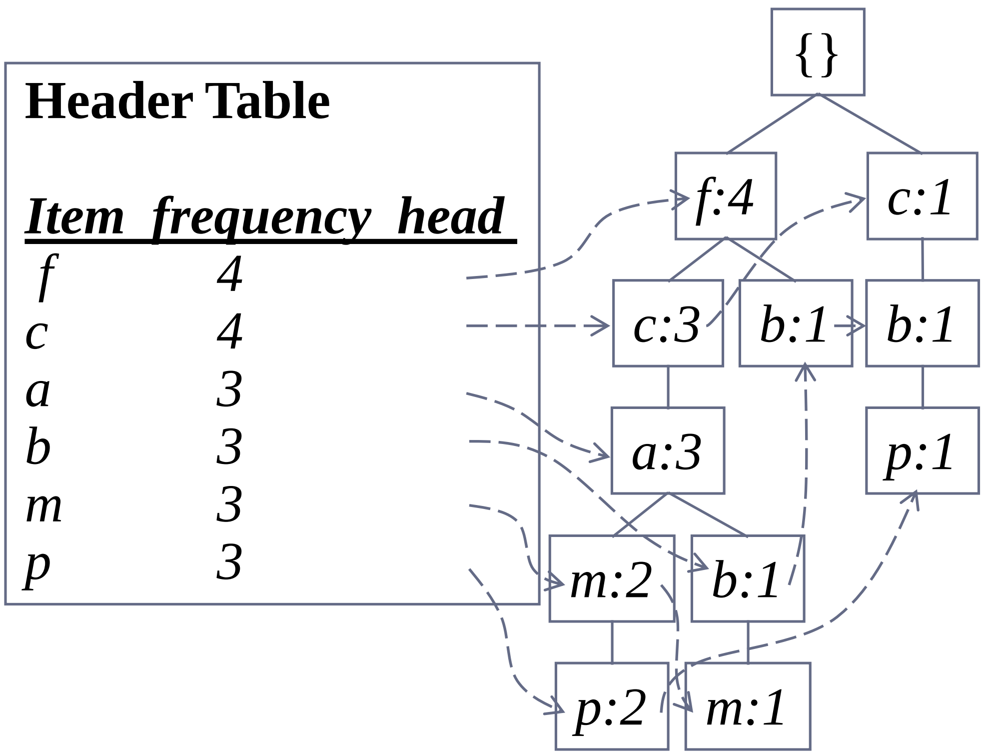
F-list = f-c-a-b-m-p

Partition Patterns and Databases

- Frequent patterns can be partitioned into subsets according to f-list
  - F-list = f-c-a-b-m-p
  - Patterns containing p
  - Patterns having m but no p
  - …
  - Patterns having c but no a nor b, m, p
  - Pattern f
- Completeness and non-redundency

Find Patterns Having P From P-conditional Database

- Starting at the frequent item header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item *p*
- Accumulate all of *transformed prefix paths* of item *p* to form *p*'s conditional pattern base

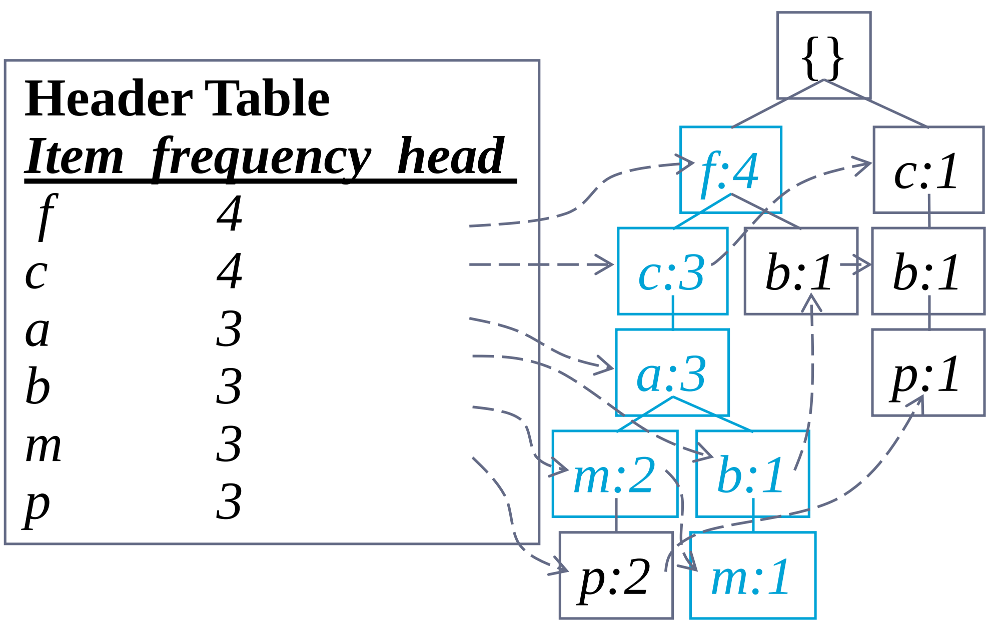


***Conditional* pattern bases**

item	***cond. pattern base***
c	***f:3***
a	***fc:3***
b	***fca:1, f:1, c:1***
m	***fca:2, fcab:1***
p	***fcam:2, cb:1***

From Conditional Pattern-bases to Conditional FP-trees

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base



m-conditional **pattern base:**

fca:2, fcab:1

{}

f:3

c:3

a:3

m-conditional **FP-tree**

All frequent patterns relate to *m*

m,

fm, cm, am,

fcm, fam, cam,

fcam

Don't forget to add back m!

Recursion: Mining Each Conditional FP-tree

{}
|
f:3
|
c:3
|
a:3

m-conditional **FP-tree**

Cond. pattern base of "am": (fc:3)

{}
|
f:3
|
c:3

am-conditional **FP-tree**

Cond. pattern base of "cm": (f:3)

{}
|
f:3

cm-conditional **FP-tree**

Cond. pattern base of "cam": (f:3)

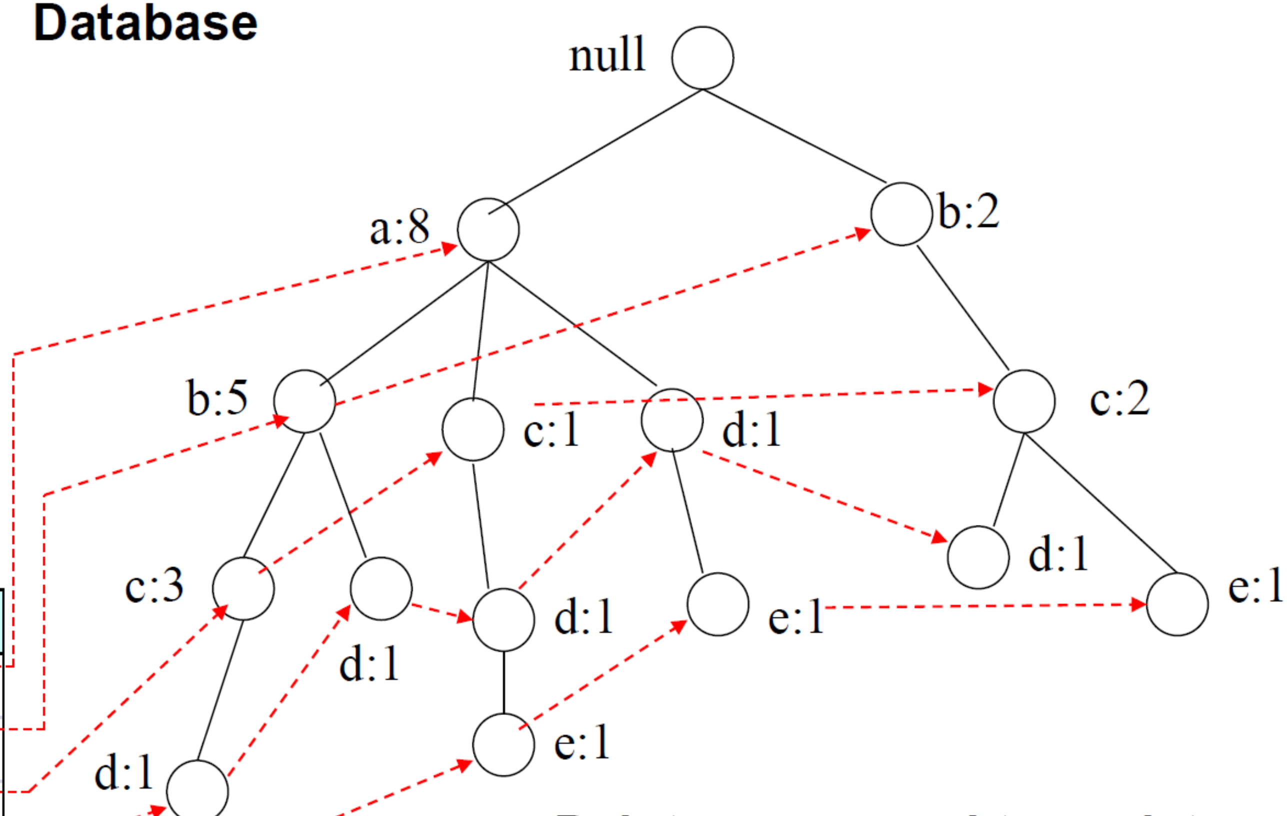
{}
|
f:3

cam-conditional **FP-tree**

Another Example: FP-Tree Construction

TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

Transaction Database



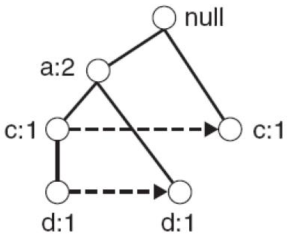
Header table

Item	Pointer
a	
b	
c	
d	
e	

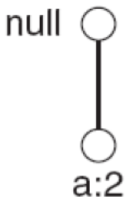
Pointers are used to assist frequent itemset generation

Mining Sub-tree Ending with e

- Conditional pattern base for e: {acd:1; ad:1; bc:1}
- Conditional FP-tree for e:



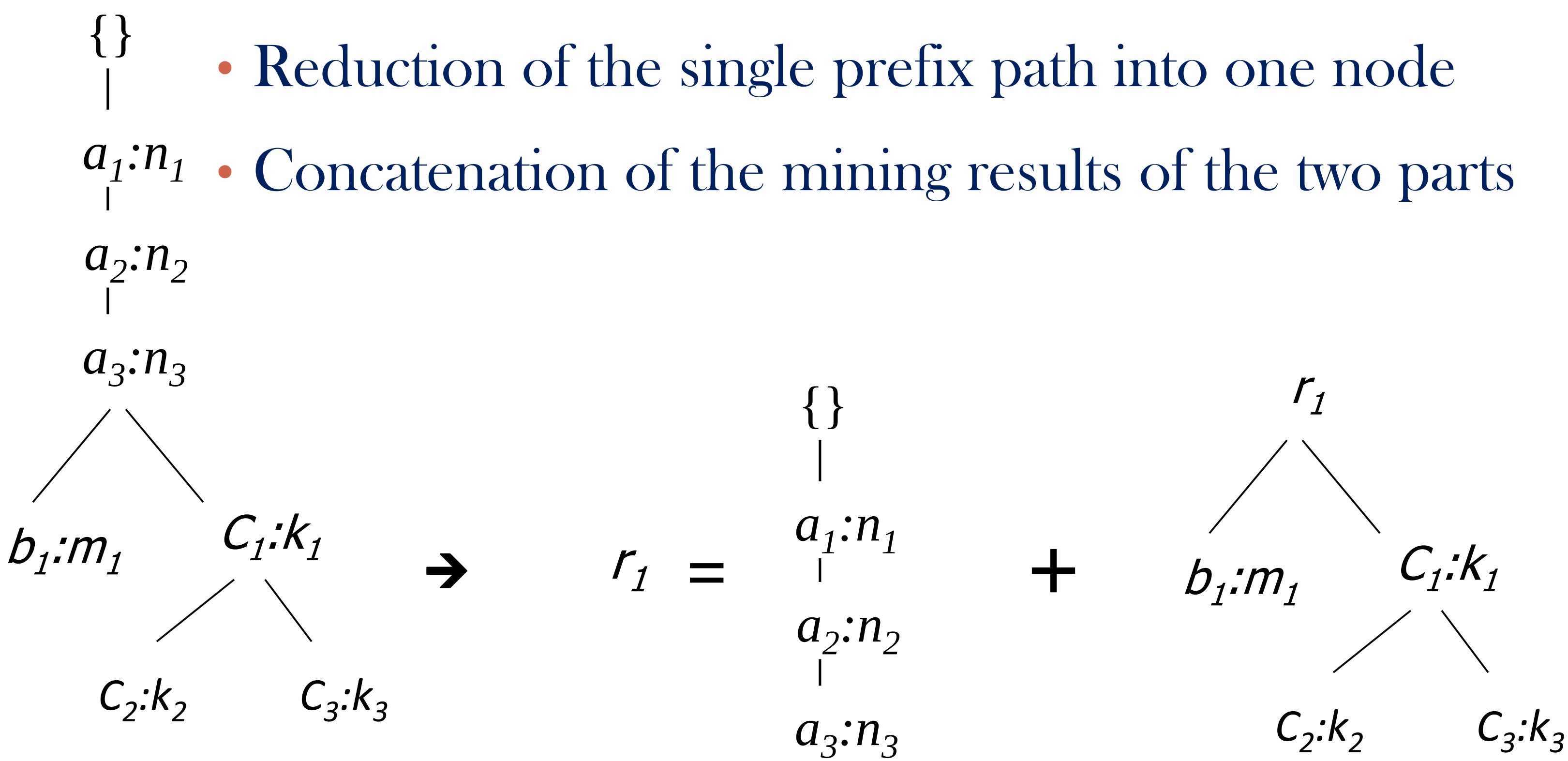
  - Conditional pattern base for de: {ac:1; a:1}
  - Conditional FP-tree for de:



  - Frequent patterns for de: {ade:2, de:2}
  - Conditional pattern base for ce: {a:1}
  - Conditional FP-tree for ce: empty
  - Frequent patterns for ce: {ce:2}
  - Conditional pattern base for ae: {∅}
  - Conditional FP-tree for ae: empty
  - Frequent patterns for ae: {ae:2}
- Therefore, all frequent patterns with e are: {ade:2, de:2, ce:2, ae:2, e:3}

A Special Case: Single Prefix Path in FP-tree

- Suppose a (conditional) FP-tree T has a shared single prefix-path P
- Mining can be decomposed into two parts
  - Reduction of the single prefix path into one node
  - Concatenation of the mining results of the two parts



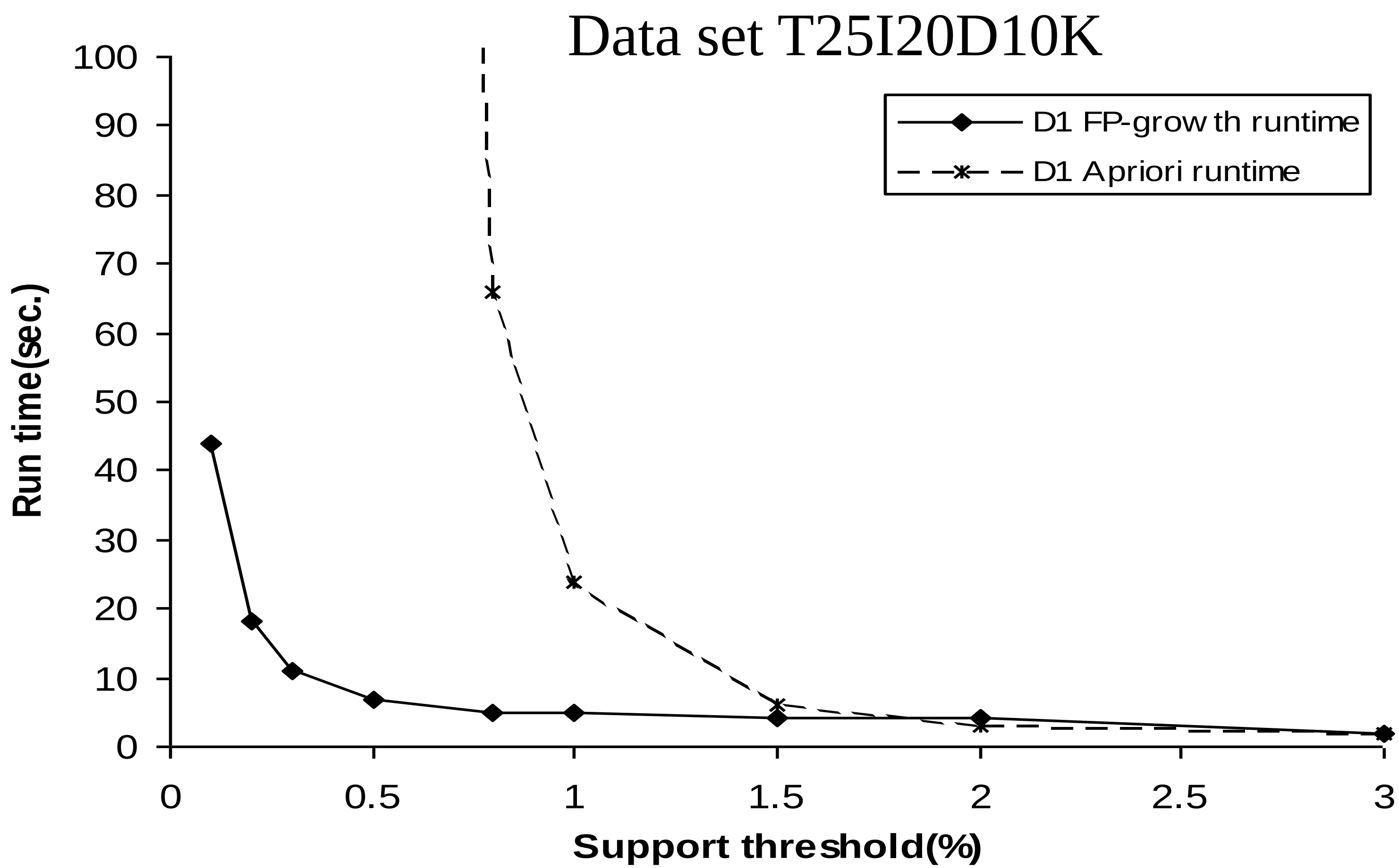
Benefits of the FP-tree Structure

- Completeness
  - Preserve complete information for frequent pattern mining
  - Never break a long pattern of any transaction
- Compactness
  - Reduce irrelevant info—infrequent items are gone
  - Items in frequency descending order: the more frequently occurring, the more likely to be shared
  - Never be larger than the original database (not count node-links and the *count* field)

*Scaling FP-growth by Database Projection

- What about if FP-tree cannot fit in memory?
  - DB projection
- First partition a database into a set of projected DBs
- Then construct and mine FP-tree for each projected DB
- Parallel projection vs. partition projection techniques
  - Parallel projection
    - Project the DB in parallel for each frequent item
    - Parallel projection is space costly
    - All the partitions can be processed in parallel
  - Partition projection
    - Partition the DB based on the ordered frequent items
    - Passing the unprocessed parts to the subsequent partitions

FP-Growth vs. Apriori: Scalability With the Support Threshold



Advantages of the Pattern Growth Approach

- Divide-and-conquer:
  - Decompose both the mining task and DB according to the frequent patterns obtained so far
  - Lead to focused search of smaller databases
- Other factors
  - No candidate generation, no candidate test
  - Compressed database: FP-tree structure
  - No repeated scan of entire database
  - Basic ops: counting local freq items and building sub FP-tree, no pattern search and matching

*Further Improvements of Mining Methods

- AFOPT (Liu, et al. @ KDD'03)
  - A "push-right" method for mining condensed frequent pattern (CFP) tree
- Carpenter (Pan, et al. @ KDD'03)
  - Mine data sets with small rows but numerous columns
  - Construct a row-enumeration tree for efficient mining
- FPgrowth+ (Grahne and Zhu, FIMI'03)
  - Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003
- TD-Close (Liu, et al, SDM'06)

*Extension of Pattern Growth Mining Methodology

- Mining closed frequent itemsets and max-patterns
  - CLOSET (DMKD'00), FPclose, and FPMax (Grahne & Zhu, Fimi'03)
- Mining sequential patterns
  - PrefixSpan (ICDE'01), CloSpan (SDM'03), BIDE (ICDE'04)
- Mining graph patterns
  - gSpan (ICDM'02), CloseGraph (KDD'03)
- Constraint-based mining of frequent patterns
  - Convertible constraints (ICDE'01), gPrune (PAKDD'03)
- Computing iceberg data cubes with complex measures
  - H-tree, H-cubing, and Star-cubing (SIGMOD'01, VLDB'03)
- Pattern-growth-based Clustering
  - MaPle (Pei, et al., ICDM'03)
- Pattern-Growth-Based Classification
  - Mining frequent and discriminative patterns (Cheng, et al, ICDE'07)

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
  - Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Generating Association Rules

ECLAT: Mining by Exploring Vertical Data Format

Similar idea for inverted index in storing text

- Vertical format: $t(AB) = \{T_{11}, T_{25}, \ldots\}$
  - tid-list: list of trans.-ids containing an itemset
- Deriving frequent patterns based on vertical intersections
  - $t(X) = t(Y)$: X and Y always happen together
  - $t(X) \subset t(Y)$: transaction having X always has Y
- Using diffset to accelerate mining
  - Only keep track of differences of tids
  - $t(X) = \{T_1, T_2, T_3\}$, $t(XY) = \{T_1, T_3\}$
  - Diffset $(XY, X) = \{T_2\}$
- Eclat (Zaki et al. @KDD'97)

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach
  - Improving the Efficiency of Apriori
- FPGrowth: A Frequent Pattern-Growth Approach
- ECLAT: Frequent Pattern Mining with Vertical Data Format
- Generating Association Rules

Generating Association Rules

- Strong association rules
  - Satisfying minimum support and minimum confidence
  - Recall: $Confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)}$
- Steps of generating association rules from frequent pattern l:
  - Step 1: generate all nonempty subsets of l
  - Step 2: for every nonempty subset s, calculate the confidence for rule $s \Rightarrow (l - s)$

Example

- $X = \{I1, I2, I5\}$:2
  - Nonempty subsets of X are: $\{I1, I2\}$: 4, $\{I1, I5\}$: 2, $\{I2, I5\}$: 2, $\{I1\}$: 6, $\{I2\}$: 7, *and* $\{I5\}$: 2
  - Association rules are:

$$
\begin{array}{ll}
\{I1, I2\} \Rightarrow I5, & confidence = 2/4 = 50\% \\
\{I1, I5\} \Rightarrow I2, & confidence = 2/2 = 100\% \\
\{I2, I5\} \Rightarrow I1, & confidence = 2/2 = 100\% \\
I1 \Rightarrow \{I2, I5\}, & confidence = 2/6 = 33\% \\
I2 \Rightarrow \{I1, I5\}, & confidence = 2/7 = 29\% \\
I5 \Rightarrow \{I1, I2\}, & confidence = 2/2 = 100\%
\end{array}
$$

Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic Concepts
- Frequent Itemset Mining Methods
- Pattern Evaluation Methods
- Summary

Misleading Strong Association Rules

- Not all strong association rules are interesting

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

- Shall we target people who play basketball for cereal ads? *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%]
- Hint: What is the overall probability of people who eat cereal?
  - 3750/5000 = 75% > 66.7%!
- Confidence measure of a rule could be misleading

Other Measures

- From association to correlation
  - Lift
  - χ^2
  - All_confidence
  - Max_confidence
  - Kulczynski
  - Cosine

Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75% > 66.7%.
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

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

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

$$lift(B, C) = \frac{2000/5000}{3000/5000 * 3750/5000} = 0.89$$

$$lift(B, \neg C) = \frac{1000/5000}{3000/5000 * 1250/5000} = 1.33$$

1: independent
>1: positively correlated
<1: negatively correlated

Correlation Analysis (Nominal Data)

- χ^2 **(chi-square) test**

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- Independency test between two attributes
  - The larger the χ^2 value, the more likely the variables are related
- The cells that contribute the most to the χ^2 value are those whose actual count is very different from the expected count under independence assumption
- Correlation does not imply causality
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

When Do We Need Chi-Square Test?

- Considering two attributes A and B
  - A: a nominal attribute with c distinct values, $a_1, \dots, a_c$
    - E.g., Grades of Math
  - B: a nominal attribute with r distinct values, $b_1, \dots, b_r$
    - E.g., Grades of Science
- Question: Are A and B related?

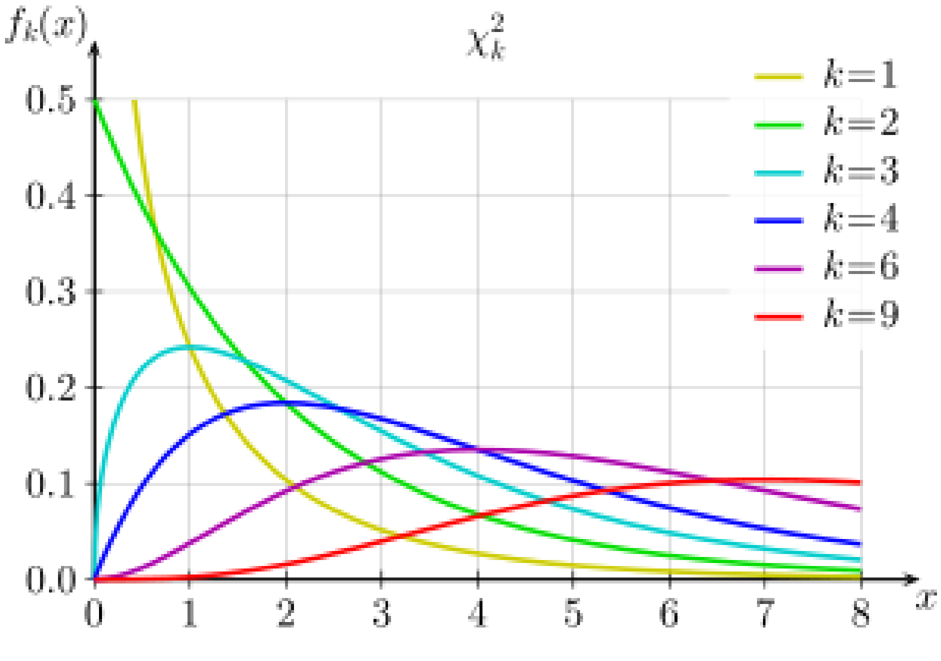
How Can We Run Chi-Square Test?

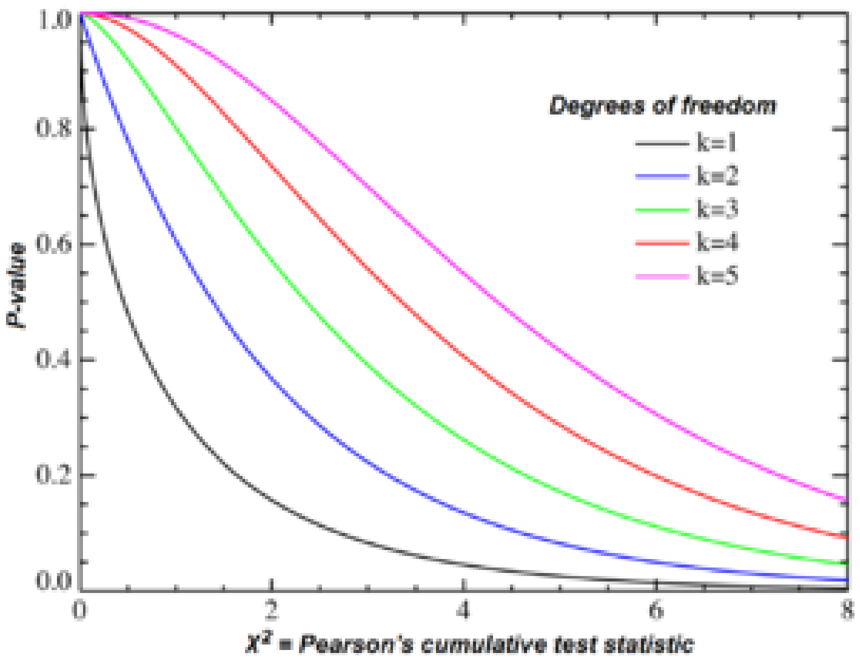
- Constructing contingency table
  - Observed frequency o_{ij}: number of data objects taking value b_i for attribute B and taking value a_j for attribute A

	a_1	a_2	…	a_c
b_1	o_{11}	o_{12}	…	o_{1c}
b_2	o_{21}	o_{22}	…	o_{2c}
…	…	…	…	…
b_r	o_{r1}	o_{r2}	…	o_{rc}

- Calculate expected frequency $e_{ij} = \frac{count(B=b_i) \times count(A=a_j)}{n}$
  - Null hypothesis: A and B are independent

- The Pearson χ^2 statistic is computed as:
  - $X^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$
  - Follows Chi-squared distribution with degree of freedom as $(r-1) \times (c-1)$





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

- χ^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

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

- It shows that like_science_fiction and play_chess are correlated in the group
  - Degree of freedom = (2-1)(2-1) = 1
  - P-value = $P(X^2 > 507.93) = 0.0$
    - Reject the null hypothesis => A and B are dependent

Are *lift* and χ^2 Good Measures of Correlation?

- Lift and χ^2 are affected by null-transaction
  - E.g., number of transactions that do not contain milk nor coffee
- All_confidence
  - $all\_conf(A,B)=\min\{P(A|B),P(B|A)\}$
- Max_confidence
  - $max\_conf(A,B)=\max\{P(A|B),P(B|A)\}$
- Kulczynski
  - $Kulc(A,B) = \frac{1}{2}(P(A|B) + P(B|A))$
- Cosine
  - $cosine(A,B) = \sqrt{P(A|B) \times P(B|A)}$

Comparison of Interestingness Measures

- Null-(transaction) invariance is crucial for correlation analysis
- Lift and χ^2 are not null-invariant
- 5 null-invariant measures

	Milk	No Milk	Sum (row)
Coffee	m, c	~m, c	c
No Coffee	m, ~c	~m, ~c	~c
Sum(col.)	m	~m	Σ

Measure	Definition	Range	Null-Invariant
$\chi^2(a,b)$	$\sum_{i,j=0,1} \frac{(e(a_i,b_j)-o(a_i,b_j))^2}{e(a_i,b_j)}$	$[0,\infty]$	No
$Lift(a,b)$	$\frac{P(ab)}{P(a)P(b)}$	$[0,\infty]$	No
$AllConf(a,b)$	$\frac{sup(ab)}{max\{sup(a),sup(b)\}}$	$[0,1]$	Yes
$Coherence(a,b)$	$\frac{sup(ab)}{sup(a)+sup(b)-sup(ab)}$	$[0,1]$	Yes
$Cosine(a,b)$	$\frac{sup(ab)}{\sqrt{sup(a)sup(b)}}$	$[0,1]$	Yes
$Kulc(a,b)$	$\frac{sup(ab)}{2}(\frac{1}{sup(a)}+\frac{1}{sup(b)})$	$[0,1]$	Yes
$MaxConf$(a,b)	$max\{\frac{sup(ab)}{sup(a)},\frac{sup(ab)}{sup(b)}\}$	$[0,1]$	Yes

Table 3. Interestingness measure definitions.

Null-transactions w.r.t. m and c

Kulczynski measure (1927)

Null-invariant

Data set	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	χ^2	$Lift$	$AllConf$	$Coherence$	$Cosine$	$Kulc$	$MaxConf$
D_1	10,000	1,000	1,000	100,000	90557	9.26	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0	1	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	670	8.44	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	24740	25.75	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	8173	9.18	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	965	1.97	0.01	0.01	0.10	0.5	0.99

Table 2. Example data sets.

Subtle: They disagree

*Analysis of DBLP Coauthor Relationships

Recent DB conferences, removing balanced associations, low sup, etc.

ID	Author a	Author b	$sup(ab)$	$sup(a)$	$sup(b)$	$Coherence$	$Cosine$	$Kulc$
1	Hans-Peter Kriegel	Martin Ester	28	146	54	0.163 (2)	0.315 (7)	0.355 (9)
2	Michael Carey	Miron Livny	26	104	58	0.191 (1)	0.335 (4)	0.349 (10)
3	Hans-Peter Kriegel	Joerg Sander	24	146	36	0.152 (3)	0.331 (5)	0.416 (8)
4	Christos Faloutsos	Spiros Papadimitriou	20	162	26	0.119 (7)	0.308 (10)	0.446 (7)
5	Hans-Peter Kriegel	Martin Pfeifle	18	146	18	0.123 (6)	0.351 (2)	0.562 (2)
6	Hector Garcia-Molina	Wilburt Labio	16	144	18	0.110 (9)	0.314 (8)	0.500 (4)
7	Divyakant Agrawal	Wang Hsiung	16	120	16	0.133 (5)	0.365 (1)	0.567 (1)
8	Elke Rundensteiner	Murali Mani	16	104	20	0.148 (4)	0.351 (3)	0.477 (6)
9	Divyakant Agrawal	Oliver Po	12	120	12	0.100 (10)	0.316 (6)	0.550 (3)
10	Gerhard Weikum	Martin Theobald	12	106	14	0.111 (8)	0.312 (9)	0.485 (5)

Table 5. Experiment on DBLP data set.

Advisor-advisee relation: Kulc: high, coherence: low, cosine: middle

- Tianyi Wu, Yuguo Chen and Jiawei Han, "[Association Mining in Large Databases: A Re-Examination of Its Measures]()", Proc. 2007 Int. Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD'07), Sept. 2007

*Which Null-Invariant Measure Is Better?

- IR (Imbalance Ratio): measure the imbalance of two itemsets A and B in rule implications

$$IR(A,B) = \frac{|sup(A) - sup(B)|}{sup(A) + sup(B) - sup(A \cup B)}$$

- Kulczynski and Imbalance Ratio (IR) together present a clear picture for all the three datasets D_4 through D_6
  - D_4 is balanced & neutral
  - D_5 is imbalanced & neutral
  - D_6 is very imbalanced & neutral

$Data$	mc	$\overline{m}c$	$m\overline{c}$	$\overline{mc}$	$all\_conf.$	$max\_conf.$	$Kulc.$	$cosine$	IR
D_1	10,000	1,000	1,000	100,000	0.91	0.91	0.91	0.91	0.0
D_2	10,000	1,000	1,000	100	0.91	0.91	0.91	0.91	0.0
D_3	100	1,000	1,000	100,000	0.09	0.09	0.09	0.09	0.0
D_4	1,000	1,000	1,000	100,000	0.5	0.5	0.5	0.5	0.0
D_5	1,000	100	10,000	100,000	0.09	0.91	0.5	0.29	0.89
D_6	1,000	10	100,000	100,000	0.01	0.99	0.5	0.10	0.99

Chapter 6: Mining Frequent Patterns, Association and Correlations

- Basic Concepts
- Frequent Itemset Mining Methods
- Pattern Evaluation Methods
- Summary

Summary

- Basic concepts
  - Frequent pattern, association rules, support-confident framework, closed and max-patterns
- Scalable frequent pattern mining methods
  - Apriori
  - FPgrowth
  - Vertical format approach (ECLAT)
- Which patterns are interesting?
  - Pattern evaluation methods

Ref: Basic Concepts of Frequent Pattern Mining

- (Association Rules) R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93.
- (Max-pattern) R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98.
- (Closed-pattern) N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
- (Sequential pattern) R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95

Ref: Apriori and Its Improvements

- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. VLDB'95.
- J. S. Park, M. S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95.
- H. Toivonen. Sampling large databases for association rules. VLDB'96.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97.
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98.

Ref: Depth-First, Projection-Based FP Mining

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. J. Parallel and Distributed Computing:02.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD’ 00.
- J. Liu, Y. Pan, K. Wang, and J. Han. Mining Frequent Item Sets by Opportunistic Projection. KDD'02.
- J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining Top-K Frequent Closed Patterns without Minimum Support. ICDM'02.
- J. Wang, J. Han, and J. Pei. CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. KDD'03.
- G. Liu, H. Lu, W. Lou, J. X. Yu. On Computing, Storing and Querying Frequent Patterns. KDD'03.
- G. Grahne and J. Zhu, Efficiently Using Prefix-Trees in Mining Frequent Itemsets, Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003

Ref: Mining Correlations and Interesting Rules

- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97.
- C. Silverstein, S. Brin, R. Motwani, and J. Ullman. Scalable techniques for mining causal structures. VLDB'98.
- P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. KDD'02.
- E. Omiecinski. Alternative Interest Measures for Mining Associations. TKDE'03.
- T. Wu, Y. Chen and J. Han, "Association Mining in Large Databases: A Re-Examination of Its Measures", PKDD'07

Ref: Freq. Pattern Mining Applications

- Y. Huhtala, J. Kärkkäinen, P. Porkka, H. Toivonen. Efficient Discovery of Functional and Approximate Dependencies Using Partitions. ICDE’98.
- H. V. Jagadish, J. Madar, and R. Ng. Semantic Compression and Pattern Extraction with Fascicles. VLDB'99.
- T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining Database Structure; or How to Build a Data Quality Browser. SIGMOD'02.
- K. Wang, S. Zhou, J. Han. Profit Mining: From Patterns to Actions. EDBT’02.