2019 EE448, Big Data Mining, Lecture 3

Fundamental Data Mining Algorithms

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

http://wnzhang.net/teaching/ee448/index.html

REVIEW

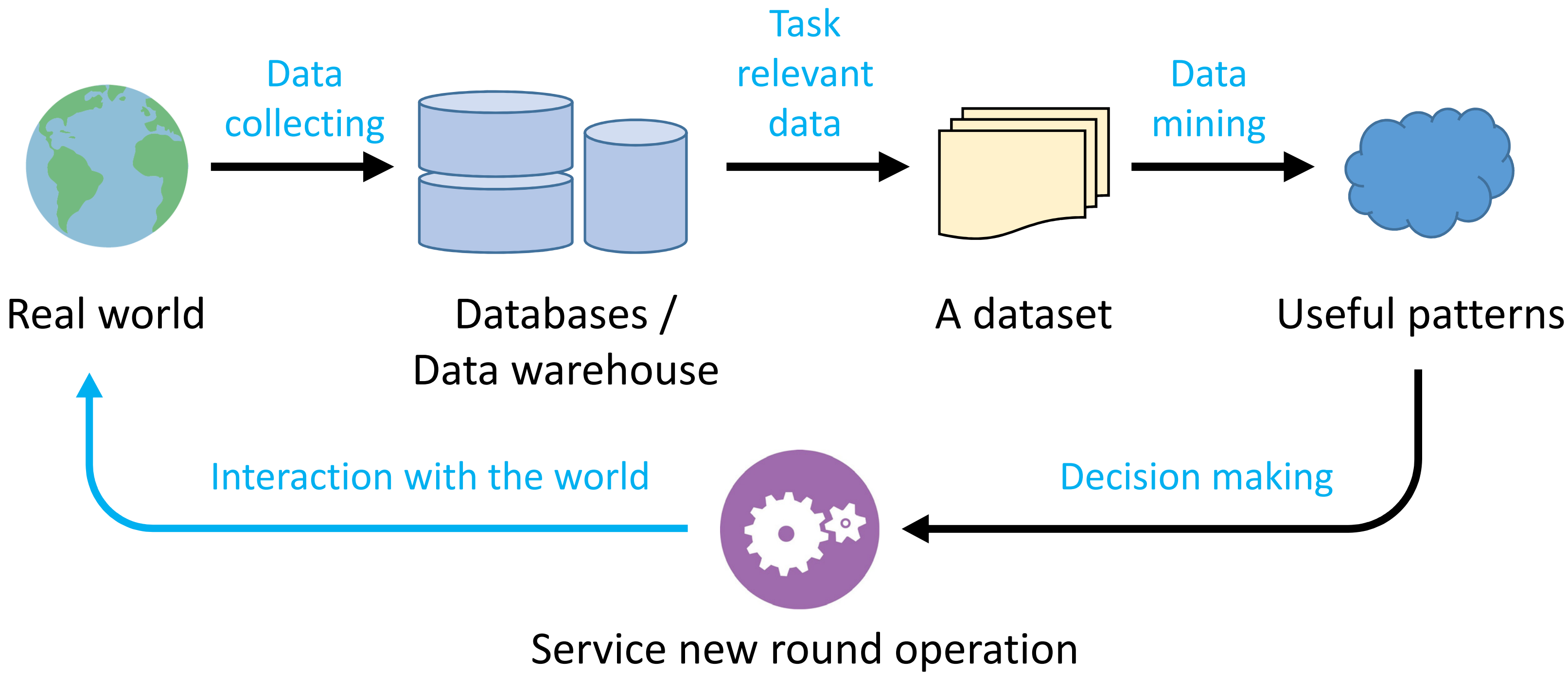
What is Data Mining?

- Data mining is about the extraction of non-trivial, implicit, previously unknown and potentially useful principles, patterns or knowledge from massive amount of data.

- Data Science is the subject concerned with the scientific methodology to properly, effectively and efficiently perform data mining
  - an interdisciplinary field about scientific methods, processes, and systems

REVIEW

A Typical Data Mining Process



- Data mining plays a key role of enabling and improving the various data services in the world
- Note that the (improved) data services would then change the world data, which would in turn change the data to mine

REVIEW

An Example in User Behavior Modeling

Interest	Gender	Age	BBC Sports	PubMed	Bloomberg Business	Spotify
Finance	Male	29	Yes	No	Yes	No
Sports	Male	21	Yes	No	No	Yes
Medicine	Female	32	No	Yes	No	No
Music	Female	25	No	No	No	Yes
Medicine	Male	40	Yes	Yes	Yes	No

Expensive data — Cheap data

- A 7-field record data
  - 3 fields of data that are expensive to obtain
    - Interest, gender, age collected by user registration information or questionnaires
  - 4 fields of data that are easy or cheap to obtain
    - Raw data of whether the user has visited a particular website during the last two weeks, as recorded by the website log

REVIEW

An Example in User Behavior Modeling

Interest	Gender	Age	BBC Sports	PubMed	Bloomberg Business	Spotify
Finance	Male	29	Yes	No	Yes	No
Sports	Male	21	Yes	No	No	Yes
Medicine	Female	32	No	Yes	No	No
Music	Female	25	No	No	No	Yes
Medicine	Male	40	Yes	Yes	Yes	No

Expensive data (Interest, Gender, Age) — Cheap data (BBC Sports, PubMed, Bloomberg Business, Spotify)

- **Deterministic view**: fit a function

  Age = f(Browsing=BBC Sports, Bloomberg Business)

- **Probabilistic view**: fit a joint data distribution

  p(Interest=Finance | Browsing=BBC Sports, Bloomberg Business)

  p(Gender=Male | Browsing=BBC Sports, Bloomberg Business)

Content of This Lecture

Prediction $X \Rightarrow Y$

- Frequent patterns and association rule mining
  - Apriori
  - FP-Growth algorithms

- Neighborhood methods
  - K-nearest neighbors

Frequent Patterns and Association Rule Mining

This part are mostly based on Prof. Jiawei Han's book and lectures

- http://hanj.cs.illinois.edu/bk3/bk3_slidesindex.htm
- https://wiki.cites.illinois.edu/wiki/display/cs512/Lectures

REVIEW

A DM Use Case: Frequent Item Set Mining

WRAPPING PAPER	0.99
INSTANT COFFEE GOLD	1.99
INSTANT COFFEE GOLD	1.99
ORANGE JUICE 1.5L	0.79
ORANGE JUICE 1.5L	0.79
RICE CRACKERS SALT	0.29
RICE CRACKERS SALT	0.29
PLAIN MARGARINE	0.44
GARDEN GLOVES	1.49
FREE RANGE EGGS	1.05
ASSORTED MUESLI	1.49
COOKIES	1.05
MACARONI	0.42
BUTTERMILK DESSERT	0.29
BUTTERMILK DESSERT	0.29
BUTTERMILK DESSERT	0.29
BUTTERMILK DESSERT	0.29
TOTAL	14.23
CASH	20.00
CHANGE	5.77

THANK YOU AND GOODBYE

Some intuitive patterns:

$$\{\text{milk, bread, butter}\}$$
$$\{\text{onion, potatoes, beef}\}$$

Some non-intuitive ones:

$$\{\text{diaper, beer}\}$$

Agrawal, R.; Imieliński, T.; Swami, A. (1993). "Mining association rules between sets of items in large databases". *ACM SIGMOD 1993*

A DM Use Case: Association Rule Mining

WRAPPING PAPER	0.99
INSTANT COFFEE GOLD	1.99
INSTANT COFFEE GOLD	1.99
ORANGE JUICE 1.5L	0.79
ORANGE JUICE 1.5L	0.79
RICE CRACKERS SALT	0.29
RICE CRACKERS SALT	0.29
PLAIN MARGARINE	0.44
GARDEN GLOVES	1.49
FREE RANGE EGGS	1.05
ASSORTED MUESLI	1.49
COOKIES	1.05
MACARONI	0.42
BUTTERMILK DESSERT	0.29
BUTTERMILK DESSERT	0.29
BUTTERMILK DESSERT	0.29
BUTTERMILK DESSERT	0.29
TOTAL	14.23
CASH	20.00
CHANGE	5.77

THANK YOU AND GOODBYE

Some intuitive patterns:

$$\{\text{milk, bread}\} \Rightarrow \{\text{butter}\}$$
$$\{\text{onion, potatoes}\} \Rightarrow \{\text{burger}\}$$

Some non-intuitive ones:

$$\{\text{diaper}\} \Rightarrow \{\text{beer}\}$$

Agrawal, R.; Imieliński, T.; Swami, A. (1993). "Mining association rules between sets of items in large databases". *ACM SIGMOD 1993*

Frequent Pattern and Association Rules

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Association rule:
  - Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of m items
  - Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of transactions that each $t_i \subseteq I$
  - An association rule is a relation as

    $$X \rightarrow Y, \text{ where } X, Y \subset I \text{ and } X \cap Y = \emptyset$$

  - Here X and Y are itemsets, could be regarded as patterns

- First proposed by Agrawal, Imielinski, and Swami in the context of frequent itemsets and association rule mining
  - R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93

Frequent Pattern and Association Rules

- 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?
  - Can we automatically classify web documents?

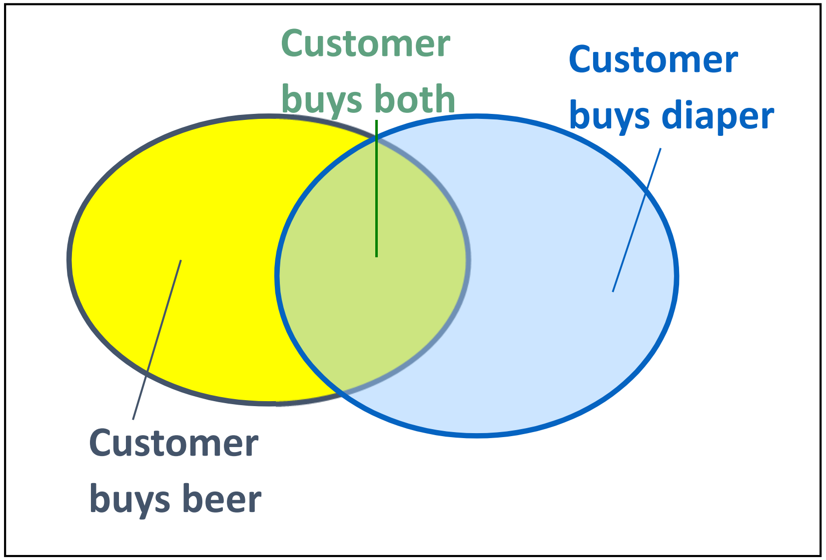
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

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
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

Basic Concepts: Frequent Patterns

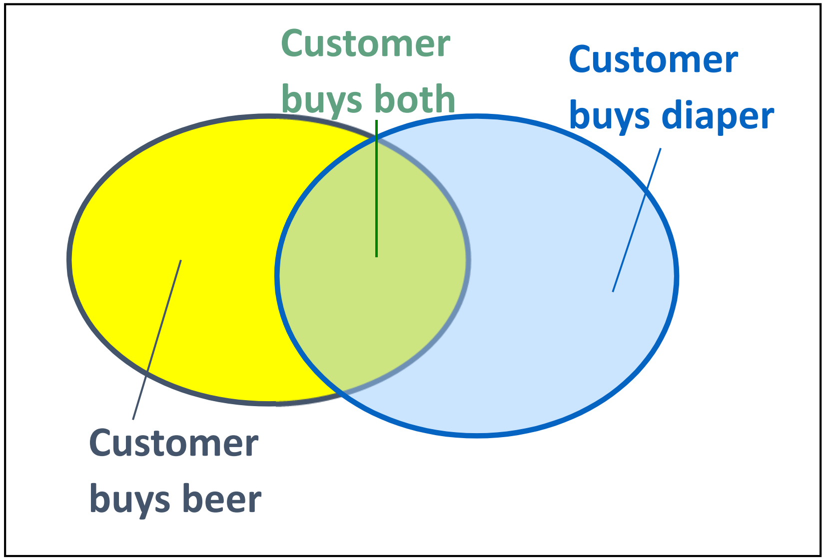
Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk



- 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 contain 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
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk



- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, s, probability that a transaction contains $X \cup Y$

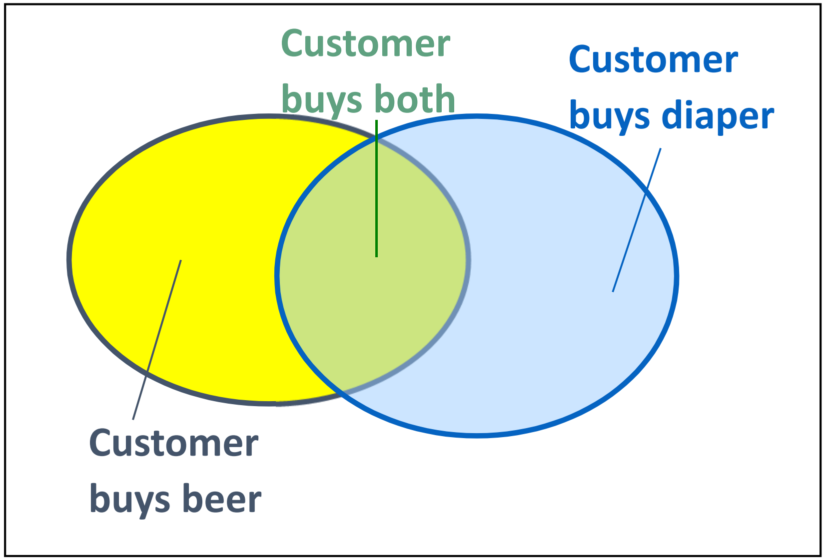
$$s = \frac{\#\{t, (X \cup Y) \subset t\}}{n}$$

  - confidence, c, conditional probability that a transaction having X also contains Y

$$c = \frac{\#\{t, (X \cup Y) \subset t\}}{\#\{t, X \subset t\}}$$

Basic Concepts: Association Rules

Tid	Items bought
1	Beer, Nuts, Diaper
2	Beer, Coffee, Diaper
3	Beer, Diaper, Eggs
4	Nuts, Eggs, Milk
5	Nuts, Coffee, Diaper, Eggs, Milk



- Set the minimum thresholds
  - *minsup* = 50%
  - *minconf* = 50%
- Frequent Patterns:
  - Beer:3, Nuts:3, Diaper:4, Eggs:3
  - {Beer, Diaper}:3
- Association rules: (many more!) *sup* *conf*
  - Beer → Diaper (60%, 100%)
  - Diaper → Beer (60%, 75%)
  - Nuts → Diaper (60%, 100%)
  - Diaper → Nuts (80%, 50%)
  - …

Closed Patterns and Max-Patterns

- A long pattern contains a combinatorial number of sub-patterns, e.g., $\{i_1, \ldots, i_{100}\}$ contains $\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 = 1.27\times10^{30}$ 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 $Y \supset X$, 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 $Y \supset X$
  - 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 = {<i_1, …, i_{100}>, < i_1, …, i_{50}>}
  - min_sup = 1.
- What is the set of closed itemset?
  - <a_1, …, a_{100}>: 1
  - <a_1, …, a_{50}>: 2
- What is the set of max-pattern?
  - <a_1, …, a_{100}>: 1
- What is the set of all patterns?
  - !!

The Downward Closure Property and Scalable Mining Methods

- The downward closure property of frequent patterns
  - Any subset 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
    - R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94
  - Frequent pattern growth (FP-growth)
    - J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00

Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach

  R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94

- FPGrowth: A Frequent Pattern-Growth Approach without candidate generation

  J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!

- Method:
  - Initially, scan data once to get frequent 1-itemset
  - Generate length (k+1)-sized candidate itemsets from frequent k-itemsets
  - Test the candidates against data
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database

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

1st 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}

2nd 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}

3rd 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 = 1; L_k !=Ø; k++) **do**
    C_{k+1} = candidates generated from L_k;
    for each transaction t in database **do**
        increment the count of all candidates in C_{k+1} that are contained in t;
    end
    L_{k+1} = candidates in C_{k+1} with min_support;
end
return $\cup_k L_k$;

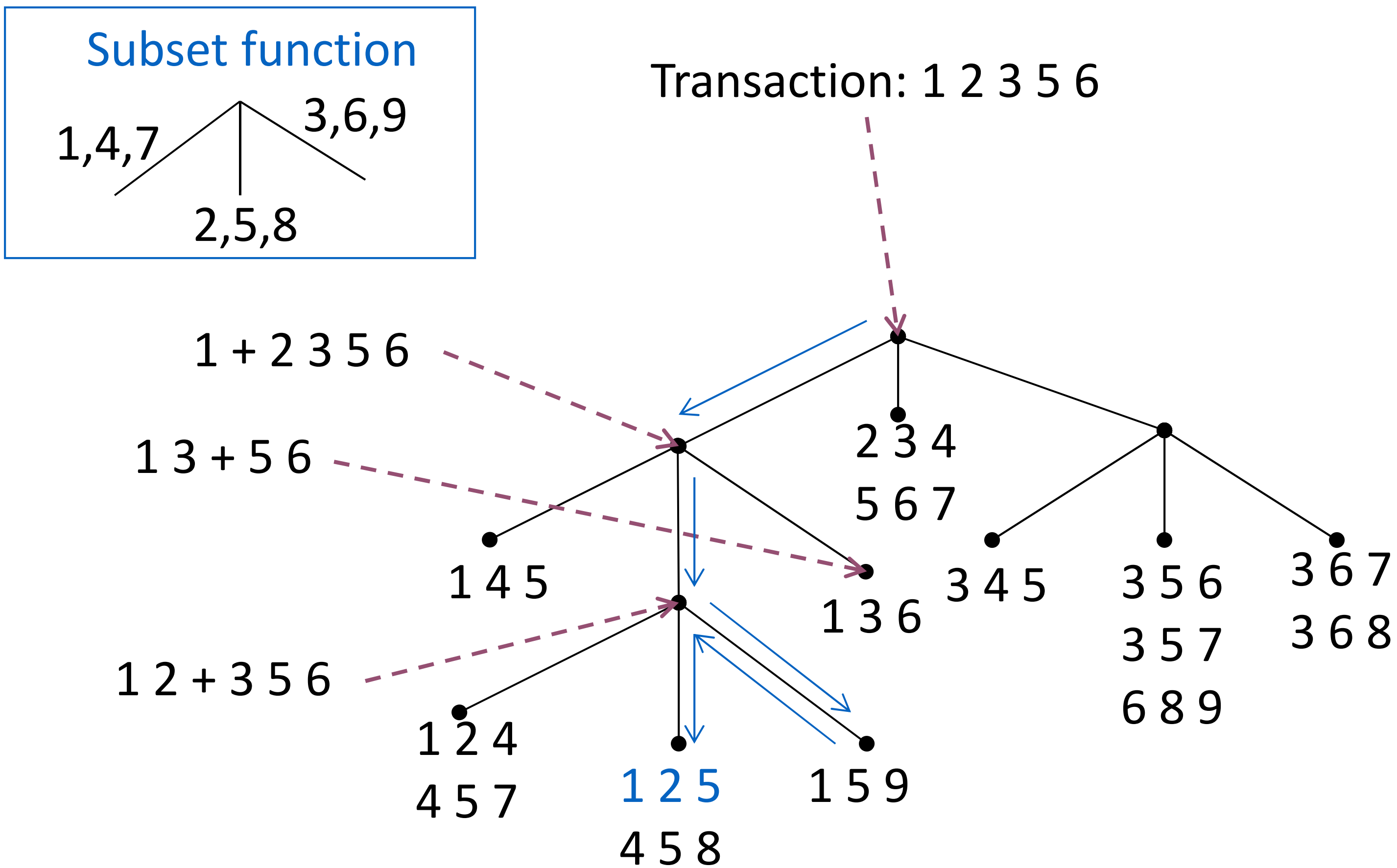
Implementation of Apriori

- How to generate candidates?
  - Step 1: self-joining L_k
  - Step 2: pruning
- Example of candidate generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Self-joining: $L_3 \times L_3$
    - *abcd* from *abc* and *abd*
    - *acde* from *acd* and *ace*
  - Pruning:
    - *acde* is removed because *ade* is not in L_3
  - C_4 = {*abcd*}

How to Count Supports of Candidates?

- Why counting supports of candidates a problem?
  - The total number of candidates can be very huge
  - One transaction may contain many candidates
- Method:
  - Candidate itemsets are stored in a hash-tree
  - Leaf node of hash-tree contains a list of itemsets and counts
  - Interior node contains a hash table
  - Subset function: finds all the candidates contained in a transaction

Counting Supports of Candidates Using Hash Tree



Scalable Frequent Itemset Mining Methods

- Apriori: A Candidate Generation-and-Test Approach

  R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94

- FPGrowth: A Frequent Pattern-Growth Approach without candidate generation

  J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00

Construct FP-tree from a Transaction Database

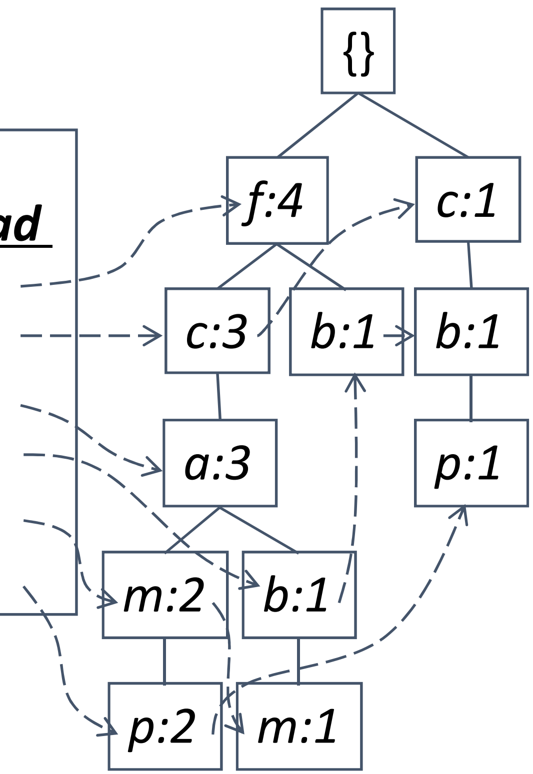
TID	***Items bought***	***(ordered) frequent items***
100	***{f, a, c, d, g, i, m, p}***	***{f, c, a, m, p}***
200	***{a, b, c, f, l, m, o}***	***{f, c, a, b, m}***
300	***{b, f, h, j, o, w}***	***{f, b}***
400	***{b, c, k, s, p}***	***{c, b, p}***
500	***{a, f, c, e, l, p, m, n}***	***{f, c, a, m, p}***

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

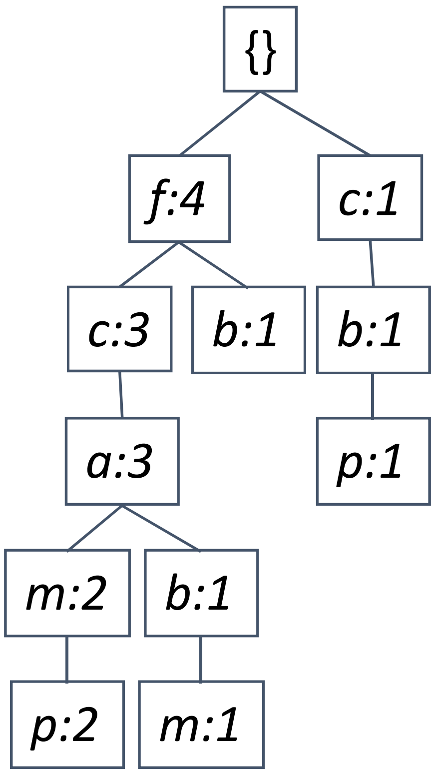
Item	***frequency***	***head***
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



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 *b* but no *m* nor *p*
  - …
  - Patterns having *c* but no *a* nor *b*, *m*, *p*
  - Pattern *f*
- Completeness and non-redundency

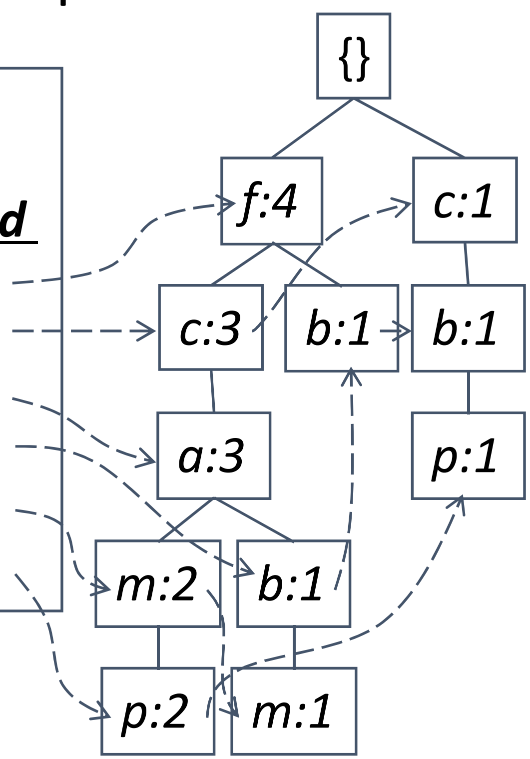


Find Patterns Having P From P-conditional Database

- 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

Header Table

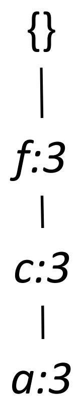
Item	***frequency***	***head***
f	*4*	
c	*4*	
a	*3*	
b	*3*	
m	*3*	
p	*3*	



Conditional **pattern bases**

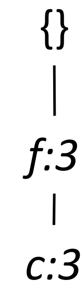
item	***cond. pattern base***
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

Recursion: Mining Each Conditional FP-tree



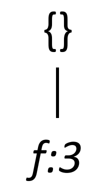
m-conditional **FP-tree**

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



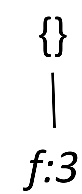
am-conditional **FP-tree**

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



cm-conditional **FP-tree**

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

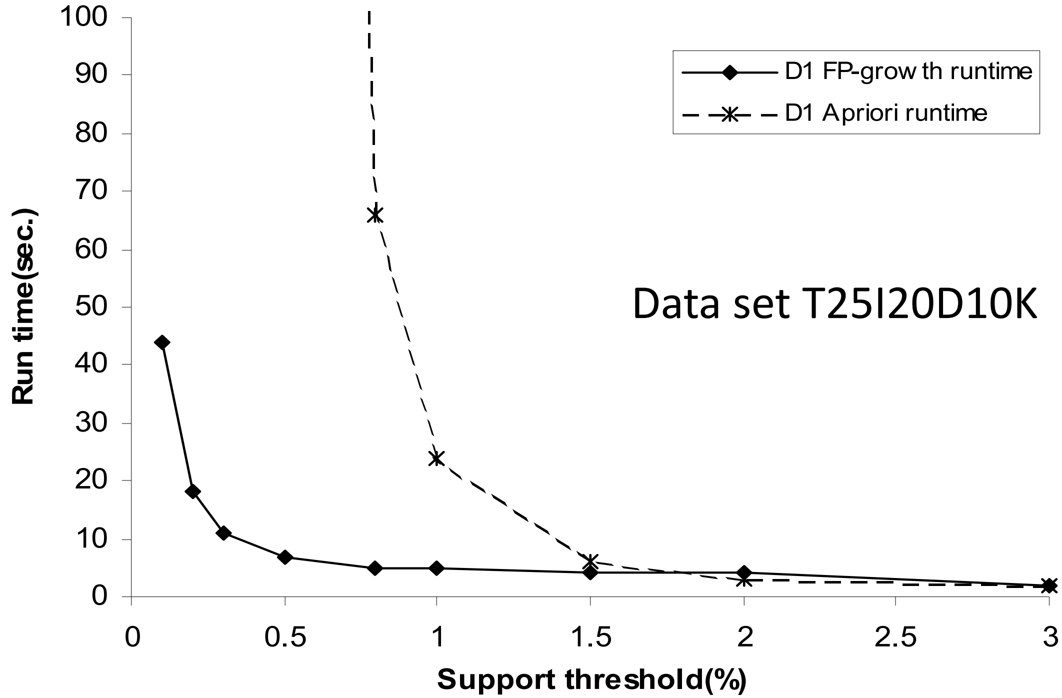


cam-conditional **FP-tree**

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

Performance of FPGrowth in Large Datasets



FP-Growth vs. Apriori

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 operations: counting local frequent items and building sub FP-tree, no pattern search and matching
- A good open-source implementation and refinement of FPGrowth
  - FPGrowth+: B. Goethals and M. Zaki. An introduction to workshop on frequent itemset mining implementations. Proc. ICDM'03 Int. Workshop on Frequent Itemset Mining Implementations (FIMI'03), Melbourne, FL, Nov. 2003

Content of This Lecture

Prediction $X \Rightarrow Y$

- Frequent patterns and association rule mining
  - Apriori
  - FP-Growth algorithms

- Neighborhood methods
  - K-nearest neighbors

K Nearest Neighbor Algorithm (KNN)

- A non-parametric method used for data prediction
  - For each input instance *x*, find *k* closest training instances $N_k(x)$ in the feature space
  - The prediction of *x* is based on the average of labels of the *k* instances

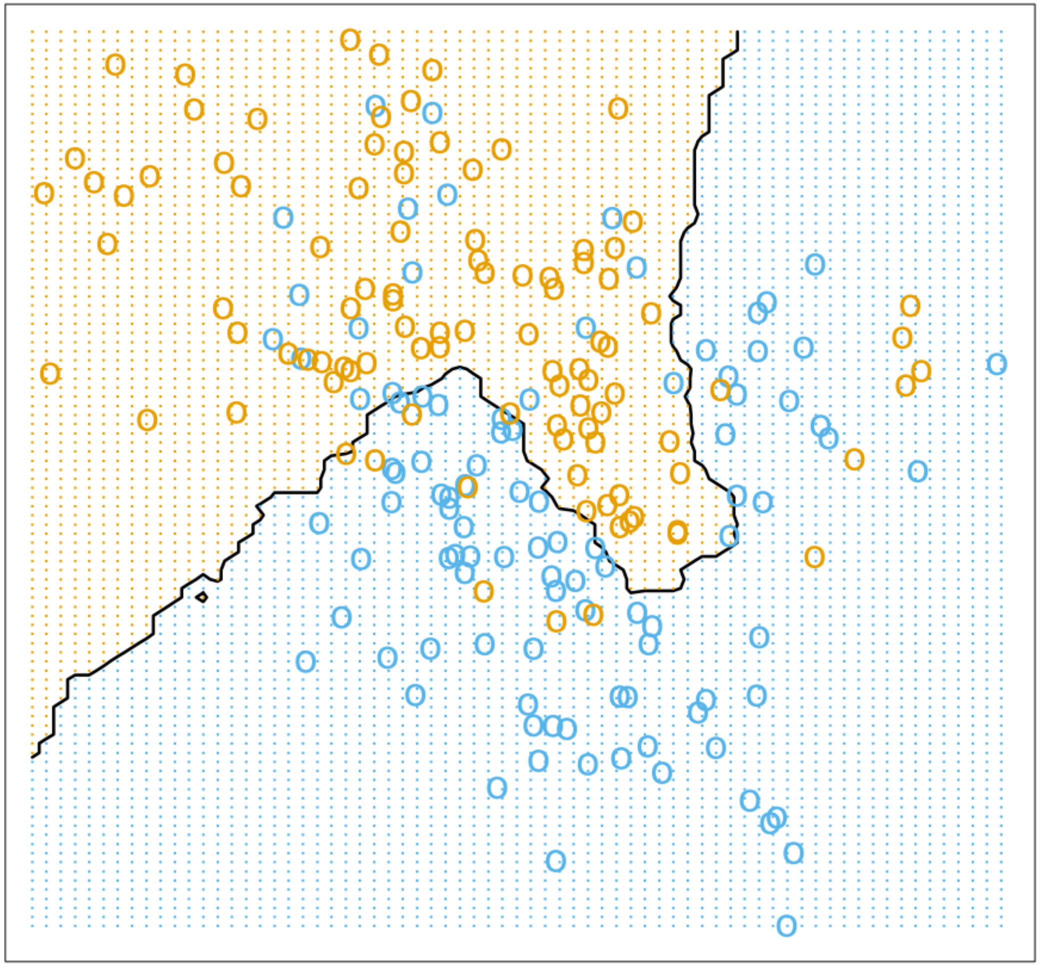
$$\hat{y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  - For classification problem, it is the majority voting among neighbors

$$p(\hat{y}|x) = \frac{1}{k} \sum_{x_i \in N_k(x)} 1(y_i = \hat{y})$$

kNN Example

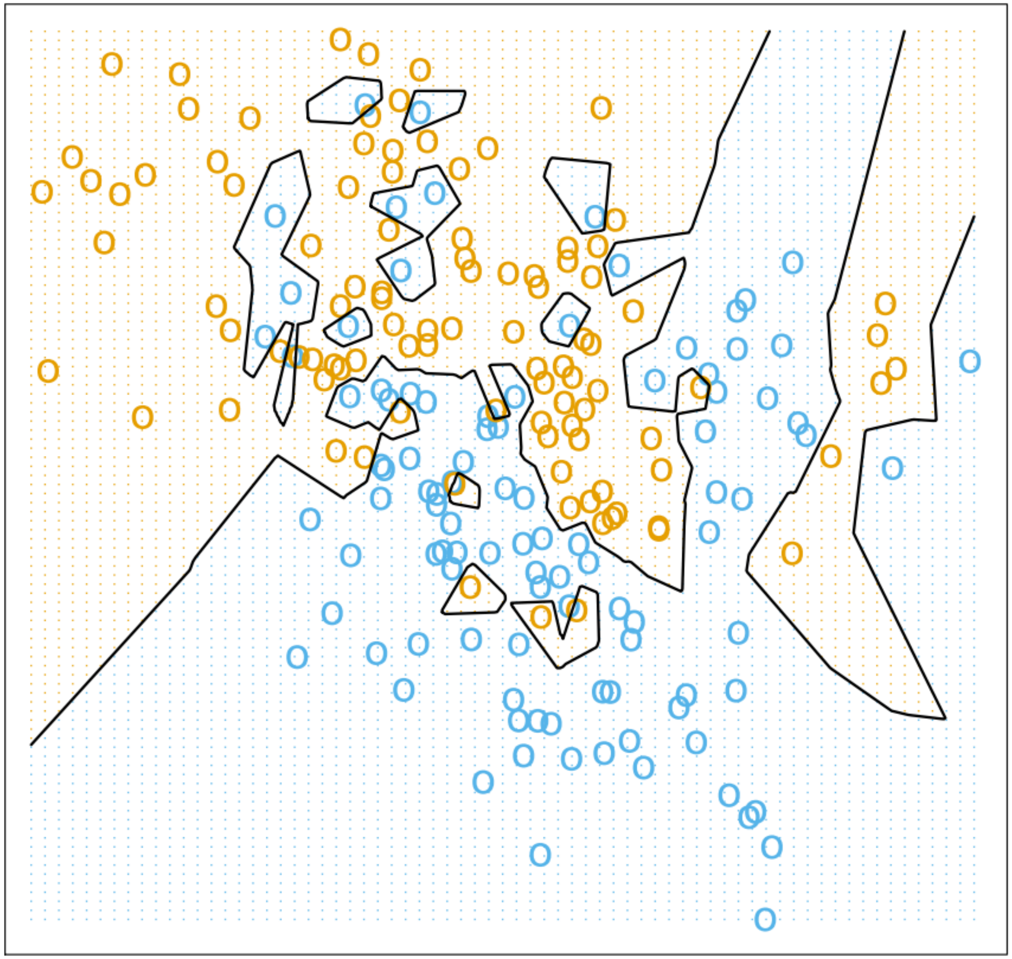
15-nearest neighbor



Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. “The Elements of Statistical Learning”. Springer 2009.

kNN Example

1-nearest neighbor



Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. “The Elements of Statistical Learning”. Springer 2009.

K Nearest Neighbor Algorithm (KNN)

- Generalized version
  - Define similarity function $s(x, x_i)$ between the input instance x and its neighbor x_i
  - Then the prediction is based on the weighted average of the neighbor labels based on the similarities

$$\hat{y}(x) = \frac{\sum_{x_i \in N_k(x)} s(x, x_i) y_i}{\sum_{x_i \in N_k(x)} s(x, x_i)}$$

Non-Parametric kNN

- No parameter to learn
  - In fact, there are *N* parameters: each instance is a parameter
  - There are *N*/*k* effective parameters
    - Intuition: if the neighborhoods are non-overlapping, there would be *N*/*k* neighborhoods, each of which fits one parameter
- Hyperparameter *k*
  - We cannot use sum-of-squared error on the training set as a criterion for picking *k*, since *k*=1 is always the best
  - Tune *k* on validation set

Efficiency Concerns

- It is often time consuming to find the *k* nearest neighbors
  - A native solution needs to go through all data instances for each prediction
- Some practical solutions
  - Build inverse index (from feature to instance). We shall get back to this later in Search Engine lecture
  - Parallelized computing (e.g., with GPU parallelization)
  - Pre-calculation with some candidate instances
    - With triangle inequality
  - Learning hashing code
  - Approximation methods

Further Reading

- Xindong Wu et al. Top 10 algorithms in data mining. 2008.

  http://www.cs.uvm.edu/~icdm/algorithms/10Algorithms-08.pdf

- C4.5, k-Means, SVM, Apriori, EM, PageRank, AdaBoost, kNN, Naïve Bayes, CART

More Thinkings

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
- The worst case complexity vs. the expected probability
  - Ex: Suppose Walmart has 10^4 kinds of products
    - The chance to pick up one product 10^{-4}
    - The chance to pick up a particular set of 10 products: $\sim 10^{-40}$
    - What is the chance this particular set of 10 products to be frequent 10^3 times in 10^9 transactions?