



COMP3420: Advanced Databases and Data Mining

Introduction to association
mining



Lecture outline

- Overview of the data mining module
- Review: data mining
- The data mining process
- What is association mining?
- Market basket analysis and association rules examples
- Basic concepts and formalism
- Basic rule measurements
- The Apriori algorithm
- Performance bottlenecks in Apriori
- Improve Apriori's efficiency

Overview of the data mining module

- Twelve lectures over five weeks
 - Association mining
 - Cluster analysis
 - Classification and prediction
 - Mining data streams and time series
 - Web data mining
 - Text data mining
 - Privacy-preserving data mining and data sharing (Peter)
- Three tutorials (practical Rattle labs)
 - Tutorial 4 in week 9: 5 May
 - Tutorial 5 in week 11: 19 May
 - Tutorial 6 in week 13: 2 June

Overview of the data mining module

- One assignment (some theoretical data mining exercise as well as practical data mining with Rattle)
- Lecturer:
 - Peter Christen (peter.christen@anu.edu.au - N330)
 - Denny (denny.denny@anu.edu.au)
- How to contact us?
 - COMP3420 Forum
 - Email comp3420@cs.anu.edu.au
 - See Peter Christen

Review: Data Mining

- Data mining (knowledge discovery from data)
 - **Automated** or convenient extraction (semi-automated) of interesting (**non-trivial, implicit, previously unknown** and **potentially useful/meaningful**) patterns or knowledge from huge amount of data
 - *“The key to success in business is to know something that nobody else knows” - Aristotle Onassis*
- Model / Pattern / Concept
 - What is model?
 - Why do we use model / pattern?
 - How do we get the model?



Data, Information, Knowledge

Product Details

Hardcover: 800 pages

Publisher: Morgan Kaufmann; 2 edition (November 3, 2005)

Language: English

ISBN-10: 1558609016

ISBN-13: 978-1558609013

Product Dimensions: 9.3 x 7.8 x 1.6 inches

Shipping Weight: 4.6 pounds ([View shipping rates and policies](#))

Average Customer Review: ★★★★★ ([29 customer reviews](#))

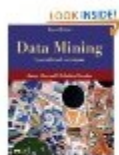
Amazon.com Sales Rank: #55,670 in Books (See [Bestsellers in Books](#))

Popular in these categories: ([What's this?](#))

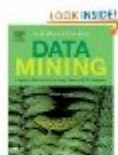
#38 in [Books](#) > [Computers & Internet](#) > [Databases](#) > [Data Mining](#)

#44 in [Books](#) > [Computers & Internet](#) > [Computer Science](#) > [Artificial Intelligence](#)

Frequently Bought Together



+



+

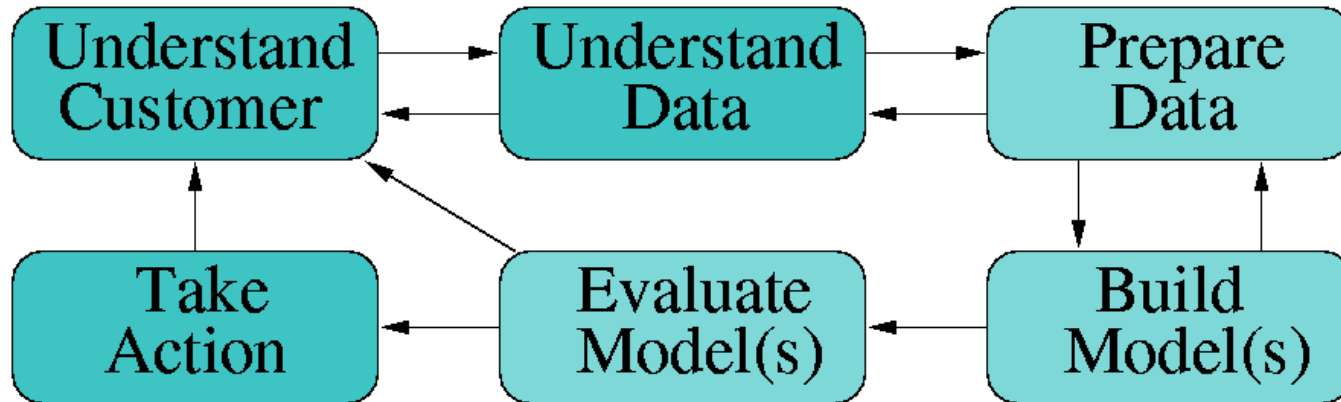


Price For All Three: \$171.11

Add all three to Cart

- This item:** [Data Mining: Concepts and Techniques, Second Edition \(The Morgan Kaufmann Series in Data Management Systems\)](#) by Micheline Kamber Jiawei Han
- [Data Mining: Practical Machine Learning Tools and Techniques, Second Edition \(Morgan Kaufmann Series in Data Management Systems\)](#) by Ian H. Witten
- [Introduction to Data Mining](#) by Pang-Ning Tan

The data mining / KDD process



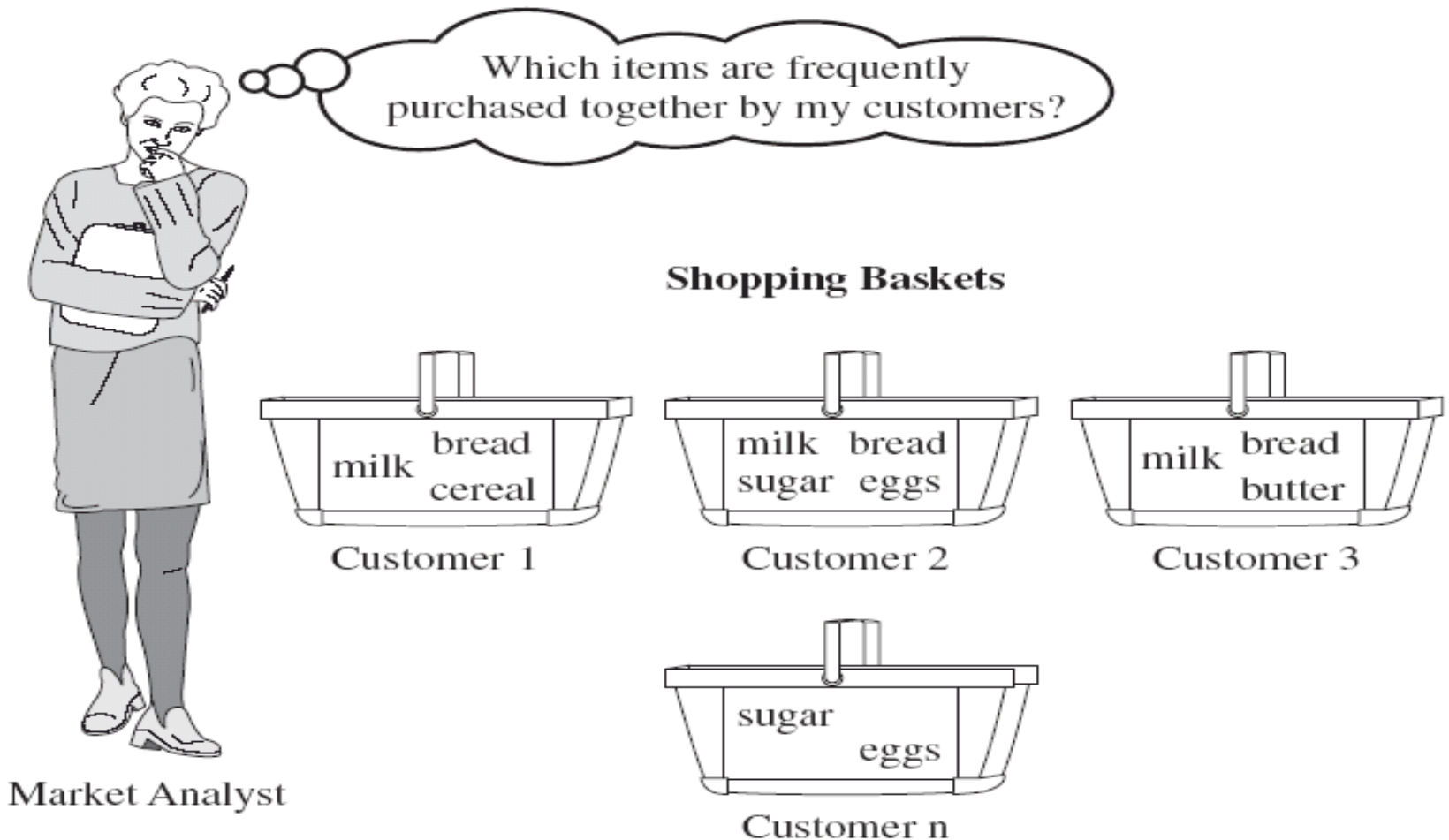
- Data mining is an interactive process
- Data mining = *Build Model(s)*
- Typically up to 90% of time and effort are spent in the first three steps!

(Follows: *CRoss Industry Standard Process for Data Mining*, <http://www.crisp-dm.org/>)

What is association mining?

- Association mining is the task of finding frequent rules / associations / patterns / correlations / causal structures within (large) sets of items in transactional (relational) databases
- *Unsupervised* learning techniques (*descriptive* data mining, not *predictive* data mining)
- A common application is *market basket analysis* which items are frequently sold together at a supermarket
 - arranging items on shelves
 - which items should be promoted together
- Other applications are
 - Web log analysis (click-stream)
 - Cross-marketing
 - Sale campaign analysis
 - DNA sequence analysis

Market basket analysis



Association rules examples

- Rules form: $body \Rightarrow head [support, confidence]$

- Market basket:

$buys(X, 'beer') \Rightarrow buys(X, 'snacks') [1\%, 60\%]$

- If a customer X purchased 'beer', in 60% she or he also purchased 'snacks'
- 1% of all transactions contain the items 'beer' and 'snacks'

- Student grades:

$major(X, 'InfTech') \text{ and } takes(X, 'COMP3420') \Rightarrow$
 $grade(X, 'D') [3\%, 75\%]$

- If a student X, who's major is 'InfTech', took the course 'COMP3420' she or he in 75% achieved a grade 'D'
- The combination 'InfTech', 'COMP3420' and 'D' appears in 3% of all transactions (records) in the database

Basic concepts

- Given:
 - A (large) database of transactions
 - Each transaction contains a list of one or more items (e.g. purchased by a customer in a visit)
- Find the rules that correlate the presence of one set of items with that of another set of items
- Normally one is only interested in rules that are frequent
 - For example, 70% of customers who buy tires and car accessories also get their car service done
Question: How can this be improved to 80%? Possibly offer special deals like a 20% reduction of tire costs when the service is done

Formalism

- Set of items $X = \{x_1, x_2, \dots, x_k\}$
- Database D containing transactions
- Each transaction T is a set of items, such that T is a subset of X
- Each transaction is associated with a unique identifier, called TID (for example, a unique number)
- Let A be a set of items (a subset of X)
- An association rule is an implication of the form $A \Rightarrow B$, where A is a subset of X and B is a subset of X , and the intersection of A and B is empty
 - No item in A can be in B , and vice versa
 - No rule of the form: $\{\text{'beer'}, \text{'chips'}\} \Rightarrow \{\text{'chips'}, \text{'peanuts'}\}$

Basic rule measurements

- A rule $A \Rightarrow B$ holds in a database D with *support* s , with s being the percentage of transactions in D that contain A and B

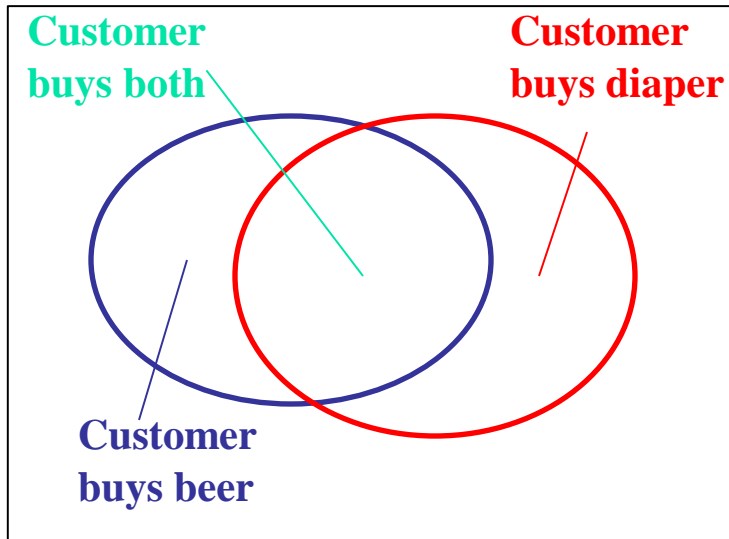
$$\text{support}(A \Rightarrow B) = P(A \cup B)$$

- The rule $A \Rightarrow B$ has a *confidence* c in a database D if c is the percentage of transactions in D containing A that also contain B

$$\text{confidence}(A \Rightarrow B) = P(B|A) = P(A \cup B) / P(A)$$

$$\text{confidence}(A \Rightarrow B) = \text{support}(A \cup B) / \text{support}(A)$$

Rule measurements example



- Find all the rules $\{X, Y\} \Rightarrow Z$ with minimum confidence and support
- Support, s , is the probability that a transaction contains $\{X, Y, Z\}$
- Confidence, c , is the conditional probability that a transaction having $\{X, Y\}$ also contains Z

Transaction ID	Items Bought
2000	a, b, c
1000	a, c
4000	a, d
5000	b, e, f

Let minimum support = 50%, and minimum confidence = 50%, so we have $([a, c])$:

- $a \Rightarrow c$ [50%, 66.6%]
- $c \Rightarrow a$ [50%, 100%]

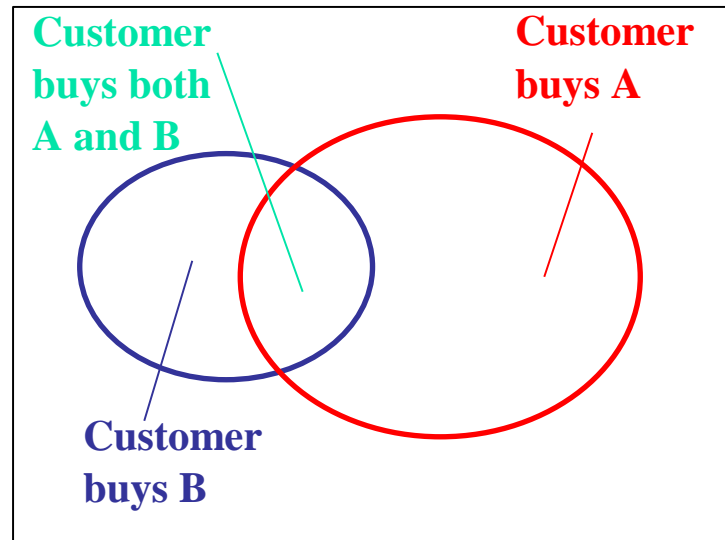
Rule measurements example (2)

Transaction ID	Items Bought
2000	a, b, c
1000	a, c
4000	a, d
5000	b, e, f

Itemset	Support
a	75.00%
b	50.00%
c	50.00%
a, c	50.00%

- Minimum support = 50% and confidence = 50%
- Rule $a \Rightarrow c$
 - support ($a \Rightarrow c$): 50%
 - confidence ($a \Rightarrow c$) = $\text{support}(a \Rightarrow c) / \text{support}(a) = 50\% / 75\% = 66.66\%$

Basic rule measurements



- Is $support(A \Rightarrow B) > support(B \Rightarrow A)$?
- Which rule has a higher confidence?
 - $confidence(A \Rightarrow B)$
 - $confidence(B \Rightarrow A)$

Mining frequent item sets

- Key step: Find the *frequent sets of items* that have *minimum support* (appear in at least xx% of all transactions in a database)
- Basic principle (*Apriori* principle): A sub-set of a frequent item set must also be a frequent item set
 - For example, if {a,b} is frequent, both {a} and {b} have to be frequent (if `beer' and 'chips' are purchased frequently together, then `beer' is purchased frequently and `chips' are also purchased frequently)
- Basic approach: Iteratively find frequent item sets with cardinality from 1 to k (k -item sets), $k > 1$
- Use the frequent item sets to generate association rules
 - For example, frequent 3-item set {a,b,c} contains rules:
 $a \Rightarrow c$, $b \Rightarrow c$, $a \Rightarrow b$, $\{a,b\} \Rightarrow c$, $\{a,c\} \Rightarrow b$, $\{b,c\} \Rightarrow a$, etc.
- We are normally only interested in longer rules

The *Apriori* algorithm (Agrawal & Srikant, VLDB'94)

- C_k : Candidate item set of size k
 L_k : Frequent item set of size k
- Pseudo-code:

$L_1 = \{\text{frequent items}\};$

for ($k = 1; L_k \neq \phi; k++$) **do begin**

$C_{k+1} =$ 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 do

$L_{k+1} =$ candidates in C_{k+1} with min_support

return $\cup_k L_k$;

The *Apriori* algorithm – An example (sup=50%)

Database D

TID	Items
100	a,c,d
200	b,c,e
300	a,b,c,e
400	b,e

Scan D

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	sup
{a, b}	1
{a, c}	2
{a, e}	1
{b, c}	2
{b, e}	3
{c, e}	2

Scan D

C_2

itemset
{a, b}
{a, c}
{a, e}
{b, c}
{b, e}
{c, e}

L_2

itemset	sup
{a, c}	2
{b, c}	2
{b, e}	3
{c, e}	2

C_3

Scan D

itemset	sup
{b, c, e}	2

L_3

itemset	sup
{b, c, e}	2

C_3

itemset
{b, c, e}

The *Apriori* algorithm – An example (2)

Database D

TID	Items
100	a,c,d
200	b,c,e
300	a,b,c,e
400	b,e

L_3

itemset	sup
{b, c, e}	2

- Minimum support = 50% and minimum confidence = 50%
- Rules:
 - $b \Rightarrow c$ [50%, 66.66%]
 - $b \Rightarrow e$ [75%, 100%]
 - $c \Rightarrow e$ [50%, 66.66%]
 - $\{b, c\} \Rightarrow e$ [50%, 100%]
 - $\{b, e\} \Rightarrow c$ [50%, 66.66%]
 - $\{c, e\} \Rightarrow b$ [50%, 100%]

Important details of the *Apriori* algorithm

- How to generate candidate sets?
 - Step 1: Self-joining L_k (C_k is generated by joining L_{k-1} with itself)
 - Step 2: Pruning (any $(k-1)$ -item set that is not frequent cannot be a subset of a frequent k -item set)
- Example of candidate generation:
 - $L_3 = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{a,c,e\}, \{b,c,d\}\}$
 - Self-joining: $L_3 * L_3$ ($\{a,b,c,d\}$ from $\{a,b,c\}$ and $\{a,b,d\}$, and $\{a,c,d,e\}$ from $\{a,c,d\}$ and $\{a,c,e\}$)
 - Pruning: $\{a,c,d,e\}$ is removed because $\{a,d,e\}$ is not in L_3
 - $C_4 = \{\{a,b,c,d\}\}$
- How to count supports for candidates?

How to generate candidate item-sets?

- Suppose the items in L_{k-1} are listed in an order (e.g. $a < b$)
- Step 1: Self-joining L_{k-1}

insert into C_k

select p.item₁, p.item₂, ..., p.item_{k-1}, q.item_{k-1}

from L_{k-1} p, L_{k-1} q

where p.item₁= q.item₁, ..., p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}

- Step 2: Pruning

forall item sets c **in** C_k **do**

forall (k-1)-sub-sets s of c **do**

if (s is not in L_{k-1}) **then** delete c from C_k

Apriori performance bottlenecks

- The core of the *Apriori* algorithm is to
 - Use frequent $(k-1)$ item sets to generate candidate frequent k item sets
 - Use database scan and pattern matching to collect counts for candidate item sets
- Candidate generation is the main bottleneck
 - 10^4 frequent 1-item sets (sets of length 1) will generate 10^7 candidate 2-item sets!
 - To discover a frequent pattern of size 100 (for example $\{a_1, a_2, \dots, a_{100}\}$) one needs to generate $2^{100} = 10^{30}$ candidates
 - Multiple scans of the database are needed ($n+1$ scans if the longest pattern is n items long)

Methods to improve *Apriori's* efficiency

- Reduce the number of scans of the database
 - Any item set that is potentially frequent in the database must be frequent in at least one of the partitions of the database
 - Scan 1: Partition database and find local frequent patterns
 - Scan 2: Consolidate global frequent patterns
- Shrink number of candidates
 - Select a sample of the database, mine frequent patterns within sample using *Apriori*
 - Scan database once to verify frequent item sets found in sample
 - Scan database again to find missed frequent patterns
- Facilitate support of counting candidates
 - For example, use special data structures like Frequent-Pattern tree (FP-tree)