COS10022 – DATA SCIENCE PRINCIPLES

Dr Pei-Wei Tsai (Lecturer, Unit Convenor)
ptsai@swin.edu.au, EN508d

SWINBURNE UNIVERSITY OF TECHNOLOGY

Week 06
Association Rules

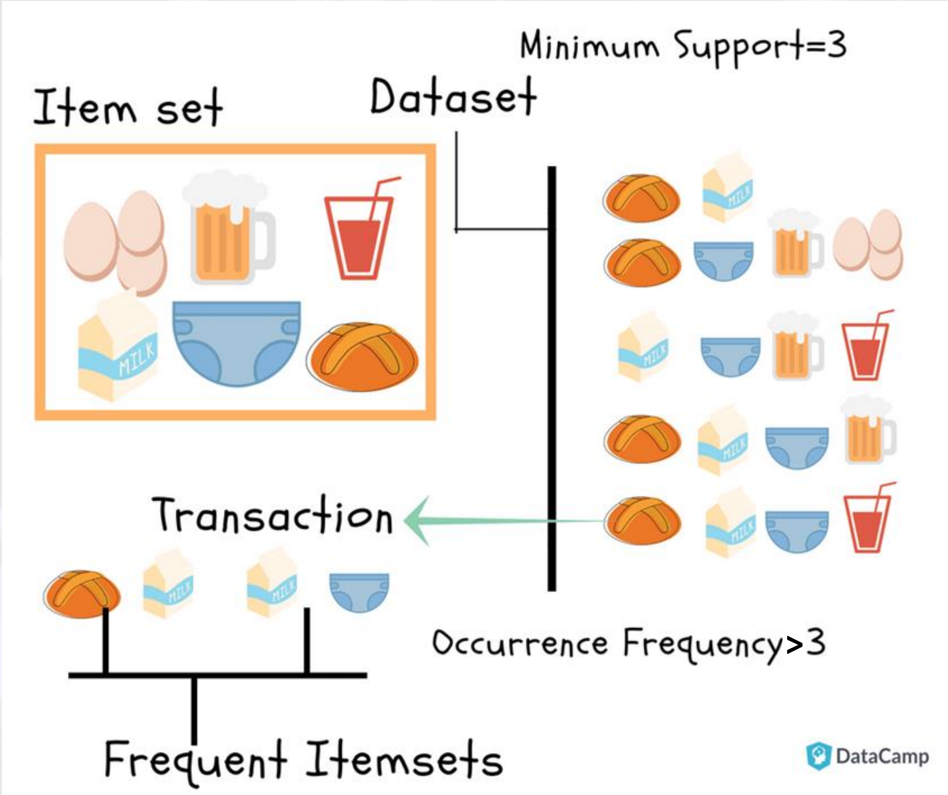
COS10022 - Data science Principles

Outline

Overview

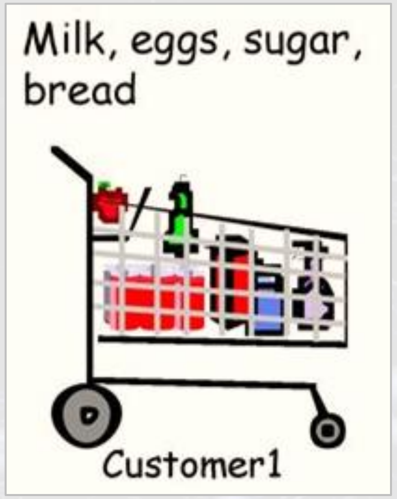
- Association rules
  - An **unsupervised** learning method.
  - A **descriptive**, **not predictive**, method.
  - Used to discover **interesting hidden relationships** in a large dataset.
  - The disclosed relationships are represented as **rules** or **frequent itemsets**.
  - Commonly used for mining transactions in database.

Market Basket Analysis



Overview

- Some questions that association rules can answer:
  - Which products tend to be purchased together?
  - Of those customers who are similar to this person, what products do they tend to buy?
  - Of those customers who have purchased this product, what other similar products do they tend to view or purchase?



To analyze customer buying habits by finding associations and correlations between the different items that customers place in their shopping baskets.

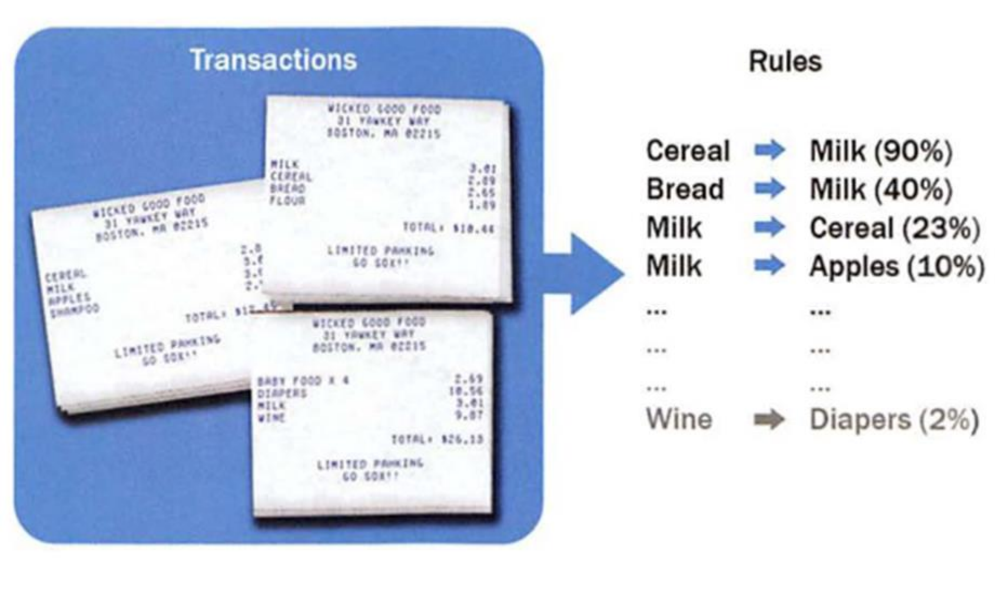
Overview

The **general logic** behind association rules:

1. A large collection of transactions (depicted as three stacks of receipts, in which each transaction consists of one or more items).

2. Association rules go through the **items** being purchased to see what items are **frequently** bought together and to discover a list of **rules** that describe the purchasing behavior.

3. The **rules** suggest that when *cereal* is purchased, 90% of the time *milk* is purchased; when *bread* is purchased, 40% of the time *milk* is purchased also; when *milk* is purchased, 23% of the time *cereal* is also purchased.



Overview

Rules

- Each rule is in the form *X* → *Y*
  - Means when Item *X* is observed, Item *Y* is also observed.

Itemset

- A collection of items or individual entities that contain some kind of relationship.
- An itemset containing k items is called a ***k*-itemset**.
  - *k*-itemset = {item 1, item 2,…,item *k*}
- Examples:
  - A set of retail items purchased together in one transaction.
  - A set of hyperlinks clicked on by one user in a single session.

ID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}
…	…

market basket transactions

{Diapers, Beer} Example of a frequent itemset

{Diapers} → {Beer} Example of an association rule

Overview

1-itemsets

2-itemsets

Apriori Algorithm

- The most fundamental algorithms for generating association rules.
- One major component of Apriori is **support**.
  - Given an Itemset *L*, the support of *L* is the percentage of transactions that contain *L*.
  - If 80% of all transactions contain itemset {bread}, then the support of {bread} is 0.8.
  - If 60% of all transactions contain itemset {bread, butter}, then the support of {bread, butter} is 0.6.

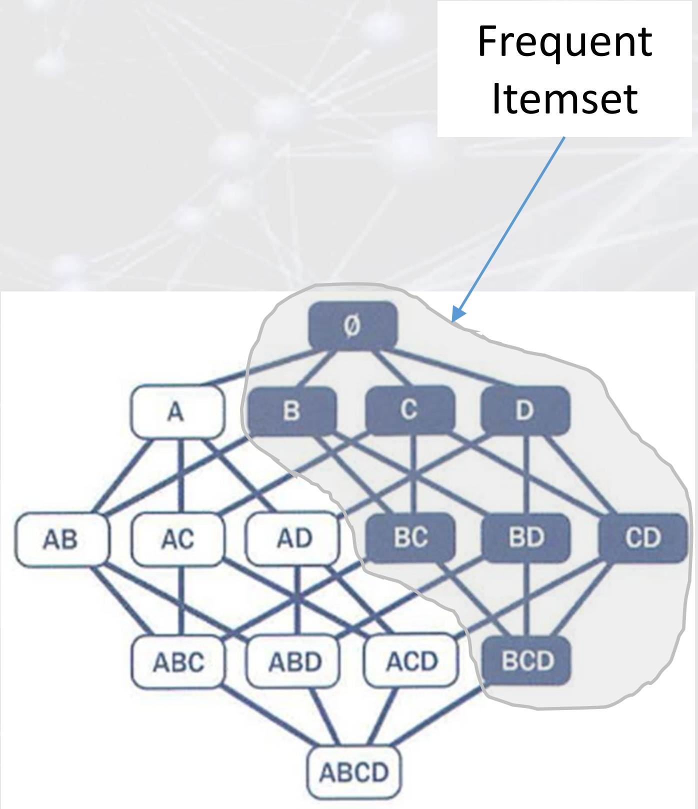
Frequent Itemset

- Items that appear together "often enough" (i.e. meets the **minimum support** criterion).
  - If the minimum support is set at 0.7, {bread} is considered a frequent itemset; whereas {bread, butter} is not considered as a frequent itemset.

Overview

Apriori Property

- Also called **downward closure property**.
- If an item is considered frequent, then any subset of the frequent itemset **must** also be frequent.
  - If 60% of the transactions contain {bread, jam}, then **at least** 60% of all the transactions will contain {bread} or {jam}.
  - If the support of {bread, jam} is 0.6, the support of {bread} or {jam} is **at least** 0.6.
- If itemset {B, C, D} is frequent, then all the subset of this itemset, **shaded in the figure**, must also be frequent itemsets.



Overview

Association Rules

- An implication expression of the form *X* → *Y*, where *X* and *Y* are non-overlapping itemsets.
  - E.g. {Milk, Diaper} → {Beer}
- Generating association rules:
  - Step 1: Find **frequent itemsets** whose occurrences exceed a predefined minimum **support** threshold.
  - Step 2: Derive **association rules** from those frequent itemsets (with the constrains of minimum **confidence** threshold).

Rule evaluation Metrics

- **Support** (***s***): No. of transactions that contain both *X* and *Y* out of total no. of transactions.
  - E.g. A support of 2% means that 2% of all the transactions under analysis show that {Milk, Diaper} and {Beer} are purchased together.
- **Confidence** (***c***): No. of transactions that contain both *X* and *Y* out of total no. of transactions **that contains *X***.
  - E.g. A confidence of 60% means that 60% of customers who purchased {Milk, Diaper} also bought {Beer}

Outline

- Overview
- Apriori Algorithm
- Evaluation of Candidate Rules
- Examples
- Application of Association Rules
- Validation and Testing
- Diagnostic

Apriori Algorithm

Creating Frequent Sets

- Apriori employs an **iterative** approach known as **a level-wise search**, where *k-itemsets* are used to explore (*k*+1)-itemsets.

- First, the set of **frequent 1-itemsets** is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted L_1.

- Next, L_1 is used to find L_2, the set of **frequent 2-itemsets**, which is used to find L_3, and so on, until no more frequent k-itemsets can be found.

- The finding of each L_k requires one full scan of the database.

Example 1

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

(5 transactions ; 6 types of items)

L_1

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

L_2

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

L_3

Itemset	Count
{Bread,Milk,Diaper}	**2**
{Bread,Milk,Beer}	**1**
{Bread,Diaper,Beer}	**2**

Triplets (3-itemsets)

MinSupp = 3/5=0.6

Apriori Algorithm

Creating Frequent Sets

- Let's define:

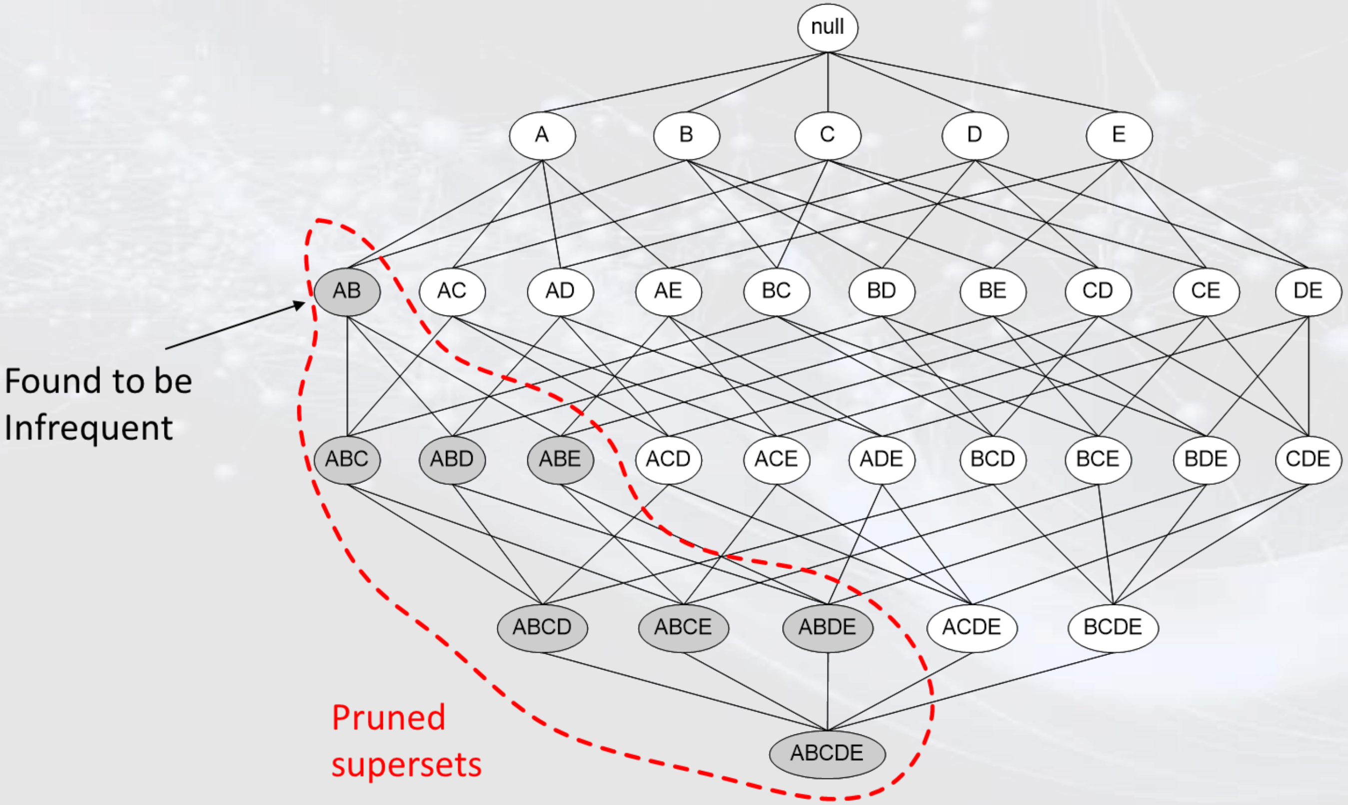
  C_k as a **candidate** itemset of size k

  L_k as a **frequent** itemset of size k

- Main steps of iteration are:
  1. Find frequent itemset L_{k-1} (starting from L_1)
  2. **Join step**: C_k is generated by joining L_{k-1} with itself (cartesian product L_{k-1} x L_{k-1})
  3. **Prune step** (Apriori Property): Any (k−1) size itemset that is not frequent cannot be a subset of a frequent k size itemset, hence should be removed from C_k
  4. Frequent set L_k has been achieved

Apriori Algorithm

Illustrating Apriori Principle



- Any subset of a frequent itemset must also be frequent.
- Itemsets that do not meet the minimum support threshold are **pruned** away.

Apriori Algorithm

Pseudo Code

L_1 = {frequent items};
for (k = 1; L_k !=$\emptyset$; 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
  L_{k+1} = candidates in C_{k+1} with min_support
  end
return $\cup_k L_k$;

Example 2

$Sup_{min} = 2$

Database TDB

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

Outline

Evaluation of Candidate Rules

The process of creating association rules is two-staged.

- First, a set of candidate rule based on frequent itemsets is generated.
  - If {Bread, Egg, Milk, Butter} is the frequent itemset, candidate rules will look like:
    - **{Egg, Milk, Butter} → {Bread}**
    - **{Bread, Milk, Butter} → {Egg}**
    - **{Bread, Egg} → {Milk, Butter}**
    - **Etc.**

- Second, the appropriateness of these candidate rules are evaluated using:
  - **Confidence**
  - **Lift**
  - **Leverage**

Evaluation of Candidate Rules

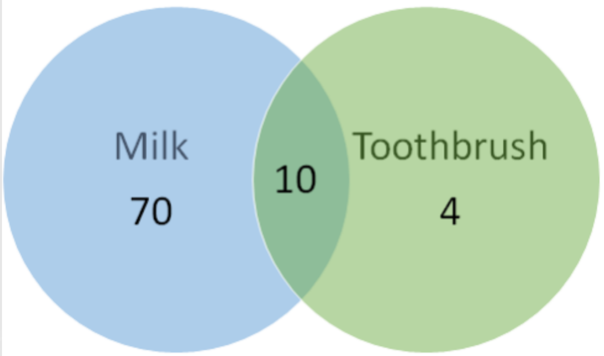
Confidence

- The measures of **certainty** or **trustworthiness** associated with each discovered rule.
- Mathematically, the percentage of transactions that contain both *X* and *Y* out of all transactions that contain *X*.

$$\text{Confidence}(\ X \rightarrow Y\ ) = \frac{\text{Support}(\ X + Y\ )}{\text{support}(\ X\ )}$$

- E.g. if {bread, eggs, milk} has support of **0.15** and {bread, eggs} also has a support of **0.15**, the **confidence of rule {bread, eggs} → {milk}** is **1**.
  - This means 100% of the time a customer buys bread and eggs, milk is brought as well. The rule is therefore correct for 100% of the transactions containing bread and eggs.

Evaluation of Candidate Rules



{Toothbrush}→{Milk}:
Confidence = 10/(10+4) = 0.7

Confidence

- A relationship may be thought of as **interesting** when the algorithm identifies the relationship with a measure of confidence greater than or equal to the **predefined threshold** (i.e. the **minimum confidence**).

- Problem with *Confidence:*
  - Given a rule *X* → *Y*, confidence considers only the antecedent (*X*) and the co-occurrence of *X* and *Y.*
  - Cannot tell if a rule contains true implication of the relationship or if the rule is purely coincidental.

https://towardsdatascience.com/association-rules-2-aa9a77241654

Evaluation of Candidate Rules

Lift

- Measures **how many times more often** *X* and *Y* occur together than expected if they are statistically independent of each other.
- A measure of how *X* and *Y* are **really related** rather than coincidentally happening together.

$$\text{Lift}(X \Rightarrow Y) = \frac{\text{support}(X + Y)}{\text{support}(X) \times \text{support}(Y)}$$

- ***Lift* = 1** if *X* and *Y* are statistically independent
- ***Lift* > 1** indicates the **degree of usefulness** of the rule
  - A larger value of lift suggests a greater strength of the association between *X* and *Y*.

Evaluation of Candidate Rules

Lift

- E.g. Assuming 1000 transactions,
  - If {milk, eggs} appears in 300, {milk} in 500, and {eggs} in 400 of the transactions, then **Lift(milk → eggs)** = 0.3 / (0.5 * 0.4) = **1.5**
  - If {milk, bread} appears in 400, {milk} in 500, and {bread} in 400 of the transactions, then **Lift(milk → bread)** = 0.4/(0.5*0.4) = **2.0**

- Therefore it can be **concluded that milk and bread have a stronger association than milk and eggs.**

Evaluation of Candidate Rules

Leverage

- Measure the difference in the **probability of *X* and *Y* appearing together** in the dataset compared to what would be expected if *X* and *Y* were statistically independent of each other.

$$\text{Leverage}(X \rightarrow Y) = \text{Support}(X + Y) - \text{Support}(X) \times \text{Support}(Y)$$

- ***Leverage* = 0** if X and Y are statistically independent
- ***Leverage* > 0** indicates the **degree of relationship** between *X* and *Y*,
  - A larger leverage value indicates a stronger relationship between *X* and *Y*.

Evaluation of Candidate Rules

Leverage

- E.g. Assuming 1000 transactions,
    - If {milk, eggs} appears in 300, {milk} in 500, and {eggs} in 400 of the transactions, then **Leverage(milk → eggs)** = 0.3 - 0.5*0.4 = **0.1**
    - If {milk, bread} appears in 400, {milk} in 500, and {bread} in 400 of the transactions, then **Leverage (milk → bread)** = 0.4 - 0.5*0.4 = **0.2**

- It again **confirms that milk and bread have a stronger association than milk and eggs.**

Summary

- Assuming 1000 transactions,
  - If {milk, eggs} appears in 300, {milk} in 500, and {eggs} in 400 of the transactions.
  - If {milk, bread} appears in 400, {milk} in 500, and {bread} in 400 of the transactions.

	{Milk} → {Egg}	{Milk} → {Bread}
Confidence	${0.3}/{0.5} = 0.6$	${0.4}/{0.5} = 0.8$
Lift	${0.3}/{(0.5 \times 0.4)} = 1.5$	${0.4}/{(0.5 \times 0.4)} = 2$
Leverage	$0.3 - (0.5 \times 0.4) = 0.1$	$0.4 - (0.5 \times 0.4) = 0.2$
	Smaller	*Greater*

Evaluation of Candidate Rules

- Confidence is able to identify trustworthy rules, but it cannot tell whether a rule is coincidental.

- Measures such as lift and leverage not only ensure interesting rules are identified but also filter out the coincidental rules.

- Support, confidence, lift and leverage ensures the discovery of interesting and strong rules from sample dataset.

Outline

- Overview
- Apriori Algorithm
- Evaluation of Candidate Rules
- Examples
- Application of Association Rules
- Validation and Testing
- Diagnostic

Example 3

Min_Supp Count = 2
Min_Conf = 50%

Transaction ID	Items
100	I1,I2
200	I2,I3,I4,I5
300	I2,I3
400	I1
500	I1,I2,I3

C1

Itemset	Support Count
I1	3
I2	4
I3	3
I4	1
I5	1

L1

Itemset	Support Count
I1	3
I2	4
I3	3

Five Market Basket transactions with items labelled as I1, I2, I3 and so on.

1. **Candidate list generation, C1**: Start by creating all individual items called candidates and calculate their support counts.
2. **Create frequent 1-itemsets, L1**: Remove candidates that fail min_sup count (i.e. I4 and I5). No supersets of infrequent itemset must be generated and tested.

Min_Supp Count = 2
Min_Conf = 50%

C2

Itemset	Support Count
I1,I2	2
I1,I3	1
I2,I3	3

L2

Itemset	Support Count
I1,I2	2
I2,I3	3

C3

Itemset	Support Count
I1,I2,I3	1

L3 is null.

3. **Generate second candidate list, C2**: C2 is generated by L1 cross join L1.
4. **Create Frequent 2-itemsets, L2**: Remove candidates that fail min_sup count.
5. **Generate third candidate list, C3**: C3 is generated by L2 cross join L2.
6. **Support count for {I1, I2, I3} fails min_supp.** Association rule mining is completed and there will be no C4 candidate list.

Transaction ID	Items
100	I1,I2
200	I2,I3,I4,I5
300	I2,I3
400	I1
500	I1,I2,I3

{L1, L2} appears in 2 transactions together.

{L2, L3} appears in 3 transactions together.

{L1, L2, I3} appears in only 1 transaction together.

L2

Itemset	Support Count
I1,I2	2
I2,I3	3

Non-empty Subsets for {I1,I2}
I2
I1

Association Rules	Confidence	
{I2}=>I1	2/4=50%	✓
{I1}=>I2	2/3=66%	✓

Non-empty Subsets for {I2,I3}
I2
I3

Association Rules	Confidence	
{I2}=>I3	3/4=75%	✓
{I3}=>I2	3/3=100%	✓

L1

Itemset	Support Count
I1	3
I2	4
I3	3

7. **Generate candidate rules:** Take the last non-empty frequent itemset (i.e. L2 = {I1, I2}, {I2, I3}) and Use all the non-empty subsets of the frequent itemset to generate association rules.

8. **Calculate confidence of each candidate rule:**

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X + Y)}{\text{support}(X)}$$

9. **Strong association rules:** Green ticked are all strong rules satisfying min_conf >=50% and min_supp count =2.

Transaction ID	Items
100	I1,I2
200	I2,I3,I4,I5
300	I2,I3
400	I1
500	I1,I2,I3

Example 4

TID	List of Items
T100	I1, I2, I5
T101	I2, I4
T102	I2, I3
T103	I1, I2, I4
T104	I1, I3
T105	I2, I3
T106	I1, I3
T107	I1, I2 ,I3, I5
T108	I1, I2, I3

- Consider a database, D , consisting of 9 transactions.
- Suppose minimum support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required be 70%.
- We have first to find out the frequent itemsets using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Step 1: Generating 1-itemset Frequent Pattern

Scan D to count of each candidate

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count (2)

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

- In the first iteration of the algorithm, each item is a member of the set of candidate.
- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets satisfying minimum support.

Step 2: Generating 2-itemset Frequent Pattern

Generate C_2 candidates from L_1

$(L_1 x L_1)$

cartesian product L_{k-1} x L_{k-1}

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2

Compare candidate support count with minimum support count

Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

Step 2: Generating 2-itemset Frequent Pattern [Cont.]

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses **L_1 *Join* L_1** to generate a candidate set of 2-itemsets, C_2.
- Next, the transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated (as shown in the middle table).
- The set of frequent 2-itemsets, L_2 , is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.
- Note: **We haven't used Apriori Property yet.**

Step 3: Generating 3-itemset Frequent Pattern

Generate C_3 candidates from L_2

$L_2 x L_2$

Itemset
{I1, I2, I3}
{I1, I2, I5}
….

C_3

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2
….	

C_3

Compare candidate support count with min support count

Itemset	Sup Count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

- The generation of the set of candidate 3-itemsets, C_3 , involves the use of the Apriori Property.
- In order to find C_3, we compute L_2 *Join* L_2.
- C_3 = L2 *Join* L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.
- Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k.

Step 3: Generating 3-itemset Frequent Pattern [Cont.]

C_3 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.

- Based on the **Apriori property** that all subsets of a frequent itemset must also be frequent, we can determine that four latter candidates cannot possibly be frequent. How ?
- For example, let's take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L_2, we will keep {I1, I2, I3} in C_3.
- Let's take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, **{I3, I5} is not a member of** L_2 and hence it is not frequent **violating Apriori Property**. Thus we will have to remove {I2, I3, I5} from C_3.
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of **result of Join operation** for **Pruning**.
- Now, the transactions in D are scanned in order to determine L_3, **consisting of those candidates 3-itemsets in** C_3 **having minimum support.**

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L_3 *Join* L_3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.
- Thus, $C_4 = \phi$, and algorithm terminates**, having found all the frequent items. This completes our Apriori Algorithm.**

- What's Next ?

  These frequent itemsets will be used to generate **strong association rules** (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets

- Back To Example:

  List all frequent itemsets, L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.

  - Lets take the 3-itemsets: {I1,I2,I5}.
  - Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.
  - The resulting candidates rules are:
    - {I1,I2} → {I5}
    - {I1,I5} → {I2}
    - {I2,I5} → {I1}
    - {I5} → {I1,I2}
    - {I2} → {I1,I5}
    - {I1} → {I2,I5}

Step 5: Generating Association Rules from Frequent Itemsets [Cont.]

- Let **minimum confidence threshold** is , say 70%.
- The resulting association rules are shown below, each listed with its confidence.
  - R1: I1 ^ I2 → I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - **R1 is Rejected**.
  - R2: I1 ^ I5 → I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - R3: I2 ^ I5 → I1
    - Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
    - R3 is Selected.

{I1,I2,I5}

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}(X + Y)}{\text{support}(X)}$$

{I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}

Step 5: Generating Association Rules from Frequent Itemsets [Cont.]

- R4: I1 → I2 ^ I5
  - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
  - R4 is Rejected.
- R5: I2 → I1 ^ I5
  - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
  - R5 is Rejected.
- R6: I5 → I1 ^ I2
  - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
  - R6 is Selected.

In this way, we have found three strong association rules.

Outline

- Overview
- Apriori Algorithm
- Evaluation of Candidate Rules
- Examples
- Application of Association Rules
- Validation and Testing
- Diagnostic

Applications of Association Rules

The term **market basket analysis** refers to a specific implementation of association rules.

- For better merchandising – products to include/exclude from inventory each month
- Placement of products
- Cross-selling
- Promotional programs—multiple product purchase incentives managed through a loyalty card program

Applications of Association Rules

- Input: the simple point-of-sale transaction data
- Output: Most frequent affinities among items

- Example: according to the transaction data…

  "Customer who bought a laptop computer and a virus protection software, also bought extended service plan 70 percent of the time."

- How do you use such a pattern/knowledge?
  - Put the items next to each other for ease of finding
  - Promote the items as a package (do not put one on sale if the other(s) are on sale)
  - Place items far apart from each other so that the customer must walk the aisles to search for it, and by doing so potentially seeing and buying other items

Applications of Association Rules

Recommender systems – Amazon, Netflix:

- Clickstream analysis from web usage log files
- Website visitors to page X click on links A,B,C more than on links D,E,F

In medicine:

- relationships between symptoms and illnesses;
- diagnosis and patient characteristics and treatments (to be used in medical DSS);
- genes and their functions (to be used in genomics projects)..

Outline

- Overview
- Apriori Algorithm
- Evaluation of Candidate Rules
- Application of Association Rules
- Examples
- Validation and Testing
- Diagnostic

Validation and Testing

- *The frequent* and *high confidence* itemsets are found by pre-specified minimum support and minimum confidence levels
- *Measures like lift* and/or *leverage* then ensure that interesting rules are identified rather than coincidental ones
- However, some of the remaining rules may be considered subjectively uninteresting because they don't yield unexpected profitable actions
    - E.g., rules like {paper} → {pencil} are not interesting/meaningful
- Incorporating *subjective knowledge* requires domain experts
- Good rules provide valuable insights for institutions to improve their business operations

Outline

- Overview
- Apriori Algorithm
- Evaluation of Candidate Rules
- Examples
- Application of Association Rules
- Validation and Testing
- Diagnostic

Diagnostics

- Although the Apriori algorithm is easy to understand and implement, some of the rules generated are **uninteresting** or **practically useless**.
- Additionally, some of the rules may be generated due to **coincidental relationships** between the variables.
- Measures like **confidence, lift, and leverage** should be used **along with human insights** to address this problem.

Diagnostics

- Another problem with association rules is that, in Phase 3 and 4 of the Data Analytics Lifecycle , the team **must specify the minimum support** prior to the model execution, which may lead to **too many or too few rules**.
- In related research, a variant of the algorithm can use a **predefined target range** for the number of rules so that the algorithm can adjust the minimum support accordingly.
- Algorithm requires a scan of the entire database to obtain the result. Accordingly, as the database grows, it takes more **time** to compute in each run.

Diagnostics

Approaches to improve Apriori's efficiency:

Partitioning:

- Any itemset that is potentially frequent in a transaction database must be frequent in at least one of the partitions of the transaction database.

Sampling:

- This extracts a subset of the data with a lower support threshold and uses the subset to perform association rule mining.

Transaction reduction:

- A transaction that does not contain frequent k-itemsets is useless in subsequent scans and therefore can be ignored.

Hash-based itemset counting:

- If the corresponding hashing bucket count of a k-itemset is below a certain threshold, the k-itemset cannot be frequent.

Dynamic itemset counting:

- Only add new candidate itemsets when all of their subsets are estimated to be frequent.

References

- EMC Education Services. (2015). *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. Wiley.

- Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

- Data Camp, ‘Market Basket Analysis using R’, retrieved from https://www.datacamp.com/community/tutorials/market-basket-analysis-r