Mining Association Rules

- It is one of techniques used to discover patterns from massive data.
- The type of data being processed by this technique is usually transaction data.
  - This technology could be used for market analysis, pattern associations.
  - Example: 90% of transactions that purchase bread and butter also purchase milk.
  - The antecedent of this rules consists of bread and butter, and the consequent consists of milk.

- Mining Association Rules: the presence of one set of items implies the presence of another set of items.

  Ex. People who purchased hammers also purchased nails.
- An association rule is an expression X=>Y, where X and Y are sets of items.
  - i.e. transactions of the database which contain X tend to contain Y.
- Given:

  (1)a database of transactions TX

  (2)each TX has a list of items purchased

Applications

- Bar-code technology can collect transaction data.
- Examples
  - Find all rules that have “Diet Coke” as consequent. These rules may help plan what the store should do to boost the sale of Diet Coke.
  - Find all rules that have “bagels” in the antecedent. These rules may help determine what products may be impacted if the store discontinues selling bagels.
  - Find all the rules relating items located on shelves A and B in the store. These rules may help shelf planning by determining if the sale of items on shelf A is related to the sale of items on shelf B.

Definitions

- Let
  - $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items.
  - D be a set of transactions
  - Each transaction T is an itemset such that $T \subseteq I$
- We say that a transaction T contains an itemset X, if $X \subseteq T$.
- An association rule is an implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$.

Support

- The Rule $X \Rightarrow Y$ has support *s* in the transaction set D if *s*% of transactions in D contain X.

- support = $\dfrac{\#\ of\ trans\ containing\ all\ the\ items\ in\ X \cup Y}{total\ \#\ of\ trans}$

- example: 30% of **<u>all</u>** transactions have bread, butter and milk.

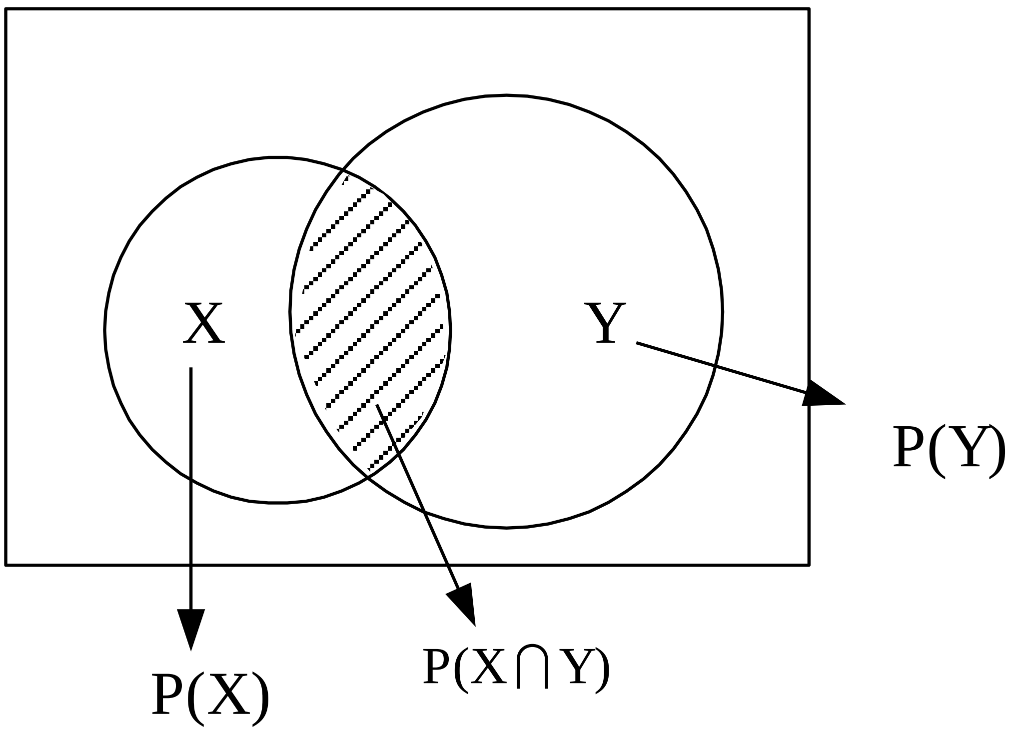
Confidence

- The Rule $X \Rightarrow Y$ holds in the transaction set D with confidence *c* if *c*% of transactions in D that contain X also contain Y.
- confidence = $\dfrac{\#\ of\ trans\ that\ contain\ both\ X\ and\ Y}{\#\ of\ trans\ containing\ X}$

$$= \frac{\mathrm{support}(X \cup Y)}{\mathrm{support}(X)}$$

- Example: 90% of transactions that purchase bread and butter also purchase milk.

Example



- $X \Rightarrow Y$

Support=$P(X \cap Y)$

Confidence=

$$\frac{\text{support}(X \cup Y)}{\text{support}(X)} = \frac{P(X \cap Y)}{P(X)}$$

- Support and Confidence
  - Support corresponds to statistical significance.
  - Confidence is a measure of the rule’s strength.

Items	**Count**
Bread	3
Butter	3
Milk	3
Bread Butter	2
Bread Milk	3
Butter Milk	2
Bread Butter Milk	2

total # of transaction = 4

Antecedent	**Consequent**	**Support**	**Confidence**
Bread	Butter	50 %	67 %
Bread	Milk	75 %	100 %
Bread	Butter, Milk	50 %	67 %
Butter	Bread	50 %	67 %
Milk	Bread	75 %	100 %
Milk	Butter	50 %	67 %
Bread Butter	Milk	50 %	100 %
Bread Milk	Butter	50 %	67 %

The Task of Mining Association Rules

- find all rules or patterns that satisfy user specified minimum support (minsup) and minimum confidence (minconf).
  - for example: 50% as minsup and 66% minconf

- find out strong rules in the transactions
  - Strong rules is defined as rules with high confidence and strong support.

How to Mining Association Rules

- Find the strong rules.
  - To find the strong rules, two tasks are required.
    - Discovery the large itemsets (the sets of itemset) that have transaction support above a pre-detemined minimum support.
    - Use the large itemsets found to generate the association rules in the database.
  - After the large itemset are identified, the corresponding association rules can be derived in a straightforward manner.
  - The performance of the mining technique is largely determined by the first task.

Methods for Mining Association Rules

- Apriori Algorithm (Agrawal & Srikant'94)
  - The original mining association rules.
  - Apriori algorithm is mainly used to search the largest itemsets.
  - Apriori *constructs a candidate set of large itemsets*, *counts the number of occurrence of each candidate itemset*, and *then determines large itemsets based on a pre-determined minimum support*.
  - Derive association rules.

Example (Apriori)

- Let min_support=50% and min_confidence=60%. Since there are four records in the table, the number of transactions above the minsup is 2 (4 * 50% = 2).

Database D

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

- *k*-itemset An itemset haves *k* items.
- L_k Set of a large *k*-itemset.
- C_k Set of a candidate *k*-itemset.

Database *D*

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Scan D $\longrightarrow$

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}

Scan D $\longrightarrow$

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}

Scan D $\longrightarrow$

C_3

Itemset	Sup.
{B C E}	2

L_3

Itemset	Sup.
{B C E}	2

Apriori's Algorithm

```
L_1 = {large 1-itemsets};
for (k=2; L_k-1≠0; k++) do begin
    C_k=apriori-gen(L_k-1); // New candidates
    forall transactions t∈D do begin
            C_t=subset(C_k,t);  // Candidates contained in t
            forall candidates c ∈C_t do
                    c.count++;
    end
    L_k={c ∈C_k | c.count≥minsup}
end
Answet=∪_k L_k;
```

- The apriori-gen function takes an argument L_{k-1}, the set of all large (*k*-1)-itemsets. It returns a superset of the set of all large *k*-itemsets. The function works. First, in the join step, we join L_{k-1} with L_{k-1};

insert into C_k
select $p.item_1$, $p.item_2$, …, $p.item_{k-1}$, $q.item_{k-1}$
from L_{k-1} p, L_{k-1} q
where $p.item_1=q.item_1$, …, $p.item_{k-2}=q.item_{k-2}$, $p.item_{k-1}<q.item_{k-1}$;

- Next, in the prune step, we delete all itemsets $c \in C_k$ such that some (k-1)-subset of c is not in L_{k-1};

forall itemsets $c \in C_k$ do
  forall (k-1)-subsets s of c do
    if $(s \notin L_{k-1})$ then
      delete c from C_k;

- For Example:

Large 3-itemsets	Candidates 4-itemsets (after join)	Candidates 4-itemsets (after pruning)
{A B C} {A B D} {A C D} {A C E} {B C D}	{A B C D} {A C D E}	{A B C D}

Algorithm AprioriTID

- The database *D* is not used for counting support after the first pass.
- Each member of the set $\overline{C}_k$ is of the from <TID, $\{X_k\}$>, where each X_k is a potentially large *k*-itemset present in the transaction with identifier TID.

Database

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

$\overline{C}_1$

TID	Set-of-Itemsets
100	{{A}, {C}, {D}}
200	{{B}, {C}, {E}}
300	{{A}, {B}, {C}, {E}}
400	{{B}, {E}}

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

$\overline{C}_2$

TID	Set-of-Itemsets
100	{{A C}}
200	{{B C}, {B E}, {C E}}
300	{{A B}, {A C}, {A E}, {B C}, {B E}, {C E}}
400	{{B E}}

L_2

Itemset	Sup.
{A C}	2
{B C}	2
{B E}	3
{C E}	2

C_3

Itemset	Sup.
{B C E}	2

$\overline{C}_3$

TID	Set-of-Itemsets
200	{{B C E}}
300	{{B C E}}

L_3

Itemset	Sup.
{B C E}	2

DHP Algorithm (Direct Hashing with Pruning)

- Apriori + hashing (Park, Chen and Yu'95)
- use hash-based method to reduce the size of C2
- Allow effective reduction on Tx database size (Tx number and each Tx size)

Example (DHP)

on the fly

C_1	count
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1
{A}
{B}
{C}
{E}

minimum support, s=2

Making a hash table

100 {A C}, {A D}, {C D}

200 {B C}, {B E}, {C E}

300 {A B}, {A C}, {A E}, {B C}, {B E}, {C E}

400 {B E}

h{{x y}} = ((order of x) * 10 + (order of y)) mod 7;

	{C E}				{B E}		{A C}
	{C E}		{B C}		{B E}		{C D}
	{A D}	{A E}	{B C}		{B E}	{A B}	{A C}
Hash table H_2	3	1	2	0	3	1	3
Hash address	0	1	2	3	4	5	6

Generating C_2

$L_1 * L_1$	# in a bucket with the itemset
{A B}	1
{A C}	3
{A E}	1
{B C}	2
{B E}	3
{C E}	3

$\longrightarrow$

C_2
{A C}
{B C}
{B E}
{C E}

Example of L_2 and D_3

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

{A C} $\rightarrow$ Discard
{B C} {B E} {C E} $\rightarrow$ Keep {B C E}
{A C} {B C} {B E} {C E} $\rightarrow$ Keep {B C E}
{B E} $\rightarrow$ Discard

$D_3 = \{<200,\ B\ C\ E>,\ <300,\ B\ C\ E>\}$

C_2	count
{A C}	2
{B C}	2
{B E}	3
{C E}	2

$\rightarrow$

L_2
{A C}
{B C}
{B E}
{C E}

s = 2

Counting support in a hash tree

Boolean Algorithm

- Boolean Algorithm (Wur & Leu ’99)
  - Similar to the Apriori algorithm.
  - By only scanning the database once and avoiding generating candidate itemsets in computing frequent itemsets.
  - Step 1: the frequent itemsets is identified.
  - Step 2: the association rules based on the identified frequent itemsets are generated.

Boolean Operations

- Logic OR and AND operations are used to compute frequent itemsets.
- Logic AND and XOR operations are applied to derive all interesting association rules based on the computed frequent itemsets.

V1	1	1	0	0
V2	1	0	1	0
V1 AND V2	1	0	0	0
V1 OR V2	1	1	1	0
V1 XOR V2	0	1	1	0

Example (Boolean)

Database D

TID	Items
100	ACD
200	BCE
300	ABCE
400	BE

Item Set I : ABCDE

Minimum support:40% (2)
Minimum confidence:50%

The Initial TITTC Table

	A	B	C	D	E	T100	T200	T300	T400	Count
A	1	0	0	0	0	1	0	1	0	2
B	0	1	0	0	0	0	1	1	1	3
C	0	0	1	0	0	1	1	1	0	3
D	0	0	0	1	0	1	0	0	0	1
E	0	0	0	0	1	0	1	1	1	3

The $TITTC_1$ Table

	A	B	C	D	E	T100	T200	T300	T400	Count
A	1	0	0	0	0	1	0	1	0	2
B	0	1	0	0	0	0	1	1	1	3
C	0	0	1	0	0	1	1	1	0	3
E	0	0	0	0	1	0	1	1	1	3

The TITTC$_2$ Table

	A	B	C	D	E	T100	T200	T300	T400	Count
AC	1	0	1	0	0	1	0	1	0	2
BC	0	1	1	0	0	0	1	1	0	2
BE	0	1	0	1	0	0	1	1	1	3
CE	0	0	1	0	1	0	1	1	0	2

The TITTC$_3$ Table

	A	B	C	D	E	T100	T200	T300	T400	Count
BCE	0	1	1	0	1	0	1	1	0	2

The final TIC Table

	A	B	C	D	E	Count
A	1	0	0	0	0	2
B	0	1	0	0	0	3
C	0	0	1	0	0	3
E	0	0	0	0	1	3
AC	1	0	1	0	0	2
BC	0	1	1	0	0	2
BE	0	1	0	0	1	3
CE	0	0	1	0	1	2
BCE	0	1	1	0	1	2

The TAR Table

Antecedent	Consequent	Support	Confidence
A	C	50 %	100 %
B	C	50 %	67 %
B	E	75 %	100 %
B	CE	50 %	67 %
C	A	50 %	67 %
C	B	50 %	67 %
C	E	50 %	67 %
C	BE	50 %	67 %
E	B	75 %	100 %
E	C	50 %	67 %
E	BC	50 %	67 %
BC	E	50 %	100 %
BE	C	50 %	67 %
CE	B	50 %	100 %