Sr. No.	Module	Detailed Contents	Hrs
1	**Business Intelligence-**	Introduction and overview of BI-Effective and timely decisions, Data Information and knowledge, BI Architecture, Ethics and BI. BI Applications- Balanced score card, Fraud detection, Telecommunication Industry, Banking and finance, Market segmentation.	**06**
2	**Prediction methods and models for BI**	Data preparation, Prediction methods-Mathematical method, Distance methods, Logic method, heuristic method-local optimization technique, stochastic hill climber, evaluation of models	**06**
3	**BI using Data Warehousing**	Introduction to DW, DW architecture, ETL Process, Top-down and bottom-up approaches, characteristics and benefits of data mart, Difference between OLAP and OLTP. Dimensional analysis- Define cubes. Drill- down and roll- up – slice and dice or rotation, OLAP models- ROLAP and MOLAP. Define Schemas- Star, snowflake and fact constellations.	**08**
4	**Data Mining and Preprocessing**	Data mining- definition and functionalities, KDD Process, Data Cleaning: - Missing values, Noisy data, data integration and transformations. Data Reduction: - Data cube aggregation, dimensionality reduction- data compression, Numerosity reduction- discretization and concept hierarchy.	**06**
5	**Associations and Correlation**	Association rule mining:-support and confidence and frequent item sets, market basket analysis, Apriori algorithm, Incremental ARM, Associative classification- Rule Mining.	**06**
6	**Classification and Prediction**	Introduction, Classification methods:-Decision Tree- ID3, CART, Bayesian classification- Baye'stheorem(Naïve Bayesian classification),Linear and nonlinear regression.	**08**
7	**Clustering**	Introduction, categorization of Major, Clustering Methods:- partitioning methods- K-Means. Hierarchical- Agglomerative and divisive methods, Model- based- Expectation and Maximization.	**08**
8	**Web mining and Text**	Text data analysis and Information retrieval, text retrieval methods, dimensionality reduction for text	**04**

KDD Process

Knowledge Discovery in Databases

-Explosive growth of data

-data collection and data availability

source:web, ecommerce,bioinformatics,simulation etc

-data drowning but starving for knowledge

-automated analysis of data is required

KDD Process

Knowledge Discovery in Databases

KDD is the process of finding useful info and patterns in data.

Data mining is the **use of algorithms** to extract the info and patterns derived by the KDD process

-extraction of interesting (nontrivial, previously unknown, and potentially useful) patterns or knowledge from huge amount of data

KDD process consists of following steps:

- Selection
- Preprocessing
- Transformation
- Data mining
- Interpretation / evaluation

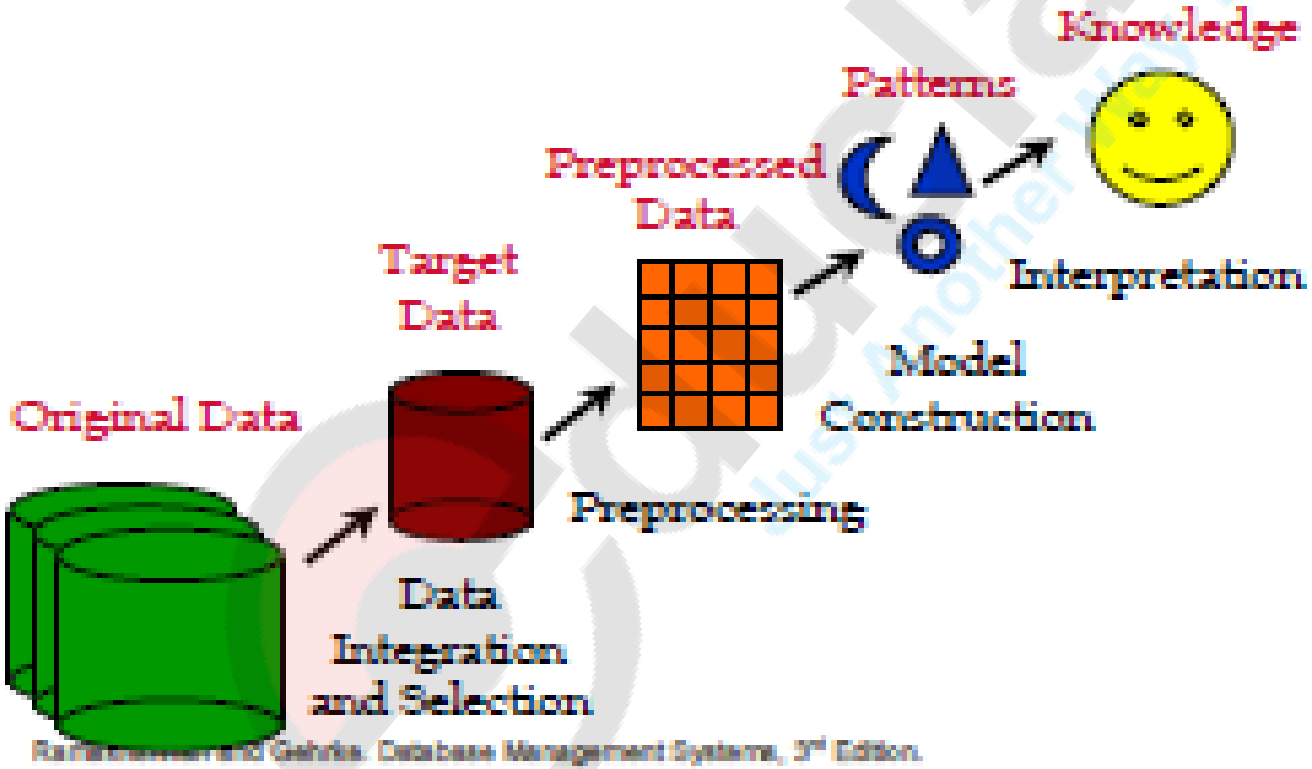
KDD Process

- Selection
  - Obtains the data from various databases, files & non electronics sources.
- Preprocessing
  - Data obtained may be incorrect or missing.
  - There may be anomalous data from multiple sources having different data types
- Transformation
  - Data is converted into a common format for processing
  - some data may be encoded, transformed to more usable format.
  - Data reduction may be used to reduce the number of possible data values being considered.

KDD Process

- Data mining
  - Data mining algo. are applied to transformed data to generate the desired results.
- Interpretation / evaluation
  - Usefulness of the results depends upon how they are presented to the user.
  - Various visualization & GUI strategies are used at this step.

Preprocessing and Mining



Ramakrishnan and Gehrke, Database Management Systems, 3rd Edition.

Association Rule

- Earliest use of data mining in supermarkets
  - Eg. 80-90's
- Market Basket transactions
  - Millions of records can be studied to study patterns in form of rules

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?— Bread and butter?!
  - What are the subsequent purchases after buying a PC?
  - 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?

- Discloses an intrinsic and important property of data sets
- Forms the 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: associative classification
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing: iceberg cube and cube-gradient
  - Semantic data compression: fascicles
  - Broad applications

- Finding frequent patterns plays an essential role in mining association, correlations and many other interesting relationship among data.
- Searches for recurring relationship in a given data set.
- Can help in various decision making processes.
- These patterns can be represented in the form of association rules

Market Basket Analysis

- Association rule algos are used to do market basket analysis to a set of data.
- The purpose of market basket analysis is to find interesting relationships among retail products. The results of a market basket analysis help retailers to :
  - design promotions
  - arrange shelf or catalog items &
  - develop cross-marketing strategies.
  - manage stock

Counting Co-occurrences

- A market basket is a collection of items purchased by a customer in a single customer transaction.
- A customer transaction consists of purchasing the items from the store by single visit.
- A common goal for retailer is to identify items that are purchased together.
  - ★ Frequent Itemsets

transid	custid	date	item	qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	5/10/99	Pen	1
113	106	5/10/99	Milk	1
114	201	6/1/99	Pen	2
114	201	6/1/99	Ink	2
114	201	6/1/99	juice	4
114	201	6/1/99	water	1

Association Rule

- Market Basket transactions
  - Millions of records can be studied to study patterns in form of rules
- Fields
  - Web usage mining
  - Banking
  - Bio informatics
  - Credit/ debit card analysis
  - Product clustering
  - Catalog design
- Places items together or provide discount
  - Train ticket → taxi ticket
- Algo
  - Apriori → candidate generation → Large Itemset
  - FP growth → fp tree generation

Frequent Item sets

- A set of items is called item set.
- Rule *support* and *confidence* are two measures of rule interestingness.
- They reflect the usefulness and certainity of discovered rules.
- Association rules are considered interesting if they satisfy both a minimum support threshold and minimum confidence threshold.
- These thresholds can be set by users or domain experts.
- The support of an item set is a fraction of transaction in the database that contains all the item from itemset.

OR

- The *support* supp(X) of an itemset X is defined as the proportion of transactions in the data set which contain the itemset

Association Rule

- Itemset
  - T1→ {1,2,3}
  - T2 → {2,3}
- Given a set of transaction T , the goal is to find all rules having
  - support >= minsup threshold
  - Confidence >= minconf threshold

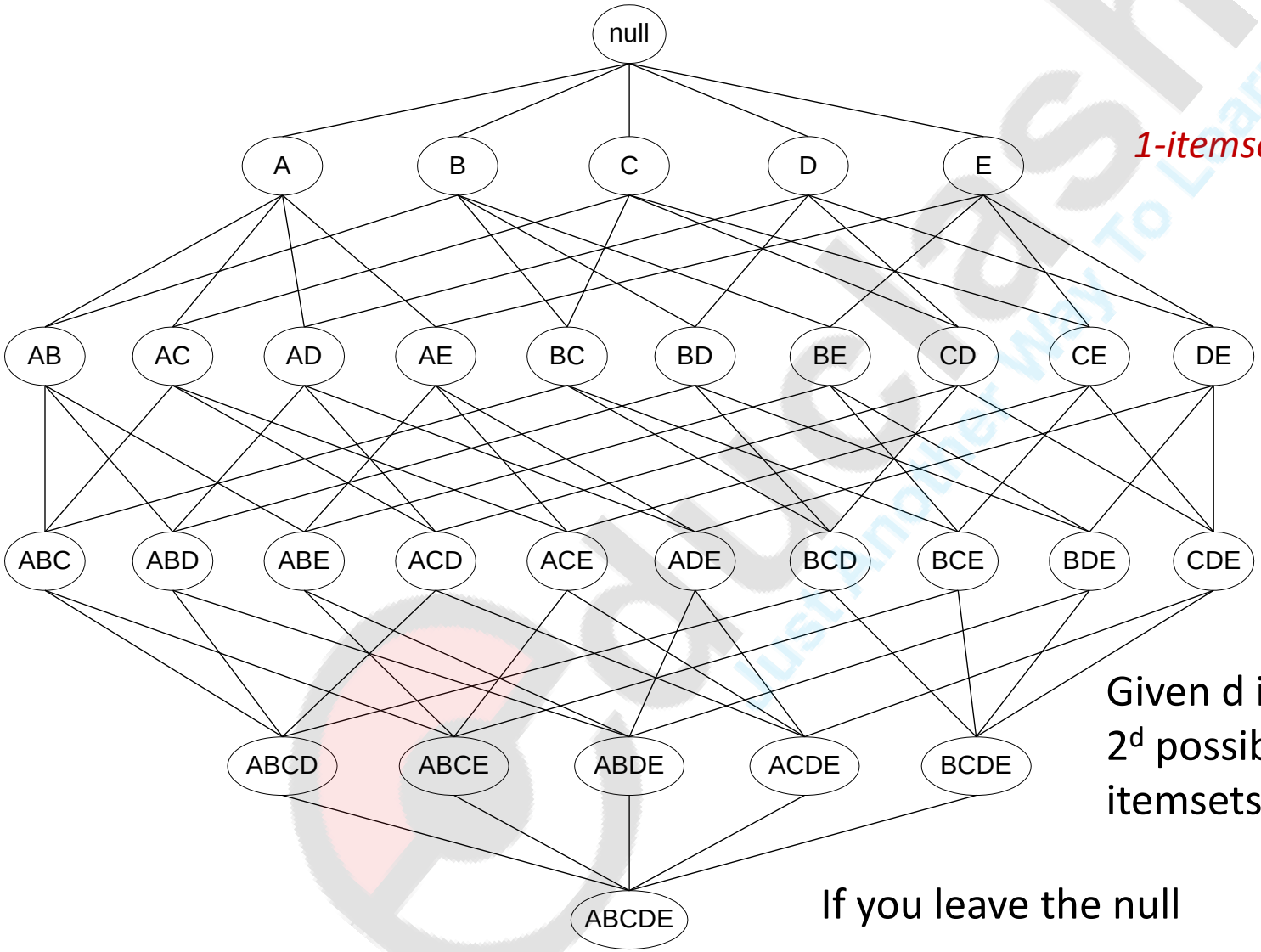
- Bread => butter [13.33%, 45 %]
- Support is 13.33%
- Confidence is 45 %

- Total transactions =150
- Bread : in 20 transactions
- Support {Bread}= [20/150]*100=13.33%
- In 20 transaction butter is in 9 transactions
- Confidence{Bread}= [9/20] *100
  =[supp(bread,butter)/supp(bread)]*100
  =45%

Frequent Item sets

- For e.g. {pen, ink} has 75% support in purchases.
- We thus conclude that pen and ink are the items which are frequently purchased together.
- On the other hand, {milk, juice} are not purchased together frequently.
- User can specify the minimum support (minsup) and find all itemsets whose support is above minsup. Such itemsets are called frequent itemset.
- These sets of items may be a singleton set.
- Let's consider the user specified minimum support as 70% then frequent items will be {pen}, {ink}, {milk}, {pen, ink}, {pen, milk}.

Frequent Itemset Generation



1-itemsets

2-itemsets

3-itemsets

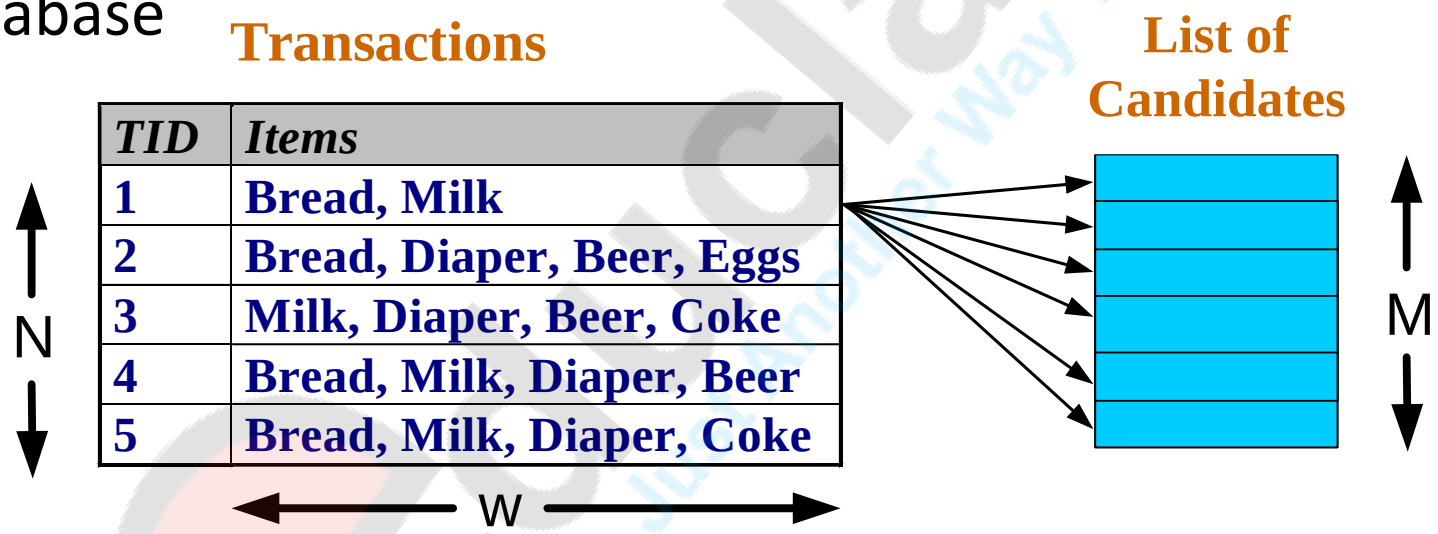
Given d items, there are 2^d possible candidate itemsets

If you leave the null

2^d -1 possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database



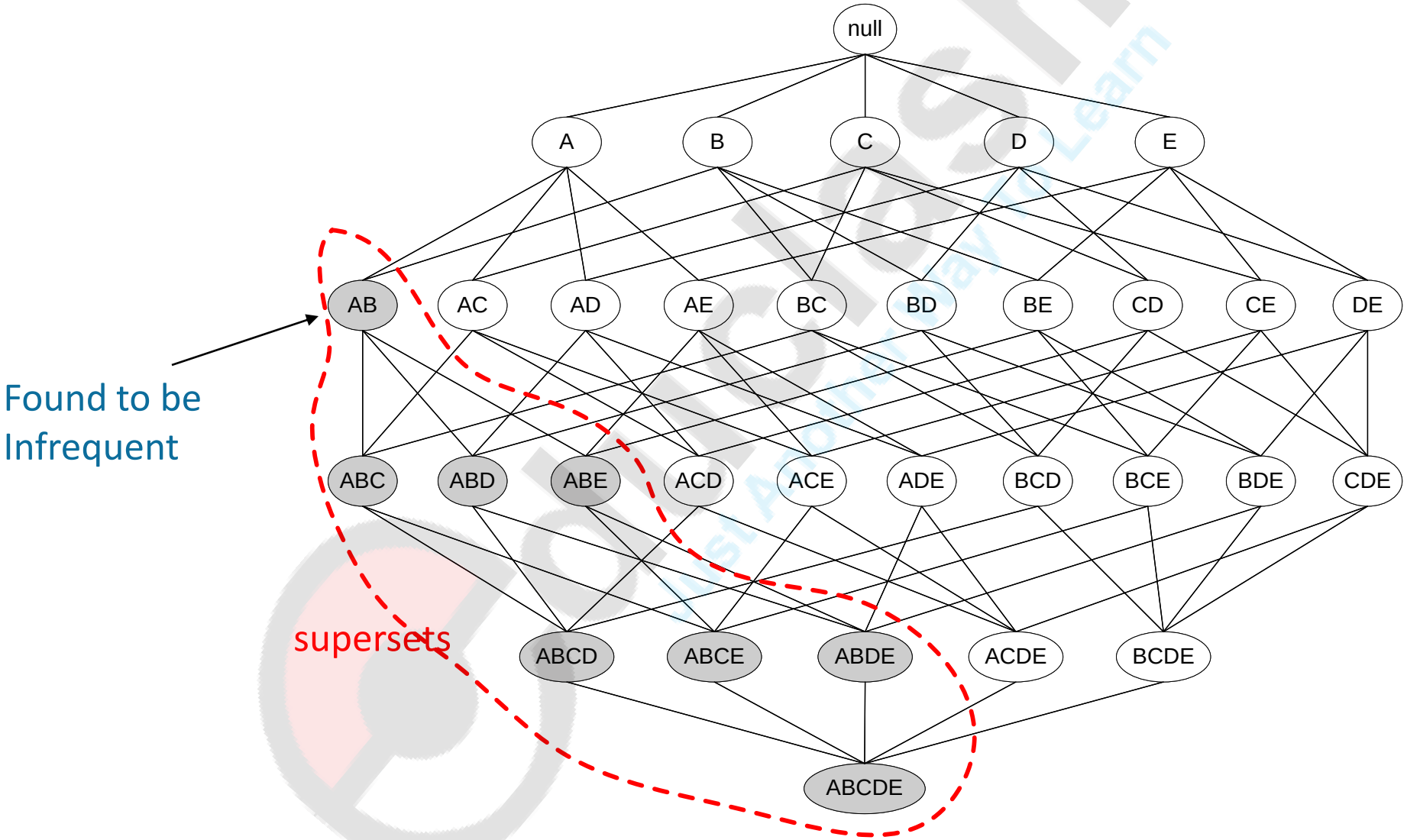
  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Algo. to identify frequent itemset

Algorithm to identify frequent itemset is based on a simple but fundamental property of frequent itemsets:

- The a Priori Property (antimonotone): Every **subset** of a frequent itemset is also a frequent itemset.
- Algo uses the prior knowledge of frequent item sets.
- Employs an iterative approach known as *level-wise search* where *K-itemsets* are used to explore *(k+1) itemsets.*
- This property helps avoid scanning through a no. of itemsets for search of freq. itemsets.
  - If AB is not frequent ABC,ABE,ABCD,ABCE,ABCDE etc are also not frequent.

Illustrating Apriori Principle



ALGO. TO IDENTIFY FREQUENT ITEMSET

For each item

check if it is frequent itemset // (appears in > minsup)

K=1

Repeat

for each new frequent itemset, I_k with K items,

generate all itemsets I_{k+1} with K+1 items, $I_k \subset I_{k+1}$

Scan all transactions once and check

If generated I_{k+1} with K+1 items are frequent

K:= K+1

Until new frequent itemsets are identified.

Association Rules

- Many algorithms have been proposed for discovering various forms of rules that briefly describe the data.
- An association rule has the form LHS ➔ RHS, where both LHS and RHS are sets of items.
- The interpretation of such a rule is that if every item in LHS is purchased in a transaction, then it is likely that the items in RHS are purchased as well.
- By examining the set of transaction in Purchase, we can identify rules of the form:

  { pen} ➔ {ink}

- This is read as "If a pen is purchased in a transaction, it is likely that the ink is also purchased in that transaction".

Example 1 :

transaction	Itemset
1	a,b,c
2	a,c
3	a,d
4	b,e,f

- Minsupport = 50%
- minConfidence= 50%

Support count=(minsupport/100) * total no. of transactions

= (50/100)*4

Support count = 2

C1

items	support
a	3
b	2
c	2
d	1
e	1
f	1

L1

items	support
a	3
b	2
c	2

C2

items	support
a,b	1
b , c	1
c, a	2

L2

items	support
c, a	2

transaction	Itemset
1	a,b,c
2	a,c
3	a,d
4	b,e,f

- Minsupport = 50%
- MinConfidence= 50%

items	support
c, a	2

Association rule	support	confidence	Confidence %
a→c	2	=support/occurrence of “a” =2/3 =0.66	66% >minconf
c→a	2	=support/occurrence of “c” =2/2 =1	100% >minconf

Final Rules are a→c and c→ a.

Example 2

- Minsupport = 60%
- minConfidence= 80%

Support count=(minsupport/100) * total no. of transa

= (60/100)*5

Support count = 3

transaction	Itemset
1	m,o,n,k,e,y
2	d,o,n,k,e,y
3	m,a,k,e,
4	m,u,c,k,y
5	c,o,o,k,i,e

C1

items	support
M	3
O	3
N	2
K	5
E	4
Y	3
D	1
A	1
U	1
C	2
I	1

L1

items	support
M	3
O	3
K	5
E	4
Y	3

C2

items	support
M,o	1
M,k	3
M,e	2
M,y	2
O,k	3
O,e	3
O,y	2
K,e	4
K,y	3
E,y	2

L2

items	support
M k	3
O, k	3
O, e	3
K,E	4
K,Y	3

Example 2:

- Minsupport = 60%
- minConfidence= 80%

transaction	Itemset
1	m,o,n,k,e,y
2	d,o,n,k,e,y
3	m,a,k,e,
4	m,u,c,k,y
5	c,o,o,k,i,e

C3

items	support
M,k,o,	1
M ke	2
Mky	2
Oke	3
Oky	2
Key	2

L3

items	support
O,k,e	3

C4 is not possible as only

Are rules

Association rule	support	confidence	Confidence %
O,k→E	3	=support/occurrence of O & K =3/3 = 1	100% >minconf
O,e → K	3	=3/3 =1	100% >minconf
K,e →o	3	=3/4=0.75	75%
E→o,k	3	=3/4=0.75	75%
K→o,e	3	=3/5=0.6	60%
O→k,e	3	=3/3=1	100%

Association Rule

- In Rule

  X ➔ Y } X is antecedent and Y is consequent

- If X appears, Y also tend to appear.
- Two important measures (Evaluation metrics) for an association rule :
  - ★Support (Prevalence)
  - ★Confidence (Predictability)

Important Measures for association rule

Support :-

- The support for an itemset (X) is the percentage of transactions that contain (X).

- The support for a rule LHS ➔ RHS
  - is fraction of transactions that contain both LHS & RHS.

$$\text{Supp (LHS} \rightarrow \text{RHS)} = \text{Supp(LHS U RHS)}$$

- supp(pen ➔ ink) is 75%.
- Pen & ink appear together in 75% transactions

Important Measures for association rule

Confidence :-

- Confidence of a rule is an indication of the strength of the rule.

$$\text{Conf (LHS} \rightarrow \text{RHS)} = \frac{\text{Sup(LHS U RHS)}}{\text{Sup(LHS)}}$$

sup (LHS U RHS) = % of transac that contains both LHS & RHS.

sup (LHS) = % of transac that contains LHS.

- Conf (pen ➔ ink) is 75%.
- Conf (pen ➔ ink) = [Sup(pen U ink) / Sup(pen)] * 100

Apriori: A Candidate Generation-and-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

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

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

C_1

1st scan →

Apriory Algorithm

Note: user specifies 2 parameters: minsupport (p %), minconf.

Step 1 (Find FIs):

- Scan all trans. and find FI(Large itemset) having one item. These have support above p%. Let these be L_1.

Iterations:

- Recursively generate L_2, L_3, etc (using previous frequent itemsets) until FIs of all sizes are generated.
- General procedure to obtain k-itemsets from (k-1) itemsets for k=2, 3, …. is as follows:
  - Create a candidate list C_k of k-itemsets by performing a join operation on pairs of (k-1) itemsets in the list L_{k-1} . A pair is combined only if the first (k-2) items are the same in both itemsets.
  - Scan the trans. (DB) to find L_k from C_k .

Step 2 (Find ARs):

- Generate the rules by dividing Large itemsets L_k into LHS and RHS part such that LHS ➔ RHS is valid association rule if it meets minconf. requirement.

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 !=∅; 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 $\cup_k L_k$;

Educlash
Just Another Way To Learn

Example 3

- Minsupport = 2
- minConf= 60%

transaction	Itemset
100	1,3,4
200	2,3,5
300	1,2,3,5

Subsets are not frequent
Even if one subset is not frequent
discard → **pruning**

C1

items	support
1	3
2	3
3	4
4	1
5	4

L1

items	support
1	3
2	3
3	4
5	5

C2

items	support
1,2	1
1,3	
1,5	
2	
	3
3,5	3

items	support
1,3	3
1,5	2
2,3	2
2,5	3
3,5	3

C3

items
1,2,3
1,2,5
1,3,5
2,3,5

L3

items	support
1,3,5	2
2,3,5	2

c4

items	support
1,2,3,5	1

←Discard as supp is 1
But if it was 2 then we would have to again prune

Subsets → {1,2,3},{1,2,5}{1,3,5},{2,3,5} where {1,2,3},{1,2,5} are not frequent hence would have to discard even otherwise.

Example 3

- Minsupport = 2
- minConf= 60%

transaction	Itemset
100	1,3,4
200	2,3,5
300	1,2,3,5
400	2,5
500	1,3,5

C1

items	support
1	3
2	3
3	4
4	1
5	4

L1

items	support
1	3
2	3
3	4
5	5

C2

items	support
1,2	1
1,3	3
1,5	2
2,3	2
2,5	3
3,5	3

L2

items	support
1,3	3
1,5	2
2,3	2
2,5	3
3,5	3

C3

items	support
1,2,3 subsets →{1,2}{1,3}{2,3}	X
1,2,5 → {1,2},{1,5},{2,5}	X
1,3,5 → {1,3},{1,5},{3,5}	2
2,3,5 → {2,3},{3,5},{2,5}	2

L3

items	support
1,3,5	2
2,3,5	2

c4

items	support
1,2,3,5	1

←Discard as supp is 1
But if it was 2 then we would have to again prune

Subsets → {1,2,3},{1,2,5}{1,3,5},{2,3,5} where {1,2,3},{1,2,5} are not frequent hence would have to discard even otherwise.

Example 3:

- minSupport = 2
- minConf= 60%

L3

items	support
1,3,5	2
2,3,5	2

Association rule	support	confidence	Confidence %
{1,3}→5	2	Supp(1,3,5)/supp(1,3)= 2/3	66.66%
{1,5}→ 3	2		
{5,3} → 1	2		
5→{1,3}	2		60.60%
3→{1,5}	2		50% reject
1→{5,3}	2		66.6%
……			

- Bread => butter [13.33%, 45 %]
- Support is 13.33%
- Confidence is 45 %

- Total transactions =150
- Bread : in 20 transactions
- Support {Bread}= [20/150]*100=13.33%
- In 20 transaction butter is in 9 transactions
- Confidence {Bread→ butter}= [9/20] *100
  =[supp(bread,butter)/supp(bread)]*100
  =45%
- Lift {x→ y} = confidence{x→ y}/supp(x)
  - Lift =1 → x has no impact on y
  - Lift>1 , +ve **correlation**, y is likely to be bought with x
  - Lift < 1,-ve **correlation** , y is likely not to be bought with x
  - Larger the lift ratio , more significant association

Confidence {Bread→ butter}=

$$P(\text{butter}|\text{bread}) = \frac{P(\text{butter and Bread})}{P(\text{Bread})}$$

Lift {Bread→ butter}=

$$= \frac{P(\text{butter and Bread})}{P(\text{Bread})P(\text{butter})}$$

Example 1

Trans id	Items
1	Card reader, Memory card, Mobile, Laptop
2	Card reader, Mobile, Laptop
3	digi cam, Laptop, LCD TV
4	Card reader, digi cam, Laptop
5	Card reader, digi cam, Mobile

- Find association rule with 50 % support and 75 % confidence for the above data.
- Find support and confidence of the following rules:
  - Laptop ➔ Card reader
  - Laptop, Mobile ➔ Card reader

Example 2

- Find association rule with 25 % support and 70 % confidence for the following data:

Trans id	Items
1	Biscuits, Bread, Cheese, Coffee, Yogurt
2	Bread, Cereal, Cheese, Coffee
3	Cheese, Chocolate, Donuts, Juice, Milk
4	Bread, Cheese, Coffee, Cereal, Juice
5	Bread, Cereal, Chocolate, Donuts, Juice
6	Milk, Tea
7	Biscuits, Bread, Cheese, Coffee, Milk
8	Egg, Milk, Tea
9	Bread, Cereal, Cheese, Chocolate, Coffee
10	Bread, Cereal, Chocolate, Donuts, Juice

Apriori Algorithm

- Apriory algorithm assumes that the database is memory resident
- Max no of scans is one more than cardinality of largest FI
- Weakness: large no of database scans are required

Apriori Adv/Disadv

- ***Advantages:***
  - Uses large itemset property.
  - Easily parallelized
  - Easy to implement.
- ***Disadvantages:***
  - Assumes transaction database is memory resident.
  - Requires up to m database scans.

Improving the efficiency of Apriori

- Hash based technique
- Transaction reduction
- Partitioning
- Sampling
- Dynamic itemset counting

Partitioning

- Basic partition algorithm reduces no of DB scans to two and divides the database into partitions such that each can be placed into main memory.
- It scans the database twice.
- In one scan it generates a set of all potentially large itemsets by scanning the database once. This set is a superset of all large itemsets.
- In second scan, counters for each of these itemsets are set up and their actual support is measured.

Partitioning Algo.

- The algorithm executes in two phases.
- In phase I, the Partition algorithm logically divides the database into a number of non-overlapping partitions.
- The partitions are considered one at a time and all large itemsets for that partition are generated. At the end of phase I, these large itemsets are merged to generate a set of all potential large itemsets.
- In phase II, the actual support for these itemsets are generated and the large itemsets are identified. The partition sizes are chosen such that each partition can be accommodated in the main memory so that the partitions are read only once in each phase.

1) P = partition_database($\mathcal{D}$)
2) n = Number of partitions
3) **for** $i = 1$ to n **begin** // *Phase I*
4) read_in_partition($p_i \in P$)
5) L^i = gen_large_itemsets(p_i)
6) **end**
7) **for** ($i = 2$; $L_i^j \neq \emptyset$, $j = 1, 2, \ldots, n$; i++) **do**
8) $C_i^G = \cup_{j=1,2,\ldots,n} L_i^j$ // *Merge Phase*
10) **for** $i = 1$ to n **begin** // *Phase II*
11) read_in_partition($p_i \in P$)
12) **for all** candidates $c \in C^G$ gen_count(c, p_i)
13) **end**
14) $L^G = \{c \in C^G | c.\text{count} \geq minSup\}$

Figure 1: Partition Algorithm

Partitioning

- Apriori algo assumes that the database is memory resident.
- The database for a large DB can be divided into partitions.
- It can improve performance by:
  - Large item set will be large in atleast one of the partitions.
  - Can help if we have limited memory
  - Parallel /distributed algos can be created , each partition handled by a separate machine.
  - Incremental generation of association rules partition by partitions

Dunham pg. no. 174

Partitioning

- Divide database into partitions $D^1, D^2, \ldots, D^p$
- Apply Apriori to each partition
- Any large itemset must be large in at least one partition.

Partitioning Algorithm

1. Divide D into partitions $D^1, D^2, \ldots, D^p$;
2. For I = 1 to p do
3. L^i = Apriori(D^i);
4. $C = L^1 \cup \ldots \cup L^p$;
5. Count C on D to generate L;

Partitioning Example

L^1 ={{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

D^1

D^2

L^2 ={{Bread}, {Milk}, {PeanutButter}, {Bread,Milk}, {Bread,PeanutButter}, {Milk, PeanutButter}, {Bread,Milk,PeanutButter}, {Beer}, {Beer,Bread}, {Beer,Milk}}

S=10%

If the data are not uniform there may be a large percentage of false candidates

Partitioning Adv/Disadv

- ***Advantages:***
  - Adapts to available main memory
  - Easily parallelized
  - Maximum number of database scans is two.
- ***Disadvantages:***
  - May have many candidates during second scan.

Comparision

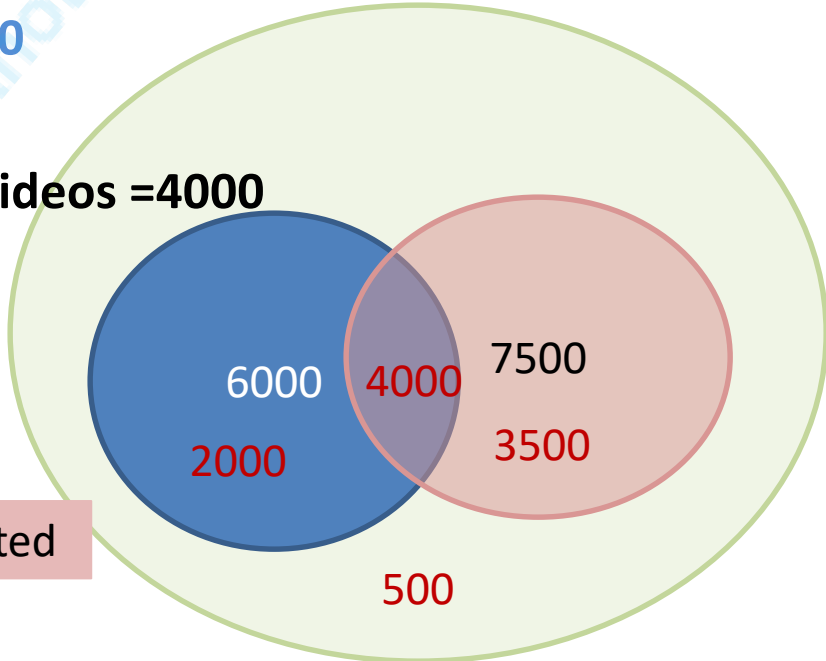
- The Partition algorithm was motivated by the need to reduce disk I/O. In this aspect it has a clear advantage over the Apriori algorithm. The Partition algorithm reads the database at most twice irrespective of (a) the minimum support and (b) the number of partitions.
- Apriori reads the database multiple number of times. The exact number depends on the minimum support and the data characteristics and cannot be determined in advance.

- Min supp and Min Conf threshold help weed out a no. of uninteresting rules.
- But even strong association rules can be uninteresting and misleading
- *How can be tell which strong association rules are really interesting?*
- Eg. *Allelectronics* has discovered rule **computer games → videos [minsupp30%,minconf 60%]**
- **Total transactions =10,000**
- **Transaction having computer games=6000**
- **Transaction having videos =7500**
- **Transactions having computer games & videos =4000**

computer games → videos [supp40%,conf 66%]

Supp= 4000/10000=40%

Conf= 4000/6000 =66%



Infact computer games and videos are –vely correlated

Correlation rules

- Defined the set of itemsets that are correlated
- Negative correlation may be useful
- If a set is correlated so is the superset
- ***Lift*** is a simple correlation measure.
- χ^2 is second correlation measure

Lift

- Bread => butter [13.33%, 45 %]
- Support is 13.33%
- Confidence is 45 %

Confidence {Bread→ butter}=
$P(\text{butter}|\text{bread}) = \frac{P(\text{butter and Bread})}{P(\text{Bread})}$

Lift {Bread→ butter}=
$= \frac{P(\text{butter and Bread})}{P(\text{Bread})P(\text{butter})}$

- Total transactions =150
- Bread : in 20 transactions
- Support {Bread}= [20/150]*100=13.33%
- In 20 transaction butter is in 9 transactio
- Confidence {Bread→ butter}= [9/20] *100
  =[supp(bread,butter)/supp(bread)]*100
  =45%
- **Lift {x→ y} = confidence{x→ y}/supp(x)**
  - **Lift =1 → x has no impact on y i.e. x and y are independent**
  - **Lift>1 , +ve correlation, y is likely to be bought with x**
  - **Lift < 1,-ve correlation , y is likely not to be bought with x**
  - **Larger the lift ratio , more significant association**

- Eg. *Allelectronics* has discovered rule **computer games → videos**
- **[minsupp30%,minconf 60%]**
- **Total transactions =10,000**
- **Transaction having computer games=6000**
- **Transaction having videos =7500**
- **Transactions having computer games & videos =4000**

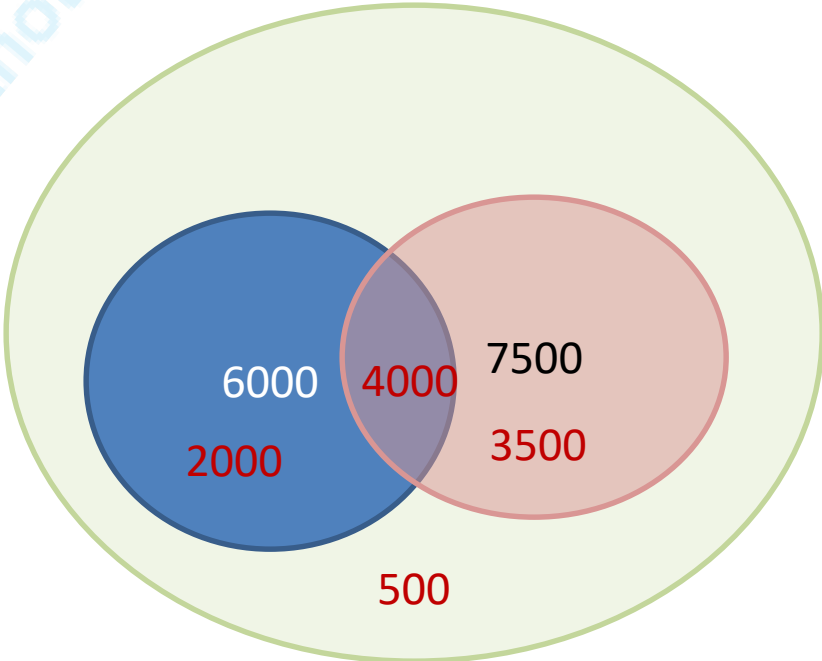
Since **lift** , 0.89 is less than 1 there is a negative correlation
This could not have been identified by **supp** and **Conf** alone

computer games → videos [supp40%,conf 66%]

Supp= 4000/10000=40%

Conf= 4000/6000 =66%

	games	No game	
Video	4000	3500	7500
No video	2000	500	2500
	6000	4000	10000



Lift { **games → videos** }=

$$= \frac{P(\text{games and videos})}{P(\text{games})P(\text{videos})}$$

$$= \frac{4000/10000}{(6000/10000)(7500/10000)} = 0.89$$

Lift =1 → x has no impact on y i.e. x and y are independent
Lift>1 , +ve correlation, y is likely to be bought with x
Lift < 1,-ve correlation , y is likely not to be bought with x
Larger the lift ratio , more significant association

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

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

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

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:

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

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

Lift

	B	B’	Total
A	15	10	25
A’	55	20	75
total	70	30	100

Lift { **A** → **B** }=

$$= \frac{P(A \text{ and } B)}{P(A)P(B)}$$

$$= \frac{15/100}{(25/100) * (70/100)} = 0.85$$

Since lift is less than 1 there is a negative correlation.

CHI - SQUARE AS PROOF OF ASSOCIATION

- Chi squared analysis is useful in determining the statistical significance level of association rules.
  - facilitate pruning of rule sets obtained using standard association rule mining techniques
  - allow identification of statistically significant rules that may have been overlooked by the mining algorithm
  - provide an analytical description of the relationship between confidence and support in terms of chi squared and lift.

CHI - SQUARE AS PROOF OF ASSOCIATION

- Despite the abundance of alternative evaluation measures, chi-squared analysis, a classical technique in statistics for determining the closeness of two probability distributions, continues to be one of the most widely used tools for statistical significance testing in many scientific circles, e.g. bioinformatics.
- It was suggested that chi-squared analysis be used to assess the statistical significance level of the dependence between antecedent and consequent in association rules.
- chi-square statistic does not by itself measure the strength of the dependence of the antecedent and consequent of a rule
- The statistical significance of an association rule may be gauged through chisquare analysis.

χ^2

Since χ^2 is greater than 1 there is a negative correlation , consistent with **lift** measure also.

- Squared difference between observed a[illegible] expected values.
- It used to measure how much a itemset (in rule) count differs from the expected
- If all values are independent chi sq value should be 0
- If >1, -ve correlation
- If <1, +ve correaltion
- $\chi^2 = \Sigma \left[\frac{(o - e)^2}{e} \right]$

Expected value less than observed

	[illegible]mes	No game	
Video	4000	3500	7500
No vid[illegible]	2000	500	2500
	6000	4000	10000

	Games (expected)	No game (expected)	
Video	(6000*7500)/10000 =**4500**	(4000*7500)/10000= **3000**	7500
No video	**6000**-4500= **1500**	**4000**-3000= **1000**	2500
	6000	**4000**	10000

$$= \frac{(4000 - 4500)^2}{4500} + \frac{(3500 - 3000)^2}{3000} + \frac{(2000 - 1500)^2}{1500} + \frac{(500 - 1000)^2}{1000}$$

$$= 555.6$$

χ^2

- Squared difference between observed and expected values.
- It used to measure how much a itemset (in rule) count differs from the expected
- If all values are independent chi sq value should be **0**.
- If >1, -ve correlation
- If <1, +ve correaltion
- $\chi^2 = \Sigma \left[\frac{(o - e)^2}{e} \right]$

	Games	No game	
Video	4000	3500	7500
No video	2000	500	2500
	6000	4000	10000

	Games (expected)	No game (expected)	
Video	(6000*7500)/10000 =**4500**	(4000*7500)/10000= **3000**	7500
No video	**6000**-4500= **1500**	**4000**-3000= **1000**	2500
	6000	**4000**	10000

$$= \frac{(4000 - 4500)^2}{4500} + \frac{(3500 - 3000)^2}{3000} + \frac{(2000 - 1500)^2}{1500} + \frac{(500 - 1000)^2}{1000}$$

$$= 555.6$$

χ^2

Since χ^2 is greater than 1 there is a negative correlation.

	B	B’	Total
A	15	10	25
A’	55	20	75
total	70	30	100

	B (expected)	B’ (expected)	
A	(70*25)/100 =**17.5**	(30*25)/100= **7.5**	25
A’	**70**-17.5= **52.5**	**30**-7.5= **22.5**	75
	70	**30**	100

$$= \frac{(15 - 17.5)^2}{17.5} + \frac{(10 - 7.5)^2}{7.5} + \frac{(55 - 52.5)^2}{52.5} + \frac{(20 - 22.5)^2}{22.5}$$

$$= 1.587$$

Incremental Mining of Association Rules

Underlying Problem

- A, B, C $\Rightarrow$ D, E (*support, confidence)*
- where

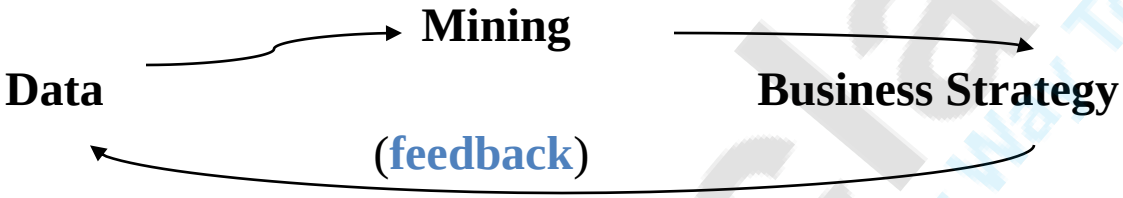
$$support = \frac{\text{count}(\{A,B,C,D,E\})}{database\_size}$$

$$confidence = \frac{\text{count}(\{A,B,C,D,E\})}{\text{count}(\{A,B,C\})}$$

- Find all **rules** which satisfy *minsupp*
- Find all **itemsets** which satisfy *minsupp*

Incremental Mining

- Mining is not a one-time operation



 Why not use previous mining results?

- Issues
  - Insertions, Deletions
  - Skew of Increment
  - Change in *minsupp*

Maintenance of Association Rules

- two approaches
  - non-incremental approach.
    - executing again an ARM algorithm on the whole updated data.
    - The advantage of this method is the free choice of a new support threshold.
    - Its drawback is ignoring old association rules making the operation more expensive.
  - incremental approach
    - uses old association rules to compute new association rules.
    - It's faster in comparison with non-incremental maintenance.
    - its drawback is that maintenance is realized on the same initial extraction support threshold.

Dunham pg. no. 182

Fast UPdate (FUP)

- An itemset that wasn't large before, can now become large only if it is large within the increment.
- we already know L, large itemsets in the old DB
- Itemset must be large in either partitions
- we need not count any item in new updated data that had infrequent subsets in the old dB

Educlash
Just Another Way To Learn