Association Rules Mining

Association Rules

- Association rules specify interesting associations or correlations.
- Most common application is in grocery/retail stores, where these rules are used to place items on shelves, target coupons to customers, and do cross selling.
- The process of discovering association rules is also known as *market basket analysis*.
- Another name for finding associations is *affinity analysis*.

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

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

An Example of Association Rule

{Diaper} → {Beer}

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

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

Definition: Association Rule

- **Association Rule**
  - An implication expression of the form X → Y, where X and Y are itemsets
  - Example:
    {Milk, Diaper} → {Beer}

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

- **Rule Evaluation Metrics**
  - Support (s)
    - Fraction of transactions that contain both X and Y
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Mining Association Rules

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

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

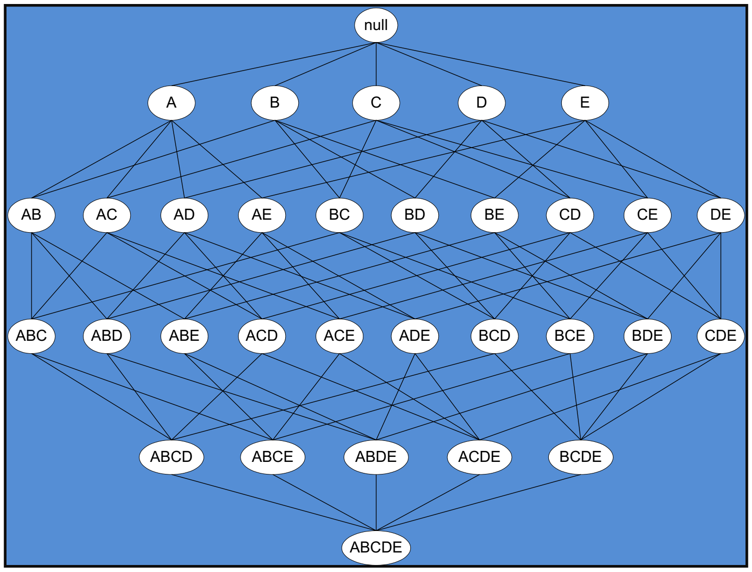
Observations:

- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support ≥ minsup
  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

Frequent Itemset Generation



Given d items, there are 2^d 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

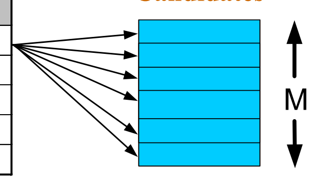
Transactions

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

N

w

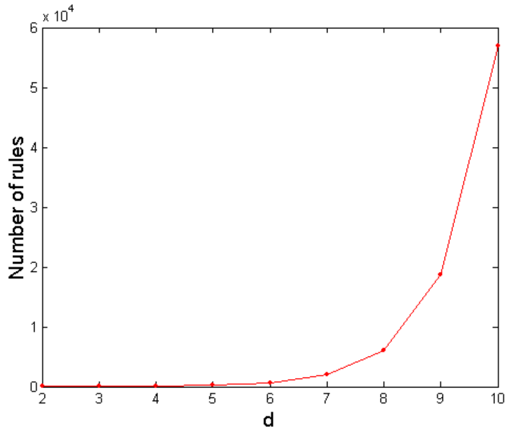
List of Candidates



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

Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right] = 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

Reducing Number of Candidates

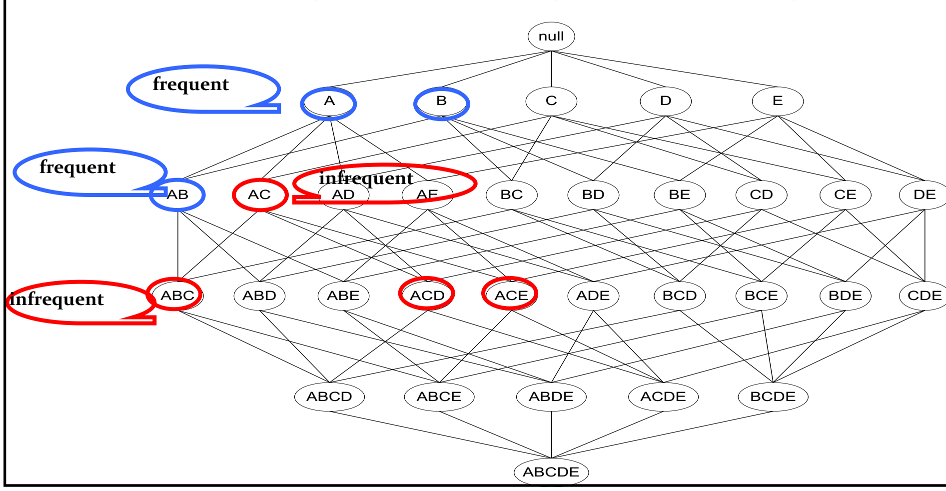
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X,Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

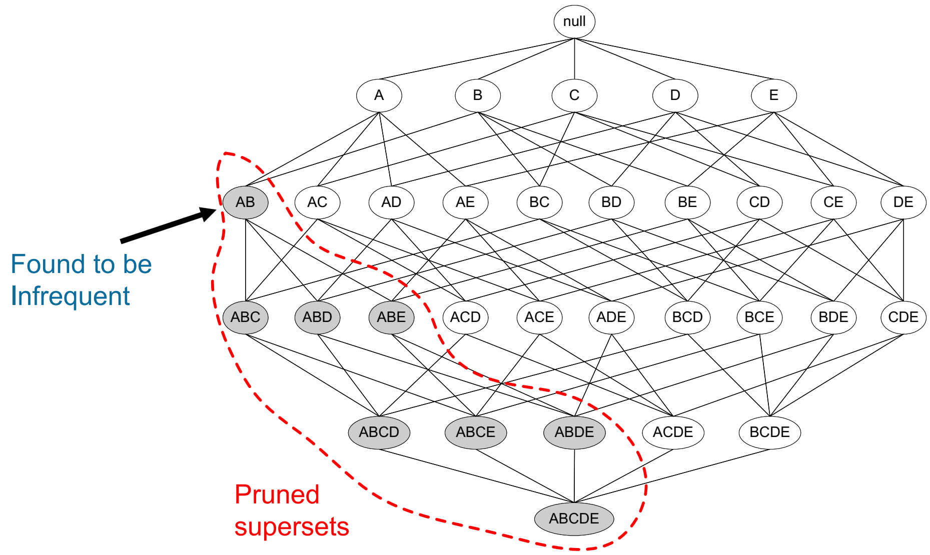
  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Apriori Principle

- If an itemset is frequent, then all of its subsets must also be frequent
- If an itemset is infrequent, then all of its supersets must be infrequent too



Illustrating Apriori Principle



Illustrating Apriori Principle

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

Items (1-itemsets)

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)

Minimum Support = 3

Triplets (3-itemsets)

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

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3 = 41$
With support-based pruning,
$6 + 6 + 1 = 13$

Apriori Algorithm

- Method:
  - Let k=1
  - Generate frequent itemsets of length 1
  - Repeat until no new frequent itemsets are identified
    - Generate length (k+1) candidate itemsets from length k frequent itemsets by join operation
    - Prune candidate itemsets containing subsets of length k that are infrequent
    - Count the support of each candidate by scanning the DB
    - Eliminate candidates that are infrequent, leaving only those that are frequent

Example

Transaction database

TID	Items
t1	I1, I2, I5
t2	I2, I4
t3	I2, I3
t4	I1, I2, I4
t5	I1, I3
t6	I2, I3
t7	I1, I3
t8	I1, I2, I3, I5
t9	I1, I2, I3

Step 1: Scan the transaction database for each item count. This will generate set *C1*, the list of candidate 1-itemsets.

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

Step 2: The frequent 1-itemsets from *C1* can be determined by comparing the count values against the required minimum support. Let us use minimum support count of 2. Then, the set of frequent 1-itemsets, *L1*, is shown below.

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

Step 3: The join operation is performed on *L1* to obtain *C2*, the set of candidate 2-itemsets. Since all 1-itemsets are frequent, pruning is not helpful here.

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

Step 4: The support for each 2-itemset is determined by scanning the database. The set of frequent 2-itemsets, *L2*, is found by comparing against the required minimum support.

2-Itemset	Count	Include in L_2
{I1, I2}	4	√
{I1, I3}	4	√
{I1, I4}	1	x
{I1, I5}	2	√
{I2, I3}	4	√
{I2, I4}	2	√
{I2, I5}	2	√
{I3, I4}	0	x
{I3, I5}	1	x
{I4, I5}	0	x

Step 5: The procedure is repeated for 3-itemsets. Performing join on *L2* results in *C3* = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}. Now we can do pruning using the apriori property because there are 2-itemsets that are not frequent. Thus, pruned *C3* = {{I1, I2, I3}, {I1, I2, I5}}. Checking the support for these 3-itemsets, we find that both of the 3-itemsets have the requisite support. Thus, *L3* = {{I1, I2, I3}, {I1, I2, I5}}.

Step 6: Continuing for 4-itemsets, we get after join only one 4-itemset, {I1, I2, I3, I5}. The application of pruning eliminates it because its subset {I2, I3, I5} is not frequent. Thus, pruned *C4* = Æ. The algorithm terminates at this point.

Step 7: Once the frequent itemsets are obtained, the association rules are generated in a straightforward manner through the following formula:
confidence (X => Y) = Pr(Y/X) = count (X U *Y)/ count (X)*

The frequent itemsets for the running example are: {I1, I2, I3} and {I1, I2, I5}. Examples of some of the association rules that can be derived are:

I1 & I2 => I5	confidence = 2/4 = 50%
I1 & I2 => I3	confidence = 2/4 = 50%
I1 & I3 => I2	confidence = 2/4 = 50%
I2 & I5 => I1	confidence = 2/2 = 100%
I1 => I2 & I5	confidence = 2/6 = 33%

Only those rules that have confidence greater than minimum needed confidence are kept; the rest are discarded.

Factors Affecting Apriori Algorithm

- Choice of minimum support threshold
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets
- Dimensionality (number of items) of the data set
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase
- Size of database
  - since Apriori makes multiple passes, run time of algorithm may increase with number of transactions
- Average transaction width
  - transaction width increases with denser data sets
  - This may increase max length of frequent itemsets

Rule Generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

    $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$, $BCD \rightarrow A$,
    $A \rightarrow BCD$, $B \rightarrow ACD$, $C \rightarrow ABD$, $D \rightarrow ABC$
    $AB \rightarrow CD$, $AC \rightarrow BD$, $AD \rightarrow BC$, $BC \rightarrow AD$,
    $BD \rightarrow AC$, $CD \rightarrow AB$,

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

Rule Generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

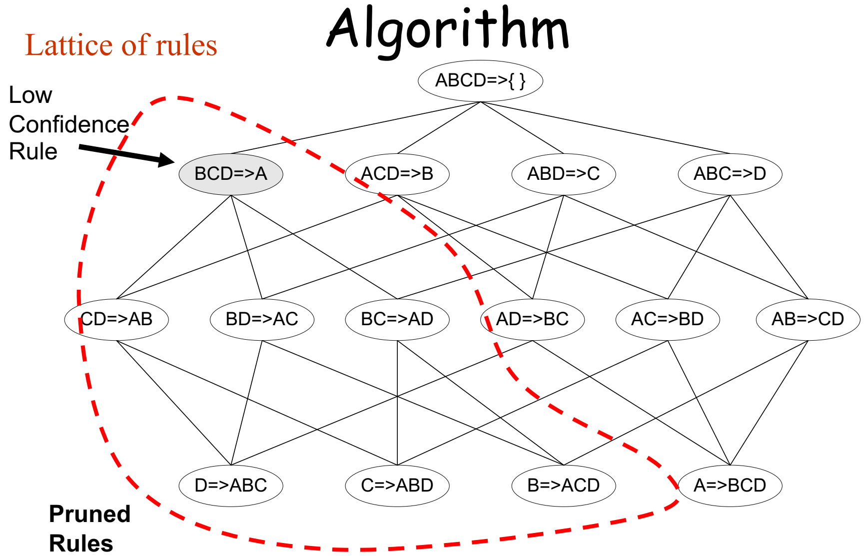
    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., L = {A,B,C,D}:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Rule Generation for Apriori Algorithm



Rule Generation for Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix
  in the rule consequent

- join(CD=>AB,BD=>AC)
  would produce the candidate
  rule D => ABC

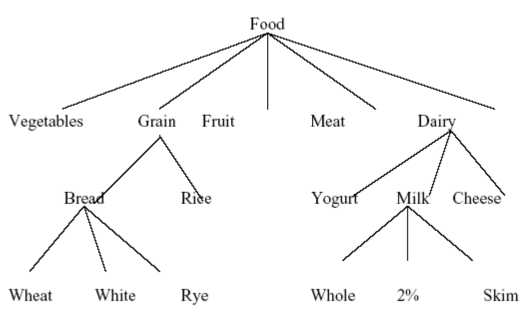
- Prune rule D=>ABC if its
  subset AD=>BC does not have
  high confidence

Effect of Support Distribution

- How to set the appropriate *minsup* threshold?
  - If *minsup* is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
  - If *minsup* is set too low, it is computationally expensive and the number of itemsets is very large
- Using a single minimum support threshold may not be effective

Multiple-Level Association Rules

These rules utilize different levels of abstractions to represent items. An example of such an abstraction is shown below in the form of a concept hierarchy. The concept hierarchy allows association rules for itemsets at any level of hierarchy. The hierarchy is traversed top-down and large itemsets generated at level *i* are used for generating large itemsets at level *i+1*. The idea of hierarchy is really useful for applications in grocery stores where an item, for example Coke or Pepsi, has several sizes and tastes, each with its own UPC.



Quantitative Association Rules

- The association rules discussed thus far are categorical rules. The items have no attributes attached to them.
- Rules for items with attributes are known as quantitative association rules. For example, we may want to find the items that customers purchase when buying expensive wines to generate a rule of the following type:

  Buys wine costing more than $50 => Buys caviar

Summary

- There is another rule mining algorithm FP-Growth
- Related to Recommendation Systems