Generating Association Rules

Association Rules Mining General Concepts

This is an example of **Unsupervised** Data Mining-- You are not trying to predict a variable.

All previous classification algorithms are considered Supervised techniques.

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.

Nominal attributes are required.

Affinity Analysis is the process of determining which things go together. This is also called **market basket analysis.**

For example, we may have the following products: Milk, Cheese, Bread, Eggs

Possible **associations** include:

1. if customers purchase milk they also purchase bread **{milk} → {bread}**
2. if customers purchase bread they also purchase milk **{bread}→ {milk}**
3. if customers purchase milk and eggs they also purchase cheese and bread **{milk, eggs} → { cheese, bread}**
4. if customers purchase milk, cheese, and eggs they also purchase bread **{milk, cheese, eggs} → {bread}**

Based on a set of transactions of customers

Note that #1 and #2 are not the same as is demonstrated in the confidence rating of each rule described below.

Implication means co-occurrence, not causality!

Definition: Frequent Itemset

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

Itemset:

A collection of 1 or more items

- {bread, milk, diaper}

Support Count:

Support count, σ, is the frequency count of occurences of the itemset

- σ({bread,milk,diaper}) = 2

Support

(similar to the idea of coverage with decision rules)

Support is the percentage of instances in the database that contain all items listed in an itemset

- For the bread AND milk cases #1 and #2 we might have σ(bread and milk) = 5000 out of 50000 instances for s=10% support or in the case of the tiny 5 items dataset, we would have σ=3 out of 5 instances for
- s=60%.

Association Rule

An association rule is an implication expression of the form **X → Y**, where X and Y are itemsets .

Example: {Milk, Diaper} → {Beer}

Confidence

(similar to the idea of accuracy with decision rules)

Each rule has an associated confidence: the conditional probability of the association.

E.g., the probability that purchasing a set of items they then purchase another set of items, so if there were 10000 recorded transactions purchasing milk, and of those 5000 purchase bread, we have 50% confidence for rule #1.

For rule #2, we might have 15000 purchasing bread, of which 5000 purchased milk, then it is 33% confidence.
In the 4 itemset example

Example:

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

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

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

Item Sets

Item sets are attribute-value combinations that meet a specified coverage requirement (minimum support). Item sets that do not make the cut are discarded.

We can also talk about minimum confidence.

Association Rules Mining Approach

Given a set of transactions, T, the goal of association rule mining is to find all rules having

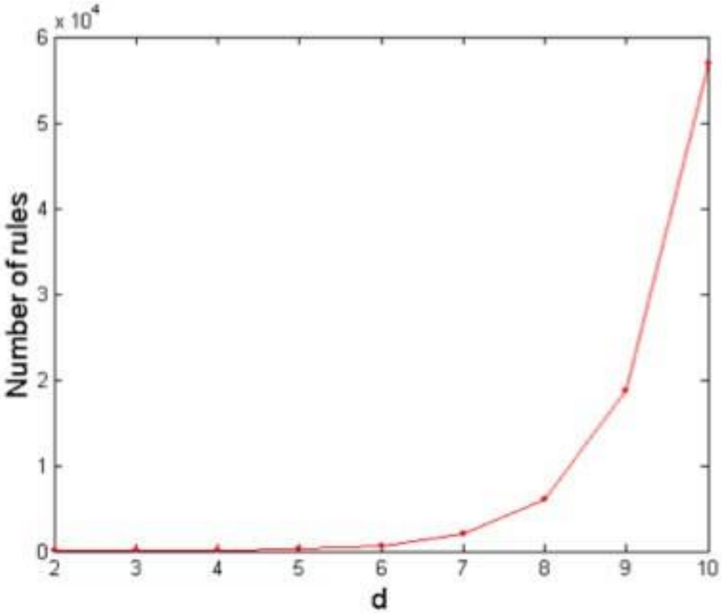
- **support ≥ minSup threshold**
- **confidence ≥ minConf threshold**

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! (exponential $O(3^n)$)

Below is a graph showing the total number of rules to consider for d unique items.



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

If d=6, R = 602 rules

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)

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

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

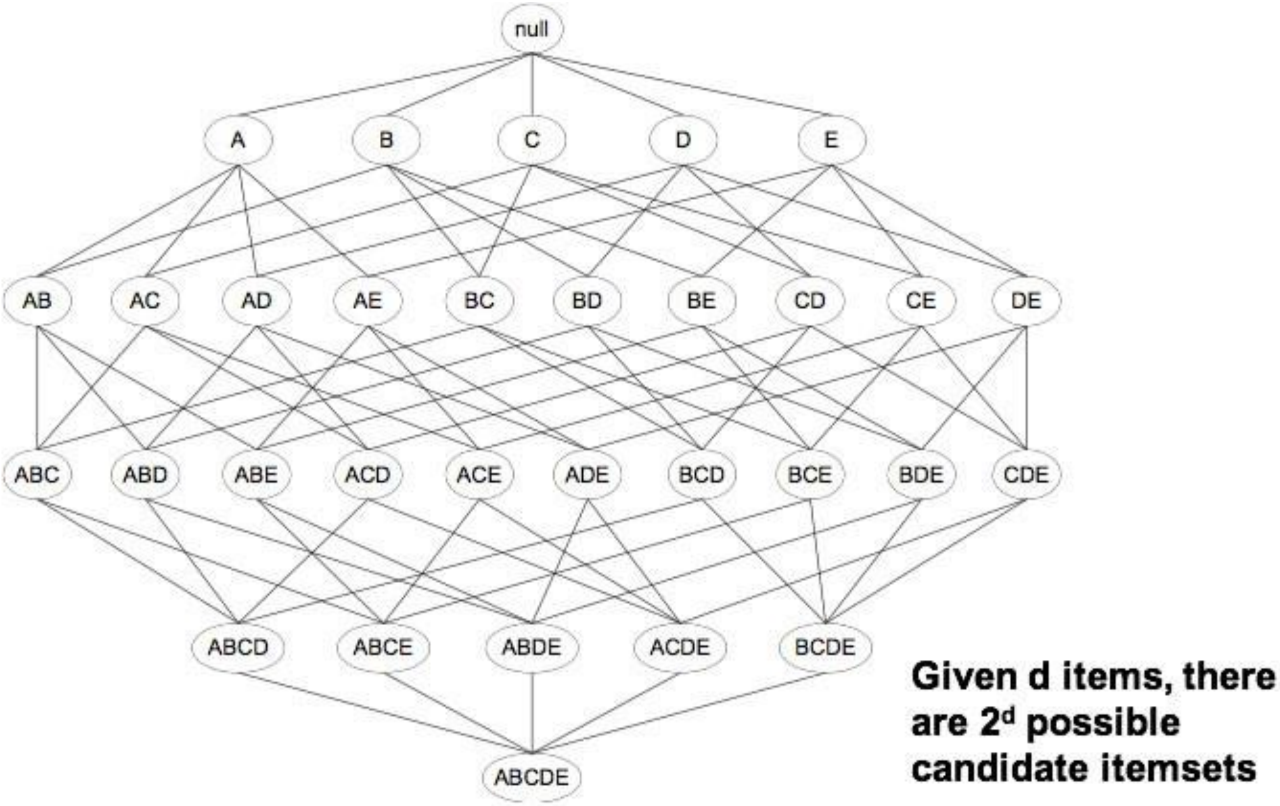
Thus, we may decouple the support and confidence requirements

Mining the Association Rules

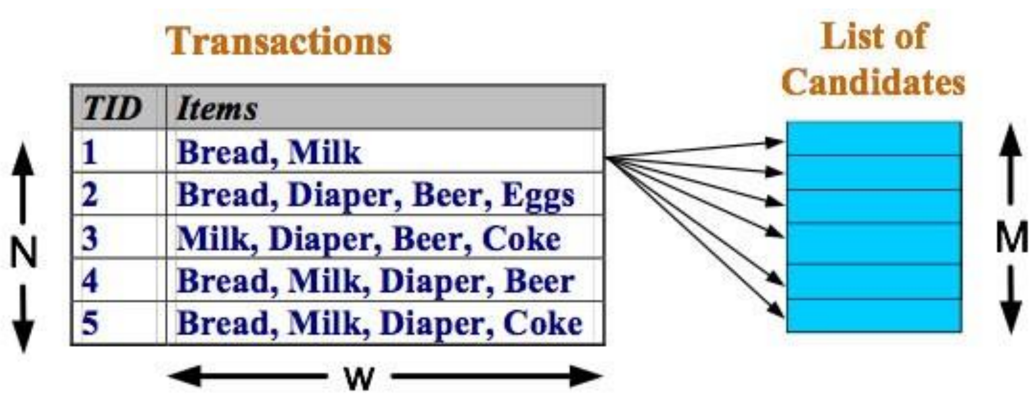
Two-step approach:

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



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 is exponential ~ O(NMw), which is expensive since M = 2^d !!!

Strategies

Reduce the number of candidates (M)Complete search: M=2^d

Use pruning techniques to reduce M (use Apriori principle, below)

Reduce the number of transactions (N)

- Reduce size of N as the size of itemset increases
- Used by DHP and vertical-based mining algorithms

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

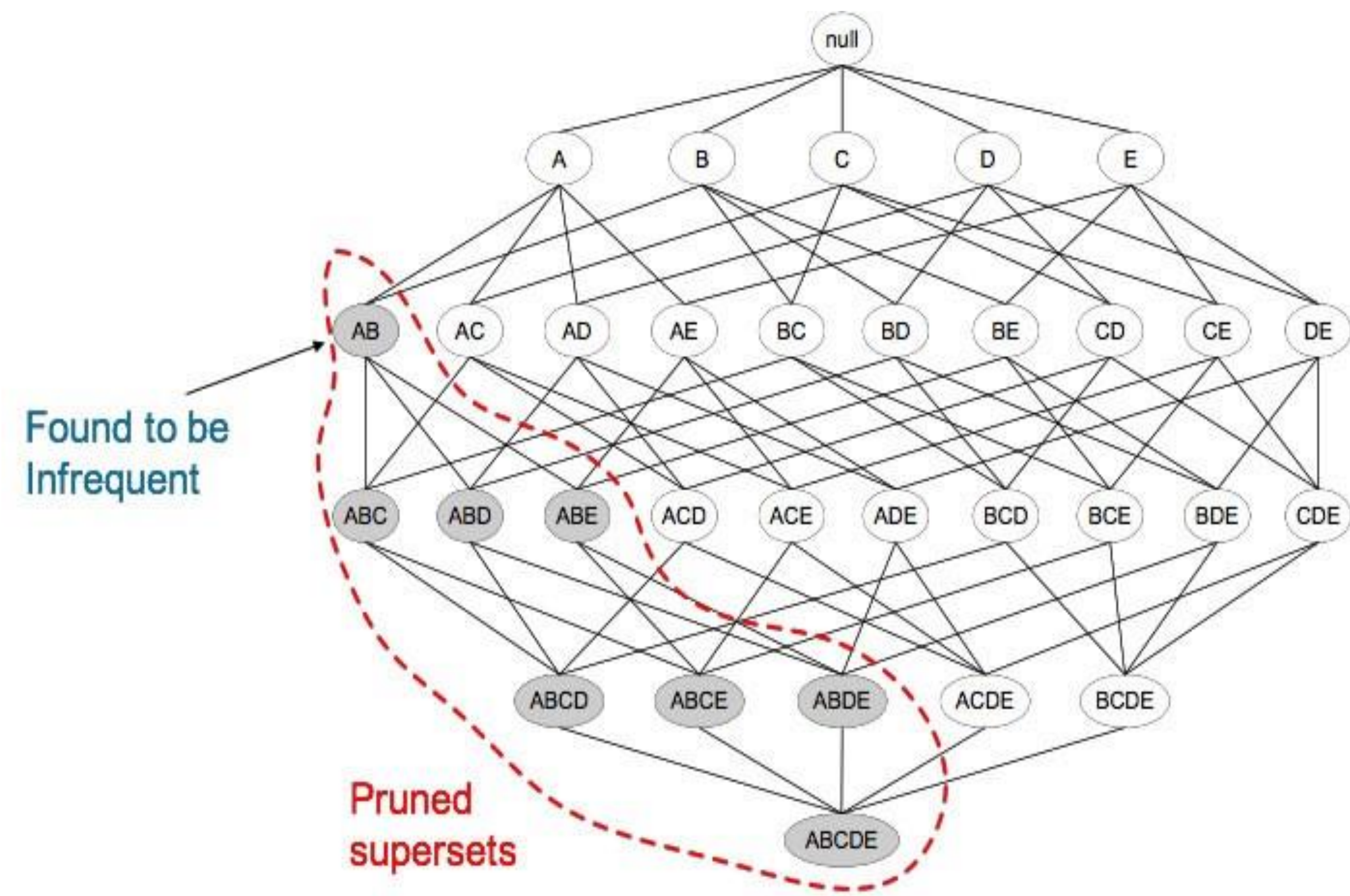
Apriori principle:

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:

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

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



Example step through

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

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

Items (1-itemsets)

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

Minimum Support = 3

If every subset is considered,
$^6C_1 + {}^6C_2 + {}^6C_3$
$6 + 15 + 20 = 41$
With support-based pruning,
$6 + 6 + 4 = 16$

The formal Apriori algorithm

F_k: frequent k-itemsets

L_k: candidate k-itemsets

Algorithm

- Let k=1
- Generate F_1 = {frequent 1-itemsets} Repeat
- until F_k is empty:
  - Candidate Generation: Generate L_{k+1} from F_k
  - Candidate Pruning: Prune candidate itemsets in L_{k+1} containing subsets of length k that are infrequent
  - Support Counting: Count the support of each candidate in L_{k+1} by scanning the DB
  - Candidate Elimination: Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

Informally, the algorithm is

Finding one-item sets easy
Use one-item sets to generate two-item sets, two-item sets to generate three-item sets, …

- Keep only those item sets that meet the support threshold at each level to prune those at higher levels.
- Then partition the retained item sets into rules and keep only those that meet the confidence threshold.

Example 2: credit card promotion database

This example considers a dataset of nominal values, although binary, both of which can be considered interesting. Unlike marketbasket where only purchases are interesting.

Single itemsets now can be twice as large than above.

Magazine Promo	Watch Promo	Life Ins Promo	Credit Card Ins.	Sex
Yes	No	No	No	Male
Yes	Yes	Yes	No	Female
No	No	No	No	Male
Yes	Yes	Yes	Yes	Male
Yes	No	Yes	No	Female
No	No	No	No	Female
Yes	No	Yes	Yes	Male
No	Yes	No	No	Male
Yes	No	No	No	Male
Yes	Yes	Yes	No	Female

Single item sets at a 40% coverage threshold:

single item sets	Number of items
A. Magazine Promo=Yes	7
B. Watch Promo=Yes	4
C. Watch Promo=No	6
D. Life Ins Promo=Yes	5
E. Life Ins Promo=No	5
F. Credit Card Ins=No	8
G. Sex=Male	6
H. Sex=Female	4

Pairing--Step 2

Now begin pairing up combinations with the same coverage threshold (again 40% here)

single item sets	Number of items
A. Magazine Promo=Yes	7
B. Watch Promo=Yes	4
C. Watch Promo=No	6
D. Life Ins Promo=Yes	5
E. Life Ins Promo=No	5
F. Credit Card Ins=No	8
G. Sex=Male	6
H .Sex=Female	4

	A	B	C	D	E	F	G	H
B	3	-						
C	4		-					
D	5			-				
E	2		4		-			
F	5		5		5	-		
G	4		4		4	4	-	
H						4		-

Resulting rules from two item sets. Consider rules in both directions:

1. (A → D)
   (MagazinePromo=Yes)→ (LifeInsPromo=Yes) at 5/7 confidence
2. (D → A)
   (LifeInsPromo=Yes) → (MagazinePromo=Yes) at 5/5 confidence
3. twenty more rules from the 10 two-item-sets (A then C, C then A, A then F, F then A, etc.)

Now apply minimum confidence threshold

If confidence threshold would be 80%, then the first rule (A → D) is eliminated.

Repeat process for 3 item set rules, then 4 item set rules, etc., but keep the support and confidence thresholds the same.

Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

Example F_3 = {ABC,ABD,ABE,ACD,BCD,BDE}

- Lexicographically ordered!

Candidate four-item sets are:

- Merge(**AB**C, **AB**D) = **AB**CD
- Merge(**AB**C, **AB**E) = **AB**CE
- Merge(**AB**D, **AB**E) = **AB**DE

Do not merge(**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2 (A C

D E) Not candidate because of no (C D E)

L_4= {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated from first method

Candidate pruning

- Prune ABCE because ACE and BCE are infrequent
- Prune ABDE because ADE is infrequent

After candidate pruning: F_4 = {ABCD}

Alternate $F_{k-1} \times F_{k-1}$ Method

Merge two frequent (k-1)-itemsets if the last (k-2) items of the first one is identical to the first (k-2) items of the second.

F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}

- Merge(A**BC, BC**D) = A**BC**D

  Merge(A**BD, BD**E) = A**BD**E
  Merge(A**CD, CD**E) = A**CD**E
- Merge(B**CD, CD**E) = B**CD**E

L_4= {ABCD,ABDE,ACDE,BCDE} is the set of candidate 4-itemsets generated from second method pruning

results in F_4 = {ABCD} why are others eliminated?

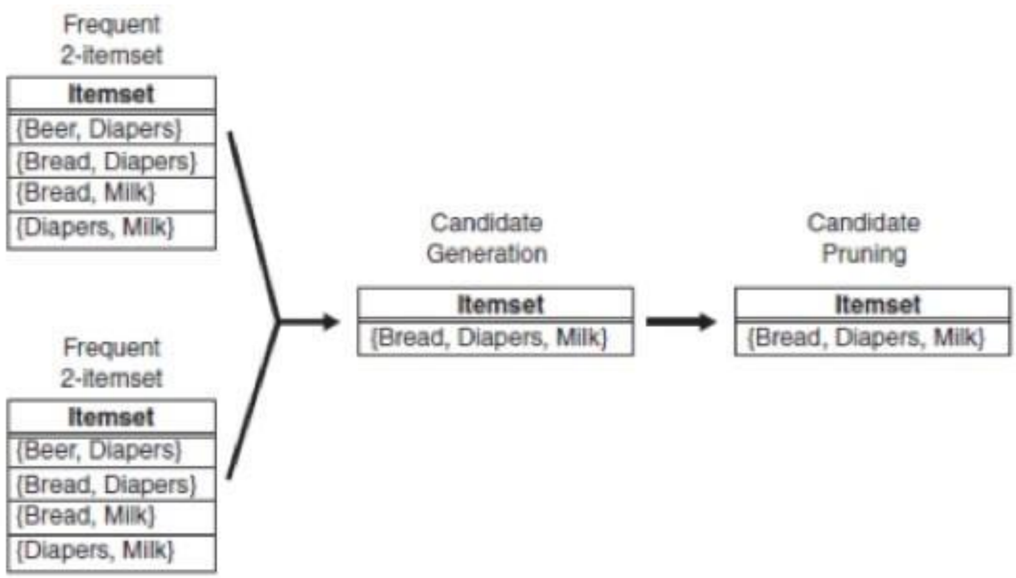


Figure 6.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

Rule generation

A three item set will be partitioned to generate 6 rules:

- An item set (A B C) generates rules
- (A & B) → C,
- (A & C) → B,
- (B & C) → A,
- A → (B & C),
- B → (A & C),

C → (A & B)

Example 4 item set L = (A B C D), the partitioning result in the following rules

- ABC → D, ABD→ C, ACD → B, BCD → A, A → BCD, B → ACD, C → ABD, D → ABC
  AB → CD, AC → BD, AD→ BC, BC → AD, BD → AC, CD → AB

  If |L| = k, then there are 2^k – 2 candidate association rules (We are ignoring L → True and True → L. Weka will include the latter!)

In general, confidence does not have an anti-monotone property.

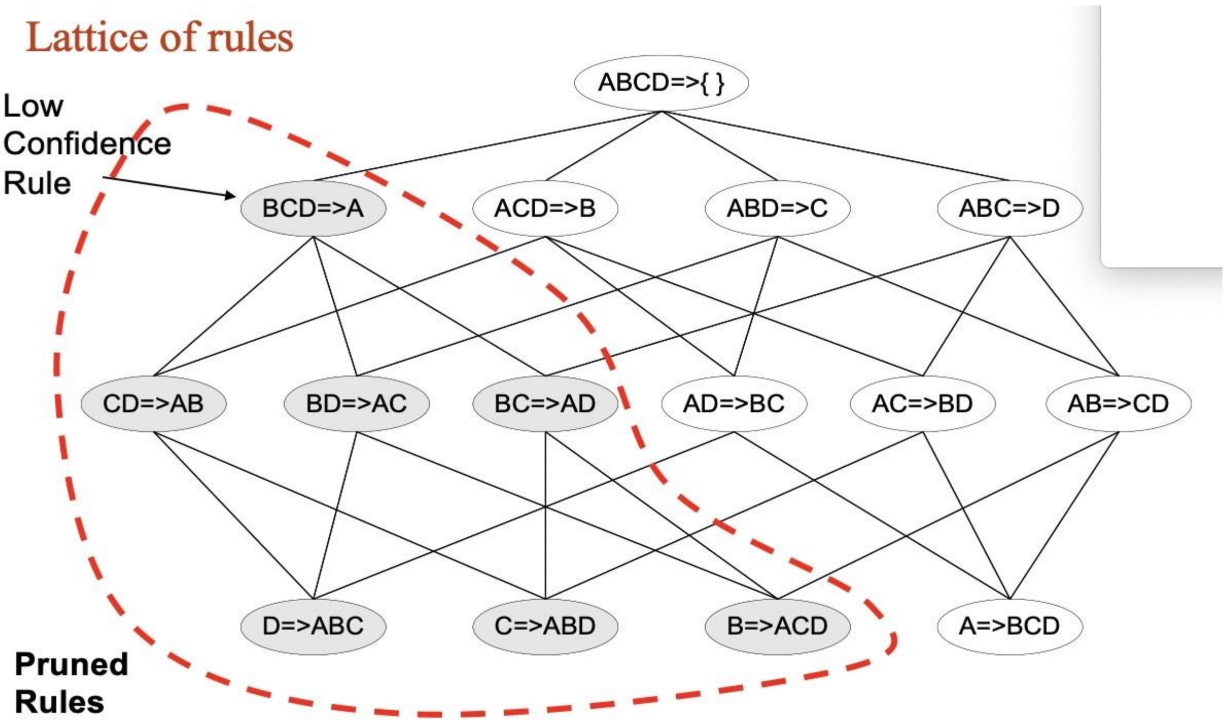
- i.e., conf(ABC →D) can be larger or smaller than conf(AB →D)

But confidence of rules generated from the same itemset has an anti-monotone property • E.g.,

Suppose {A,B,C,D} is a frequent 4-itemset:

conf(ABC → D) ≥ conf(AB →CD) ≥ conf(A → BCD)

Confidence is anti-monotone w.r.t. number of items on the RHS of the rule. conf = σ(itemset) / σ(lhs)



Weather example

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

One-item sets	Two-item sets	Three-item sets	Four-item sets
Outlook = Sunny (5)	Outlook = Sunny Temperature = Hot (2)	Outlook = Sunny Temperature = Hot Humidity = High (2)	Outlook = Sunny Temperature = Hot Humidity = High Play = No (2)
Temperature = Cool (4)	Outlook = Sunny Humidity = High (3)	Outlook = Sunny Humidity = High Windy = False (2)	Outlook = Rainy Temperature = Mild Windy = False Play = Yes (2)
...	...	...	...

In total: (with minimum support of two)

- 12 one-item sets,
- 47 two-item sets,
- 39 three-item sets,
- 6 four-item sets
- 0 five-item sets

Once all item sets with minimum support have been generated, we can turn them into rules
Example:

- Humidity = Normal, Windy = False, Play = Yes (4)

Seven (2^N-1) potential rules (6 useful ones)

If Humidity = Normal and Windy = False → Play = Yes (4/4)
If Humidity = Normal and Play = Yes → Windy = False (4/6)
If Windy = False and Play = Yes → Humidity = Normal (4/6)
If Humidity = Normal → Windy = False and Play = Yes (4/7)
If Windy = False → Humidity = Normal and Play = Yes (4/8)
If Play = Yes → Humidity = Normal and Windy = False (4/9)
?? If True → Humidity = Normal and Windy = False and Play = Yes (4/12)

Factors Affecting Complexity of Apriori

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 and traversals of hash tree (number of subsets in a transaction increases with its width)

Support Counting of Candidate Itemsets

Scan the database of transactions to determine the support of each candidate itemset

Must match every candidate itemset against every transaction--- expensive operation

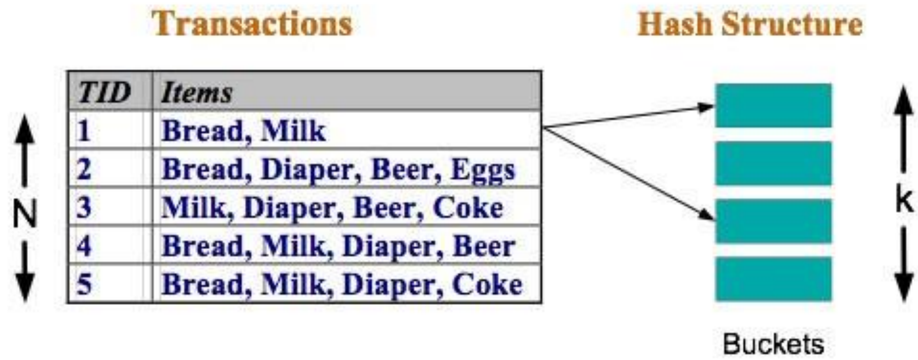
The highlighted itemset support? Search all transactions....

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

Itemset
{ Beer, Diaper, Milk}
{ Beer,Bread,Diaper}
{Bread, Diaper, Milk}
{ Beer, Bread, Milk}

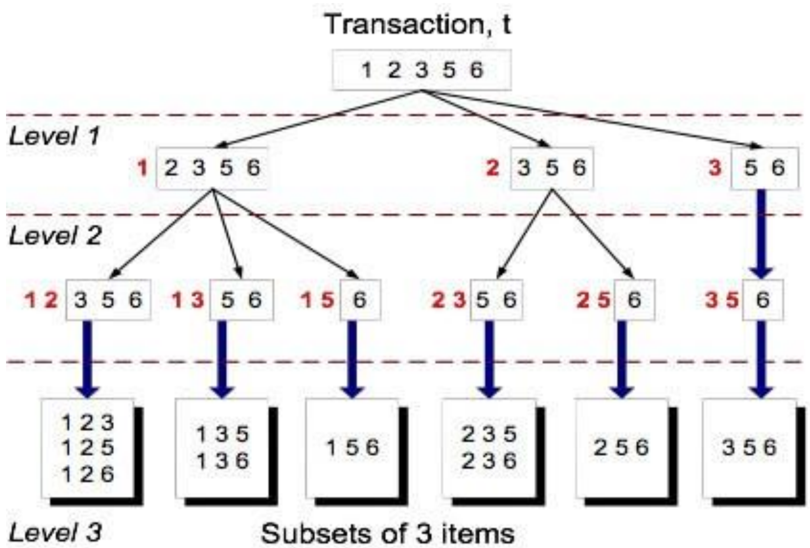
To reduce number of comparisons, store the candidate itemsets in a hash structure

Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

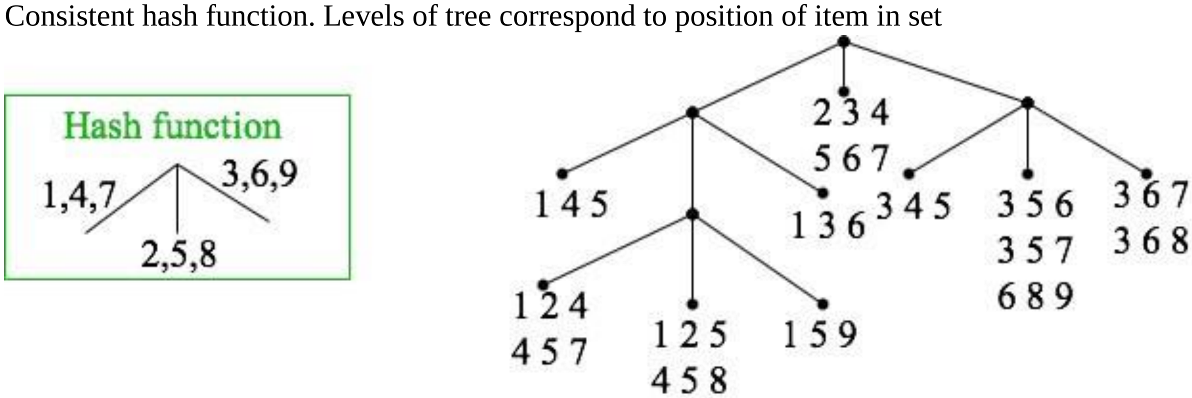


Suppose you have 15 candidate itemsets of length 3:
{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8}, {1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5}, {3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

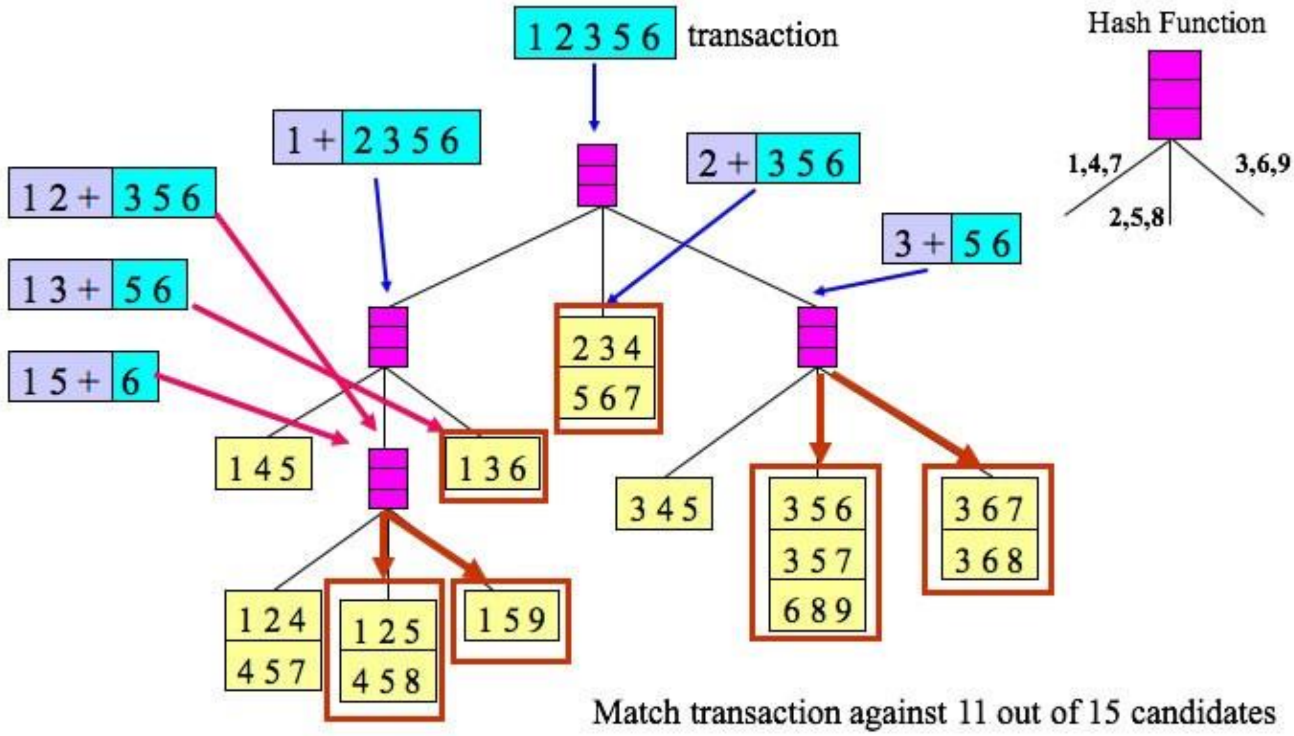
How many of these itemsets are supported by transaction (1,2,3,5,6)?



Consistent hash function. Levels of tree correspond to position of item in set



Matching transaction (1 2 3 5 6) leads to the buckets that contain the item sets to which counts can be incremented.



General Considerations

Association rules do not require identification of dependent variables first. This is a good example of information discovery.

Not all rules may be useful. We may have a rule that exceeds our confidence level, but the item sets are also high in probability so not much new information is revealed. The lift is low.

> If customers purchase milk, they also purchase bread (conf. level of 50%) but if 70% of all purchases involves milk and 50% of purchases include bread, the rule is of little use.

Two types of relationships of interest:

1. association rules that show a lift in product sales for a particular product where the lift in sales is the result of is association with one or more other products--may conclude that marketing may use this information
2. association rules that show a lower than expected confidence for a particular association--may conclude that the products involved in the rule are competing for the same market.

Start with high thresholds to see what rules are found; then reduce the levels as needed.