Frequent pattern mining: association rules

CS434

What Is Frequent Pattern Mining?

- Frequent pattern: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What are the subsequent purchases after buying a PC?
  - What kinds of DNA are sensitive to this new drug?
- Broad applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis
  - Web log (click stream) analysis
  - DNA sequence analysis

Association rules

Data: 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

Example of Association Rules

{Diaper} $\rightarrow$ {Beer},
{Milk, Bread} $\rightarrow$ {Eggs,Coke},
{Beer, Bread} $\rightarrow$ {Milk},

Implication means **co-occurrence, not causality!**

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

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 $X \rightarrow Y$, where X and Y are itemsets
  - Example:
    $\{Milk, Diaper\} \rightarrow \{Beer\}$
- Rule Evaluation Metrics
  - Support (s)
    - Fraction of transactions that contain both X and Y: P(X ^ Y)
  - Confidence (c)
    - Measures how often items in Y appear in transactions that contain X : P(Y|X)

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:

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

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

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

Problem definition: Association Rules Mining

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

- **Inputs:**
  Itemset $X=\{x_1, \ldots, x_k\}$,
  thresholds: *min_sup, min_conf*
- **Output**:
  All the rules $X \rightarrow Y$ having:
  support (P(X^Y)) ≥ *min_sup*
  confidence (P(Y|X))≥ *min_conf*

Let min_sup = 50%, min_conf = 50%:

$A \rightarrow C$ (50%, 66.7%)

$C \rightarrow A$ (50%, 100%)

Brute-force solution

- List all possible association rules
- Compute the support and confidence for each rule
- Prune rules that fail the *min_sup* and *min_conf* thresholds

$\Rightarrow$ 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} $\rightarrow$ {Beer} (s=0.4, c=0.67)
{Milk,Beer} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Beer} $\rightarrow$ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk,Beer} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper,Beer} (s=0.4, c=0.5)

Observations:

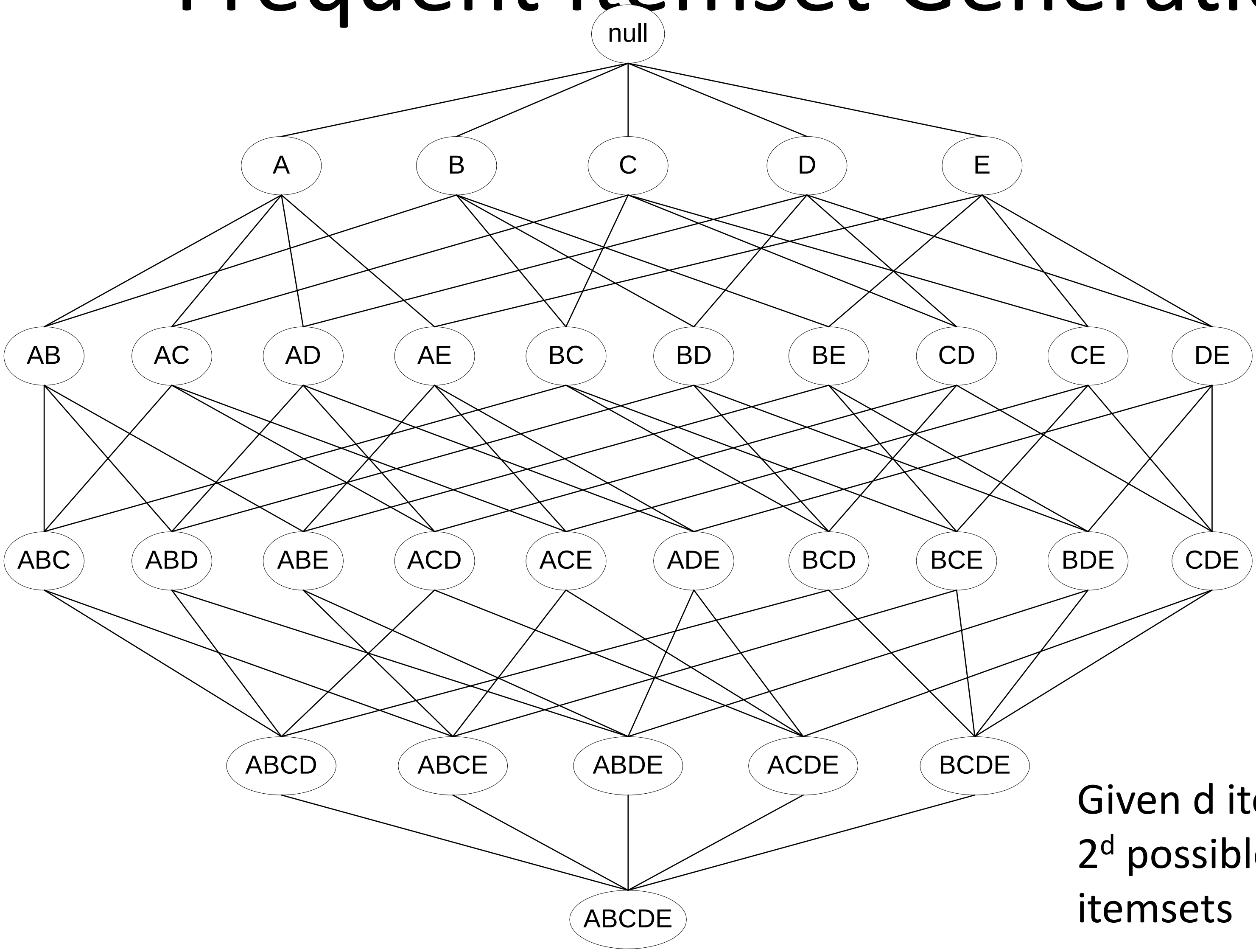
- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements
- We can first find all frequent itemsets that satisfy the support requirement

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ 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

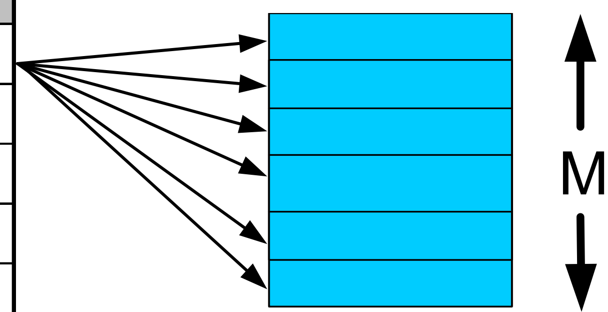
Transactions

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 $M = 2^d$!!!

Reducing Number of Candidates

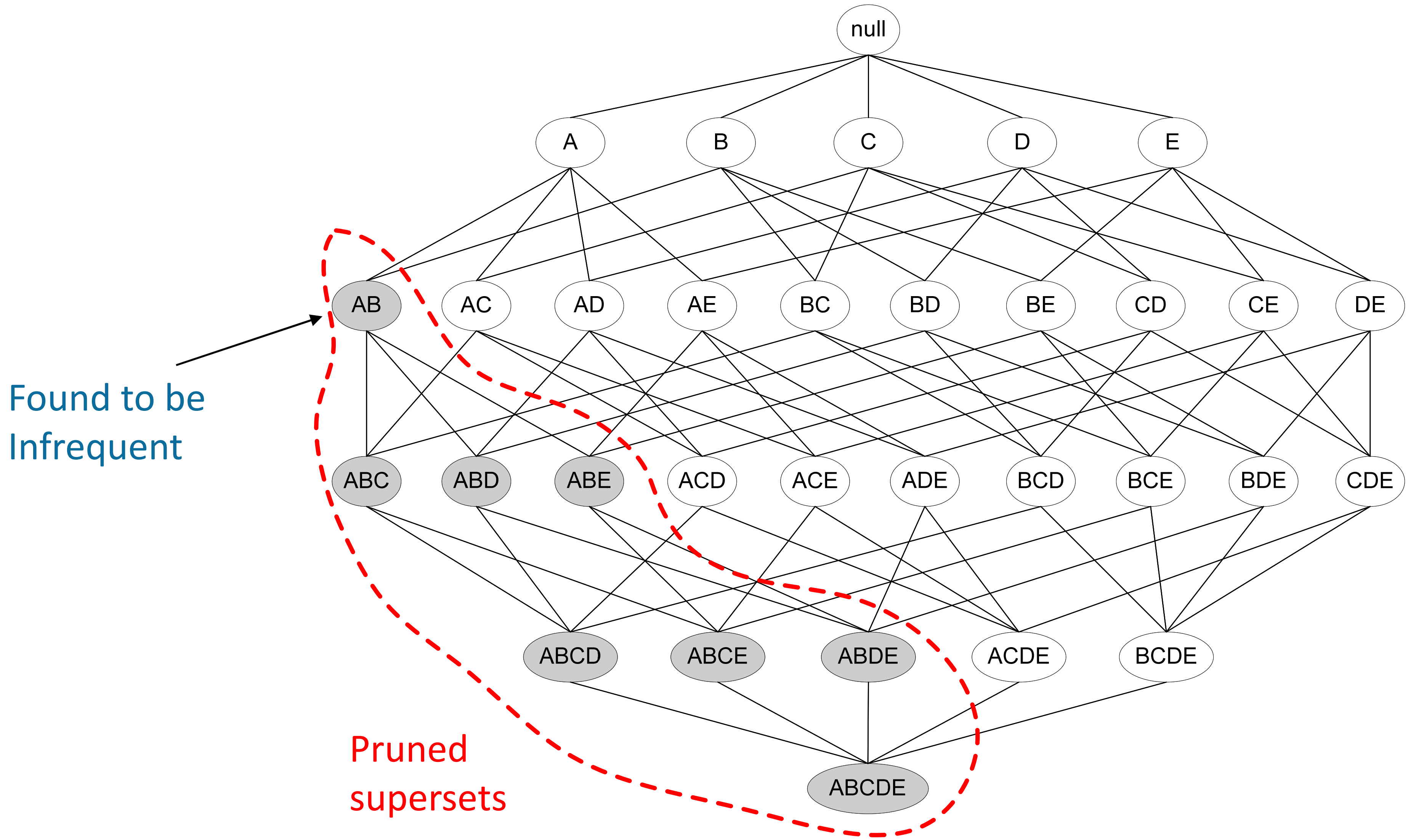
- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
  - If **{beer, diaper, nuts}** is frequent, so is **{beer, diaper}**
  - i.e., every transaction having {beer, diaper, nuts} also contains {beer, diaper}

- 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

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)

Min Support count = 3

Triplets (3-itemsets)

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

If every subset is considered,

$$C_1^6 + C_2^6 + C_3^6 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$

The Apriori Algorithm

- Identify all frequent itemsets (with given *minsup*)
- 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
    - 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

The Apriori Algorithm

- Pseudo-code:

  C_k: Candidate itemset of size k
  L_k : frequent itemset of size k

  L_1 = {frequent items};
  for ($k = 1$; $L_k \mathrel{!=} \emptyset$; 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 $\cup_k L_k$;

How to Generate Candidates?

- Suppose the items in L_k are listed in an order (e.g., alphabetic ordering)
- Step 1: self-joining L_k

  For all itemsets **p** and **q** in L_k such that

  $\boldsymbol{p}.item_i = \boldsymbol{q}.item_i$ for I = 1, 2, …, k-1 and $\boldsymbol{p}.item_k < \boldsymbol{q}.item_k$

  Add to $\boldsymbol{C_{k+1}}$

  $\boldsymbol{p}.item_1, \boldsymbol{p}.item_2, \ldots, \boldsymbol{p}.item_k, \boldsymbol{q}.item_k$

- Step 2: pruning

  For all ***itemsets c in*** $\boldsymbol{C_{k+1}}$ do

  For all (k)-subsets ***s*** of c do

  if *(**s** is not in* L_k*)* then delete ***c*** from C_{k+1}

Important Details of Apriori

Self-joining rule:
1. we join two itemsets if and only if they only differ by their last item
2. When joining, the items are always ranked based on a fixed ordering of the items (e.g., alphabetic ordering)

- Example of Candidate-generation
  - L_3=*{abc, abd, acd, ace, bcd}*
  - Self-joining: L_3*L_3
    - ***abcd*** from ***abc*** and ***abd***
    - ***acde*** from ***acd*** and ***ace***
  - Pruning:
    - ***acde*** is removed because ***ade*** is not in L_3
  - C_4={***abcd***}

Why not abd, and acd -> abcd?

Why should this work?

- How can we be sure we are not missing any possible itemset?
- This can be seen by proving that for every possible frequent k+1-itemset, it will be included using this self-joining process

Proof
For any k +1 item set S (with items ranked), it will be included by joining the following two subsets:

1. S_k = {the first k items of S}
2. S'_k = *S* with the *k*-th item removed

Clearly S_k and S'_k are frequent, and differ by only the last item.
So they must satisfy the self-join condition and S_k ∩ S’_k = S

The Apriori Algorithm—An Example

$Sup_{min} = 2/4$

Database TDB

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

1^{st} 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}

2^{nd} 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

{A, B, C}?

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     – Generate all itemsets whose support $\geq$ minsup

  2. Rule Generation
     – Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
     – Enumerate all possible rules from the frequent itemset and out these of high confidence

Example: Generating rules

- **Min_conf = 80%**

Database TDB

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

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

L2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

$A \rightarrow C$: 100%
$C \rightarrow A$: 66.7%
$B \rightarrow C$: 66.7%
$C \rightarrow B$: 66.7%
$B \rightarrow E$: 100%
$E \rightarrow B$: 100%
$C \rightarrow E$: 66.7%
$E \rightarrow C$: 66.7%

L3

Itemset	sup
{B, C, E}	2

$B, C \rightarrow E$: 100%
$B, E \rightarrow C$: 66.7%
$C, E \rightarrow B$: 100%

Frequent-Pattern Mining: Summary

- Frequent pattern mining—an important task in data mining
- "Scalable" frequent pattern mining methods
  - Apriori (Candidate generation & test)
- The Apriori property has also been used in mining other type of patterns such as sequential and structured patterns
- Problem: frequent patterns are not necessarily interesting patterns
  - Bread -> milk is not really interesting although it has high support and confidence
  - Many other measures of interestingness exist to address this problem
    - Such as "unexpectedness"

Comparing Association rule with Supervised learning

- Supervised learning
  - Have predefined class variable
  - Focus on difference one class from another
- Association rule mining
  - Do not have predefined target class variable
  - Right hand side of the rule can have many items
  - We could place the class variable C on the right hand side of a rule, but it does not focus on differentiating classes, but more on characterizing a class

What you need to know

- What is an association rule?
- What are the support and confidence of a rule?
- The apriori property
- How to find frequent itemset using the aprioir property
  - The Candidate Generation : self-join, and prune
  - Why is it correct?
- How to produce association rules based on frequent itemsets?