Associations and Frequent Item Analysis

Outline

- Transactions
- Frequent itemsets
- Subset Property
- Association rules
- Applications

Transactions Example

TID	Produce
1	MILK, BREAD, EGGS
2	BREAD, SUGAR
3	BREAD, CEREAL
4	MILK, BREAD, SUGAR
5	MILK, CEREAL
6	BREAD, CEREAL
7	MILK, CEREAL
8	MILK, BREAD, CEREAL, EGGS
9	MILK, BREAD, CEREAL

Transaction database: Example

TID	Products
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

ITEMS:

Instances = Transactions

A = milk
B= bread
C= cereal
D= sugar
E= eggs

Transaction database: Example

Attributes converted to binary flags

TID	Products
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

TID	A	B	C	D	E
1	1	1	0	0	1
2	0	1	0	1	0
3	0	1	1	0	0
4	1	1	0	1	0
5	1	0	1	0	0
6	0	1	1	0	0
7	1	0	1	0	0
8	1	1	1	0	1
9	1	1	1	0	0

Definitions

- Item: *attribute*=*value* pair or simply *value*
  - usually attributes are converted to binary *flags* for each value, e.g. **product="A"** is written as **"A"**
- Itemset I : a subset of possible items
  - Example: I = {A,B,E} (order unimportant)
- Transaction: (TID, itemset)
  - TID is transaction ID

Support and Frequent Itemsets

- Support of an itemset
  - sup(I) = no. of transactions t that support (i.e. contain) I
- In example database:
  - sup ({A,B,E}) = 2, sup ({B,C}) = 4
- Frequent itemset I is one with at least the minimum support count
  - sup(I) >= *minsup*

SUBSET PROPERTY

- **Every subset of a frequent set is frequent!**
- Q: Why is it so?
- A: Example: Suppose {A,B} is frequent. Since each occurrence of A,B includes both A and B, then both A and B must also be frequent
- Similar argument for larger itemsets
- Almost all association rule algorithms are based on this subset property

Association Rules

- Association rule R : *Itemset1 => Itemset2*
  - *Itemset1, 2* are disjoint and *Itemset2* is non-empty
  - meaning: if transaction includes *Itemset1* then it also has *Itemset2*
- Examples
  - A,B => E,C
  - A => B,C

From Frequent Itemsets to Association Rules

- *Q: Given frequent set {A,B,E}, what are possible association rules?*
  - A => B, E
  - A, B => E
  - A, E => B
  - B => A, E
  - B, E => A
  - E => A, B
  - __ => A,B,E (empty rule), or true => A,B,E

Classification vs Association Rules

Classification Rules	Association Rules
▪ Focus on one target field	▪ Many target fields
▪ Specify class in all cases	▪ Applicable in some cases
▪ Measures: Accuracy	▪ Measures: Support, Confidence, Lift

Rule Support and Confidence

- Suppose $R : I => J$ is an association rule
  - sup (R) = sup (I ∪ J) is the *support count*
    - support of itemset I ∪ J (I or J)
  - conf (R) = sup(J) / sup(R) is the *confidence* of R
    - fraction of transactions with I ∪ J that have J
- Association rules with minimum support and count are sometimes called "***strong***" rules

Association Rules Example:

- *Q: Given frequent set {A,B,E}, what association rules have minsup = 2 and minconf= 50% ?*

  A, B => E : conf=2/4 = 50%

  A, E => B : conf=2/2 = 100%

  B, E => A : conf=2/2 = 100%

  E => A, B : conf=2/2 = 100%

Don't qualify

A =>B, E : conf=2/6 =33%< 50%

B => A, E : conf=2/7 = 28% < 50%

__ => A,B,E : conf: 2/9 = 22% < 50%

TID	List of items
1	**A, B, E**
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	**A, B, C, E**
9	A, B, C

Find Strong Association Rules

- A rule has the parameters *minsup* and *minconf*:
  - sup(R) >= *minsup* and conf (R) >= *minconf*
- Problem:
  - Find all association rules with given *minsup* and *minconf*
- First, find all frequent itemsets

Finding Frequent Itemsets

- Start by finding one-item sets (easy)
- *Q: How?*
- A: Simply count the frequencies of all items

Finding itemsets: next level

- Apriori algorithm (Agrawal & Srikant)
- Idea: use one-item sets to generate two-item sets, two-item sets to generate three-item sets, …
  - If (A B) is a frequent item set, then (A) and (B) have to be frequent item sets as well!
  - In general: if X is frequent *k*-item set, then all (*k*-1)-item subsets of X are also frequent

  ⇒Compute *k*-item set by merging (*k*-1)-item sets

An example

- Given: five three-item sets

  `(A B C), (A B D), (A C D), (A C E), (B C D)`
- Lexicographic order improves efficiency
- Candidate four-item sets:

  `(A B C D)` **`Q: OK?`**

A: yes, because all 3-item subsets are frequent

`(A C D E)` **`Q: OK?`**

A: No, because (C D E) is not frequent

Generating Association Rules

- Two stage process:
  - Determine frequent itemsets e.g. with the Apriori algorithm.
  - For each frequent item set *I*
    - for each subset *J* of *I*
      - determine all association rules of the form: *I-J => J*
- Main idea used in both stages : subset property

Example: Generating Rules from an Itemset

- Frequent itemset from golf data:

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

- Seven potential rules:

```
If Humidity = Normal and Windy = False then Play = Yes          4/4
If Humidity = Normal and Play = Yes then Windy = False          4/6
If Windy = False and Play = Yes then Humidity = Normal          4/6
If Humidity = Normal then Windy = False and Play = Yes          4/7
If Windy = False then Humidity = Normal and Play = Yes          4/8
If Play = Yes then Humidity = Normal and Windy = False          4/9
If True then Humidity = Normal and Windy = False and Play = Yes 4/12
```


Rules for the weather data

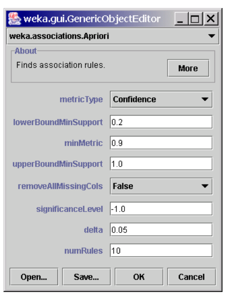
- Rules with support > 1 and confidence = 100%:

	Association rule		Sup.	Conf.
1	Humidity=Normal Windy=False	⇒Play=Yes	4	100%
2	Temperature=Cool	⇒Humidity=Normal	4	100%
3	Outlook=Overcast	⇒Play=Yes	4	100%
4	Temperature=Cold Play=Yes	⇒Humidity=Normal	3	100%
...	...	...	...	...
58	Outlook=Sunny Temperature=Hot	⇒Humidity=High	2	100%

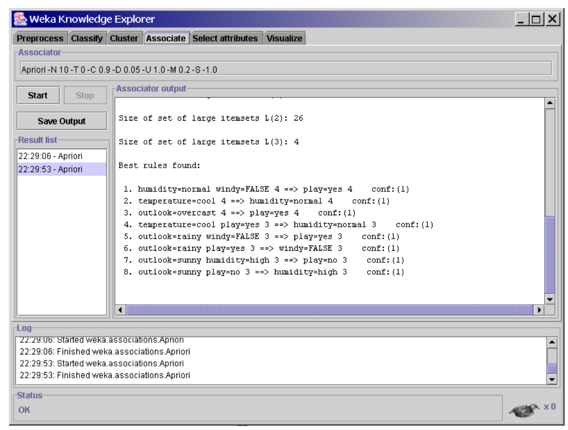
- In total: 3 rules with support four, 5 with support three, and 50 with support two

Weka associations

File: weather.nominal.arff
MinSupport: 0.2



Weka associations: output



Filtering Association Rules

- Problem: any large dataset can lead to very large number of association rules, even with reasonable Min Confidence and Support
- Confidence by itself is not sufficient
  - e.g. if all transactions include Z, then
  - any rule I => Z will have confidence 100%.
- Other measures to filter rules

Association Rule LIFT

- The *lift* of an association rule *I => J* is defined as:
  - lift = P(J|I) / P(J)
  - Note, P(I) = (support of I) / (no. of transactions)
  - ratio of confidence to expected confidence
- Interpretation:
  - if lift > 1, then I and J are positively correlated

    lift < 1, then I are J are negatively correlated.

    lift = 1, then I and J are independent.

Other issues

- ARFF format very inefficient for typical *market basket data*
  - Attributes represent items in a basket and most items are usually missing
- Interestingness of associations
  - find unusual associations: Milk usually goes with bread, but soy milk does not.

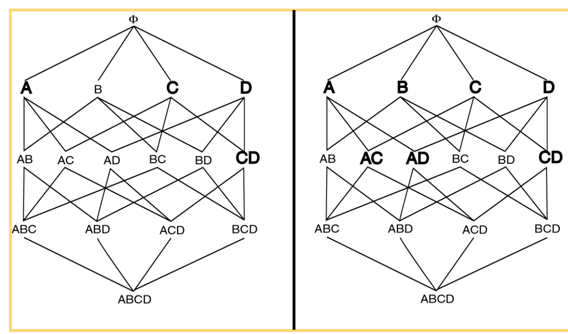
Beyond Binary Data

- Hierarchies
  - drink → milk → low-fat milk → Stop&Shop low-fat milk …
  - find associations on any level
- Sequences over time
- …

Sampling

- Large databases
- Sample the database and apply Apriori to the sample.
- ***Potentially Large Itemsets (PL):*** Large itemsets from sample
- ***Negative Border (BD^-):***
  - Generalization of Apriori-Gen applied to itemsets of varying sizes.
  - Minimal set of itemsets which are not in PL, but whose subsets are all in PL.

Negative Border Example



PL $\quad$ PL $\cup BD^-$(PL)

Sampling Algorithm

1. D_s = sample of Database D;
2. PL = Large itemsets in D_s using smalls;
3. C = PL $\cup$ BD^-(PL);
4. Count C in Database using s;
5. ML = large itemsets in BD^-(PL);
6. If ML = $\varnothing$ then done
7. else C = repeated application of BD^-;
8. Count C in Database;

Sampling Example

- Find AR assuming s = 20%
- D_s = { t_1,t_2}
- Smalls = 10%
- PL = {{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}
- BD^-(PL)={{Beer},{Milk}}
- ML = {{Beer}, {Milk}}
- Repeated application of BD^- generates all remaining itemsets

Sampling Adv/Disadv

- ***Advantages:***
  - Reduces number of database scans to one in the best case and two in worst.
  - Scales better.
- ***Disadvantages:***
  - Potentially large number of candidates in second pass

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. $\quad L^i = \text{Apriori}(D^i)$;
4. $C = L^1 \cup \ldots \cup L^p$;
5. Count C on D to generate L;

Partitioning Example

	Transaction	Items
D^1	t_1	Bread,Jelly,PeanutButter
	t_2	Bread,PeanutButter
D^2	t_3	Bread,Milk,PeanutButter
	t_4	Beer,Bread
	t_5	Beer,Milk

S=10%

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

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

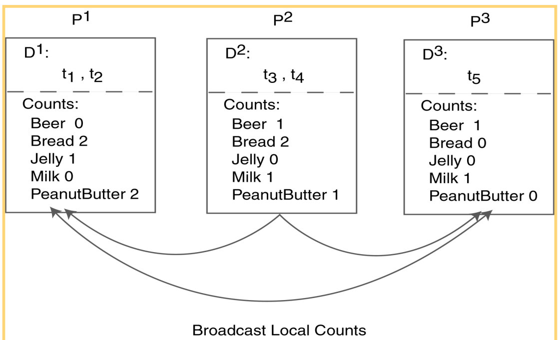
Partitioning Adv/Disadv

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

Count Distribution Algorithm(CDA)

1. Place data partition at each site.
2. In Parallel at each site do
3. $\quad C_1$ = Itemsets of size one in I;
4. $\quad$ Count C_1;
5. $\quad$ Broadcast counts to all sites;
6. $\quad$ Determine global large itemsets of size 1, L_1;
7. $\quad$ i = 1;
8. $\quad$ Repeat
9. $\quad\quad$ i = i + 1;
10. $\quad\quad C_i = \text{Apriori-Gen}(L_{i-1})$;
11. $\quad\quad$ Count C_i;
12. $\quad\quad$ Broadcast counts to all sites;
13. $\quad\quad$ Determine global large itemsets of size i, L_i;
14. $\quad$ until no more large itemsets found;

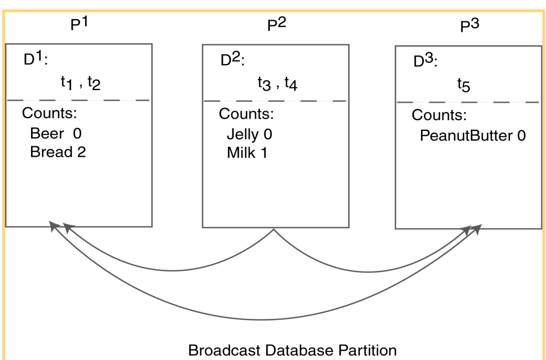
CDA Example



Data Distribution Algorithm(DDA)

1. Place data partition at each site.
2. In Parallel at each site do
3. Determine local candidates of size 1 to count;
4. Broadcast local transactions to other sites;
5. Count local candidates of size 1 on all data;
6. Determine large itemsets of size 1 for local candidates;
7. Broadcast large itemsets to all sites;
8. Determine L_1;
9. i = 1;
10. Repeat
11. i = i + 1;
12. C_i = Apriori-Gen(L_{i-1});
13. Determine local candidates of size i to count;
14. Count, broadcast, and find L_i;
15. until no more large itemsets found;

DDA Example



Applications

- Market basket analysis
  - Store layout, client offers
- ...

Application Difficulties

- Wal-Mart knows that customers who buy Barbie dolls have a 60% likelihood of buying one of three types of candy bars.
- What does Wal-Mart do with information like that? 'I don't have a clue,' says Wal-Mart's chief of merchandising, Lee Scott
- See - KDnuggets 98:01 for many ideas www.kdnuggets.com/news/98/n01.html
- Diapers and beer urban legend

Summary

- Frequent itemsets
- Association rules
- Subset property
- Apriori algorithm
- Application difficulties