Mining Association Rules

- KDD from a DBMS point of view
  - The importance of efficiency
- Market basket analysis
- Association Rules
- The Apriori Algorithm
- Other types of association rules

Knowledge Discovery in Databases

- Or *Data Mining*
- Emerged from several perspectives, e.g., statistical large scale data analysis, machine learning, and *DBMS*
- From a DBMS point of the view, the question often is, what information can be gathered from an existing DB *efficiently*
- One idea, boolean association rules:
  Beer ⇒Diapers [support=10%, confidence=60%]
- Mining association rules – methods to find such rules efficiently

Market Basket Analysis

- Organizations gather *market baskets* – lists of products that are purchased by customers

Trans	Market Basket (Items)
100	Apples, Bread, Diapers, Fish
200	Apples, Cat Food, Diapers, Garbage Bags
300	Bread, Diapers, Fish
400	Apples, Bread, Fish
500	Cat Food, Fish
600	Bread, Cat Food, Garbage Bags
700	Apples, Bread, Garbage Bags
800	Bread, Fish
900	Apples, Bread, Garbage Bags
1000	Bread, Garbage Bags

How are Market Baskets Used?

- Look for combinations of products
- For example, note that Bread and Diapers occur together, we can:
  - Put the Bread near the Diapers so that if a customer buys one they will buy the other
  - Put the two items far apart in the store (hoping that having purchased one they will wander across the store looking for the other and perhaps pick up an intervening item – garbage bags)
  - Put out a coupon for one item in the hopes of also selling the other item

How Do We Characterize Sets of Items?

- Association rule:
  Items X ⇒ Items Y [support = S%, confidence = C%]
- Read X implies Y with support S% and confidence C%
- X and Y are sets of items that appear in the same market baskets with:

$$Support(X \Rightarrow Y) = \frac{\#\ times\ X\ and\ Y\ occur\ in\ the\ same\ basket}{total\ number\ of\ baskets}$$

$$Confidence(X \Rightarrow Y) = \frac{\#\ times\ X\ and\ Y\ occur\ in\ the\ same\ basket}{\#times\ X\ occurs\ in\ a\ basket}$$

Association Rules (Boolean)

- Apples ⇒ Bread [Sup=40% (4/10), Conf=80% (4/5)
- Cat Food ⇒ Garb Bags [Sup=20% (2/10), Conf=67% (2/3)

Trans	Market Basket (Items)
100	Apples, Bread, Diapers, Fish
200	Apples, *Cat Food*, Diapers, Garbage Bags
300	Bread, Diapers, Fish
400	Apples, Bread, Fish
500	*Cat Food*, Fish
600	Bread, *Cat Food*, Garbage Bags
700	Apples, Bread, Garbage Bags
800	Bread, Fish
900	Apples, Bread, Garbage Bags
1000	Bread, Garbage Bags

Mining Association Rules

- Which ones to pick?
  - Every set of items that appears has some level of support (but in many cases minimal)
- General approach: set a minimum support threshold (minimum % of tuples the set of items must appear in)
  - Find all such sets and sort them by support and confidence
- Key question: how do we find those sets of items that have appropriate support efficiently?

Apriori

- Agrawal and Srikant
- Efficient mechanism for finding association rules by efficiently finding sets of items (itemsets) that meet a minimal support criterion
- Builds itemsets of size K that meet the support criterion by using the items of size K 1
- Makes use of certain principles regarding itemsets to limit the work that needs to be done
- Each size of itemsets constructed in one pass through the DB
- Very popular association rule mining algorithm

Apriori Properties

- For some support threshold S
  - If any itemset I meets the threshold S, then any non-empty subset of I also meets the threshold
  - If any itemset I does not meet the threshold S, any superset (including 0 or more other items) will also not meet the threshold
- These properties can be used to prune the search space and reduce the amount of work that needs to be done

The Apriori Algorithm

```
L1 = the set of itemsets of size 1 that meet the
   minimal threshold criterion
K=2

Repeat
   CK = AprioriGen(LK-1,S)  // Generate candidates
   Initialize count for each candidate in CK to 0
   For each tuple in the DB
       Increment count for any matching item in CK
   LK = items from CK with count ≥ S
Until (LK is empty) or (K = max # of items)
```


AprioriGen

- Assume that itemsets are represented as ordered list of items

```
AprioriGen(LK-1,S)
   CK = {}
   for each I1 in LK-1
     for each I2 in LK-1
       if (first K-2 items of I1, I2 same) ∧
          (last item of I1, I2 differs) then
         form item I consisting of first K-2 items
           of I1 and last item of I1 and I2
         if (all subsets of I meet minimum
             support threshold S)
           add I to CK
  return CK
```


Apriori Example

- Set support threshold to 20% (2 items)

Tran	Items
100	A B D F
200	A C D G
300	B D F
400	A B F
500	C F
600	B C G
700	A B G
800	B F
900	A B G
1000	B G

Construct L1, items with count ≥ 2

L_1

Itemset	Count
A	5
B	8
C	3
D	3
F	5
G	5

Apriori Example

- From L_1 construct C_2 (candidates of size 2) – all 15 are candidates

Tran	Items
100	A B D F
200	A C D G
300	B D F
400	A B F
500	C F
600	B C G
700	A B G
800	B F
900	A B G
1000	B G

L_1

Itemset	Count
A	5
B	8
C	3
D	3
F	5
G	5

C_2

Itemset	Count
A B	?
A C	?
A D	?
A F	?
A G	?
B C	?
B D	?
B F	?
B G	?
C D	?
C F	?
C G	?
D F	?
D G	?
F G	?

Apriori Example

- Count actual tuples for C_2 and eliminate any candidates not meeting S

Tran	Items
100	A B D F
200	A C D G
300	B D F
400	A B F
500	C F
600	B C G
700	A B G
800	B F
900	A B G
1000	B G

L_1

Itemset	Count
A	5
B	8
C	3
D	3
F	5
G	5

C_2

Itemset	Count
A B	4
A C	1
A D	2
A F	2
A G	3
B C	1
B D	2
B F	3
B G	3
C D	1
C F	1
C G	2
D F	2
D G	1
F G	0

Apriori Example

- Construct candidates C_3 from L_2 eliminating candidates that can not meet threshold based on L_2

Tran	Items
100	A B D F
200	A C D G
300	B D F
400	A B F
500	C F
600	B C G
700	A B G
800	B F
900	A B G
1000	B G

L_2

Itemset	Count
A B	4
A D	2
A F	2
A G	3
B D	2
B F	3
B G	3
C G	2
D F	2

C_3

Itemset	Count
A B D	?
A B F	?
A B G	?
A D F	?
A D G	?
A F G	?
B D F	?
B D G	?
B F G	?

Apriori Example

- Consult actual data to count for candidates in C_3 and eliminate any candidates that don't have enough support

Tran	Items
100	A B D F
200	A C D G
300	B D F
400	A B F
500	C F
600	B C G
700	A B G
800	B F
900	A B G
1000	B G

C_3

Itemset	Count
A B D	1
A B F	2
A B G	2
A D F	2
B D F	2

Apriori Example

- Construct candidates from L_3 for C_4 eliminating any that do not can not meet the minimum support threshold based on L_3

Tran	Items
100	A B D F
200	A C D G
300	B D F
400	A B F
500	C F
600	B C G
700	A B G
800	B F
900	A B G
1000	B G

L_3

Itemset	Count
A B F	2
A B G	2
A D F	2
B D F	2

C_4

Itemset	Count
A B F G	?

Constructing Association Rules

- Every remaining item in one of the sets L_K has enough support to be a rule
- For each item I in some L_K, separate the items into two non empty sets (generally two or more ways to do this, each corresponding to association rule)
  - Support of rule is based on count for item I
  - Confidence is obtained by dividing count for item I by count for subset on the left of the rule you form (note that due to the apriori properties, this item is guaranteed to have been counted in some set L_K)

Improving Efficiency

- Hashing – especially useful for constructing L_2 level,
  - While tuples read to construct the L_1 counts, determine hash value for each pair of items in the tuple and add 1 to the count for that hash bucket
  - When considering candidates, eliminate any whose hash bucket does not contain enough tuples for support
- Transaction (tuple) reduction – any tuple that does not contain any L_K frequent items will not contain any L_{K+1} frequent items and can be eliminated
- And many more …

Other Variations

- Quantitative Association Rules

  Age [X,31..39] $\wedge$ Income[X,$40K..$50K] $\Rightarrow$ Buy [X,Computer]
  - Key question, how to construct bins (what quantitative values go together)
- Multilevel Association Rules
  - May not be a lot of support for very specific things (very few cases of buying Dell Pent 3 1 GB memory … and Logitech Mouse)
  - But there may be support as you consider the general categories of items (Dell Pent 3 is a computer, Logitech is a type of mouse, lots of support for buying Computer and Mouse together) – can have a multilevel hierarchy and look for rules at various levels of that hierarchy

Association Rules

- Mining can proceed efficiently, but what exactly does an association rule tell us?
- Need to consider if the expected support is really any better than we would expect to see from random distribution
  - Does A and B having 25% support follow from A and B each having 50% support on their own?