Computer Science 477/577

Association Rule Mining: Apriori

Lecture 12

Database and Rule Assumptions

- Assume a database comprised of *n transactions*
  - Each of which is a set of *items*
- Transaction might correspond to a set of purchases made by a customer
  - Examples
    - {milk, cheese, bread}
    - {fish, cheese, bread, milk, sugar}
- Goal: *association rules*
  - Examples: ‘buying fish and sugar is often associated with buying milk and cheese’,
- As before only want rules that meet certain criteria for ‘interestingness’
  - Specified later.

Database and Rule Assumptions

- Not interested in the quantity of cheese or the number of cans of dog food etc. bought.
- Do not record the items that a customer did *not* buy
- Not interested in rules that include a test of what was *not* bought,
  - ‘Customers who buy milk but do not buy cheese generally buy bread’.
  - We only look for rules that link items that were actually bought.

Terminology and Notation

- Let *m* be the number possible items that can be bought
- Let *I* denote the set of all possible items.
- In practice, *m* can easily be many hundreds or even many thousands.
- Depends on whether a company decides to consider (for example) all the meat it sells as a single item 'meat'
  - Or as a separate item for each type of meat ('beef', 'lamb', 'chicken' etc.)
  - Or as a separate item for each type and weight combination.
- Possible itemset extremely large
  - $2^{|I|}$

Convention

- The items in a transaction (or any other itemset) are listed in standard order
  - May be alphabetical or something similar, e.g.
  - Always write {cheese, fish, meat},
    - Not {meat, fish, cheese} etc.
- Harmless and reduces and simplifies calculations needed to discover 'interesting'

Transaction number	Transactions (itemsets)
1	{a, b, c}
2	{a, b, c, d, e}
3	{b}
4	{c, d, e}
5	{c}
6	{b, c, d}
7	{c, d, e}
8	{c, e}

Database with eight transactions

Itemset Support

- *Support count* of an itemset *S*, or *count* the number of transactions in the database matched by *S*.
- An itemset *S matches* a transaction *T* (which is itself an itemset) if *S* is a subset of *T*
  - All the items in *S* are also in *T*.
  - Example: *{*bread, milk*}* matches the transaction *{*cheese, bread, fish, milk, wine*}*.
- If *S* = *{*bread, milk*}* has a support count of 12, written as count(*S*) = 12, 12 of the transactions in the database contain both the items bread and milk.
- We define the *support* of an itemset *S*, written as support(*S*), to be the proportion of itemsets in the database that are matched by *S*,
  - The proportion of transactions that contain all the items in *S*.
  - Support(*S*) = count(*S*)/*n*,
    - *n* is the number of transactions in the database.

Association Rules

Transaction number	Transactions (itemsets)
1	{a, b, c}
2	{a, b, c, d, e}
3	{b}
4	{c, d, e}
5	{c}
6	{b, c, d}
7	{c, d, e}
8	{c, e}

- Example
  - When items *c* and *d* are bought item *e* is often bought
- We can write this as the rule

$$\{c,d\} \rightarrow \{e\}$$

  - Arrow is read as 'implies'
  - A *prediction*
- The rule *cd* → *e* is typical of most if not all of the rules used in Association Rule Mining
  - Not invariably correct.
  - Satisfied for transactions for transactions 2, 4 and 7
  - But not 6

More Terminology and Notation

- *Support count* of an itemset *S*, or just the *count* of an itemset *S*,
  - The the number of transactions in the database matched by *S*.
- An itemset *S* *matches* a transaction *T* (which is itself an itemset) if *S* is a subset of *T*,
  - All the items in *S* are also in *T*. For example itemset
  - {bread, milk} matches the transaction {cheese, bread, fish, milk, wine}.
- If an itemset *S* = {bread, milk} has a support count of 12
  - count(*S*) = 12 or count({bread, milk}) = 12,
- 12 of the transactions in the database contain both the items bread and milk.

Support

- *Support* of an itemset *S*, support(*S*), is proportion of itemsets in the database that are matched by *S*,
  - The proportion of transactions that contain all the items in *S*.
- Alternatively we can look at it in terms of the frequency with which the items in *S* occur together in the database.
- So we have support(*S*) = $\frac{count(S)}{n}$
  - Where *n* is the number of transactions in the database.

Association Rules

- Itemsets are sets, but ignore set-theoretic notation.
- The presence of items *c*, *d* and *e* in transactions 2, 4, and 7 can support other rules such as

$$c \rightarrow ed$$

- and

$$e \rightarrow cd$$

  - (which do not have to be invariably correct)
- $count(L)$ = 4 and $count\ (L \cup R)\ =\ 3.$
- 8 transactions in the database → calculations are
  - $Support(L)\ =\ count(L)/8\ =\ 4/8$
  - $Support(L\ \cup\ R)\ =\ count(L\ \cup\ R)/8\ =\ 3/8$

Rule Confidence

- Confidence of a rule can be calculated either by
  - $\mathrm{Confidence}(L \rightarrow R) = \mathrm{count}(L \cup R)/\mathrm{count}(L)$

  or by
  - $\mathrm{Confidence}(L \rightarrow R) = \mathrm{support}(L \cup R)/\mathrm{support}(L)$
- Typically reject any rule for which the support is below a minimum threshold value called *minsup*
  - Typically 0.01 (i.e. 1%)
- Also to reject all rule with confidence below a minimum threshold value called *minconf*, typically 0.8 (i.e. 80%).
- For the rule *cd* → *e*, the confidence is $\mathrm{count}(\{c, d, e\})/\mathrm{count}(\{c, d\})$
  - Which is 3/4 = 0.75.

Exercise

- Only one rule has confidence about minsup, ≥ 0.8

Rule $L \rightarrow R$	count$(L \cup R)$	count(L)	confidence$(L \rightarrow R)$
$de \rightarrow c$	3	3	1.0
$ce \rightarrow d$	3	4	0.75
$cd \rightarrow e$	3	4	0.75
$e \rightarrow cd$	3	4	0.75
$d \rightarrow ce$	3	4	0.75
$c \rightarrow de$	3	7	0.43

Generating Rules

- Terminology
  - *Frequent itemset* to mean any itemset for which the value of support is greater than or equal to *minsup*.
  - The terms *supported itemset* and *large itemset* are often used instead of frequent itemset.
- Basic but very inefficient method for generating rules from transaction database
- 1. Generate all supported itemsets $L \cup R$ with cardinality at least two.
- 2. For each such itemset generate all the possible rules with at least one item on each side and retain those for which confidence $\geq minconf$.

Computing Rules with Basic Method

- The number of possible itemsets $L \cup R$ is the same as the number of possible subsets of *I*, the set of all items, which has cardinality *m*.
  - There are 2^m such subsets.
  - *m* have a single element
  - One has no elements (the empty set).
- Thus the number of itemsets $L \cup R$ with cardinality at least 2 is $2^m - m - 1$.
- If *m* is (unrealistically) 20 the number of itemsets $L \cup R$

$$2^{20} - 20 - 1 = 1,048,555.$$

- If is (still unrealistically) 100 the number of itemsets $L \cup R$ is

$$2^{100} - 100 - 1 \approx 10^{30}$$

- Generating and testing all rules impossible

A Priori Algorithm

- Theorem 1
  - If an itemset is supported, all of its (non-empty) subsets are also supported.
    - I.e., every subset of a frequent set is frequent
- Theorem 2
  - If $L_k = \emptyset$ (the empty set) then L_{k+1}, L_{k+2}, etc. must also be empty.
- Generate the supported itemsets in ascending order of cardinality
  - All those with one element first
  - Then all those with two elements, etc.
- At each stage, the set L_k of supported items of cardinality k is generated from the previous set L_{k-1}
- If at any stage L_k is $\emptyset$, the empty set we know that L_{k+1}, L_{k+2} etc. must also be empty

Generating new Rule Candidates

- Use L_{k-1} to form a *candidate set* C_k
  - Itemsets of cardinality *k*.
- C_k must be constructed so as to all the supported itemsets of cardinality *k*
  - May contain some other itemsets that are not supported.
- Next we need to generate L_k as a subset of C_k.
- Discard some of the members of C_k as possible members of L_k by inspecting the members of L_{k-1}.
- Check the remainder against the transactions in the database to establish support values.
- Only those itemsets with support greater than or equal to *minsup* are copied from C_k into L_k.

Pseudo-code

```
Create L_1 = set of supported itemsets of cardinality one
Set k to 2
while (L_{k-1} ≠ ∅) {
    Create C_k from L_{k-1}
    Prune all the itemsets in C_k that are not
        supported, to create L_k
    Increase k by 1
}
The set of all supported itemsets is L_1 ∪ L_2 ∪ ··· ∪ L_k
```

- To start the process we construct C_1,
  - Set of all itemsets comprising just a single item,
  - Make a pass through the database counting the number of transactions that match each of these itemsets.
  - Divide these counts by the number of transactions in the database
    - Checking for minsup each single-element itemset.
      - Discard all those with support < *minsup* to yield L_k.
- Continue until L_k is empty.

AprioriGen - Generating C_k from L_k

- Assume that L_4 is the list

  {{p, q, r, s}, {p, q, r, t}, {p, q, r, z}, {p, q, s, z}, {p, r, s, z}, {q, r, s, z}, {r, s, w, x}, {r, s, w, z}, {r, t, v, x}, {r, t, v, z}, {r, t, x, z}, {r, v, x, y}, {r, v, x, z}, {r, v, y, z}, {r, x, y, z}, {t, v, x, z}, {v, x, y, z}}
  - Seventeen itemsets of cardinality four.
- Six pairs of elements that have the first three elements in common.
- Each combination causes to be placed into C_5

First itemset	Second itemset	Contribution to C_5
$\{p,q,r,s\}$	$\{p,q,r,t\}$	$\{p,q,r,s,t\}$
$\{p,q,r,s\}$	$\{p,q,r,z\}$	$\{p,q,r,s,z\}$
$\{p,q,r,t\}$	$\{p,q,r,z\}$	$\{p,q,r,t,z\}$
$\{r,s,w,x\}$	$\{r,s,w,z\}$	$\{r,s,w,x,z\}$
$\{r,t,v,x\}$	$\{r,t,v,z\}$	$\{r,t,v,x,z\}$
$\{r,v,x,y\}$	$\{r,v,x,z\}$	$\{r,v,x,y,z\}$

AprioriGen

- The pruning step where each of the subsets of cardinality four of the itemsets in C_5 are examined:

Itemset in C_5	Subsets all in L_4?
$\{p, q, r, s, t\}$	No, e.g. $\{p, q, s, t\}$ is not a member of L_4
$\{p, q, r, s, z\}$	Yes
$\{p, q, r, t, z\}$	No, e.g. $\{p, q, t, z\}$ is not a member of L_4
$\{r, s, w, x, z\}$	No, e.g. $\{r, s, x, z\}$ is not a member of L_4
$\{r, t, v, x, z\}$	Yes
$\{r, v, x, y, z\}$	Yes

- Eliminate first, third and fourth itemsets from C_5, making the final version of candidate set C_5

$$\{\{p, q, r, s, z\}, \{r, t, v, x, z\}, \{r, v, x, y, z\}\}$$

- The three itemsets in C_5 checked against the database
  - Establish which are supported.

Example

- Assume a database with 100 items and a large number of transactions.
- Construct C_1
  - Itemsets with a single member.
- A pass though the database to establish the support count for each of the 100 itemsets in C_1 and from these calculate L_1,
  - Set of supported itemsets
    - Comprise just a single member
- Assume that L_1 has just 8 of these members, namely {a}, {b}, {c}, {d}, {e}, {f}, {g} and {h}.
- Can now form candidate itemsets of cardinality two.

Generating two-item Sets

- In generating C_1 from L_1 all pairs of (single-item) itemsets in L_1 are considered to match at the 'join' step,
  - Nothing to the left of the rightmost element of each one that might fail to match.
- In this case the candidate generation algorithm gives us as members of C_1 all the itemsets with two members drawn from the eight items *a*, *b*, *c*, . . . , *h*.
- Candidate itemset of two elements cannot include any of the other 92 items from the original set of 100, e.g. {*a, z*}
  - For each, one of its subsets would not be supported.

Generating two-item Sets

- There are 28 possible itemsets of cardinality 2 that can be formed from the items a, b, c, . . . , h.
- They are
  - {a, b}, {a, c}, {a, d}, {a, e}, {a, f}, {a, g}, {a, h}, {b, c}, {b, d}, {b, e}, {b, f}, {b, g}, {b, h}, {c, d}, {c, e}, {c, f}, {c, g}, {c, h}, {d, e}, {d, f}, {d, g}, {d, h}, {e, f}, {e, g}, {e, h}, {f, g}, {f, h}, {g, h}.

A Second Pass

- Reject any itemsets that have support less than *minsup*.
- Assume only 6 of the 28 itemsets with two elements turn out to be supported,
  - L_2 = {{a, c}, {a, d}, {a, h}, {c, g}, {c, h}, {g, h}}.
- The algorithm for generating C_3 now yields just four members
  - {a, c, d}, {a, c, h}, {a, d, h}, {c, g, h}.
- Check subsets are supported.
  - Itemsets {a, c, d} and {a, d, h} fail
    - Subsets {c, d} and {d, h} are not members of L_2.
- Possible members: {a, c, h} and {c, g, h} are possible members of L_3

Third Pass

- A third pass through the database finds the itemsets {a, c, h} and {c, g, h}.
- Assume they both turn out to be supported,
  - So L_3 = {{a, c, h}, {c, g, h}}.
- We now need to calculate C_4.
- No members,
  - Two members of L_3 have no element in common.
- Since C_3 is empty, by Theorem 2, L_3 must also be empty
- Found all the itemsets of cardinality at least two with three passes through the database.
- Needed to find the support counts for $100 + 28 + 2 = 130$
  - A vast improvement over checking through the total number of possible itemsets for 100 items
    - 10^{30}

Generating Rules

- The set of all supported itemsets with at least two members is the union of L_2 and L_3
  - {{a, c}, {a, d}, {a, h}, {c, g}, {c, h}, {g, h}, {a, c, h}, {c, g, h}}.
- Eight itemsets.
- Next need to generate the candidate rules
  - Determine which have a confidence value greater than or equal to *minconf*.

Improvements

- Apriori has substantial efficiency problems
  - When there are a large number of transactions,
  - Large number of items
  - Or both.
- Main problems is the large number of candidate itemsets generated during the early stages of the process.
- If the number of supported itemsets of cardinality one (the members of L_1) is a large *N*,
  - Number of candidate itemsets in C_2, $\frac{N(N-1)}{2}$ can be very large.
- A fairly large (but not huge) database may comprise over 1,000 items and 100,000 transactions.
  - 800 supported itemsets in L_1, of itemsets in C_2 is $800 \times 799/2$, which is approximately $320{,}000$.

Generating Rules for a Supported Itemset

- If $L \cup R$ has k elements, generate possible rules $L \to R$
  - Check their confidence value.
- Method: generate all possible right-hand sides in turn.
- Each one must have at least one and at most $k - 1$ elements.
- Elements not on the RHS must be on the LHS
- Example: for {c,d,e}: 6 possible rules.
- The number of ways of selecting *i* items from the *k* in a supported itemset of cardinality *k* for the right-hand side of a rule is given by $\binom{i}{k} = \frac{k!}{(k-i)!i!}$
  - Also denoted ${}_iC_k$
- Total number of rules $\sum_{i=1}^{k-1} \binom{k}{k-1}$

Reducing Rules

- If k is, 10, this number is manageable.
- For $k = 10$ there are $2^{10} - 2 = 1022$ possible rules.
- For $k = 20$ it is $1{,}048{,}574$
- Theorem 3
- Transferring members of a supported itemset from the left-hand side of a rule to the right-hand side cannot increase the value of rule confidence
- A rule is *confident* if the confidence of a rule ≥ *minconf*
  - Otherwise, *unconfident*.
- Theorem 3 two important results:
  - Any superset of an unconfident right-hand itemset is unconfident.
  - Any (non-empty) subset of a confident right-hand itemset is confident

Reducing Rules

- Any superset of an unconfident right-hand itemset is unconfident.
- Any (non-empty) subset of a confident right-hand itemset is confident
- Search space of RHS reduced
  - Similar to Apriori
  - Considerable reduction in the number of candidate rules
- Generate confident right-hand side itemsets of increasing cardinality
- If at any stage there are no more confident itemsets of a certain cardinality there cannot be any of larger cardinality
  - Rule generation process can stop.