CS4445 DATA MINING B Term 2014. WPI

Solutions to HOMEWORK 4. Association Rules Using Apriori. By Artem Gritsenko.

1. "Join" condition or "merge" is the process of generation new candidate k-itemset based on the frequent (k-1)-itemsets found in the previous iteration. The idea is to merge a pair of frequent (k-1)-itemsets from the previous iteration of the Apriori algorithm into k-itemset if and only if their first k-2 items are the same. This results in adding the last item of the first merging (k-1)-itemset to the second (k-1)-itemset and creating a new k-itemset.
2. "Subset" condition or "candidate pruning" is the process that allows to reduce the number of itemsets (from the "merge" step) for which we need to perform support counting by removing infrequent itemsets from consideration. This is done by ensuring that all the possible (k-1)-subsets of the k-itemset (constructed with "merge" procedure) are frequent, that is they are among the frequent (k-1)-itemsets from the previous iteration of the Apriori algorithm.
3. We start the Apriori algorithm on the dataset by producing a list of all possible 1-itemsets (that in this case are all possible attribute-value pairs in the dataset). For each itemset, support is calculated as the frequency of the itemset occurring in the dataset.
   The next step is to perform support-based pruning strategy that will eliminate all the itemsets that have a low support value (we use minimum support =3 by the problem definition). This step allows us to eliminate infrequent rules.

Support counts for candidate 1-itemsets					
Buying=vhigh	2	**Maint=med**	**6**	**Safety=med**	**10**
Buying=high	**4**	**Maint=low**	**4**	**Safety=high**	**5**
Buying=med	**11**	**Persons=2**	**6**	**Class=unacc**	**15**
Buying=low	**5**	**Persons=4**	**11**	**Class=acc**	**5**
Maint=vhigh	**9**	**Persons=more**	**5**	Class=good	2
Maint=high	**3**	**Safety=low**	**7**		

Here is the resulting list of frequent 1-itemsets.

Frequent 1-itemsets
Buying=high
Buying=med
Buying=low
Maint=vhigh
Maint=high
Maint=med
Maint=low
Persons=2
Persons=4
Persons=more
Safety=low
Safety=med
Safety=high
Class=unacc
Class=acc

On the next steps the algorithm generates new candidate k-itemsets using the frequent (k-1)-itemsets from previous iteration. The “merge” and “candidate pruning” procedures do not apply at this step because the 1-itemsets from previous iteration have empty prefixes. That is why we just use all possible combinations of 1-itemsets to generate candidate 2-itemsets. Here and further to reduce duplicates we will use lexicographical ordering of the attributes (the order in which attributes appear in the dataset) when generating candidate itemsets. As we use minimum support = 3, so we prune all the itemsets with support less than 3.

Support counts for candidate 2-itemsets			
Buying=high, Maint=vhigh	0	Maint=vhigh, Persons=2	2
Buying=high, Maint=high	1	**Maint=vhigh, Persons=4**	**3**
Buying=high, Maint=med	2	**Maint=vhigh, Persons=more**	**4**
Buying=high, Maint=low	1	Maint=vhigh, Safety=high	0
Buying=high, Persons=2	1	**Maint=vhigh, Safety=med**	**7**
Buying=high, Persons=4	**3**	Maint=vhigh, Safety=low	2
Buying=high, Persons=more	0	**Maint=vhigh, Class=unacc**	**6**
Buying=high, Safety=high	2	**Maint=vhigh, Class=acc**	**3**
Buying=high, Safety=med	0	Maint=high, Persons=2	1
Buying=high, Safety=low	2	Maint=high, Persons=4	2
Buying=high, Class=unacc	**3**	Maint=high, Persons=more	0
Buying=high**,** Class=acc	0	Maint=high, Safety=high	1
Buying=med, Maint=vhigh	**6**	Maint=high, Safety=med	0
Buying= med, Maint=high	1	Maint=high, Safety=low	2
Buying= med, Maint=med	2	**Maint=high, Class=unacc**	**3**
Buying= med, Maint=low	2	Maint=high, Class=acc	0
Buying= med, Persons=2	2	Maint=med, Persons=2	1
Buying= med, Persons=4	**6**	**Maint=med, Persons=4**	**4**
Buying= med, Persons=more	**3**	Maint=med, Persons=more	1
Buying= med, Safety=low	**5**	Maint=med, Safety=high	2
Buying= med, Safety=med	**6**	Maint=med, Safety=med	2
Buying= med, Safety=high	0	Maint=med, Safety=low	2
Buying= med, Class=unacc	**7**	**Maint=med, Class=unacc**	**3**
Buying= med, Class=acc	**4**	Maint=med, Class=acc	1
Buying=low, Maint=vhigh	1	Maint=low, Persons=2	2
Buying= low, Maint=high	1	Maint= low, Persons=4	2
Buying= low, Maint=med	2	Maint= low, Persons=more	0
Buying= low, Maint=low	1	Maint= low, Safety=high	2
Buying= low, Persons=2	**3**	Maint= low, Safety=med	1
Buying= low, Persons=4	1	Maint= low, Safety=low	1
Buying= low, Persons=more	1	Maint= low, Class=unacc	1
Buying= low, Safety=low	0	**Maint= low, Class=acc**	**3**
Buying= low, Safety=med	2	**Safety=low, Class=unacc**	**7**
Buying= low, Safety=high	**3**	Safety=low, Class=acc	0
Buying= low, Class=unacc	**3**	**Safety=med, Class=unacc**	**5**
Buying= low**,** Class=acc	1	**Safety=med, Class=acc**	**5**
Persons=2, Safety=low	0	**Safety=high, Class=unacc**	**3**

Persons=2, Safety=med	**3**	Safety=high, Class=acc	0
Persons=2, Safety=high	**3**		
Persons=2, Class=unacc	**6**		
Persons=2, Class=acc	0		
Persons=4, Safety=low	**5**		
Persons=4, Safety=med	**4**		
Persons=4, Safety=high	2		
Persons=4, Class=unacc	**6**		
Persons=4, Class=acc	**3**		
Persons=more, Safety=low	2		
Persons= more, Safety=med	**3**		
Persons= more, Safety=high	0		
Persons= more, Class=unacc	**3**		
Persons= more, Class=acc	2		

Here is the list of frequent 2-itemsets that survived the support-based pruning. To generate the candidate 3-itemsets we use the "merge" procedure described above. We will merge pair of 2-itemsets only if their first 1 item is identical. The 2-itemsets with the same first item marked with the same color in the table below.

Frequent 2-itemsets	
Buying=high, Persons=4	3
Buying=high, Class=unacc	3
Buying=med, Maint=vhigh	6
Buying= med, Persons=4	6
Buying= med, Persons=more	3
Buying= med, Safety=low	5
Buying= med, Safety=med	6
Buying= med, Class=unacc	7
Buying= med, Class=acc	4
Buying= low, Persons=2	3
Buying= low, Safety=high	3
Buying= low, Class=unacc	3
Persons=2, Safety=med	3
Persons=2, Safety=high	3
Persons=2, Class=unacc	6
Persons=4, Safety=low	5
Persons=4, Safety=med	4
Persons=4, Class=unacc	6
Persons=4, Class=acc	3
Persons= more, Safety=med	3
Persons= more, Class=unacc	3
Maint=vhigh, Persons=4	3
Maint=vhigh, Persons=more	4
Maint=vhigh, Safety=med	7
Maint=vhigh, Class=unacc	6
Maint=vhigh, Class=acc	3

Maint=high, Class=unacc	3
Maint=med, Persons=4	4
Maint=med, Class=unacc	3
Maint= low, Class=acc	3
Safety=low, Class=unacc	7
Safety=med, Class=unacc	5
Safety=med, Class=acc	5
Safety=high, Class=unacc	3

Here are all the possible 3-itemsets generated from 2-itemsets with "merge" procedure (within each colored group). Before running support-based pruning we want to make sure that all the 2-item subsets of these 3-itemsets are frequent. Based on the Apriori principle if a subset of the itemset is infrequent than the itemset is infrequent. So, we apply "subset" condition to the 3-itemsets. Because we know that the subset with the 1st and 2nd items and the subset with the 2nd and 3rd items came from frequent itemsets, we only need to make sure that the subset with the 2nd and 3rd items is among the frequent 2-itemsets from the previous iteration. The itemsets that did not meet the conditions are crossed over in the table below.

As in the previous steps, we perform support-based pruning leaving only the 3-itemsets that have support larger or equal to 3.

Support counts for candidate 3-itemsets	
Buying=high, Persons=4, Class=unacc	2
Buying=med, Maint=vhigh, Persons=4	2
Buying= med, Maint=vhigh, Persons=more	2
~~Buying= med, **Maint=vhigh, Safety=low**~~	
Buying= med, Maint=vhigh, Safety=med	**4**
Buying= med, Maint=vhigh, Class=unacc	**4**
Buying= med, Maint=vhigh, Class=acc	2
Buying= med, Persons=4, Safety=low	**3**
Buying= med, Persons=4, Safety=med	**3**
Buying= med, Persons=4, Class=unacc	**3**
Buying= med, Persons=4, Class=acc	**3**
~~Buying= med, **Persons= more, Safety=low**~~	
Buying= med, Persons= more, Safety=med	1
Buying= med, Persons= more, Class=unacc	2
~~**Buying= med, Persons=more, Class=acc**~~	
Buying= med, Maint=vhigh, Safety=med	**4**
~~Buying= med, **Safety=low, Class=acc**~~	
Buying= med, Safety=low, Class=acc	**5**
Buying= med, Safety=med, Class=unacc	2
Buying= med, Safety=med, Class=acc	**4**
Buying= low, Persons=2, Safety=high	2
Buying= low, Persons=2, Class=unacc	**3**
Buying= low, Safety=high, Class=unacc	2
Persons=2, Safety=med, Class=unacc	**3**

Persons=2, Safety=high, Class=unacc	**3**
Persons=4, Safety=low, Class=unacc	**5**
~~Persons=4, **Safety=low, Class=acc**~~	
Persons=4, Safety=med, Class=unacc	1
Persons=4, Safety=med, Class=acc	**3**
Persons=more, Safety=med, Class=unacc	1
Maint=vhigh, Persons=4, Safety=med	**3**
Maint=vhigh, Persons=4, Class=unacc	1
Maint=vhigh, Persons=4, Class=acc	2
Maint=vhigh, Persons=more,Safety=med	2
Maint=vhigh,Persons=more,Class=unacc	**3**
~~Maint=vhigh, **Persons= more, Class=acc**~~	
Maint=vhigh, Safety=med, Class=unacc	**4**
Maint=vhigh, Safety=med, Class=acc	**3**
Maint=med, Persons=4, Class=unacc	2

Here is the list of the frequent 3-itemsets. To generate candidate 4-itemsets we apply "merge" procedure again. We merge 3-itemsets that have first 3 items identical to each other. The groups of 3-itemsets that have first 3 items identical are marked with different colors.

Frequent 3-itemsets	
Buying= med, Maint=vhigh, Safety=med	4
Buying= med, Maint=vhigh, Class=unacc	4
Buying= med, Persons=4, Safety=low	3
Buying= med, Persons=4, Safety=med	3
Buying= med, Persons=4, Class=unacc	3
Buying= med, Persons=4, Class=acc	3
Buying= med, Safety=low, Class=unacc	5
Buying= med, Safety=med, Class=acc	4
Buying= low, Persons=2, Class=unacc	3
Persons=2, Safety=med, Class=unacc	3
Persons=2, Safety=high, Class=unacc	3
Persons=4, Safety=low, Class=unacc	5
Persons=4, Safety=med, Class=acc	3
Maint=vhigh, Persons=4, Safety=med	3
Maint=vhigh,Persons=more,Class=unacc	3
Maint=vhigh, Safety=med, Class=unacc	4
Maint=vhigh, Safety=med, Class=acc	3

Merging of the 3-itemsets (within the same color band following the merge condition) produces 5 candidate 4-itemsets. We perform "candidate pruning" to eliminate infrequent itemsets. We check if all possible combinations of 3-items subsets present among the frequent 3-itemsets.

Support counts for candidate 4-itemsets	
~~Buying= med, Maint=vhigh, Safety=med, Class=unacc~~	
~~Buying= med, Persons=4, Safety=low, Class=acc~~	
Buying= med, Persons=4, Safety=low, Class=unacc	**3**
~~Buying= med, Persons=4, Safety=med, Class=unacc~~	
Buying= med, Persons=4, Safety=med, Class=acc	**3**

For the 1st itemset Buying= med, Maint=vhigh, Safety=med, Class=unacc we check:

1) Maint=vhigh, Safety=med, Class=unacc - **frequent**
2) Buying= med, Safety=med, Class=unacc - **infrequent**

We don't need to check the condition for Buying= med, Maint=vhigh, Class=unacc and Buying= med, Maint=vhigh, Class=unacc as we know they are frequent.

For the 2nd itemset Buying= med, Persons=4, Safety=low, Class=acc in the same manner we check:

1) Buying= med, Safety=low, Class=acc - **infrequent**
2) Persons=4, Safety=low, Class=acc - **infrequent**

For the 3rd itemset Buying= med, Persons=4, Safety=low, Class=unacc we check:

1) Buying= med, Safety=low, Class=unacc - **frequent**
2) Persons=4, Safety=low, Class=unacc- **frequent**

For the 4th itemset Buying= med, Persons=4, Safety=med, Class=unacc we check:

1) Buying= med, Safety=med, Class=unacc - **infrequent**
2) Persons=4, Safety=med, Class=unacc - **infrequent**

For the 5th itemset Buying= med, Persons=4, Safety=med, Class=acc we check:

1) Buying= med, Safety=med, Class=acc - **frequent**
2) Persons=4, Safety=med, Class=acc – **frequent**

Frequent 4-itemsets	
Buying= med, Persons=4, Safety=low, Class=unacc	3
Buying= med, Persons=4, Safety=med, Class=acc	3

We stop the process of the frequent itemset generation as we can't construct candidate 5-itemsets from these frequent 4-itemsets as they don't satisfy the merge condition.

2. **Interestingness (or goodness) metrics:**

$$supp(X => Y) = supp(X \& Y) = P(X \& Y)$$

$$conf(X => Y) = P(Y \mid X) = \frac{supp(X => Y)}{supp(X)}$$

$$lift\ (X => Y) = \frac{P(Y|X)}{P(Y)} = \frac{P(X \& Y)}{P(X)P(Y)} = \frac{supp(X \& Y)}{supp(X)supp(Y)} = \frac{supp(X => Y)}{supp(X)supp(Y)}$$

$$conv(X => Y) = \frac{P(not\ Y)}{P(not\ Y \mid X)} = \frac{1 - P(Y)}{1 - P(Y|X)} = \frac{1 - supp(Y)}{1 - conf(X => Y)}$$

$$leverage(X => Y) = supp(X => Y) - supp(X)supp(Y) = supp(X \& Y) - supp(X)supp(Y)$$

For Rule 1: Buying= med, Persons=4 => Safety=low, Class=unacc (see rule construction below):

lift(Rule 1) = lift() = 0.5 / (7/22) = 1.57

Lift compares the probability that Y occurs in a data instance given that X occurs, against the probability of Y occurring in a data instance in general. Hence the value of the lift for the first rule tells us that that Y is 1.57 times more likely to occur when X occurs than in general.

leverage(Rule1) = 3/22 – (6/22 * 7/22) = 0.049

Leverage measures the difference of antecedent and consequent of the rule appearing together in the data set and what would be expected if antecedent and consequent where statistically independent.

conviction(Rule1)=(1-7/22) / (1-0.5) = 1.36

Conviction compares the probability that the consequent doesn't appear in a data instance against the probability that the consequent doesn't appear in a data instance if the antecedent appears. Hence the value of conviction for rule 1 tells us that Y is 1.36 times more likely to not occur in a data instance than when X occurs in the data instance.

3. RULE GENERATION

The last step of the Apriori algorithm is the rule generation. Rules can be generated by dividing each itemset on two subsets X and Y and creating the rule X=>Y. All possible combinations of these subsets will create new rules. We will create the rules for the 1st 4-itemset that we generated above. We will evaluate the rules by calculating lift and confidence.

Itemset for rule generation
Buying= med, Persons=4, Safety=low, Class=unacc

$$conf(X => Y) = \frac{supp(X => Y)}{supp(X)}$$

Rule 1: Buying= med, Persons=4 => Safety=low, Class=unacc

conf(Rule1) =(3/22) / (6/22) = 0.5

lift(Rule 1) =0.5 / (7/22) = 1.57

Rule 2: Buying= med, Safety=low => Persons=4 Class=unacc

conf(Rule2)= (3/22) / (5/22)=0.6

lift(Rule2)=0.6 / (6/22) = 2.2

Rule 3: Buying= med, Class=unacc => Persons=4, Safety=low

conf(Rule3) = (3/22) / (7/22) = 0.43

lift(Rule3) =0.43 / (5/22) = 1.892

Rule 4: Persons=4, Class=unacc => Buying= med, Safety=low

conf(Rule4) = (3/22) / (6/22) =0.5

lift(Rule4)=0.5 / (5/22) = 2.2

Rule 5: Persons=4, Safety=low => Buying= med, Class=unacc

conf(Rule5) = (3/22) / (5/22) = 0.6

lift(Rule5) = 0.6 / (7/22) = 1.892

Rule 6: Safety=low, Class=unacc => Buying= med, Persons=4

conf(Rule6) = (3/22) / (7/22) = 0.43

lift(Rule6) = 0.42 / (6/22) = 1.57

4. **Weks's Implementation of the Apriori Algorithm:**
   We will discuss the following parameters of the Weka implementation of the Apriori algorithm:
   1) lowerBoundMinSupport
   2) upperBoundMinSupport
   3) delta
   4) metricType
   5) minMetric
   6) numRules

The last parameter limits the number of the rules that the Apriori algorithm is supposed to find. Weka's implementation of the Apriori algorithm runs until it finds a user-specified number of rules indicated by numRules parameter. If the algorithm can't find at least this amount of rules indicated by the user with certain min support threshold, it decreases the min support threshold and runs the algorithm over again. The first 3 parameters tune the way the iterative algorithm works. UpperBoundMinSupport is the initial min support value with which the algorithm starts. If it can't find the required amount of rules it decreases min support by delta. The process is repeated iteratively until the required number of rules is found or the min support threshold reaches the value of lowerBoundMinSupport. Metric type and min metric are used in the rule generation. Metric type is the metric that is used to evaluate the rules (lift, confidence, conviction or leverage). The minMetric sets the threshold for the rule to be generated or to be discarded.