7.5.

Section 7.2.4 presented various ways of defining negatively correlated patterns. Consider Definition 7.3: “Suppose that itemsets X and Y are both frequent, that is, sup(X) ≥ min sup and sup(Y) ≥ min sup, where min sup is the minimum support threshold. If (P(X|Y) + P(Y|X))/2 < , where is a negative pattern threshold, then pattern X Y is a negatively correlated pattern.” Design an efficient pattern growth algorithm for mining the set of negatively correlated patterns.

*Answer:*

Proposed algorithm:

1. Firstly, generate a list of frequent itemsets, either using apriori or fp-growth algorithm. From this list, we also know support value of each itemset.
2. For each itemset with length of *k*, generate all possible combination of (*k*, *k-1*, …, 2, 1)-itemsets.
3. Iterate through the transaction data set *D* to calculate support value of each itemset. We can generate all support values just by iterating through *D* once.
4. After that, find pairs of itemsets that are negatively correlated according to Kulzcynski definition below:
5. Notice that the Kulzcynski definition can be rearranged as shown above. Since we already collect the support value of each possible itemsets in step 3 and we have the support value of the original *k*-itemset, we can find every pair of subsets of the *k*-itemset and see if they are negatively correlated.

Suppose we have a frequent itemset *I* and a pair of its subset *s* and (*I-s)*. We get the support value of *I* when we generate all frequent itemsets using apriori algorithm. The itemset *s* and (*I*-*s*) are two possible itemset combinations from *I.* In step 3, we calculate the support value of *s* and (*I*-*s*). Hence, if is less than , itemset *s* and (*I-s*) are negatively correlated.

7.9.

*Answer:*

Suppose P1 and P2 are two closed patterns with a support of and respectively. is a set of transactions that contain pattern P1, for example *t1, t2, t3, t4, t5,* and *t6.* is a set of transactions that contain pattern P2, such as *t1, t2, t3, t7, t8,* and *t9*. We would expect that two close patterns appear in almost identical set of transactions. This means that we expect that will be very close to . If P1 and P2 are completely different pattern, they will not have any intersection in their transaction sets, hence . On the other hand, if P1 and P2 are completely identical, they will appear in a similar set of transaction and will satisfy . Therefore, the formula *Pat\_dist* covers the possible distance between P1 and P2 and normalize the value to [0,1].

7.10. Design an efficient algorithm that compresses a large set of patterns into a small compact set. Discuss whether your mining method is robust under different pattern similarity definitions.

*Answer*:

A well-known algorithm to do pattern compression is a greedy algorithm.[[1]](#footnote-1) The following pseudocode describes the algorithm.

Input: (1) a collection of frequent patterns FP, (2) a minimum support, M, and (3) a quality measure for clustering, .

Output: The set of representative patterns

BEGIN

for each *P FP so that support (P) M*

Insert *P* into the set *E*;

for each *Q FP so that Q covers P*

Insert *P* into the set *Q*;

while *E*

Find a RP that maximizes |set(RP)|;

for each *Q* set(RP)

Remove *Q* from *E* and the remaining sets;

Output RP;

END

8.3.

Given a decision tree, you have the option of (a) converting the decision tree to rules and then pruning the resulting rules, or (b) pruning the decision tree and then converting the pruned tree to rules. What advantage does (a) have over (b)?

*Answer:*

In the prepruning method described in method (b), we will remove a subtree completely so it cannot be used again in the algorithm. Method (b) also brings a problem of choosing appropriate threshold. High thresholds might result in an oversimplified tree. Method (a) is less restrictive since we only prune the rule and keep the tree unmodified.

8.5. Given a 5-GB data set with 50 attributes (each containing 100 distinct values) and 512 MB of main memory in your laptop, outline an efficient method that constructs decision trees in such large data sets. Justify your answer by rough calculation of your main memory usage.

*Answer*:

If we use the RainForest method, we will need an AVC-set for every attribute at each tree node. Since the root node include all the attributes, the biggest amount of space is needed to compute the AVC-set for the root node. Since we have 100 distinct values and 50 attributes, we will have 100 x 50 x C of values to be stored, assuming we have C classes. These values are long integers. For modern computer, storing around 5000 long integers is not a problem. For other nodes, we will assume that each node on the lower level will have one less attributes so we will have to store 100 x C x of values in total to create a tree. To increase the efficiency and reduce the usage of memory, we can compute the AVC-set for nodes at the same level of the tree in parallel.

8.7. (a) How would you modify the basic decision tree algorithm to take into consideration the count of each generalized data tuple (i.e., of each row entry)?

*Answer*:

We can consider each tuple as a single item. After we count the count of each tuple, we calculate the attribute selection measure, for example information gain. The count is used to determine the most common class among the tuples. The tree is constructed by following a common classification algorithm.

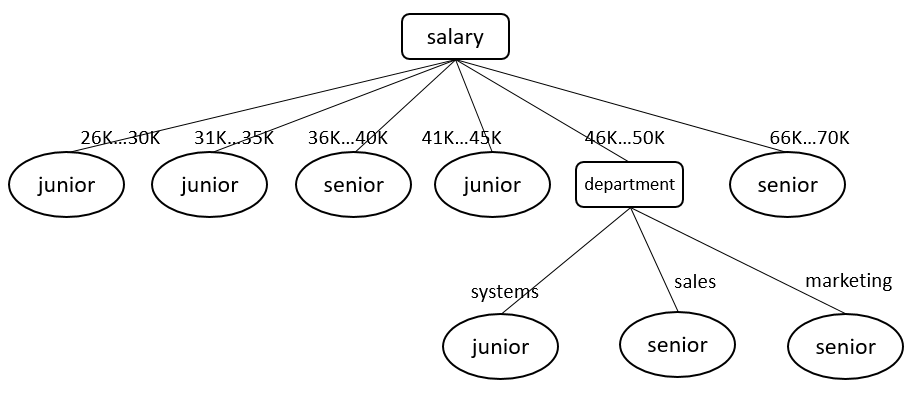
(b) Use your algorithm to construct a decision tree from the given data.

*Answer:*

We have 6 juniors and 5 seniors. The expected information needed to classify a tuple is bits. Next, we will compute the expected information requirement for each attribute.

Hence, we will get the highest information gain if we partition the data using *salary* attribute. The information gain is 0.899 – 0.3615 = 0.5375.

To further partition the data, we now look at all records that have a salary value in the range of 46K – 50K. Among these records, the department attributes match exactly the status so we will divide the subtree using this attribute. The final decision tree is shown as follows.



(c) Given a data tuple having the values “systems,” “26 . . . 30,” and “46–50K” for the attributes department, age, and salary, respectively, what would a naive Bayesian classification of the status for the tuple be?

Since P(X|senior) = 0, Bayesian classifier would give a prediction probability of 0. On the other hand, the value of P(X|junior) is 0.018. This would result in the prediction value of 0.678. Hence, a naïve Bayesian classification would predict junior as a class.

8.12. The data tuples of Figure 8.25 are sorted by decreasing probability value, as returned by a classifier. For each tuple, compute the values for the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Compute the true positive rate (TPR) and false positive rate (FPR). Plot the ROC curve for the data.

*Answer*:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tuple # | Class | Prob | TP | FP | TN | FN | TPR | FPR |
| 1 | P | 0.95 | 1 | 0 | 5 | 4 | 0.2 | 0 |
| 2 | N | 0.85 | 1 | 1 | 4 | 4 | 0.2 | 0.2 |
| 3 | P | 0.78 | 2 | 1 | 4 | 3 | 0.4 | 0.2 |
| 4 | P | 0.66 | 3 | 1 | 4 | 2 | 0.6 | 0.2 |
| 5 | N | 0.6 | 3 | 2 | 3 | 2 | 0.6 | 0.4 |
| 6 | P | 0.55 | 4 | 2 | 3 | 1 | 0.8 | 0.4 |
| 7 | N | 0.53 | 4 | 3 | 2 | 1 | 0.8 | 0.6 |
| 8 | N | 0.52 | 4 | 4 | 1 | 1 | 0.8 | 0.8 |
| 9 | N | 0.51 | 4 | 5 | 0 | 1 | 0.8 | 1.0 |
| 10 | P | 0.4 | 5 | 5 | 0 | 0 | 1 | 1.0 |

8.14. Suppose that we want to select between two prediction models, M1 and M2. We have performed 10 rounds of 10-fold cross-validation on each model, where the same data partitioning in round i is used for both M1 and M2. The error rates obtained for M1 are 30.5, 32.2, 20.7, 20.6, 31.0, 41.0, 27.7, 26.0, 21.5, 26.0. The error rates for M2 are 22.4, 14.5, 22.4, 19.6, 20.7, 20.4, 22.1, 19.4, 16.2, 35.0. Comment on whether one model is significantly better than the other considering a significance level of 1%.

*Answer*:

1. Xin, Dong, et all. “Mining Compressed Pattern Sets”. VLDB 2005. [↑](#footnote-ref-1)