1. （必填）自己提出的问题的理解（罗列全部）：
2. 提出的问题1：Function MScandidate-gen()的剪枝方法有改进，生成的项目集是完备还是有例外，如果有的话在Algorithm MS-Apriori()又是怎么排除例外情况的，换句话说，Fk和Ck有什么区别

讨论后的理解：Ck生成了备选集，备选集只满足任意(k-1)子集也是备选集，而Fk限制到备选集的每一个项目支持度都是大于等于最小支持度的。之前对项目的支持度和最小支持度理解有误。

1. 提出的问题2：MS-genRules()将对genRules()怎样作出改进？

讨论后的理解：引入多种最小支持度之后，规则生成会产生头部问题，所以规则生成算法在计算置信度时要特别注意。

1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：
2. 问题3：为什么说在整个计算过程中，我们并不需要将整个数据集加载到内存？有什么算法是必须要把整个数据集都放进去的嘛，为什么要这么做？

自己的理解：因为计算支持度只需要一个一个计数，每次统计的是一个项目集即可，我们不需要把所有项目集都存入内存。但是，有些算法想要构建一个字典并使用它时，我们不得不将整个数据集放在内存里提高查询速度。

1. 问题4：怎么得到Apriori算法是指数级的

自己的理解：频繁集要讨论的是++…+,他们之和是2n.

5、问题5：为什么“不把同时含有频繁项目和稀有项目的项目集作为频繁项目生成”？

自己的理解：因为这样做是没有意义的，频率差太多的项目之间关联不大。

1. （必填）读书计划

1、本周完成的内容章节：2.1-2.5

2、下周计划：2.6-2.9

四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1、读书摘要及理解（选做）

## 1 Association Rules

Let I= {i1, i2, …, im} be the set of items and T= {t1, t2, …, tn} be the set of transactions, where ti ⊆I. An association rule is of the form: X→Y, where X⊆I, Y⊆I, X∩Y=∅.The strength of an association rule can be measured by support and confidence. Support value of X→Y is the fraction of the items contain X∪Y in T(n items in total),which can be computed by

Confidence value of X→Y is an estimation of conditional probability, which can be computed by

Our task is to find all of the rules whose actual support and confidence are greater than the minimum threshold. Note that many algorithms just generate a subset of association rules rather than a complete one.

## 1.1 Apriori Algorithm

Apriori algorithm works both in the frequent itemsets generating and confident association rules mining in the generated frequent itemsets.

## 1.1.1 Frequent Itemsets Generating

Apriori algorithm is based on the downward closure property to generate frequent itemsets and to make it efficient, the items are sorted in lexicographic order. There is a level-wise search in 3 steps in each subsequent pass k.

Step 1. Use candidate-gen() function to build the candidates set Ck with frequent itemsets Fk-1

Step 2. Scan the transaction set T to count the actual support of each candidates c

Step 3. Find out the frequent k-itemsets Fk

The output is F that contains all of the frequent itemsets.

The candidate-gen() function has two steps:

**1.Join Step**: Use two frequent (k-1)-itemsets f1,f2 to generate a candidate c(f1 and f2 have the same items except the last one),then add c to Ck

**2.Prune Step**: Check all of the (k-1)-subsets of c. If one of them is not in Fk-1,then according to the downward closure property, c cannot be frequent ,so delete it from Ck

Note that this flexible algorithm can stop in a certain pass and its final output is unique(different from that of classification or clustering algorithm).However, it may still cause interestingness problem.

## 1.1.2 Association Rules Generation

To generate rules of each frequent itemsets f, for each non-empty subset α of f, we output the rule (f-α)→α if

Due to that for a rule (f − α) → α to hold, all rules of the form (f − αsub) → αsub must also hold, where αsub is a non-empty subset of α, because the support count of (f − αsub) must be less than or equal to the support count of (f − α). So we first generate all rules with one item in the consequent from the frequent itemset f then use the consequents of these rules and the function candidate-gen() to generate all possible consequents with two items that can appear in a rule, and so on.

## 1.2 Mining with Multiple Minimum Supports

For a single support value cannot define some very different items, we use multiple minimum item support(MIS) instead. The Apriori algorithm is replaced by MS-Apriori algorithm to deal with the new problem and the new algorithm will degrade into Apriori algorithm when the MIS values are all the same.

MS-Apriori algorithm is also based on level-wise search but there is a difference in the second level and we sort the items in I in ascending order of their MIS values. In the MScandidate-gen() function, the join step is the same as that in the candidate-gen() function while the prune step is different. And in the rules generating part, a problem of the head item is solved by counting an extra support.

## 1.3 Mining Class Association Rules

In some applications, the user is interested in only rules with some fixed target items on the right-hand side.

Let T be a transaction data set consisting of n transactions. Each transaction is labeled with a class y. Let I be the set of all items in T, Y be the set of all class labels (or target items) and I ∩ Y =∅. A class association rule (CAR) is an implication of the form X → y, where X ⊆ I, and y∈Y. There are two differences:

1.CAR has only one item in subsequence while AR has no limit

2.y must come from class label set Y and can not appear as condition

Unlike normal association rules, CARs can be mined directly in a single step. The key operation is to find all ruleitems that have support above minsup. A ruleitem is of the form: (condset, y), where condset ⊆ I is a set of items, and y∈Y is a class label.

Note that multiple minimum support can be applied to CAR mining ,so can multiple minimum confidence.

## 2.Sequential Patterns

Let I = {i1, i2, …, im} be a set of items. A sequence is an ordered list of itemsets. Recall an itemset X is a non-empty set of items X ⊆ I. We denote a sequence s by 〈a1a2…ar〉, where ai is an itemset, which is also called an element of s. We denote an element (or an itemset) of a sequence by {x1, x2, …, xk}, where xj∈ I is an item. We assume without loss of generality that items in an element of a sequence are in lexicographic order. An item can occur only once in an element of a sequence, but can occur multiple times in different elements. The size of a sequence is the number of **elements** (or itemsets) in the sequence. The length of a sequence is the number of **items** in the sequence. A sequence of **length** k is called a k-sequence.

## 2.1 Mining Sequential Patterns Based on GSP

This section describes two algorithms for mining sequential patterns based on the GSP algorithm: the original GSP, which uses a single minimum support, and MS-GSP, which uses multiple minimum supports.

## 2.1.1 Original GSP

GSP works in almost the same way as the Apriori algorithm and the main difference is in the candidate generation, candidate-gen-SPM():

**1. Join step**. Candidate sequences are generated by joining Fk−1 with Fk−1. A sequence s1 joins with s2 if the subsequence obtained by dropping the first item of s1 is the same as the subsequence obtained by dropping the last item of s2. The candidate sequence generated by joining s1 with s2 is the sequence s1 extended with the last item in s2.

There are two cases:

• the added item forms a separate element if it was a separate element in s2, and is appended at the end of s1 in the merged sequence, and

• the added item is part of the last element of s1 in the merged sequence otherwise.

**In short, append the item just as how it was put.**

When joining F1 with F1, we need to add the item in s2 both as part of an itemset and as a separate element. That is, joining 〈{x}〉 with 〈{y}〉 gives us both 〈{x, y}〉 and 〈{x}{y}〉. Note that x and y in {x, y} are ordered.

**2.** **Prune step**. A candidate sequence is pruned if any one of its (k−1)- subsequences is infrequent (without minimum support).

## 2.1.2 MS-GSP

Let MIS(i) be the MIS value of item i. The minimum support of a sequential pattern P is the lowest MIS value among the items in the pattern.

Unlike that in MS-Apriori, where the first item in each itemset has the lowest MIS value, in sequential pattern mining the item with the lowest MIS value may appear anywhere in a sequence. This causes problems for joining and we introduce the MScandidate-gen-SPM() function to deal with that:

**1 Join Step**. Candidate sequences are generated by joining Fk−1 with Fk−1.

**2** if the MIS value of the first item in a sequence (denoted by s1) is less than (<)the MIS value of every other item in s1 then // s1 and s2 can be equal

Sequence s1 joins with s2 if (1) the subsequences obtained by dropping the second item of s1 and the last item of s2 are the same, **and** (2) the MIS value of the last item of s2 is greater than that of the first item of s1. Candidate sequences are generated by extending s1 with the last item of s2:

• if the last item l in s2 is a separate element then

{l} is appended at the end of s1 as a separate element to form a candidate sequence c1.

if (the length and the size of s1 are both 2) AND (the last item of s2 is greater than the last item of s1) then // maintain lexicographic order

l is added at the end of the last element of s1 to form another candidate sequence c2.

• else if **(**(the length of s1 is 2 and the size of s1 is 1) AND (the last item of s2 is greater than the last item of s1)**)** OR **(**the length of s1 is greater than 2**)** then the last item in s2 is added at the end of the last element of s1 to form the candidate sequence c2.

**3** elseif the MIS value of the last item in a sequence (denoted by s2) is less than(<) the MIS value of every other item in s2 then

A similar method to the one above can be used in the reverse order.

**4** else use the Join Step in candidate-gen-SPM()

**5 Prune step**: A candidate sequence is pruned if any one of its (k−1)-subsequences is infrequent (without minimum support) except the subsequence that does not contain the item with strictly the lowest MIS value.

## 2.2 Mining Sequential Patterns Based on PrefixSpan

We now introduce another sequential pattern mining algorithm, called PrefixSpan, which does not generate candidates. Different from the GSP algorithm, which can be regarded as performing breadth-first search to find all sequential patterns, PrefixSpan performs depth-first search.

By scanning the projected database once, PrefixSpan finds all possible one item extensions to the prefix .Then we can recursively find the subsets by forming their corresponding projected databases. Also, the PrefixSpan algorithm can be adapted to mine with multiple minimum supports and we call the modified algorithm MS-PS.

r-PrefixSpan() is almost the same as PrefixSpan with one important difference. During each recursive call, either the prefix or every sequence in the projected database must contain ik. Another minor difference is that the support difference constraint needs to be checked during each projection as sup(ik) may not be the lowest in the pattern.

2.3 Generating Rules from Sequential Patterns

This section introduces only three types, sequential rules, label sequential rules and class sequential rules, which have been used in Web usage mining and Web content mining.

## 2.3.1 Sequential Rules

A sequential rule (SR) is an implication of the form, X → Y, where Y is a sequence and X is a proper subsequence of Y. Given a minimum support and a minimum confidence, according to the downward closure property, all the rules can be generated from frequent sequences without going to the original sequence data.

## 2.3.2 Label Sequential Rules

Sequential rules may not be restrictive enough in some applications. We introduce a special kind of sequential rules called label sequential rules. A label sequential rule (LSR) is of the form, X → Y, where Y is a sequence and X is a sequence produced from Y by replacing some of its items with wildcards. A wildcard is denoted by an “\*” which matches any item. These replaced items are usually very important and are called labels. The labels are a small subset of all the items in the data.

Note that due to the use of wildcards, frequent sequences alone are not sufficient for computing rule confidences. Scanning the data is needed. Notice also that the same pattern may appear in a data sequence multiple times. Rule confidences thus can be defined in different ways according to application needs. The wildcards may also be restricted to match only certain types of items to make the label prediction meaningful and unambiguous.

## 2.3.3 Class Sequential Rules

Class sequential rules (CSR) are analogous to class association rules (CAR). Let S be a set of data sequences. Each sequence is also labeled with a class y. Let I be the set of all items in S, and Y be the set of all class labels, I ∩ Y = ∅. Thus, the input data D for mining is represented with {(s1, y1), (s2, y2), …, (sn, yn)}, where si is a sequence in S and yi ∈Y is its class. A class sequential rule (CSR) is of the form X → y, where X is a sequence, and y ∈ Y. A data instance (si, yi) is said to **cover** a CSR, X → y, if X is a subsequence of si. A data instance (si, yi) is said to **satisfy** a CSR if X is a subsequence of si and yi = y.