Lab 4
Mining Association Rules

Overview: In the previous labs we have experimented with different classification algorithms – *J48* and *IBk* – and learned how different parameter settings and filters impact their results. This week we turn our attention to *association learning*, an unsupervised form of data mining (after getting a brief introduction to it in the last lab). A key objective of this activity is to gain a thorough understanding of the terms *support* and *confidence*.

1. The *Apriori* algorithm

As we have learned, unsupervised association learning has the objective of generating rules that suggest strong associations between different attributes of the input dataset. These rules are differentiated from classification rules by the absence of a designated class attribute. The *Apriori* algorithm is the primary means for mining such rules.

I will give an overview of this algorithm prior to our lab activity – and soon we will return to it in detail near the end of our coverage of Chapter 4. For now, the key terms to understand are *item sets*, *support*, and *confidence*.

An *item set* is simply a collection of attribute-value pairs of different sizes that appear in the dataset. For example, `outlook=sunny` is a single-item item set from the weather data. This item set happens to occur 5 times in the weather dataset. That means it has *support* (or *coverage*) of 5. A two-item item set is `outlook=sunny; humidity=high`. This has support of 3 in the dataset. Make sure you follow this before continuing.

The *Apriori* algorithm first generates all such item sets with a minimum specified support. Since there are only 14 instances in the weather data, an appropriate level of support is 2 in this case (2/14 = 14% -- no support below 10% is considered interesting).

The algorithm then constructs rules out of all the item sets with sufficient support. In its simplest form, this involves organizing each item set into as many combinations as possible. For example, the item set above generates `outlook=sunny` →`humidity=high`, and `humidity=high`→`outlook=sunny`. These rules – which have support of 3 – are then assessed for their *confidence*. That is support (3) divided by the number of instances in the dataset to which the rule applies. That would be 3/5 (80%) in the first case, and 3/7 (42.9%) in the second case. If the minimum confidence is 90%, neither of these rules would pass muster.

2. Association Rule Mining

To get a feel for how to apply *Apriori*, start by mining rules from the *weather.nominal.arff* dataset. Note that this algorithm expects data that is purely nominal: If present, numeric attributes must be discretized first. After loading the data in the Preprocess panel, click the *Start* button in the Associate panel to run *Apriori* with default options. It outputs 10 rules, ranked according to the confidence measure given in parentheses after each one. The number following a rule's antecedent shows how many instances satisfy the antecedent; the number following the conclusion shows how many instances satisfy the entire rule (this is the rule's "support"). Because both numbers are equal for all 10 rules, the confidence of every rule is exactly 1.

In practice, it can be tedious to find minimum support and confidence values that give satisfactory results. Consequently, Weka's *Apriori* runs the basic algorithm several times. It uses the same user-specified minimum confidence value throughout, given by the *minMetric* parameter. The support level is expressed as a proportion of the total number of instances (14 in the case of the weather data), as a ratio between 0 and 1. The minimum support level starts at a certain value (*upperBoundMinSupport*, default 1.0). In each iteration the support is decreased by a fixed amount (*delta*, default 0.05, 5% of the instances) until either a certain number of rules has been generated (*numRules*, default 10 rules) or the support reaches a certain "minimum minimum" level (*lowerBoundMinSupport*, default 0.1) – because rules are generally uninteresting if they apply to less than 10% of the dataset. These four values can all be specified by the user.

This sounds pretty complicated, so we will examine what happens on the weather data. The Associator output text area shows that the algorithm managed to generate 10 rules. This is based on a minimum confidence level of 0.9, which is the default and is also shown in the output. The *Number of cycles performed*, which is shown as 17, indicates that *Apriori* was actually run 17 times to generate these rules, with 17 different values for the minimum support. The final value, which corresponds to the output that was generated, is 0.15 (corresponding to 0.15 x 14 = 2 instances).

By looking at the options in the Generic Object Editor window, you can see that the initial value for the minimum support (*upperBoundMinSupport*) is 1 by default, and that delta is 0.05. Now 1 - 17 x 0.05 = 0.15, so this explains why a minimum support value of 0.15 is reached after 17 iterations. Note that *upperBoundMinSupport* is decreased by delta before the basic *Apriori* algorithm is run for the first time.

The Associator output text area also shows how many frequent items sets were found, based on the last value of the minimum support that was tried (0.15 in the example). In this case, given a minimum support of two instances, there are 12 item sets of size 1, 47 items sets of 2, 39 item sets of size 3 and six item sets of size 4. By setting *outputItemSets* to *true* before running the algorithm, all those different item sets and the number of instances that support them are shown. Try it out!

Exercise 1 Based on the output, what is the support for this item set?

```
Outlook=rainy  humidity = normal  windy=FALSE play=yes
```

Exercise 2 Suppose you want to generate all rules with a certain confidence and minimum support. This can be done by choosing appropriate values for *minMetric*, *lowerBoundMinSupport*, and *numRules*. What is the total number of possible rules for the weather data for each combination of values in the lab report table below?

3. Mining a Real-World Dataset

Now consider a real-world dataset, *vote.arff*, which gives the votes of 435 U.S. congressmen on 16 key issues gathered in the mid-1980s, and also includes their party affiliation as a binary attribute. This is a purely nominal dataset with some missing values (corresponding to abstentions). It is normally treated as a classification problem, the task being to predict party affiliation based on voting patterns. However, association-rule mining can also be applied to this data to seek interesting associations. More information on the data appears in the comments in the ARFF file.

Exercise 3 Run *Apriori* on this data with default settings. Comment on the rules that are generated. Several of them are quite similar. Be sure you can identify the support and confidence values for each rule. What do you observe about these values?
Exercise 4 It is interesting to see that none of the rules in the default output involve *Class=Republican*. Why do you think that is? Be specific.

4. Market Basket Analysis

Market basket analysis aims to discover interesting purchasing patterns in large datasets of transactional records. Typically these are the contents of individual shoppers' baskets in a supermarket, recorded at the checkout. Interesting patterns could be exploited in the store, such as special offers and product layout.

The "supermarket" dataset (*supermarket.arff*) in Box is a real world transaction data set from a small New Zealand supermarket. Each instance represents a customer transaction – products purchased and the departments involved. The data contains 4,500 instances and 220 attributes. Each attribute is binary and either has a value (*t* for true) or no value ("?" for missing).

The attributes are aggregated to the department level, so, for example, "bread and cake" covers several different products, and a value of *t* indicates that the customer's shopping cart contained at least one product from that department. (Unfortunately, not all departments are named.)

There is a nominal class attribute called *total* that indicates whether the transaction was less than $100 (low) or greater than $100 (high). However, we are not aiming to create a predictive model for *total*. Instead, we are interested in what items were purchased together. The aim is to find useful patterns in the data that may or may not be related to the predicted attribute.

Load the file *supermarket.arff* into the Explorer. Use Edit to study the dataset and make sure you understand it. Then use Apriori to mine this supermarket checkout data for associations. See if you can discover anything interesting.

Exercise 5 Experiment with *Apriori* and investigate the effect of the various parameters described before. There is nothing to write up for this exercise, but spend some time on it. You should be able to identify the rules, and where support and confidence are displayed. In general, you should understand what is going on here.

Lab 4: Mining Association Rules

Name: ___________________________________

Exercise 1: Make sure you've set *outputItemSets* to true.

Exercise 2: Set the *numRules* parameter to 500 for this one.

Minimum Confidence	Minimum Support	Number of Rules
0.9	**0.3**	
0.9	**0.2**	
0.9	**0.1**	
0.8	**0.3**	
0.8	**0.2**	
0.8	**0.1**	
0.7	**0.3**	
0.7	**0.2**	
0.7	**0.1**	

Briefly explain the pattern that you see in the number of rules generated.

Exercise 3: Any comment/observation will do, as long as it shows some thought. An easy thing to comment on is why the rules are all so similar.

Comment/Observation:

How do support and confidence factor in this set of rules? (This is just a review question. Explain why these particular rules "made the cut" using their support and confidence values.)

Exercise 4: This is a math question, not an opinion question!
HINT: What was the minimum support necessary to generate this set of rules? How many instances from the dataset does this support represent? That is how you'll find your answer.