CS4300Spring 2016

Project #4: Decision tree

Submission: Please submit all files through MyGateway. Please do not email final solutions to me, but feel free to ask me questions and code by email.

In this project you will be implementing a decision tree:
In this project you will be implementing a decision tree for a dataset. Decision trees are one of the most popular methods of machine learning and if written correctly, this project could be a step towards experimenting with decision trees to solve many problems.

The pseudocode for the decision tree learning algorithm is in the slides and the textbook, but in this I will give an outline for some ways to start implementing a decision tree.

For starters, a decision tree is a tree that splits on decision variables, also called features. When the decision tree learning algorithm splits a dataset on a decision variable, this will require that you are able to split up your dataset along that feature.

In this project you will be working with lists. You will be working with a list of lists of test data. Each of these lists would represent one piece of sample data and the classification. These lists would not have to include all the decision variables at later stages, but would start full. For example, {(1,0),(2,1),(3,0),(4,1),(0,1)}
would represent a sample piece of example data where decision variable 1 was 0 (false), decision variable 2 was true (1), decision variable 4 was true and the classification of this data was true (0 in this means it is the classification of the data set). Feel free to use your own scheme for this.

Your end goal is to come up with a function that takes in a list of sample lists and then returns a decision tree on these features. One way to start this is by implementing some functions that work with these lists:

`boolean belongsTo(Sample sample, int feature)` that returns whether or not a test sample contains the feature.

`ListOfTwoSamples split(ListofSamples sampleList, int feature)` that takes a list of sample data and returns two lists, the first being every member of sampleList that had this feature as false (and modifies each of these members to remove that feature), the second being every member of sample that had this feature as true (also modifies each of these members to remove that feature).

`double IGOnSplit(ListOfSamples sampleList, int feature)` that takes in a list of sample data and returns the information gain on splitting on this feature.

Your overall task is to write a function that will create a boolean decision tree, with the leaves representing a classification (true or false decision). This should probably be recursive, ending when you hit a point where the split on any decision variable would result in a loss in information or you no longer have any decision variables to split on left. Note that the number of decision variables should be the maximum size of your tree, but your tree might not necessarily have that depth.

Also you will need a function that takes in this tree and a set of Sample data and then queries the tree to determine how that data should be classified (this is for actually using it, once the tree is built). You should think about whether this function correctly classify all your test data that you used to build it.

For input, your program should accept a plain text file in the following format:

```
<numDecisionVariables> <totalData>
<List of decisions delimited by a space> <classification>
```

For example:

```
3 5
1 1 1 0
1 1 0 1
0 0 1 1
1 1 0 0
```

Which would represent a problem with 3 decision variables and 5 total samples, with the first data sample being 3 true inputs and a result of false.

Your output should consist of a text visual representation of your tree and you should then go into a loop and allow me to query your decision tree to make a classification. For example, on the above problem, you should prompt me to enter 3 boolean values (as 0 or 1) and then you would output the

I suggest using STL Lists or Maps if using C++, or JCF Lists or Maps if using Java. If you are using javascript, switch to a real language.

Initially to test, I would suggest some boolean function problem. Calculate a small data set and test it. Then see how it does on novel inputs that weren't used as samples.