CS 540-3: Homework 4

Decision Trees

70 pts total

Release date: April 3, 2016
Due date: April 15, 2016 at 11:59PM

Directions: Please submit *one* file: DecisionTreeImpl.java file to Moodle by the deadline specified. Your code also **MUST** be executable on the CS Linux machines with the command specified and finish running WITHIN FIVE MINUTES. Failure to observe these requirements will *result in a 0 for the assignment*!!! Please be sure you understand the course policies on plagiarism before consulting with other students. See either the TA or instructor if you have any questions.

1 Implementation of Decision Trees [70 pts]

In this question, you are required to build a classification decision tree for categorical attributes. The programming part includes building a tree from a training dataset and pruning the learned decision tree with a tuning dataset.

You are required to implement four methods and one member for the class DecisionTreeImpl. These are as outlined in Fig 1.

```
public class DecisionTreeImpl extends DecisionTree {
        private DecTreeNode root;
        DecisionTreeImpl(DataSet train);
        DecisionTreeImpl(DataSet train, DataSet tune);
        public String classify(Instance instance);
        public void rootInfoGain(DataSet train);
}
```

Figure 1: **DecisionTreeImpl.java**

1.1 Dataset

In each dataset file, there will be a header, as shown in Fig 2, that describes the information about the dataset, which is shown as below. First, there will be several lines starting with // which provide some descriptions and comments for the dataset file. Then, the line starting with "%%" will list all class labels. Finally, the lines with the prefix of "##" will give the name of the attribute and all its possible values. We have written the dataset loading part for you according to this header, so do NOT change it.

```
// Some descriptions and comments for the dataset files
%%,0,1
##,A1,1,2,3
##,A2,1,2,3
##,A3,1,2
##,A4,1,2,3
##,A5,1,2,3,4
##,A6,1,2
```

Figure 2: Example dataset.

1.2 Implementation Details

1.2.1 Predefined data types

For this programming part, we have defined four data types to assist your coding, which are `Instance`, `DataSet`, `DecTreeNode`, `DecisionTreeImpl`. Their data members and methods are all carefully commented; therefore, it should not be hard to understand their meanings and usage.

However, there is one thing you should keep in mind during the implementation. **In order to speed up the comparing and reduce the memory cost, all attributes, their values and class labels are stored as integer indices rather their string value in the `Instance` class**. For example, attribute A1 has three possible values 1, 2 and 3. As A1 is the first attribute introduced in the dataset, it is identified by index 0. The three possible values of A1 are identified as indices 0, 1 and 2 respectively. Similarly an integer is used to represent the class labels in the `Instance` object. In this way, we are able to translate the string value of attributes, values and class labels into indices. To access the actual string name of the class labels and attribute names, you have to access the lists `labels` and `attributes` respectively declared in `DataSet` class. To find the string name of the attribute value you have to access Map `attributeValues` in the same class.

1.2.2 Building the tree

In both `DecisionTreeImpl` methods, you are first required to build the decision tree with the training data. You can either refer to the pseudo code in Figure 18.5 on page 702 of the textbook or the course slides. To finish this part, you may need to write a recursive function as the `DecisionTreeLearning` in the textbook or `buildtree` in the slides.

1.2.3 Pruning

For constructor `DecisionTreeImpl(DataSet train, DataSet tune)`, after building the tree you also need to prune it with the help of the tuning dataset. You can refer to the pseudo code in the slides. To accomplish this, you will likely need to make your own `Prune` function. In order to change the label of the internal nodes to be the majority vote of training instance that reach

that node, you may also need to add some extra parameters to the function `Prune`.

1.3 Classification

The `public String classify(Instance instance)` takes an instance as its input and gives the classification output of the built decision tree as a string. You do not need to worry about printing. That part is already handled within the code.

1.3.1 Breaking Ties

You may encounter a case where a given node has the same number of examples with each label (e.g., 2 positives and 2 negatives) and you need to make a prediction. Although the textbook suggests randomly breaking ties, in this assignment you will choose the class label with the lowest index to break the tie.

1.3.2 About Printing and Information Gain at Root

The only printing you would need to do is within the method `public void rootInfoGain(DataSet train)`. For each attribute print the output one line at a time, first the name of the attribute and then the actual information gain. The gains should be given in the order the attributes appear in the training dataset. An example is given in: output0.txt. You need to print your results with 5 decimal places in decimal representation, so you may use `System.out.format("%.5f", arg)` for printing.

1.4 How to test

We will test your program on multiple training/tuning/testing cases, and the format of testing commands will be like this:

```
java HW4 <modeFlag> <trainFile> <tuneFile> <testFile>
```

where `trainFile`, `tuneFile`, `testFile` are the names of training, tuning, testing dataset which contain the features of corresponding instances. `modeFlag` is an integer valued from 0 to 4, controlling what the program will output:

- 0: print the information gain for each attribute at the root node

- 1: create a decision tree from a training set, print the tree
- 2: create a decision tree from a training set, print the classifications for instances in a test set
- 3: create a decision tree from a training set then prune, print the tree
- 4: create a decision tree from a training set then prune, print the classifications for instances in a test set

In order to facilitate your debugging, we are providing you with sample input files and the corresponding output files. They are `monks-train0.txt`, `monks-tune0.txt`, and `monks-test0.txt` for the input, and `output0.txt` to `output4.txt` for the output, as seen in the zip file. So here is an example command:
java HW4 1 monks-train0.txt monks-tune0.txt monks-test0.txt

You are **NOT** responsible for any file input or console output (other than `public void rootInfoGain (DataSet train)`). We have written the class `HW4` for you, which will load the data and pass it to the method you are implementing.

As part of our testing process, we will use the file you submit to us, add in the other class files, and call `javac HW4.java` to compile your code, and call the main method of `HW4` with parameters of our choosing. Make sure that your code runs on one of the computers in the department because we will conduct our test on such computers.

1.5 Deliverables

Hand in your modified version of the code skeleton we provide you with your implementation of the decision tree algorithm.