CS 540-1: Introduction to Artificial Intelligence
Homework Assignment # 2

Assigned: 9/26
Due: 10/10 before class

Hand in your homework:

This homework includes a written portion and a programming portion. Please type the written portion and hand in the file in pdf format and name it as **WrittenPart.pdf**. The first page of the pdf must include a header with: **your name, Wisc username, class section, HW# , date and, if handed in late, how many days late it is**. The programming portion will be required to write in Java, and all files needed to run your program, including any support files but input/output/debug files, should be submitted. Finally, please create a folder named as <**Wisc username**>_**HW#**, put **all your codes** as well as **WrittenPart.pdf** into the folder and compress it as <**Wisc username**>_**HW#.zip**. This final zip file should then be uploaded to the appropriate place on the course Moodle website.

Late Policy:

All assignments are due at the beginning of class on the due date. One (1) day late, defined as a 24-hour period from the deadline (weekday or weekend), will result in 10% of the total points for the assignment deducted. So, for example, if a 100-point assignment is due on a Wednesday 9:55 a.m., and it is handed in between Wednesday 9:55 a.m. and Thursday 9:55 a.m., 10 points will be deducted. Two (2) days late, 25% off; three (3) days late, 50% off. No homework can be turned in more than three (3) days late. Written questions and program submission have the same deadline. A total of two (2) free late days may be used throughout the semester without penalty.

Assignment grading questions must be raised with the instructor within one week after the assignment is returned.

Collaboration Policy:

You are to complete this assignment individually. However, you are encouraged to You are to complete this assignment individually. However, you are encouraged to discuss the general algorithms and ideas with classmates, TAs, and instructor in order to help you answer the questions. You are also welcome to give each other examples that are not on the assignment in order to demonstrate how to solve problems. But we require you to:

- not explicitly tell each other the answers
- not to copy answers or code fragments from anyone or anywhere
- not to allow your answers to be copied
- not to get any code on the Web

In those cases where you work with one or more other people on the general discussion of the assignment and surrounding topics, we suggest that you specifically record on the assignment the names of the people you were in discussion with.

Question 1: Simple Optimization [8]

Find the global minimum value of the following function where $x \in [-3.5, 4]$.

$$f(x) = -x^5 + \frac{5}{2}x^4 + \frac{75}{3}x^3 + 5$$

Question 2: 3-NN Classification [20]

In this given classification problem there are three classes: class1, class2 and class3. The number of feature dimensions is 2. The following is a training set for this problem space:

Class	Training Set Points
class1	(1, 1), (2,2), (1,2)
class2	(3,2.5), (2.7, 2.8), (1.8, 3.2)
class3	(0.9, 4), (2.8, 4.1)

Now use 3-NN with 1-norm as the distance measure to solve the following problems. 1-norm distance between two points x_i and x_j is defined as:

$$\delta(x_i, x_j) = \sum_{k=1}^{d} |x_{ik} - x_{jk}|; \text{ where } d \text{ is the number of features}$$

Show the steps of your calculation. If a tie between two classes arise then declare the class with the higher index as the output class.

a) Classify the following points: (1.5, 1.5), (3, 2.2), (1.3, 3).

b) Add a class2 point (2.5,2.7) to the training set. Now classify the points from question 2a) again.

Question 3: From Rules to Decision Trees [12]

In a given classification problem there are two classes only: class_1 and class_2. There are three features: A, B and C. Among these A and B are discrete features with the following possible values:

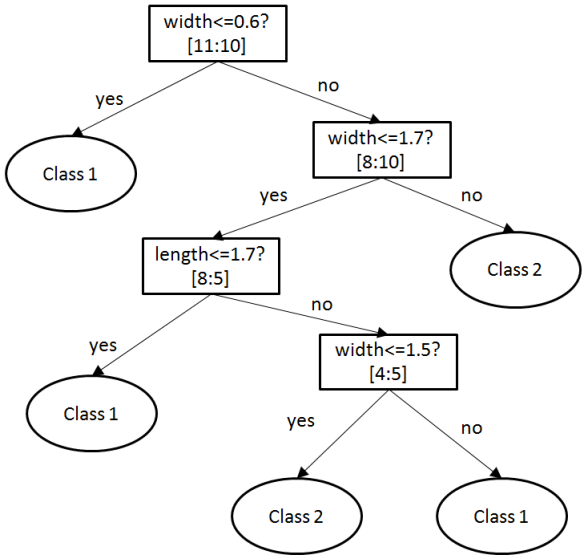
A=red, blue B=true, false

C on the other hand is a continuous feature. Given the following possible logical statement/rule, draw a corresponding decision tree.

IF((A=red **AND** C<4.5) **OR** (A=blue **AND** C $<$ 8.1 **AND** B=false))
THEN class=class_1
ELSE IF((A=red **AND** C$\geq$4.5) **OR** (A=blue **AND** C $\geq$ 8.1) **OR**
(A=blue **AND** C $<$ 8.1 **AND** B=true))
THEN class=class_2

Question 4: Pruning a Decision Tree by Hand [20]

A given classification task has two classes namely class 1 and class 2. There are only two continuous features: width and length. Given a set of training examples the following decision tree was formulated.



Here each internal node represents a condition check on one of the features. It also indicates how many training examples of each class reached that internal node. For example the [11:10] under the root means that there are 21 examples (11+10) in total. The 11 indicates that 11 of those examples belong to class 1. The 10 indicates that the remaining 10 examples belong to class 2. Leaf nodes represent the class that they belong to.

Now you are provided with the following tune set:

width	length	class
1.55	2.2	class 2
0.5	1.4	class 1
1.8	0.99	class 1
1.6	1.8	class 2
1.4	1.5	class 1

Now answer the following questions.

a) [5] What will be the accuracy of this tune set with the given decision tree?

b) [12] What will be the accuracy if one internal node is now pruned? Show the accuracy after removing each internal node one at a time.

c) [3] Now, based on these values, which internal node should be pruned first? Draw the decision tree after that particular node is pruned.

Question 5: Programming Decision Tree [40]

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 follows:

```
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);
}
```

Here, `DecisionTreeImpl(DataSet train)` learns the decision tree from the given training dataset, while `DecisionTreeImpl(DataSet train, DataSet test)` not only learns the tree but also prunes it by validating on tuning dataset. Other than these, `classify` predicts the instance's label using the trained decision tree. `rootInfoGain(DataSet train)` prints the information gain (one in each line) for all the attributes based on the train set. Finally, the root of your tree should be stored in the member `root` we have declared for you. The next sections describe other aspects of the programming part in detail.

Dataset

Our datasets come from *MONK's Problem* [1], which was the basis of a first international comparison of learning algorithms. This dataset contains 432 instances, each of which has a 6-dimensional feature and one binary class label. All attributes in the feature vector are categorical though the number of possbile values may vary from different attributes, which are specified at the header of each dataset file.

In each dataset file, there will be a header that describes the information about the dataset, which is shown as below. First, there will be several lines starting // 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 its all 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
```

[1] `https://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems`

Implementing Details

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.

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 page 15 of Prof. Zhu's slide. To finish this part, you may need to write a recursive function as the `DecisionTreeLearning` in the textbook or `buildtree` in the slide.

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 page 43 of Prof. Zhu's slide. 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`.

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.

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. Besides you need to print your results with 5 decimal places in decimal representation, so you may use `System.out.format("%.5f", arg)` for printing.

.

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 HW2 <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-testo.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 HW2 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 HW2 for you, which will load the data and pass it to the method you are implementing.

As part of our testing process, we will unzip the file you return to us, remove any class files, call `javac HW2.java` to compile your code, and call the main method of HW2 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.

Deliverables

1. Hand in (see the cover page for details) your modified version of the code skeleton we provide you with your implementation of the decision tree algorithm. Also include any additional java class files needed to run the program.

2. Optionally, in the written part of your homework, add any comments about the program that you would like us to know.