CARNEGIE MELLON UNIVERSITY

10601A/C-F18: Homework #4 - “Decision Trees”

TA: Anuva Agarwal (anuvaa@andrew.cmu.edu)
TA: Wendy Summer (wsummer@andrew.cmu.edu)

Anuva’s Office Hours:

Fri. 9/28	6:00-7:00pm	GHC 5Floor
Sat. 9/29	11:00-12:00pm	GHC 5Floor
Sun. 9/30	11:00-12:00pm	GHC 5Floor
Mon. 10/1	5:00-7:00pm	GHC 5Floor
Tues. 10/2	11:00-12:00pm	GHC 5Floor

Wendy’s Office Hours:

Wed. 9/26	12:00-1:00pm	GHC 8Floor
Thur. 9/27	3:30-4:30pm	GHC 8Floor
Fri. 9/28	10:00-11:00am	GHC 8Floor
Sun. 9/30	10:30-11:30am	GHC 5Floor
Tues. 10/2	3:30-5:30pm	GHC 8Floor

Assigned: Wednesday, September 26, 2018.
Due: 11:59:59pm on Tuesday, October 2, 2018.
You are allowed up to 15 submissions
Late Penalty: 20% per day.

Course Policies

Previously Used Assignments

Some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions, or elsewhere. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be, or may have been, available online, or from other people or sources. **It is explicitly forbidden to use any such sources, or to consult people who have solved these problems before. It is explicitly forbidden to search for these problems or their solutions on the internet.** You must solve the homework assignments completely on your own. **For programming assignments, this means you must write your programs completely by yourself, and not use any code from any source whatsoever.** I will be actively monitoring your compliance, and any violation will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

Collaboration Among Students

The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. **The actual solution must be done by each student alone.**

The purpose of programming assignments in this course is to make sure you truly understand the relevant techniques. In my more than 20 years of teaching, I have found no better way to achieve this than by having each student struggle by him/herself to implement these techniques "from scratch". For this reason, in the case of programming assignments **all code must be written by each student alone.** We will strictly enforce this policy, by carefully inspecting your code using sophisticated detection techniques. You have been warned!

The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved. Specifically, each assignment solution must include answering the following questions:

- Did you receive any help whatsoever from anyone in solving this assignment? (Yes / No). If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4").
- Did you give any help whatsoever to anyone in solving this assignment? (Yes / No). If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2").
- Did you find or come across code that implements any part of this assignment? (Yes / No) (See below policy on "found code"). If you answered 'yes', give full details (book & page, URL & location within the page, etc.).

If you gave help after turning in your own assignment and/or after answering the questions above, you must update your answers before the assignment's deadline, if necessary by emailing the TA in charge of the assignment.

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism.

Policy Regarding "Found Code"

You are encouraged to read books and other instructional materials, both online and offline, to help you understand the concepts and algorithms taught in class. These materials may contain example code or pseudo code, which may help you better understand an algorithm or an implementation detail. However, when you implement your own solution to an assignment, you must put all materials aside, and write your code **completely on your own, starting "from scratch"**. Specifically, you may not use any code you found or came across. **If you find or come across code that implements any part of your assignment, you must disclose this fact in your collaboration statement even if you didn't use it.**

Duty to Protect One's Work

Students are responsible for pro-actively protecting their work from copying and misuse by other students. If a student's work is copied by another student, the original author is also considered to be at fault and in gross violation of the course policies. It does not matter whether the author allowed the work to be copied or was merely negligent in preventing it from being copied. When overlapping work is submitted by different students, **both students will be punished.**

To protect future students, do not post your solutions publicly, neither during the course nor afterwards.

Severe Punishment of Violations of Course Policies

All violations (even first one) of course policies will **always** be reported to the university authorities, will carry **severe** penalties, usually **failure** in the course, and can even lead to **dismissal** from the university. This is not an idle threat–it is my standard practice. You have been warned!

0 Introduction & General Instructions

The goal of this assignment is for you to implement a learner, entirely from scratch. We are going to try decision trees in two domains. For simplicity, all variables are discretized into just two categories.

The programs you write will be automatically graded using the CMU Autolab system. You may write your programs in **Python**, **Java**, **C**, or **C++**. However, you should use the same language for all parts below.

Download from autolab the tar file ("Download handout"). The tar file will contain all the data that you will need in order to complete this assignment. In addition, you will also need to create the following files before you submit (see the sections below for more details):

- `inspect.{py|java|c|cpp}`
- `decisionTree.{py|java|c|cpp}`
- `Q0.txt`
- `Q1_education.txt`
- `Q1_politicians.txt`
- `Q2.txt`
- `Q3.txt`
- `Q4.txt`
- `Q5.txt`
- `Q6.txt`
- `Q7.txt`
- `Q8.txt`
- `Q9.txt`
- `Q10.txt`
- `collaboration.txt`

Do not modify the structure of the directory or rename the files therein. Answer the questions below by completing the corresponding file(s), and then compress all files listed above into a tar "hw4.tar' by running:

```
$ tar -cvf hw4.tar *.{py|java|c|cpp} *.txt
```

DO NOT put the above files in a folder and then tar that folder. You must compress the files directly into a tar file and submit it to the Autolab online.

You are allowed a maximum of 25 submissions until the deadline (see front page of this handout). **If you need an extension, please contact the TA-in-charge as soon as you are aware of such need and provide your reason.**

Besides this writeup, you are also provided with a handout tarball "hw4.tar" containing all the data you need and some example output files. To untar the file, use command

```
$ tar -xvf hw4.tar
```

This assignment contains mainly two tasks:

The first task is to predict whether a US politician is a member of the Democrat or Republican party, based on their past voting history. Attributes (covariates, predictors) are short descriptions of bills that were voted on, such as **Aid_to_nicaraguan_contras** or **Duty_free_exports**. Values are given as **'y'** for yes votes and **'n'** for no votes. Check the .csv files to see more.

The second task is to predict the final **grade** (A, not A) for high school students. The attributes (covariates, predictors) are student grades on 5 multiple choice assignments **M1** through **M5**, 4 programming assignments **P1** through **P4**, and the final exam **F**. Check the .csv files to see the attribute values.

All right, you ready to finish this assignment in an hour with only one submission? Great! Ready! Set! Implement!

Ok not so fast. Let's start with some warmup. Before you begin, think about whether decision trees are appropriate for these two tasks. Write down your thoughts in a text file **Q0.txt**: do you think a decision tree will work for the politician dataset? What about for the education dataset?

Next, inspect the training data manually; look for any unusual findings and think about which variables seem useful. In **Q1_politicians.txt** make your best guess about which variables are useful for the politician task. Do the same in **Q1_education.txt**.

We've provided you with attributes and labels split into training and testing data in files "politicians*.csv" and "education*.csv". Throughout, we show results for "example1.csv" and "example2.csv," a small, purely for demonstration version of the politicians dataset, with attributes **Anti_satellite_test_ban** and **Export_south_africa**. The format is comma separated, one row per observation, and one column per attribute. The first line of the file contains the name of each attribute, and the class is always the last column. Your program will take a training and a testing dataset as input.

Ok! Now! You are finally ready for some real warmup!

1 THE MOST AUTHENTIC WARMUP

First, let's think a little bit about decision trees. Note that, throughout this homework, we will use the convention that the leaves of the trees do not count as nodes, and as such are not included in calculations of depth and number of splits. (For example, a tree which classifies the data based on the value of a single attribute will have depth 1, and contain 1 split.) Please answer the following questions by writing in text files:

1.1 QUICK REVIEW FOR THE ID3 ALGORITHM

As addressed in the lecture, there are potentially many decision trees that express the same function. Usually we prefer the **simplest** tree (this could be the tree with least amount of nodes, least depth, or any other property one refers to as being simple).

ID3 algorithm is commonly used for generating a simple tree. The intuition behind the algorithm is to minimize the uncertainty after we query an attribute (e.g. asking an yes/no question). In practice, we minimize the uncertainty by maximizing the information gain given the queried attribute (or equivalently, minimizing the entropy conditioned on the queried attribute).

Now let's do a quick review on how the algorithm works. Given the data in the table below, where A and B take values from $\{0,1\}$, C takes value from $\{-1,0,1\}$, and output O takes value from $\{0,1\}$, please answer the following questions.

A	B	C	O
1	1	0	0
1	0	1	0
1	0	-1	0
0	0	1	1
0	1	-1	1
0	0	0	0

Q2.txt (1 pt.) What is the entropy of O, i.e. $H(O)$, in bits? (Please include only one number rounded to the third decimal place, e.g. 0.123, in your file)

Q3.txt (1 pt.) What is the entropy of O given the attribute A, i.e. $H(O|A)$, in bits? (Please include only one number rounded to the third decimal place, e.g. 0.123, in your file)

Q4.txt (1 pt.) What is the entropy of O given the attribute B, i.e. $H(O|B)$, in bits? (Please include only one number rounded to the third decimal place, e.g. 0.123, in your file)

Q5.txt (1 pt.) What is the entropy of O given the attribute C, i.e. $H(O|C)$, in bits? (Please include only one number rounded to the third decimal place, e.g. 0.123, in your file)

Q6.txt (1 pt.) Based on the results calculated above, What attribute should we query first, so that we can learn the most about the data distribution (maximize the information gain)? (Please include only one character in your file)

We then put the attribute in the root node, split the data to child nodes, and then run the same algorithm on each child node until the tree is consistent with the data (or until some preselected criteria are met).

1.2 More on the ID3 Algorithm

Consider the following data, where each variables takes value from $\{0,1\}$. Please answer the following questions.

X	Y	Z	Class
1	1	1	0
0	1	1	1
1	0	1	1
0	0	1	0
1	1	0	0
0	0	0	0

Q7.txt (2 pts.) Consider the dataset given above. Which attribute (X, Y, or Z) would a decision tree algorithm pick first to branch on, if its criteria is Mutual Information? (Please include only one character in your file)

Q8.txt (1 pts.) If the same algorithm continues until the tree is consistent with the data, what would the depth of the tree be? (Include only one number in your file)

Q9.txt (1 pts.) What depth is the shortest decision tree which is consistent with the data? (Include only one number in your file)

Now let's start doing some programming. Write a program inspect.{py|java|c|cpp} to calculate the label entropy at the root (i.e. the entropy of the labels before any splits) and the error rate (the percent of incorrectly classified instances) of classifying using a majority vote (picking the label with the most examples). You do not need to look at the values of any of the attributes to do these calculations, knowing the labels of each example is sufficient.

The command which will be used to run your program on the server, depending on the programming lanaguage that you use, is:

For Python:

```
$ python inspect.py testFileName
```

For Java:

```
$ java inspect.java testFileName
```

For C:

```
$ gcc inspect.c; ./a.out testFileName
```

For C++:

```
$ g++ inspect.cpp; ./a.out testFileName
```

The sample output, using python just as an example, for this task is:

```
$ python inspect.py example1.csv
entropy: 0.996316519559
error: 0.464285714286
```

2 THE REAL MEAT: TRAINING THE TREE

Ok now this is the real thing. In decisionTree.{py|java|c|cpp}, implement a decision tree learner with the following guidelines. (As a reference, consult Mitchell, Chapter 3, page 56.)

A few points:

- Use mutual information to determine which attribute to split on.
- Be sure you're correctly weighting your calculation of mutual information. For a split on attributeX, $I(Y;X) = H(Y) - H(Y|X) = H(Y) - P(X=0)H(Y|X=0) - P(X=1)H(Y|X=1)$. Equivalently, you can calculate $I(Y;X) = H(Y) + H(X) - H(Y,X)$.
- As a stopping rule,only split on an attribute if the mutual information is ≥0.1.
- Do not grow the tree beyond depth 2. Namely, split a node only if the mutual information is ≥ 0.1 and the node is **the root or a direct child of the root**.
- Use a majority vote of the labels at each leaf to make classification decisions. In case of a tie, you can chose to break it at random.
- Note that different datasets have different labels, take care to handle this.

Hints on getting started: write helper functions to calculate entropy and mutual information. Write a function to train a stump (tree with only one level). The correct tree and output format for the example data are shown below, where we are training on example1.csv and testing on example2.csv. For the politician data, use **"+"** for Party = "democrat" and **"-"** for Party = "republican". With the education data, use **"+"** for final grade = "A" and **"-"** for final grade = "not A". Don't worry about the order in which you list the left and right children, the autograder will take care of it. Your program should be named decisionTree and take two arguments, a training file, and a test file.

For Python:

```
$ python decisionTree.py trainFileName testFileName
```

For Java:

```
$ java decisionTree.java trainFileName testFileName
```

For C:

```
$ gcc decisionTree.c; ./a.out trainFileName testFileName
```

For C++:

```
$ g++ decisionTree.cpp; ./a.out trainFileName testFileName
```

Sample output using Python:

```
$ python decisionTree.py example1.csv example2.csv
[15+/13-]
Anti_satellite_test_ban = y: [13+/1-]
| Export_south_africa = y: [13+/0-]
| Export_south_africa = n: [0+/1-]
Anti_satellite_test_ban = n: [2+/12-]
| Export_south_africa = y: [2+/7-]
| Export_south_africa = n: [0+/5-]
error(train): 0.0714285714286
error(test): 0.142857142857
```

However, you should be careful that the tree might not be full. Here's what happens when we train on example2.csv and test on example1.csv:

```
$ python decisionTree.py example2.csv example1.csv
[13+/15-]
Anti_satellite_test_ban = y: [9+/0-]
Anti_satellite_test_ban = n: [4+/15-]
| Export_south_africa = y: [4+/10-]
| Export_south_africa = n: [0+/5-]
error(train): 0.142857142857
error(test): 0.107142857143
```

The numbers in brackets give the number of positive and negative labels from the training data in that part of the tree. The last two numbers are the error rate on the training data and the error rate on the testing data. Make sure your answers are accurate to within 0.01.

3 EVALUATION

Train and test a decision tree for the politician dataset and the education dataset. Which is more accurate on the training data? Which is more accurate on the testing data? Write down your observations in **Q10.txt**.

In addition to the politician and education datasets, autolab will test your code on two more datasets, which will not be shown to you. One set contains information about various cars, and whether or not consumers decided to buy them. The other contains data about songs, and whether or not they became top hits. The data will be in .csv files similar to the ones provided, again with the class as the last column. Shown below are the attributes and the values they can take that you should include in your code:

Music data:

- `Attribute:year('before1950'or'after1950')`
- `Attribute:solo('yes'or'no')`
- `Attribute:vocal('yes'or'no')`
- `Attribute:length('morethan3min'or'lessthan3min')`
- `Attribute:original('yes'or'no')`
- `Attribute:tempo('fast'or'slow')`
- `Attribute:folk('yes'or'no')`
- `Attribute:classical('yes'or'no')`
- `Attribute:rhythm('yes'or'no')`
- `Attribute:jazz('yes'or'no')`
- `Attribute:rock('yes'or'no')`
- `Class Label:hit('yes'or'no')`

Cars data:

- `Attribute:buying('expensive'or'cheap')`
- `Attribute:maint('high'or'low')`
- `Attribute:doors('Two'or'MoreThanTwo')`
- `Attribute:length('morethan3min'or'lessthan3min')`
- `Attribute:person('Two'or'MoreThanTwo')`
- `Attribute:boot('large'or'small')`
- `Attribute:safety('high'or'low')`
- `Class Label:class('yes'or'no')`

Please ensure your solution can handle data with these values, and use "yes" for "+".

4 Some Logistical Information

Please ensure you have completed the following files for submission and follow the instructions described in the "General Instructions" section to submit:

```
collaboration.txt
Q0.txt
Q1_education.txt
Q1_politicians.txt
Q2.txt
Q3.txt
Q4.txt
Q5.txt
Q6.txt
Q7.txt
Q8.txt
Q9.txt
Q10.txt
inspect.{py|java|c|cpp}
decisionTree.{py|java|c|cpp}
```

Note: Please make sure the programming language that you use is consistent within this assignment (e.g. don't use C++ for inspect and Python for decisionTree).