CIS 3700 (Winter 2018): Introduction to Intelligent Systems

ASSIGNMENT 3

Instructor: Y. Xiang | Assigned: Thur, Mar 15, 2018 | Due: Thur, Mar 29, 2018

Problems (Total: 36 marks)

1. (4 marks) (Types of machine learning) Classify following learning scenarios into supervised, unsupervised, and reinforcement learning. Briefly explain the reason of each classification.

   (a) A parrot is taught to say different things to different people.

   (b) A scientist investigates what causes global warming.

   (c) A biology student learns to identify plants.

   (d) A bowling player trains to play indoor ten-pin bowling competitively.

2. (3 marks) (Decision tree as function) While driving on the fastest lane at a three-lane highway, the truck behind wants to pass by tailing closely. Draw a decision tree for deciding whether to turn immediately into the middle lane. Define the meaning of each attribute clearly.

3. (3 marks) (Decision tree and Boolean function) Let a Boolean function $f(x)$ be defined as $(R \vee \neg S) \Rightarrow (P \wedge Q)$, where x is the Boolean vector $x = (R, S, P, Q)$. Draw a decision tree for the function $f(x)$, subject to the following requirements.

   The root node (attribute) should be P. Attribute S cannot be the parent node for Q or R. Attribute R cannot be the parent node for Q. For each non-leaf node, label the left outgoing link by T (true), and the right outgoing link by F (false). The number of nodes should be as fewer as possible.

4. (3 marks) (Information measure) Answer the following questions.

   (a) A query q has possible answers V_1, V_2 and V_3 of probabilities $P(V_1) = 0.2$, $P(V_2) = 0.3$ and $P(V_3) = 0.5$, respectively. Before knowing the answer w to the query, what is the amount of information that the answer w contains?

   (b) A query y has possible answers yes or no. A training set on y contains 24 positive examples and 6 negative examples. What is the amount of information that the answer w to y contains?

5. (3 marks) (Attribute test) A function $f(x)$ has possible values Y and N, where x is a vector of attributes. One attribute in x is A, with possible values A_1 and A_2.

   A training set for $f(x)$ has 64 positive and 24 negative examples. Among them, 30 examples have $A = A_1$ and the remaining 58 examples have $A = A_2$. For examples where $A = A_1$, 25 of them are positive and 5 of them are negative. For examples where $A = A_2$, 39 of them are positive and 19 of them are negative.

   (a) Consider the task to classify an object described by attribute vector x. How much information is needed to classify x, based on frequencies of $f(x)$ values in the training set only?

   (b) What is the value of $Remainder(A)$?

   (c) What is the value of $Gain(A)$?

6. (20 marks) (Decision tree learning) Develop a decision tree learning agent *DTLearn* in Java.

   [**Input**] The input to *DTLearn* consists of 2 text files: $SchemeFile$ and $DataFile$. Together, they specifies a training set of examples. Each example $(x, f(x))$ consists of a vector x of attributes and a function value $f(x)$.

   $SchemeFile$ contains general information about the attributes and the function, as exemplified below. The 1st line of $SchemeFile$ is a count N of the paragraphs that follow. Each subsequent paragraph defines one attribute, and the last paragraph defines the function. Adjacent paragraphs are separated by a single blank line. Hence, the count N is the number of attributes plus one. In the training set below, each x consists of 4 attributes: P, Q, R, and S. Each attribute paragraph starts with the name of the attribute, followed by the number of its possible values, followed by the names of values. The function paragraph has the similar format.

Example $SchemeFile$:

```
5

P
2
F T

Q
2
F T

R
2
F T

S
2
F T

Func
2
F T
```

Example $DataFile$:

```
P Q R S Func
F T F F T
F T F T T
F T T F T
F T T T T
T F F F T
T F F T T
T T F F T
T T F T F
```

$DataFile$ contains all examples in the training set, as exemplified above. Its first line lists attribute and function names, in the same order as they appear in $SchemeFile$. Each subsequent line contains one example $(x, f(x))$, specified by values of attributes in the same order as the first line, followed by the function value $f(x)$. The values are separated by white spaces.

[**Compilation and execution**] The program should be compiled with all source code in dir \3700A3FirstName and issuing the command below at \3700A3FirstName, where FirstName is the first name of the student.

```
javac DTLearn.java
```

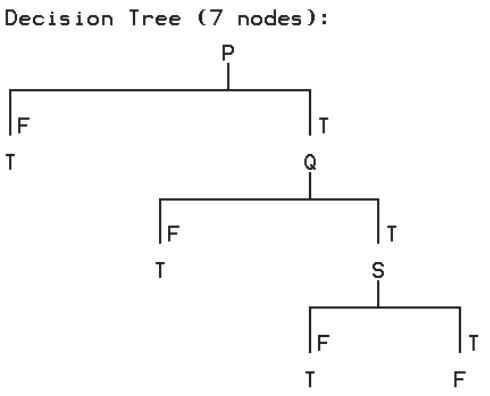
The agent should be started by the following command issued at \3700A3FirstName, where there should be no restriction to file paths $SchemeFile$ and $DataFile$.

```
java DTLearn SchemeFile DataFile
```

[**Program output**] As learning proceeds, echoes should be made as illustrated below. (1) In each round of attribute test, where all remaining attributes are tested, echo each attribute tested and the gain obtained. (2) At end of each round of attribute test, echo the attribute selected. (3) At end of learning, plot the decision tree,

Example echo (left) and tree plot (right):

```
Learning starts:
  Test P: gain = 0.1379259
  Test Q: gain = 0.056047797
  Test R: gain = 0.056047797
  Test S: gain = 0.1379259
            Select attribute P
  Test Q: gain = 0.31127918
  Test R: gain = 0.0
  Test S: gain = 0.31127918
            Select attribute Q
  Test R: gain = 0.0
  Test S: gain = 1.0000035
            Select attribute S
```



The decision tree should be drawn as a tree with each non-leaf node labeled by an attribute name, each link outgoing from an attribute labeled by a value of the attribute, and each leaf node labeled by a function value. A simple way (although not perfect) to plot the decision tree in plaintext is the following:

```
P
0

0           0
|           |
|F          |T
v           v
T           Q
1           2

2           2
|           |
|F          |T
v           v
T           S
3           4

4           4
|           |
|F          |T
v           v
T           F
5           6
```

An index is plotted under each node, e.g., 0 under P and 2 under Q. At the top of each link, the index of its endpoint is plotted, e.g., the 2nd and 3rd 0. Print the plot on paper, and manually connect identical indexes by straight lines, e.g., connecting 1st 0 to 2nd and 3rd. The result is equivalent to the tree in the above echo.

[**Class implementation**] The implementation should consist of the following Java classes.

(a) Attribute: It maintains name of an attribute and a list of possible values.

(b) Scheme: It maintains a list of attributes and the function. Each attribute or the function is an Attribute object. It should contain method(s) to load $SchemeFile$ so that a Scheme object can be instantiated.

(c) Example: It maintains attribute values and function value of an example. For efficiency, each value should be represented as an index into the list of possible values of the corresponding attribute (or function). Hence, an example is maintained as an array of integers, representing attribute values ordered identically as in Scheme, followed by function value.

(d) Sample: It maintains a set of training examples, each of which is an Example object. It should contain methods corresponding to $getInfo()$, $getRemainder()$, and $getAttribute()$ as discussed in lectures.

(e) Node.java: A node in a decision tree. It should contain the label of the node, a pointer to the node's parent, the label for the link incoming from the parent, and additional information that helps to plot the node.

(f) DTLearn: This is where the agent starts. It implements the main learning algorithm from lectures.

(g) UTIL: This class may contain any generic static utility methods, such as loading strings from files, testing membership of an element in a list, and so on.

[**Implementation and test cases**] Debug with simple $SchemeFile$ and $DataFile$ first. Some test files are available in the course website.

Implement your agent with necessary error checking to avoid garbage-in and garbage-out. For example, if attributes in $DataFile$ differ from those in $SchemeFile$, or differ in order, or attribute values in examples are undefined in $SchemeFile$, the agent should echo error and terminate.

[**Submission**] Place all Java source files in dir `\3700A3LnameFname`, where Lname is your last name and Fname is your first name. Compress the files into a single file 3700A3LnameFname.zip and submit the zip file.