CSI3120: Programming Language Concepts

Fall 2014

Assignment 1

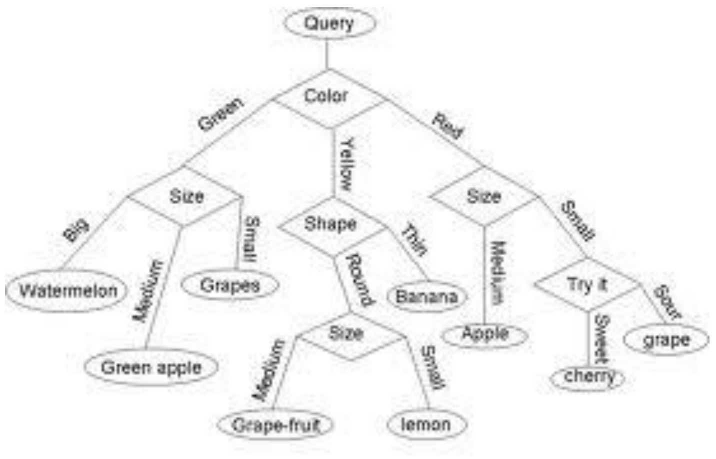
Due Date: October 2, 2014 Due Time: 23:59

This assignment is intended as a review of Scheme and Prolog programing. In addition, you will have to answer a couple of written questions about the history of programing languages and programing language evaluation.

Because Scheme and Prolog were languages created for Symbolic Artificial Intelligence (AI) tasks, the two programming questions will be AI questions. In particular, you will be introduced to decision trees. The Scheme question will have you program a simple (though perhaps not very effective[1]) decision tree induction algorithm in Scheme while the second question will have you turn medical knowledge expressed in the form of a decision tree into an interactive expert system in Prolog.

What is a decision tree?

The figure below is an example of a decision tree[2]:



This decision tree helps people identify fruit according to their physical properties. The properties are indicated in diamond shapes. Labeled arcs emanate from each of the diamonds, which represent the possible values of each property (e.g., in this particular environment, the property “Color” can take values “Green”, “Yellow” or “Red”). The leaves of the tree are indicated in oval shapes and, in this case, represent the identity of the fruit (“Watermelom”, “Grapefruit”, etc…). More generally, properties are called: “features” or “attributes” and identities are called: “classes” or “categories”. Given a sufficient

[1] If this interests you and you would like to be introduced to more effective techniques, then please, come and talk to me. This is what I do when I don’t teach csi3120!!!!

[2] Source: https://www.projectrhea.org/rhea/index.php/Lecture_21_-_Decision_Trees(Continued)_OldKiwi (retrieved on Sept 8, 2014 at 9:48am)

amount of data, good decision trees can be automatically induced from instances described in terms of attribute-value pairs and an associated class. In the tree above and examples of instances are:

"Color=Yellow, Size=D/C, Shape=Thin, Try= D/C, Class= Banana

Color = Red, Size =Small, Shape = D/C, Try = Sour, Class = Grape "

The process of inducing a decision tree that way belongs to the AI subfield of machine learning or data mining. In Question 1, you will be inducing a decision tree. There are many better decision tree inducing algorithms than the one I propose you use, but I wanted to keep your Scheme programing assignment short and sweet! ☺ In Question 2, you will take a human-induced decision tree (akin to a conversation you may have had with a human expert, in this case a medical doctor), extract the essential knowledge from the tree in order to encode it in Prolog, and build an interactive system that asks patients what their symptoms are and in doing so diagnose their disease. This process belongs to the AI subfield of Expert-Systems.

Problem 1: Scheme

The purpose of this problem is to use Scheme to induce a number of random binary decision trees, test them on a testing set and select the one that performs best on the testing set.

In order to build the trees, you do not need to know the data. You are only given the number of features in the data set (10 in this particular problem) and know that each feature can take one of two values: true or false. You must choose features, at random, to place in the tree. You should also generate stopping conditions at random. The only restriction is that the same feature cannot appear multiple times on any of the tree paths.

Here is some code that will allow you to generate a random number in the "scm" implementation of Scheme.

```
(define random
 (let ((a 69069) (c 1) (m (expt 2 32)) (seed 19380110))
   (lambda new-seed
    (if (pair? new-seed)
        (set! seed (car new-seed))
        (set! seed (modulo (+ (* seed a) c) m)))
    (/ seed m))))
```

```
(define (randint . args)

  (cond ((= (length args) 1)

         (floor (* (random) (car args))))

        ((= (length args) 2)

         (+ (car args) (floor (* (random) (- (cadr args) (car args))))))

        (else (error 'randint "usage: (randint [lo] hi)"))))
```

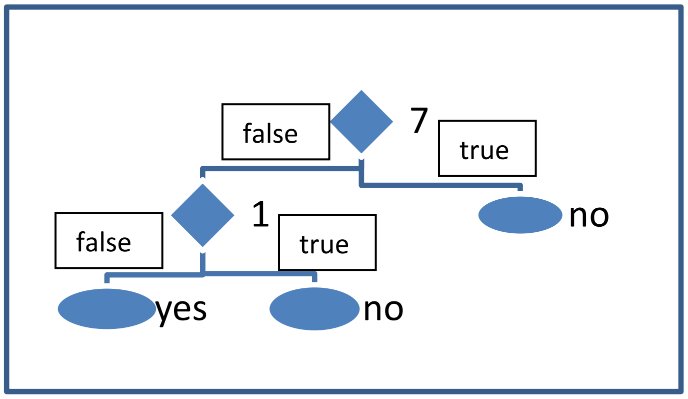
(randint -5 10) generates a random number between -5 and 10.

Please implement the following function: if the number generated by "(randint -5 10)" lies in interval [-5, -2), stop the generation process and return class "no", if it lies in interval [-2, 0], stop the generation process and return class "yes". If the number *n* generated by "(randint -5 10)" is greater than 0 and if it is not in the path of the going from the root of the tree to the current location, then the current node takes value *n*, and the function is called recursively twice: once to build the left subtree of the current node and once to build its right subtree.

Here is an example of how a tree is built:

1. A feature is chosen randomly by (randint -5 10): 7
2. 7 becomes the root of the tree. The function is called recursively twice, on the left and on the right sides (steps 3 and 4, respectively) .
3. The function is called on the left side. The value returned by (randint -5 10) is 1. The function is called recursively twice, on the left and the right sides (steps 5 and 6, respectively).
4. The function is called on the right side. The value returned by (randint -5 10) is -3. The function returns: "no" and stops.
5. The function is called on the left side. The value returned by (randint -5 10) is 7. 7 is already on the path. (randint -5 10) is called again. The value returned by (randint -5 10) is 0. The function returns: "yes" and stops.
6. The function is called on the right side. The value returned by (randint -5 10) is -3. The function returns: "no" and stops.

Note: the convention we use is that the left branch of a node corresponds to the feature value "false" of the feature represented by that node and that the right branch of a node corresponds to the feature value "true" of the feature represented by that node. The tree constructed in this example looks as follows:



You will also need to generate three more functions:

1. A function that takes a tree and a test example (e.g., (#f #t #f #f #t #t #t #f #t #f yes)) and returns the predicted class according to the tree. (The class shown in the example is the "true" class). For example, the tree above returns "no" on the example since feature 7 has value #t which leads to "no"). The tree is therefore making an error on this example.
2. A function that takes a tree and a database (list) of examples and returns a score for that tree on the database (e.g., ((#f #t #f #f #t #t #t #f #t #f yes) (#f #f #f #t #t #t #f #f #t #f yes) (#t #t #t #f #t #t #f #t #f #f no)). This function returns 2 on this example since the class of the last two examples were predicted correctly by the tree.
3. A function that generates many trees (100), scores them all, and returns the best one along with its score.

Problem 2: Prolog

For this question, you are given the medical decision tree found at:

http://familydoctor.org/familydoctor/en/health-tools/search-by-symptom/cold-flu.html

You will need to encode the tree in Prolog rules, paying particular attention to the fact that each disease must include certain sets of symptoms and exclude others. For example, in the case of the Flu, you must include the symptoms of 1. and 3., but exclude those of 2. (meaning that "nasal_drainage" is now possible). In certain cases, there are various combinations of and/or relations between symptoms. In one case, we have the expression: "a combination of symptoms including <list>". Please, interpret this as "at least two from <list>". All the relations must be coded properly. Please, make sure that your logic is correct. For example, to have strep throat, you require a headache and a sore throat. You can still have a headache or a sore throat if you do not have strep throat. However, you can't have both. In 3., the symptoms must appear suddenly. "suddenly" is not a symptom. Treat it differently.

This assignment is typical of projects that build an expert system in Prolog: In particular, the rules were generated by human beings. They are easily interpreted by a human, but less so by a computer. It is your job to translate them.

A few examples (as you can see, you need to pay good attention to details):

hasSymptoms(nathalie, gradual, [headache, fever, sore_throat]).

```
?- hasDisease(nathalie, Disease).

Disease = maybe_strep_throat ;

false.

hasSymptoms(nathalie, gradual, [headache, fever, nasal_drainage, sore_throat] ).

?- hasDisease(nathalie, Disease).

false.

hasSymptoms(nathalie, sudden, [fever, muscle_aches, runny_nose] ).

?- hasDisease(nathalie, Disease).

Disease = maybe_flu ;

false.

hasSymptoms(nathalie, sudden, [fever, muscle_aches, headache, runny_nose, sore_throat] ).

?- hasDisease(nathalie, Disease).

false.
```

Question 1: History of Programing Languages Read chapter 2 before answering.

A. What languages (list the main ones along with their general purposes and characteristics) existed at the time when Lisp was created?
B. Same as above but replace "Lisp" by "Prolog"
C. In each case, give a general discussion of why researchers felt it was necessary to create new "odd" languages such as Lisp and Prolog. Is the world better for it?

Question 2: Language Evaluation

Think (carefully) about how you would implement the programs for Problem 1 and 2 using Java. [You may want to start drafting the programs in Java, though you do not need to implement them. I just want you to think about how to do this, deeply].

In the context of each of these problems, what would be the pros and cons of using Lisp (Scheme), Prolog and Java? [Please, make sure you answer this question using the language evaluation criteria discussed in class] Was it wise for me to ask you to program Problem 1 in Scheme and Problem 2 in Prolog? Would the opposite have been better? Should I have forgotten about Scheme and Prolog and have asked you to do it all in Java?

Please, present your arguments carefully and feel free to disagree with me. You won't lose points for that as long as your arguments are well justified!