CSCI 2235
Programming Project 02 - Classification Trees

Assigned: October 08, 2019
Due: November 01, 2019 @ 23:00h

Purpose

- Work with trees
- Basic introduction Machine Learning

Problem Statement

You are going to build an application that learns from its user to identify and classify things. The example I did is classifying animals. You can do that as well, or you can think up your own application. I will first describe animal classification for people that don't want to come up with their own application.

I can classify animals by their characteristics (it is in the water, it doesn't have legs, it has gills, it is a fish). You can imagine representing these characteristics in a tree like structure where internal nodes are characteristics (furry, 2 legs, lives in water) and leaf nodes are animals (dog, human, fish). The easiest (and recommended) way is to have your classification tree be a binary tree (see Figure 1 below). Thus, each characteristic can be answered as yes or no (i.e., each internal node has exactly two children). Your application will start with a very basic tree you hard code in. That initial tree will have one internal node (the root) and two children. The user will then try to identify an animal by answering queries from your application. If your application eventually settles on the correct animal, great, if not, it will inquire as to the differentiating trait between the animal being identified and the leaf it had settled on. **NOTE: You must provide the information that I provide in my output in this circumstance (e.g., "I don't know any furry, not squeaky, bipedal animals that aren't a human.")**. This amounts to your position in the classification tree. It will then create a new internal node and new leaf node to represent that trait as well as the new animal. Below you will find a sample run for my program. You must also be capable of saving your tree to a file when the program exists and be capable of reloading the tree when it begins again. This enables your AI to keep learning across sessions.

You do not need to do animal classification. If you can think of a better idea, go for it. You must replicate the functionality described as shown in the output. As a bare minimum, you need a classification tree, you need to print out the position when adding a new node, and you need to save to and read from a file. The same grading criteria holds for non-animal and animal classification. If you have any questions, run ideas past me.

The binary tree will be implemented in a class called **LinkedBinaryTree<E>** which implements the interface **BinaryTree<E>**. To process traversing the nodes of the tree you will need to implement the following traversal methods:

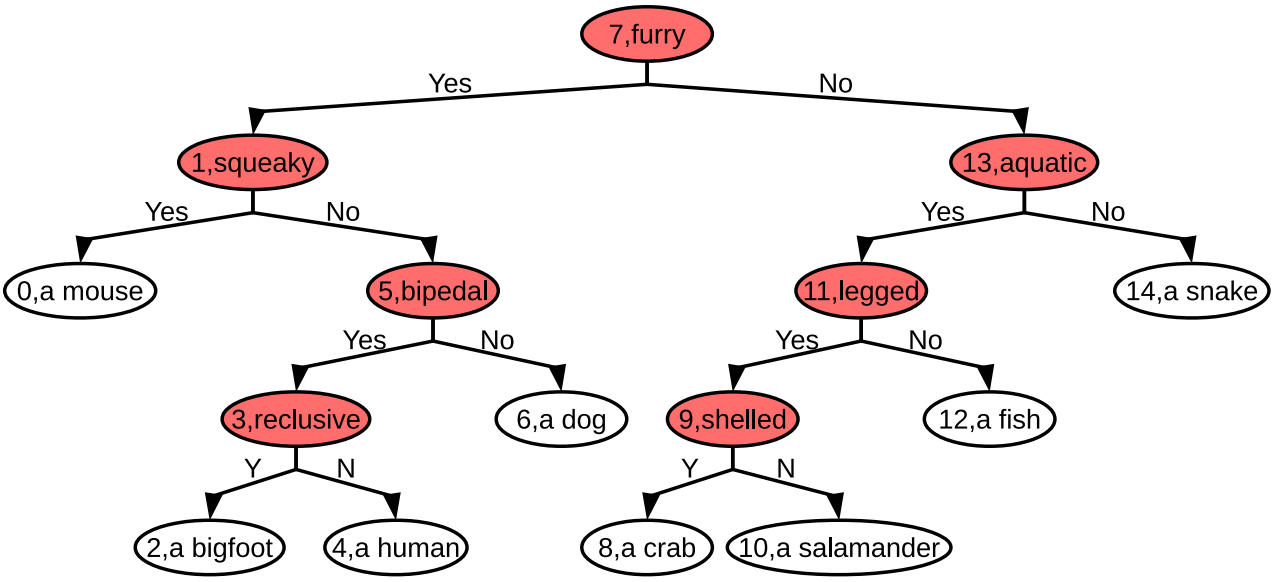

Figure 1: An example classification tree.

- DepthFirst and its strategies:
  - PreOrder
  - PostOrder
  - InOrder
- BreadthFirst

Each of these traversals will be implemented in the **edu.isu.cs2235.traversals** package. Each Traversal must extend the **AbstractTraversal** class and implement the methods (not defined in AbstractTraversal) defined in the **TreeTraversal** interface.

Also note that I have provided you with the **EnumeratedSaveCommand** and **EnumerationCommand** classes both of which extend the **TraversalCommand** class. These are found in the **edu.isu.cs2235.traversals.commands** package. Traversal commands are provided to a traversal through the **setCommand()** method. As the traversal processes the tree (visits each node) the command's **execute()** method should be called to execute it's functionality:

- **EnumerationCommand** – Sets the number for a given node, based on the order of traversal. This command when used from a traversal will effect the renumbering of nodes in the tree. This implies one of two things, either the node holds both a number and a value relating to classification, or the node holds some class which contains both pieces of data. The numbering should reflect the traversal and the ordering that presents.
- **EnumeratedSaveCommand** – Used as the command for a traversal which will write the contents of a visited node to a file. This generates the file which when used should write out the file that can be used to save the tree for later use. (See the example file below). When writing the file out each line of the file must be in the following format:
  - [parent num]:[datum num]:[side][datum prompt]
  - Where:
    - Square brackets are not included
    - parent num: is the number of the parent node, as assigned by **EnumerationCommand**
    - datum num: is the number of the node being visited (Root's parent is null and should have a -1)

    * side: is a indicator of whether the current node is the left or right side of its parent (Root is always considered to be the right side of the parent), and is marked with a `l` or a `r`.
    * datum prompt: the value used to prompt the user
    * Example: `3:4:r:a human` – "a human" is node number 4 which is the right child of node number 3

As part of your LinkedBinaryTree class you will need to implement a **BinaryTreeNode** class which implements the **Node** interface.

Assignment

1. You will need to create a class to manage the questioning and to build the tree. I have included a class called `ClassificationTree`, but you can rename it to whatever suits your needs given your choice in the classification type.
2. In the **ClassificationTree** class there is a field called **tree** which is of type **BinaryTree<Datum>**, do not remove this (though you may change the type as it suits you), as this is the tree backend for your code. You will need to build this tree in order to complete the program.
3. There are several save methods and a load method that are used to save and retrieve your tree from a file. I have implemented these methods, but they rely on you implementing the following components:
    - Your implementations of the preorder, inorder, postorder, and breadthfirst traversals.
4. See below for an example of the Input/Output file format for an example Console Output for a run of the program.

Submission

Zip compress your project directory into a file named: [first_name]_[last_name].zip (Note: square brackets not included). Submit the file to Moodle by the deadline noted above.

Grading (50 points)

- 5 Points – **LinkedBinaryTree** implemented correctly
- 5 Points – **BreadthFirstTraversal** implemented correctly
- 5 Points – **InOrderTraversal** implemented correctly
- 5 Points – **PostOrderTraversal** implemented correctly
- 5 Points – **PreOrderTraversal** implemented correctly
- 5 Points – Animals that are in the tree are successfully found
- 5 Points – An internal node and leaf node are correctly added when a new animal is encountered
- 5 Points – The position in the tree is printed when a new node is to be inserted.
- 5 Points – The interface is clean and presentable. I understand that readability may depend on user input, but SOME input should make it very readable.
- 5 Points – The saving and loading from a text file works.

Hints

1. There are efficient ways of doing things and inefficient ways. Having a good plan together before you start writing code. Also, I use both a queue and stack at various points in the code. Keep those tools in mind as you design a solution.
2. Note: If at any point you want to use a queue, stack, list, ..., you are allowed to use either your prior developed implementations or the ones provided by Java.
3. The traversals used to save and number the tree can make it more or less difficult to read the tree in from the file. Keep that in mind.

Example of Input/Output File

The input/output file format is [parent num]:[datum num]:[l **for** left side of parent, r **for** right side of parent, root is always r][datum prompt]

```
-1:7:r:furry
7:1:l:squeaky
7:13:r:aquatic
1:0:l:a mouse
1:5:r:bipedal
13:11:l:legged
13:14:r:a snake
5:3:l:reclusive
5:6:r:a dog
11:9:l:shelled
11:12:r:a fish
3:2:l:a bigfoot
3:4:r:a human
9:8:l:a crab
9:10:r:a salamander
```

Example Console Output

```
run:
Provide a file for loading.
Filename: a

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal a dog? (Y/N) > Y
Good.

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal a dog? (Y/N) > N
I 'dont know any furry animals that 'arent a dog.
What is the new animal? > mouse
What characteristic does a mouse have that a dog does not? > squeaky

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
```

```
Is this animal squeaky? (Y/N) > N
Is this animal a dog? (Y/N) > N
I 'dont know any furry, not squeaky animals that 'arent a dog.
What is the new animal? > human
What characteristic does a human have that a dog does not? > bipedal

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal a human? (Y/N) > N
I 'dont know any furry, not squeaky, bipedal animals that 'arent a human.
What is the new animal? > bigfoot
What characteristic does a bigfoot have that a human does not? > reclusive

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > N
Is this animal a snake? (Y/N) > N
I 'dont know any not furry animals that 'arent a snake.
What is the new animal? > fish
What characteristic does a fish have that a snake does not? > aquatic

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > N
Is this animal aquatic? (Y/N) > Y
Is this animal a fish? (Y/N) > N
I 'dont know any not furry, aquatic animals that 'arent a fish.
What is the new animal? > salamander
What characteristic does a salamander have that a fish does not? > legged

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > N
Is this animal aquatic? (Y/N) > Y
Is this animal legged? (Y/N) > Y
Is this animal a salamander? (Y/N) > N
I 'dont know any not furry, aquatic, legged animals that 'arent a salamander.
What is the new animal? > crab
What characteristic does a crab have that a salamander does not? > shelled

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > N
Is this animal aquatic? (Y/N) > Y
Is this animal legged? (Y/N) > Y
Is this animal shelled? (Y/N) > Y
Is this animal a crab? (Y/N) > Y
Good.

Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > Y
Is this animal reclusive? (Y/N) > N
Is this animal a human? (Y/N) > Y
Good.
```

```
Do you have another animal to identify? (Y/N) > Y
Is this animal furry? (Y/N) > Y
Is this animal squeaky? (Y/N) > N
Is this animal bipedal? (Y/N) > N
Is this animal a dog? (Y/N) > Y
Good.

Do you have another animal to identify? (Y/N) > N
Enter a file name to save the tree to:
Filename: tree.txt
```