Assignment 2
CS 6375: Machine Learning

Fall 2015

Due: 11:59 p.m., Sunday September 27 via eLearning

Inducing Decision Trees

In this homework you will implement and test the decision tree learning algorithm (See Mitchell, Chapter 3).
You can use one of the following approaches:

- R package **rpart**. You can read about it here - http://www.statmethods.net/advstats/cart.html.
  A good example is available http://www.math.uwaterloo.ca/ rwoldfor/software/R-code/iris-rpart.R.
- Weka. You can read about decision trees using Weka here and at other resources online.
- 50% Bonus if you write your own code. Your C/C++/Java implementations should compile on Linux gcc/g++ compilers.

- You will be using the following datasets to test your code:
  - Dataset 1 - Uploaded on eLearning. It's divided into 3 parts: training (for learning the model), validation (for pruning), test (for testing performance).
  - Dataset 2 - Uploaded on eLearning. It's divided into 3 parts: training (for learning the model), validation (for pruning), test (for testing performance).

If you use any package:

For both the datasets do the following

- Create a decision tree model.
- You should find a way to print a summary of your model. For example, the package **rpart** has the *printcp* function.
- Plot your model with sufficient details. For example, **rpart** has the *plotcp* and *plot* functions.
- Find a way to prune your model. For example, **rpart** has the *prune* function. You should also plot a post-pruned tree.
  Note: It is up to your judgement to find good parameters for pruning. You do not need to use the pruning dataset.
- Test your model using the test dataset. In **rpart**, you can use the function *predict*.

What to Turn in

- A README file specifying which package you used and the various parameters used.
- Your source file
- The output including the plots and model summary.

If you write your own code:

- Implement the decision tree learning algorithm. As discussed in class, the main step in decision tree learning is choosing the next attribute to split on.
  You can choose **any one** of the two heuristics below for selecting the next attribute.

  1. Information gain heuristic (See Class slides, Mitchell Chapter 3).
  2. Variance impurity heuristic described below.

     Let K denote the number of examples in the training set. Let K_0 denote the number of training examples that have $class = 0$ and K_1 denote the number of training examples that have $class = 1$. The variance impurity of the training set S is defined as:

     $$VI(S) = \frac{K_0}{K}\frac{K_1}{K}$$

     Notice that the impurity is 0 when the data is pure. The gain for this impurity is defined as usual.

     $$Gain(S, X) = VI(S) - \sum_{x \in Values(X)} \Pr(x) VI(S_x)$$

     where X is an attribute, S_x denotes the set of training examples that have $X = x$ and $\Pr(x)$ is the fraction of the training examples that have $X = x$ (i.e., the number of training examples that have $X = x$ divided by the number of training examples in S).

- Implement the post pruning algorithm given below as Algorithm 1 (See also Mitchell, Chapter 3).

- Implement a function to print the decision tree to standard output. We will use the following format.

```
wesley = 0 :
| honor = 0 :
| | barclay = 0 : 1
| | barclay = 1 : 0
| honor > 0 :
```

```
| | tea = 0 : 0
| | tea = 1 : 1
wesley = 1 : 0
```

According to this tree, if wesley = 0 and honor = 0 and barclay = 0, then the class value of the corresponding instance should be 1. In other words, the value appearing before a colon is an attribute value, and the value appearing after a colon is a class value.

- Once we compile your code, we should be able to run it from the command line. Your program should take as input the following six arguments:

```
.\program <L> <K> <training-set> <validation-set> <test-set> <to-print>
L: integer (used in the post-pruning algorithm)
K: integer (used in the post-pruning algorithm)
to-print:{yes,no}
```

It should output the accuracies on the test set for decision trees constructed using any of the two heuristics as well as the accuracies for their post-pruned versions for the given values of L and K. If to-print equals yes, it should print the decision tree in the format described above to the standard output.

What to Turn in

- Your code and a Readme file for compiling the code.

On the two datasets provided:

- Report the accuracy on the test set for decision trees constructed using any one of the two heuristics mentioned above.
- Choose 10 suitable values for L and K (not 10 values for each, just 10 combinations). For each of them, report the accuracies for the post-pruned decision trees constructed using any one of the two heuristics.

Additional Questions

Answers to the following questions will not be graded. However, they will serve as practice questions for your midterm/final.

1. Represent the following functions using decision trees: $Y = A \vee B$, $Y = (A \vee B) \wedge (B \vee C) \wedge (A \vee C)$ and $Y = (A \vee B) \wedge \neg A \wedge \neg B$

2. Compute the information gain of Attribute A.

A	B	C	Class
0	0	0	+
0	0	1	+
0	1	0	+
0	0	0	-
0	1	0	-
1	0	0	+

3. What will be the training set error for a decision tree classifier on this training data using only the attribute A.

4. True or False? Can you always convert any arbitrary non-binary decision tree to a binary decision tree. A binary decision tree is a decision tree in which all splits are binary (i.e., branching factor = 2). Justify your answer.

5. Consider two decision trees that perfectly represent the target function. One has maximum depth h and other has maximum depth g, where $h > g$. Which decision tree will you prefer and why?

   (a) The tree with maximum depth h

   (b) The tree with maximum depth g

   (c) I need more information.

Algorithm 1: Post Pruning

Input: An integer L and an integer K
Output: A post-pruned Decision Tree
begin
  Build a decision tree using all the training data. Call it D;
  Let $D_{Best} = D$;
  for $i = 1$ *to* L **do**
    Copy the tree D into a new tree D';
    M = a random number between 1 and K;
    for $j = 1$ *to* M **do**
      Let N denote the number of non-leaf nodes in the decision tree D'. Order the nodes in D' from 1 to N;
      P = a random number between 1 and N;
      Replace the subtree rooted at P in D' by a leaf node.
      Assign the majority class of the subset of the data at P to the leaf node.;
      /* For instance, if the subset of the data at P contains 10 examples with $class = 0$ and 15 examples with $class = 1$, replace P by $class = 1$ */
    end
    Evaluate the accuracy of D' on the validation set;
    /* accuracy = percentage of correctly classified examples */
    if D' *is more accurate than* D_{Best} **then**
      $D_{Best} = D'$;
    end
  end
  return D_{Best};
end