Machine Learning 15-681,15-781
Tom Mitchell, Fall 1998

Homework Assignment 2
Due at beginning of class on October 1, 1998

Overview: This assignment gives you an opportunity to experiment with a decision tree learning program. You are asked to experiment with the simple *PlayTennis* learning data described in Chapter 3 of the textbook, and then to experiment with a larger set of data describing voting records of congressmen/women from the U.S. House of Representatives.

Code setup: Copy all files from the directory `/afs/cs.cmu.edu/project/theo-3/mlc/hw2/` to your own directory. Now connect to your directory and type `gmake` to compile the code. You are ready to go. The resulting executable will be called `dt`.

Code documentation: On your directory you will find the file `README.dt`, which contains instructions on how to run the program.

Problems?: We have tried our best to assure the code is bug-free and will run on any Unix machine. However, it is impossible to test it on all machines. If you encounter a problem, first check the course web page for updates and announcements about the assignment. If you still have a problem, then contact `dmarg+@cs.cmu.edu` .

Part I: The *PlayTennis* data.

The training data for *PlayTennis* from Table 3.2 in the textbook is available on the file `play-tennis.ssv`.

1. Take a look at the training data on `play-tennis.ssv`, then run the decision tree learning program on all the training examples to be sure it produces the correct decision tree for this data (it should produce the decision tree shown in Figure 3.1 of the textbook). Now try running it on a randomly chosen subset containing half of the examples for training, and using half for test. What are the training and test accuracies?

True or false? Answer questions 2, 3, and 4 true or false. If true, demonstrate it by running the decision tree learner, and turn in the parameter settings you used and your data set. If false, explain why. When answering questions 2, 3, and 4, assume that (a) the target concept is the concept described by the decision tree of Figure 3.1 of the text. (b) all training examples you add *must* be consistent with this target concept, and (c) you are not allowed to include post pruning when learning decision trees.

2. It is possible to get ID3 to further elaborate the tree below the rightmost leaf in Figure 3.1 (and make no other changes to the tree), by adding a single new (correct) training example to the original fourteen examples.

3. It is possible to get ID3 to learn an *incorrect* tree (i.e., a tree that is not equivalent to the target concept of Figure 3.1) by adding new *correct* training examples to the original fourteen.

4. It is possible to get ID3 to include the attribute *Temperature* in the learned tree, even though the true target concept is independent of *Temperature*.

Part II: The *Voting record* data.

The file `vote.ssv` contains a set of 435 training examples, each describing the voting record of one member of the U.S. House of Representatives. The first attribute of each example describes the political party of this representative, and the remaining attributes indicate their yes/no/absent vote for each bill considered by congress. You will use this data to learn a decision tree that predicts the political party of the representative, based on his/her vote.

5. Use the voting data to train a decision tree to predict political party (Democrat or Repulican) based on the voting record. Use 25% of the members of congress for training, and the rest for test (no pruning). Rerun this experiment several times and notice the impact of different random splits of the data into training and test sets. Report the sizes and accuracies of these trees over half a dozen distinct runs.

6. Measure the impact of training set size on the accuracy and the size of the learned tree (no pruning; use 30% of the data for testing). Consider training set sizes in the range 0–40% (include at least the values .02, .1, .2, .3, and .4 for training fractions). Because of the high variance due to random splits, repeat the experiment with ten different random seeds for each training set size, then report the mean, maximum, and minumum accuracies at each training set size. Also measure the mean, max, and min tree size. Turn in two plots, showing how accuracy varies with training set size, and how the number of nodes in the final tree varies with training set size. Indicate maximum, minimum, and mean values for each training set size.

7. Measure the impact of pruning on accuracy and size of the learned trees, under the same conditions as in question 6. Use the same training set sizes, but this time use 30% of the data for pruning, and 30% for testing. Again plot the mean, maximum, and minumum accuracy for each training set size, and plot the mean, maximum, and minimum tree size for each training set size. What is the impact, if any, of post-pruning the tree?

8. The code as provided uses information gain to select attributes while growing the tree. Modify it to instead select attributes at random, and study the effect of this change. Look at the function `MaxGainAttribute()` found in the file `entropy.c`. `MaxGainAttribute()` returns the attribute to be selected. Replace this by a completely random attribute selector. Now learn trees using 30% of the data for training and 30% for test. Run the learner ten times using pruning and tens times without pruning. Compare to the results you saw in steps 5, 6, and 7 above. What is the impact of learning with randomly selected attributes?

- **Extra credit - optional!** Pick your own data set and try out this program (warning: the code can only learn trees that predict boolean attributes, although the other attributes that describe instances may be continuous or have an arbitrary number of possible discrete values).

- **Extra credit - optional!** Implement a rule post-pruning method and add it to the source code. How does its performance compare to that of reduced error pruning?

- **Extra credit - optional!** Think up another interesting extra credit problem, state it, and do it.