CSCI 374 — Machine Learning and Data Mining
Oberlin College — Fall 2016

Homework #1: Decision Trees

Important Dates

Assigned: September 21

Snapshot 1: September 28 (11:59 PM)

Snapshot 2: October 5 (11:59 PM)

Final Due Date: October 10 (11:59 PM)

Assignment

In this assignment, you will practice:

1) implementing machine learning algorithms from scratch,
2) experimenting with those algorithms on a variety of provided data sets with different properties,
3) analyzing the results of those experiments to evaluate the performance of the different implemented learning algorithms with respect to different data sets, and
4) writing a technical report detailing (i) how your implementation works, (ii) your experimental setup, (iii) the results of your experiments, and (iv) any implications or lessons learned from your implementation and results.

In particular, you will implement the two or three machine learning algorithms discussed in class for learning decision tree representations of a supervised learning classifier: ID3, C4.5, and (optionally) CART. Through implementing the algorithms (rather than re-using existing implementations), you will gain a better understanding of how decision trees are learned, how they can be used, as well as the differences between various algorithms for learning decision trees and their relative advantages and disadvantages.

Acceptable Programming Languages

You can use either the **Java** or **Python** programming languages to complete this assignment.

Data Sets

For this assignment, you will use three pre-defined data sets in CSV files that can be downloaded from the "Course Content/Homework 1" folder on Blackboard:

3) **monks1.csv**: A data set describing two classes of robots using all nominal attributes and a binary label. This data set has a simple rule set for determining the label: `if head_shape = body_shape V jacket_color = red, then yes, else no.`

This data set is useful for debugging your implementations and verifying their correctness. Monks1 was one of the first machine learning challenge problems (http://www.mli.gmu.edu/papers/91-95/91-28.pdf). This data set comes from the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems

4) **opticalDigit.csv**: A data set of optical character recognition of numeric digits from processed pixel data. Each instance represents a different 32x32 pixel image of a handwritten numeric digit (from 0 through 9). Each image was partitioned into 64 4x4 pixel segments and the number of pixels with non-background color were counted in each segment. These 64 counts (ranging from 0-16) are the 64 attributes in the data set, and the label is the number from 0-9 that is represented by the image. This data set is more complex than the Monks1 data set, but still contains only nominal attributes and a nominal label. This data set comes from the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

5) **hypothyroid.csv**: A data set describing patient health data using a mix of nominal and continuous attributes that can be used to diagnose the health of a patient's thyroid into four possible labels. This data set is more complex in the types of attributes and the number of labels than the other two data sets. This data set comes from Weka 3.8: http://www.cs.waikato.ac.nz/ml/weka/

The file format for each of these data sets is as follows:

- The first row contains a comma-separated list of the names of the label and attributes
- Each successive row represents a single instance
- The first entry (before the first comma) of each instance is the label to be learned, and all other entries (following the commas) are attribute values.
- Some attributes are strings (representing nominal values), some are integers, and others are real numbers. Each label is a string.

Program Behavior

Your program should behave as follows:

1) It should take as input three parameters:
   a. The path to a file containing a data set (e.g., monks1.csv)
   b. The name of the algorithm to use for training (see below for more details)
   c. A random seed as an integer
2) Next, the program should read in the data set as a set of instances
3) The instances should be split into training and test sets (using the random seed input to the program)
4) The training set should be fed into the specified machine learning algorithm to construct a decision tree fitting the training data
5) The learned decision tree should be evaluated using the test set created in Step 3.

6) The confusion matrix counted during Step 5 should be output as a file with its name following the pattern: results_<DataSet>_<Algorithm>_<Seed>.csv (e.g., results_monks1_ID3_12345.csv).

The file format for your output file should be as follows:

- The first row should be a comma-separated list of the possible labels in the data set, representing the list of possible predictions of the decision tree. **This row should end in a comma**.
- The second row should be a comma-separated list of the counts for the instances predicted as the different labels whose true label is the first possible label, ending with the name of the first possible label (and not a final comma).
- The third row should be a comma-separated list of the counts for the instances predicted as the different labels whose true label is the second possible label, ending with the name of the second possible label (and not a final comma).
- Etc. for the remaining possible labels

For example, the confusion matrix:

Predicted Label			
Yes	No		
200	100	Yes	Actual
50	250	No	Label

would be output as:

Yes,No,
200,100,Yes
50,250,No

The output for your program should be consistent with the random seed. That is, if the same seed is input twice, your program should learn the exact same tree and output the exact same confusion matrix. You are free to also output other files, too, if you wish (e.g., a file describing the learned tree).

Experiments

There are two options for completing this assignment:

Option #1: Implement each of the ID3, C4.5, and CART algorithms, then use the three data sets to conduct the following experiments:

1) Pick a single random seed (include it in your report) and run each learning algorithm on each data set (with the exception of do not run ID3 on the hypothyroid data set since it contains numeric data), then compare the resulting performance of the learned decision

trees from each algorithm. For each data set, how do the accuracies? Remember to use 95% confidence intervals in your comparisons.

2) Pick one data set, then learn 30 different decision trees with each algorithm, and calculate the average accuracy per algorithm across the 30 runs. To do so, use 30 different random seeds to generate 30 different training sets and 30 different trees. Then, compare the average accuracy across those 30 runs with the confidence intervals found in Experiment 1 above to answer the following questions for each algorithm:
    a. How close was the average accuracy across the 30 runs to the original accuracy found in Experiment 1?
    b. Does the average accuracy fall within or outside the confidence interval found in Experiment 1?
    c. Are the average accuracies across algorithms closer or farther apart than the original accuracies computed for Experiment 1?

    Only calculate standard errors and confidence intervals in Experiment 1 and not for your 30 additional runs in Experiment 2.

The goal of Experiment 1 is to investigate how the different algorithms compare on different data sets and gain practice evaluating their differences. The goal of Experiment 2 is to gain additional understanding into how confidence intervals measure the performance of machine learning algorithms.

For Option #1, the names of the algorithms to use as input to your program should be ID3, C4.5, and CART.

Option #2: Implement the ID3 algorithm, as well as three variants of C4.5: (1) full C4.5, (2) C4.5 without pruning, and (3) C4.5 without using *SplitInformation* when determining the best attribute (only use *Gain* as in ID3). Then, using the three data sets, conduct the same two experiments as in Option #1, except consider all three variants of C4.5 (and leave out CART) in both Experiment 1 and 2.

In particular, add the following analyses:

- For the monks1.csv and opticalDigit.csv data sets, draw the root and children of the trees found by ID3 and C4.5 without pruning. Compare any similarities or differences between the trees.
- For the monks1.csv and opticalDigit.csv data sets, compare the attributes found at the top of the tree in ID3 and in the most accurate rules found by (full) C4.5. Do the same attributes appear in both? What differences do you find?
- For all three data sets, compare full C4.5 to C4.5 without pruning to evaluate the benefits of pruning on total accuracy on the test set.
- For all three data sets, compare full C4.5 and C4.5 without *SplitInformation* to evaluate any possible benefits on total accuracy on the test set caused by considering *SplitInformation* when choosing the best attribute for each node.

For Option #2, the names of the algorithms to use as input to your program should be ID3, C4.5, C4.5NP (for C4.5 without pruning), C4.5NSI (for C4.5 without SplitInformation)

Snapshots

Since the homework assignment is multiple weeks long, there are two intermediate deadlines to help you make sure you complete the entire assignment on time:

Snapshot 1 (due Wednesday September 28 at 11:59 PM): your program should be capable of:

- Inputting the program parameters described above
- Reading a data set into a set of instances
- Splitting the data set into training and test sets (using the random seed)
- Running the ID3 algorithm
- Outputting the confusion matrix from testing the learned tree

Snapshot 2 (due Wednesday October 5 at 11:59 PM): your program should additionally be capable of:

- Running the C4.5 algorithm

For each snapshot, your code (and associated Makefile and README described below) should be organized in a ZIP file and turned in on Blackboard. Your zip file should be named:

<OCCSUserName>_SnapshotX.zip

For example, Alice Student's second snapshot would be named: astudent_Snapshot2.zip

Final Handin

Before the assignment due date (Monday October 10 at 11:59 PM), you will turn in:

1) A ZIP file (named as your OCCS username) containing:
    a. Your source code
    b. A Makefile for compiling your source code
    c. A README file
2) Your technical report as a PDF file, named the same as your ZIP file.

Your Makefile must be able to compile your source code into an executable program that behaves as described above. Your README file should describe the different source code files used by your program, as well as instructions for running your program and finding its output file(s).

Your technical report should contain:

- An introduction describing the assignment and the contents of the report (provide the reader with the background needed to understand the rest of the report)
- A description of your implementation (what did you create?)
- A description of your experimental setup (what did you run and for what purpose?)
- A discussion of the results (what did you find, why did you find that, and what are the implications?)
- A conclusion summarizing the report and assignment

Grading

The homework will be graded as follows:

- Snapshot 1: 5%
- Snapshot 2: 5%
- Implementation Correctness and Documentation: 50%
- Report: 40%

Honor Code

Each student is to complete this assignment individually. However, since the assignment is a mini-project in scope, students are encouraged to collaborate with one another to discuss the abstract design and processes of their implementations. For example, please feel free to discuss the pseudocode for each learning algorithm to help each other work through issues understanding exactly how the learning algorithms work. You might also want to discuss the processes used to generate the training and test sets from the read in data set. Or, you might need to discuss how to work with the input and output files.

At the same, since this is an individual assignment, no code can be shared between students, nor can students look at each other's code. All discussions should be limited to abstract details and not implementation-specific concerns. For example, no discussing of the code used in the classes used to represent a decision tree, nor the lines of code used to build the trees from training data. Furthermore, the source code of existing machine learning libraries (e.g., Weka for Java, scikit-learn for Python) must not be consulted. Any violation of the above will be considered an Honor Code violation.

If you have any questions about what is permissible and what is not, please discuss with the professor. Please also feel free to stop by office hours to discuss the homework assignment if you have any questions or concerns.