This homework is individual work. All assignments are due at the beginning of class on the due date. Assignments turned in late will drop 10 points for each period of 24 hours for which the assignment is late. In addition, no assignments will be accepted after the solutions have been made available.

1 Implementing TDIDT Learner

Implement the following TDIDT learning algorithm in a programming language of your choice. Given is a training sample $S = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$. You can assume that all features, as well as the output space Y are binary.

TDIDT(S):

1. IF (train set S is "sufficiently pure")
   (a) return leaf labeled with majority label in S
2. ELSE IF (all examples in train set S have the same attribute vector)
   (a) return leaf labeled with majority label in S
3. ELSE
   (a) pick A as the "best decision attribute" for next node
   (b) FOR each value v_i of A, create new descendant of node
      i. $S_i = \{(\vec{x}, y) \in S$: attribute A of $\vec{x}$ has value $v_i\}$
      ii. set subtree t_i for v_i to $TDIDT(S_i)$
   (c) return tree with A as root and t_i as subtrees

A dataset is available on the course website, at http://www.cs.cornell.edu/courses/cs478/2004sp/. It consists of two files: a training set "train.dat" and a test set "test.dat". The task is to learn a rule for how students will evaluate the quality of a teaching assistant. The label 1 stands for "good", the label 0 stands for "bad". A TA is described by the attributes

- english: 0: non-native_speaker, 1: native_speaker.
- semester: 0: regular, 1: summer.
- class size: 0: small(< 20), 1: big(≥ 20).
- advisor: 1:TA_advisor_teaches_class, 0:other_teacher.
- experience: 0:none, 1:some.
- gpa: 0: low, 1: high.

Examples are given in the following form:

```
1 1 0 1 0 0 1
1 0 0 1 1 0 0
```

Each line stands for one example. The first example describes a TA who is a native english speaker, for a course in the summer semester, with a small class size. The TA's advisor teaches the class, the TA has no experience begin a TA and has a low GPA. The students evaluated this TA to be a "good" TA. Make sure that your implementation can read this format.

Part a (40 POINTS)

Implement the TDIDT algorithm. The program should print the learned decision tree to the screen. Use the following interpretations for "sufficiently pure" and "best decision attribute":

- "sufficiently pure": all examples in S have the same label y.
- "best decision attribute": use $Gain_y(S, A)$ for selection.

Include a print-out of the program in your write-up.

Part b (10 POINTS)

Implement a program that classifies a file of examples based on a fixed decision tree. The decision tree can be "hard-coded" into the program. The program should output the error rate of the decision tree on the examples in the file.

Print the program (for some example tree) and include it in your write-up.

Part c (10 POINTS)

Train the algorithm on the dataset "train.dat" (using your solution to part a). Use the learned tree to classify the examples in "test.dat" (using your solution to part b). Report the fraction of training errors and test errors the learned tree makes. Also, report the tree that you learned. What do you notice about this tree?

2 Attribute Selection Criteria (10 POINTS)

Use some other selection criterion for "best decision attribute" instead of $Gain_y(S, A)$. Describe the criterion you selected and implement it. Repeat the experiment from Part 1c for your new criterion. Again, report error rates and the tree that you learned.

3 Overfitting (10 POINTS)

Instead of growing the decision tree until the subsamples become pure (or all feature vectors are equal), use a different criterion for "sufficiently pure". In particular, implement the criterion to stop whenever the entropy of node is less than δ. Repeat the experiment from Part 1c for your new criterion and at least 3 reasonable values for δ. Report error rates and the trees that you learned, and comment on the results.

4 Significance Tests (20 POINTS)

Use a suitable statistical significance test based on the test error rates to address the question, whether the tree generated in Problem 2 has a true error rate that is significantly different (95% confidence) from the tree in Problem 1c. Similarly, decide whether any of the trees from Problem 3 have a significantly different true error rate compared to the tree in Problem 1c. Explain how you got to your answer.

If you did not get results for Problem 1c, Problem 2, or Problem 3, use the data available on the course home page.