CS181 Assignment 1: Decision Trees

Out Monday February 4th
Due Friday February 15th

Submit by noon via iSites dropbox.

General Instructions

You may work with one other person on this assignment. Each group should turn in one writeup. This assignment consists of a theoretical component and an experimental component. The experimental component requires you to write code and analyze the effectiveness of different algorithms you implement. For the experimental results, we have specified what graphs and tables we expect to see. For questions that do not explicitly specify graphs or tables, you are not required to present data in this form.

In this assignment, you will develop a classifier for medical data. You will be working with a database of instances describing patients who have been tested for breast cancer. You will develop a classifier that can determine whether growths are malignant or benign, based on the results of tests taken by a patient. The dataset was derived from the Wisconsin breast cancer corpus, obtained from the UC Irvine machine learning repository at `http://archive.ics.uci.edu/ml/`. The UCI repository is an important collection of many of the most frequently used machine learning benchmarks.

You can find the dataset for this assignment, as well as code, at `http://www.seas.harvard.edu/courses/cs181/files/hw1.tar.gz`. The data can be found in `bcw.dat`, while `noisy.dat` contains the same data with a certain amount of random "noise" added. Each dataset contains a total of 100 samples. Each sample in the data consists of 9 attributes, each of which ranges from 1 to 10, as well as a binary label that is 0 or 1. The file `breast-cancer-wisconsin.names` describes the attributes and also contains information abou the history of the dataset.

1. **[15 Points] Decision Trees and ID3**

   (a) **[5 Points]** Suppose that the ID3 algorithm is in the middle of training on a data set, and there are seven instances remaining, with four positive and three negative instances. It has the choice of splitting on two binary attributes A and B. When A is true, there are two positive and two negative instances, while when A is false there are two positive and one negative instances. Meanwhile

when B is true there is one positive and one negative instance, while when B is false there are three positive and two negative instances.

Which attribute will ID3 choose to split on? Show the information gain calculations. For each of the two possible splits, present an informal and brief argument that the split is more useful than the other. What does this example show about the inductive bias of ID3?

(b) **[5 Points]**

Use your work in part (a) to show a tree that ID3 might construct for the following dataset, in which there are four binary attributes for a binary classification task. You do not need to show the information gain computations, but you should briefly justify why a particular attribute was chosen at each point in the tree. In case a tie needs to be broken, indicate which other attribute(s) could have been chosen.

A	B	C	D	Label
1	0	1	0	0
1	0	0	0	1
0	0	0	1	0
1	0	0	1	1
0	1	1	1	1
1	1	0	1	0
0	0	0	1	1

(c) **[5 Points]**

By eyeballing the data, find a simpler tree that has the same training error as the one produced by ID3. What can we learn from this example about the ID3 algorithm?

2. **[25 Points] ID3 with Pruning**

We have provided an implementation of ID3, but this implementation is very basic. In partiular, the implementation does not perform any pruning and as a result is susceptible to overfitting. In this problem, you will implement pruning with a validation set and perform experiments to test its effectiveness.

In the `hw1` directory from the provided tar file (http://www.seas.harvard.edu/courses/cs181/files/hw1.tar.gz), we have provided an implementation of the ID3 algorithm that does no pruning. Along with an implementation of the ID3 algorithm, we have included code in the `main` function that parses command line arguments and loads the data file. The Python code reads the data file into a `DataSet` structure, which has fields for all of the relevant quantities. The command line interface is

```
python main.py [-n] [-p valSetSize]
```

where `-n` indicates that the noisy data set should be used and `-p` signals that pruning should be done with the specified validation set size. In order to use the support code we have provided, you should call the `learn` function, which takes a single argument: a `DataSet` object. It returns a `DecisionTree` structure that is the result of calling ID3 on this training set. The `DecisionTree` structure is defined in `dtree.py`.

(a) **[5 Points]**
Complete the `main` function and implement ten-fold cross validation on the datasets we have provided. For convenience, we have stored two contiguous copies of data in the data structures so you can geta complete view of the dataset starting at any point in the first set of 100 samples and looking at the next 100 samples. You will also need to write a function that returns the score of a learned tree on a given dataset. Let *0-1 accuracy* for a given classifier and set of examples denote the (# correctly classified) / (total # of instances). Let *cross validated [test/training] performance* mean the average [test/training] performance over the ten trials of cross validation. What is the average cross-validated training and test performance over the ten trials for the non-noisy dataset? What is the cross-validated training and test performance over the ten trials for the noisy dataset? [Hint: The training and test performance for both datasets should be in the range (0.75, 1.0)].

(b) Create a pruning function that does bottom-up validation set pruning, i.e., it checks whether child nodes should be pruned before it checks their parents.

i. **[10 Points]** Provide a graph of the cross-validated training and test performance for both the full dataset and the non-noisy dataset for validation set sizes in the range [1, 80]. Sample code for creating a graph in Python is included in the **Hints** section below.

ii. **[2 Points]** How does the cross-validated performance of the validation set pruning vary with the size of the validation set?

iii. **[2 Points]** Does the validation set pruning improve the cross-validated performance of ID3 on these data?

iv. **[6 Points]** Is overfitting an issue for these data? Justify your answer.

3. **[30 Points] Boosting**

The boosting paradigm is another way to overcome the overfitting difficulty. In this problem, you will implement AdaBoost and experiment with various boosting possibilities. You will need to make several changes to the ID3 algorithm in order to implement boosting and run the experiments below. You might also need to change other parts of the support code.

We have provided the parsing of command line arguments that may be useful for your experiments. In particular, we parse the `-b` and `-d` flags. `-b` specifies the number of boosting rounds, and the `-d` flag specifies the depth of the weak learner. The full command line specification is as follows:

```
python main.py [-n] [-p valSetSize] [-b boostingRounds] \
               [-d weakLearnerDepth]
```

We will parse these flags for you, but you are responsible for using them to implement boosting. Additionally, we have already added support for example weights. Again, these weights are not currently used or updated, so you have to make appropriate changes.

(a) **[6 Points]** In order to implement boosting, you need to define an information gain criterion that takes example weights into account. Define such a criterion and discuss the reasons for your choice. Consider a set of examples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$, where the label of the first example is $y_1 = 1$, but all others are zero, i.e., $y_n = 0, \forall n > 1$. What is the weighted entropy of the set, if the weight $w_1 = 0.5$ and the others are $0.5/(N-1)$?

Here are the concrete steps needed to implement boosting:

- Modify the ID3 algorithm so that it works with weighted training instances. In particular, you need to use the weighted information gain criterion you defined above when deciding what attributes to split on. Additionally, you need to take weights into account when choosing the majority class at a given node.
- Write a function that takes a weighted set of decision trees rather than a single decision tree. This function should classify an instance by having the different trees vote on the label of the instance.
- Write the boosting wrapper that uses the AdaBoost algorithm to learn a voting hypothesis consisting of a set of weighted decision trees. The boosting algorithm should take the number of rounds as a parameter.
- Modify the `Learn` function to take an integer-valued depth cutoff as a parameter. This cutoff specifies the maximum depth to which any branch of the tree may be grown. (The depth of the root node is zero.) Do not allow the tree to contain any path longer than this depth cutoff. This modification will allow you to use depth-K decision trees as the weak learner for boosting and to experiment with different values of K. Depth-1 decision trees are decision stumps and have one branching decision.

i. With AdaBoost implemented, please analyze the effectiveness of boosting:

A. **[5 Points]** How does the maximum depth of the weak learner affect cross-validated test performance for boosting (try depths of 1 and 2) on both datasets? Provide a table that shows cross-validated test performance on both datasets for 10 and 30 rounds of boosting and depths 1 and 2. How can we explain these results?

B. **[9 Points]** Provide a graph of cross-validated test performance of the decision trees learned by boosting over different numbers of boosting rounds (between 1 and 30). For these data, fix the depth fo the weak learners to 1 and provide results for both the non-noisy and noisy datasets. How do the results vary with the number of boosting rounds? Is this expected based on our theoretical discussion of boosting in class?

C. **[5 Points]** How does the cross-validated test performance of boosting compare with that of ID3 with and without pruning? (Only comparisons for the non-noisy dataset are required).

D. **[5 Points]** Examine the cross-validated training and test performance for boosting with weak learners of depth 1 over a number of rounds in [1, 15]. Is there a relationship between the training and test performance?

4. **[8 Points] Tree Analysis**
   Choose a particularly effective decision tree and examine the structure of the tree, mapping attribute indices to qualitative descriptions using the `.names` file. Present the tree you chose along with the methodology used to generate the tree. Which attributes are most important to benign/malignant decisions?

Hints

- Here is sample code to create a graph using matplotlib:

```
import matplotlib.pyplot as plt
from pylab import *

plt.clf()

xs = range(5)
ys = [3, 5, 1, 10, 8]

p1, = plt.plot(xs, ys, color='b')

plt.title('sample graph')
plt.xlabel('x-coordinate')
plt.ylabel('y-coordinate')
plt.axis([0, 4, 0, 12])
```

```
plt.legend((p1,), (’data’,), ’lower right’)
savefig(’figure.pdf’) # save the figure to a file
plt.show() # show the figure
```