Program 3

Decision tree learning

Introduction

This project requires you to implement a decision tree learning algorithm, run experiments on real datasets, and write a report explaining your experimental results. For full credit your program will have to handle data sets with an arbitrary number of (continuous) numeric-valued features.

Your program should be able to interpret the data format specified below, produce a human readable version of the decision tree formed from a data set, and be able to use the decision tree to classify new testing instances (producing, of course, interesting statistics on the output trees such as accuracy and size).

Software

You need to develop software to learn a decision tree from a training data set, report the size of the learned tree, run that tree on a data set (both the training data, and test data), producing statistics on accuracy.

You are free to construct whatever user interface for your program you like, but you must fully document your interface. We are providing data files (described later) containing the feature/class names, the training data, the testing data, and pruning data (for extra credit).

Your software must use the information gain splitting approach (first developed in the ID3 algorithm). Furthermore, you must handle continuous input attributes/features; the classifications will still be discrete (represented here as integers, so everything is numeric and easy to parse).

Data

You'll be examining the behavior of your DTL algorithm on several data sets. The data sets are all represented in a standard format, consisting of four files per data set. The first file, `XXX.names`, describes the categories and features of the dataset, and is formatted as follows (comma separated values, although the lines are not equal length; this is not a spreadsheet format):

```
cat1, cat2, cat3, ... catn
feature1, feature2, feature3, ... featurek
```

Where `cat1 ... catn` are the n names of the output categories (integers), and `feature1 ... featurek` are the names of the input features (strings). For this programming assignment, you don't strictly need this file unless you are doing the extra credit. Also, for this assignment, you may assume all features will be numeric-valued, taking on continuous values (we've recoded any discrete string features as integers). Categories are also numeric, represented as discrete integers.

The other files of each data set, `XXX.train`, `XXX.prune`, and `XXX.test`, contain the actual data instances, formatted at one instance per line, in CSV (Comma Separated Value) format as follows:

```
i1fv1, i1fv2, ... i1fvk, i1cat
i2fv1, i2fv2, ... i2fvk, i2cat
…
iNfv1, iNfv2, ... iNfvk, iNcat
```

Where `i1fv1` *(instance 1 feature value 1)* represents the value of the first feature for the first instance, and `i1cat` is the category value of the first instance. You can assume that there will be no missing feature values. Most programming languages will have a library for reading in CSV files.

For each problem we have provided training, pruning and test sets. Usually training is 60% of the data, pruning is 20% of the data and testing is 20% of the data. These datasets come from the UCI Machine Learning Repository.

The data sets you will be examining are:

BALLOONS: These are four very very small files you can use to test your program on. They are simple enough that you will know the correct answer (see balloon.info) and each feature is binary and there are only two classes. There are no test or pruning sets provided, this data is provided for you to test your basic learning algorithm.

HYPOTHYROID: This is a medical database representing the classification of 3163 patients with and without hypothyroid disease at the Garavan Institute of Sydney, Australia.

MESSIDOR: This dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not. All features represent either a detected lesion, a descriptive feature of a anatomical part or an image-level descriptor.

Your Mission...

You will need to hand in both a writeup report and code to the TA. Deliverables for this project are:

- Code to implement the decision tree learning algorithm for the data file formats given above. A README file, with simple, clear instructions on how to run your code.
- Testing statistics for the application of your learning algorithm to each of the provided data sets. At a minimum you should provide training set accuracy, test set accuracy, and the size of the decision tree (number of nodes and max depth). If you prune (extra credit, see below), you should also provide these numbers both before and after pruning (showing that the tree got smaller, but that accuracy is nearly as good or better after pruning).
- A report analyzing the behavior of your algorithm on each data set, including any unusual or anomalous (in your opinion) behavior.

In the report you should answer the following questions:

General Questions [10 points]:

1. For each of the data sets was the decision tree algorithm able to memorize the training data (i.e., get 100% accuracy when used to classify the training data)? If not, can you give at least one reason that you think it was not able to?

2. Could you use your system to, say, buy stocks or bet on horses? Why or why not?

Balloon [30 points]:

1. Demonstrate your program on these small data sets. Draw out the trees you generated for the 4 balloon examples. Note that there are no testing or pruning files—we assume your tree will be perfect in these very simple examples.

Hypothyroid and Messidor[30 + 30 points]:

1. What is the majority class accuracy for the test set? (i.e. what is the accuracy to just say "benign" always?)

2. How does you tree's test accuracy compare to the majority accuracy?

3. What do the top 2 levels (the root and the level just below) look like (draw them)?

EXTRA CREDIT [20 points]

Use reduced-error pruning. This is a pruning criterion that replaces a subtree with a single leaf when that leaf (which classifies the instance as the majority class of all instances seen at that leaf during pruning) represents a lower error rate than does the subtree. The pruning procedure thus has the following steps:

Grow the tree fully on the training data.

In a postorder recursive fashion traverse the decision tree, replacing a non-leaf node with a leaf node only when the error rate of the subtree node exceeds the error rate of the leaf node. Measure the error rate on a separate set of instances (called the pruning data).

EXTRA CREDIT [10 points]

Explore the effect of Bagging. Choose the data set for which you performed the WORST, and see if you can improve your results via bagging.

EXTRA CREDIT [20 points]

Explore the effect of Boosting. Choose the data set for which you performed the WORST, and see if you can improve your results via implementing the AdaBoost ensemble algorithm.