Georgia Tech
CSE6242 / CS4803-DVA: Data and Visual Analytics
Spring 2013, Polo Chau
Homework #2, Due: March 26, 2013, 1:30PM

You will

- implement *random forest* from scratch, and
- explore visualizations (using D3) to help you better understand how random forest behave in binary and multiclass classification problems.

You will submit a single archive file; detailed submission instructions are in the last section.

Task 1: Random forest [20 points]

Implement random forest using a programming language of your choice. For details on random forest, see Chapter 15 in the "Elements of Statistical Learning" book. The main parameters in a random forest are:

1. How many attributes of the whole set of attributes do you select to find a split?
2. When do you stop splitting leaf nodes?
3. How many trees should be in the forest?

No matter how you choose to answer questions 1 and 2, be sure to specify and justify your choices in your **`readme.txt`**. Regarding question 3, implement your code so that you can keep track of the training and test error as you increase the number of trees in the forest.

Deliverables. [20 points] Set up your code in a way such that we can run it with our test set (for grading); see the last section for more details. Any implementation-related item should be submitted in a directory named **`code/`**.

Task 2: Binary classification performance visualization [20 points]

For the binary classification training and test sets we have provided to you, visualize the training and test error using D3. This visualization should depict how the training and test error behaves when adding more trees to your random forest. (You should be able to obtain greater than 90% accuracy.)

Deliverables.

1. **[7.5 points]** Two data files containing the training and test error values for your random

forest as the number of trees increases. Name these files **`bin-class-training-error.csv`** and **`bin-class-test-error.csv`** , respectively. You are free to choose how to specifically represent the error data in these files.

2. **[7.5 points]** An html file using D3, named **`bin-class-viz.html`**, that reads these error files to visualize the training error and test error of the trained random forest as the number of trees increases.
3. **[5 points]** A text file named **`bin-class-discussion.txt`** where you discuss your design choices, the performance of random forest on this data set (both training and test performance), and your final choice for the number of trees in the random forest for this dataset. Discuss the choices you have made for how to visualize this data and why these choices are appropriate.

Task 3: Visualizing multiclass classification training performance [20 points]

For the multiclass classification training set we have provided to you, visualize the training error using D3. This visualization should depict how the training error behaves when you add more trees to your random forest. One way to visualize errors in multiclass classification problems is to use a confusion matrix, in the style of Figure 2 in the EnsembleMatrix paper.

Deliverables.

1. **[5 points]** A file named **`multi-class-training-error.csv`** containing error information from the training of your random forest, as the number of trees increases. (Your accuracy should surpass 90% at some point.) You are free to choose how to specifically represent the error data in this file.
2. **[10 points]** An html file using D3, named **`multi-class-training-viz.html`**, that reads this error file to visualize the performance of the trained random forest as the number of trees increases. You must choose how exactly you will do this. Possibilities include:
    - a series of confusion matrices showing the performance for 10, 20, 30, ... trees
    - an interactive confusion matrix that somehow allows you to slide to discrete values like 10, 20, 30, ... trees; you are welcome to code in Javascript for this
    - some variation of a small multiple graphics

    You can also use and/or combine other visualization techniques, or even create your own. Whatever you do, justify your choices.
3. **[5 points]** A text file named **`multi-class-discussion.txt`** where you discuss the performance of random forest on this data set and your final choice for the number of trees in the random forest. Discuss the choices you have made for how to visualize this data and why these choices are appropriate.

Task 4: Multiclass classification prediction and visualization [20 points]

We will test the performance of a trained random forest on a test set we have kept to ourselves (we = Pari, Sooraj and Polo). Please provide a script/program that reads test data from *stdin* (command prompt) and writes to *stdout* (screen) the labels predicted by a trained random forest with **only** 50 trees, trained on the training set we provided you. The output should have the predicted label of each test point, one label per line. For example, if the input consists of 100 test points, the output should contain 100 lines, each containing the predicted label of the corresponding test point.

To accomplish this, you will have to either

a. train a random forest of 50 trees ahead of time and save it for later use (in a format of your choosing, located in **`code/`** somewhere), or
b. train a new random forest of 50 trees on the original training set each time we run your script

We recommend (a); in this case, your script/program simply loads the trained random forest and queries the test points on it.

After this, we would also like to visualize the test error. You must use D3 to read a confusion matrix as input and produce a visualization of it. The confusion matrix will be in a file named **`multi-class-confusion-matrix.csv`**. You may find the d3.csv function handy. Here is an example of what the format of this file will look like:

```
L1,L2,L3,L4,L5,L6
8,0,2,0,1,0
1,6,1,4,0,1
0,1,9,0,0,0
0,1,0,8,1,0
0,0,0,0,9,1
0,0,0,0,1,9
```

Each row in the confusion matrix corresponds to the *actual class* and each column corresponds to the *predicted class*. The n^{th} row and n^{th} column each refer to the n^{th} label. In this example, the entry at the 2^{nd} row and 4^{th} column is 4, which indicates that four test instances with true class `L2` were predicted to be of class `L4`.

Deliverables.

1. **[12 points]** A script named **`multi-class-predict.`*`{ext}`*** that reads from stdin test data in csv format and prints the predicted labels to stdout. We will use the script as

follows:

```
./multi-class-predict.{ext} < test-data.csv > predicted-labels.out
```

The test data will be in the same format as the training data, minus the labels. The *.{ext}* just refers to whatever file extension you are using (for example, **.sh** for Bash scripts).

2. **[8 points]** An html file using D3, named **`multi-class-test-error-viz.html`**, that visualizes a confusion matrix. This file should visualize the confusion matrix located in a file named **`multi-class-confusion-matrix.csv`** (that we will supply during grading), which will contain the confusion matrix we create from your predicted labels, in the format specified above. The confusion matrix file will be in the same directory as this html file.

Submission details

Submit the deliverable as **`hw2-{GT-USERNAME}.tar.gz`** on T-Square. For example, if your GT username is *jdoe3*, your submission file should be named **`hw2-jdoe3.tar.gz`**. Please specify the name(s) of any students you have collaborated with on this assignment, using the text box on the T-Square submission page for this assignment.

The submission directory structure.
The submitted archive should contain a single directory named **`hw2-{GT-USERNAME}`**. The directory should only have the following files (assuming GT-USERNAME is *jdoe3*):

- `hw2-jdoe3/readme.txt`
  Here you specify the tools and programming language for your implementation, and instructions for compiling your code. You may also put relevant comments here.
- `hw2-jdoe3/code/`
  This directory should contain any code, saved training models and all other implementation-related items.
- `hw2-jdoe3/bin-class-training-error.csv`
- `hw2-jdoe3/bin-class-test-error.csv`
- `hw2-jdoe3/bin-class-viz.html`
- `hw2-jdoe3/bin-class-discussion.txt`
- `hw2-jdoe3/multi-class-training-error.csv`
- `hw2-jdoe3/multi-class-training-viz.html`
- `hw2-jdoe3/multi-class-discussion.html`
- `hw2-jdoe3/multi-class-predict.{ext}`
- `hw2-jdoe3/multi-class-test-error-viz.html`

Please adhere to the directory structure and naming convention specified here. In case your

submission does not comply with this, *it will be returned to you ungraded*. You would need to resubmit in the specified form to be graded. While your resubmission will be graded, the delay resubmission will be counted as a late submission.

Data for you (# instances x # features)

For both binary and multiclass cases, you should be able to obtain greater than 90% accuracy.

1. Binary classification training set (around 6000 x 5000; each row contains features corresponding to an image of one of the two classes)
    - Training points
    - Training labels
2. Binary classification test set (around 1000 x 5000)
    - Test points
    - Test labels
3. Multiclass classification training set (around 4000 x 64; 10 classes; each row contains features extracted for a handwritten “digit”, e.g., 2, 5, etc.)
    - Training points
    - Training labels

Data we will test on (labels not provided to you)

1. Multiclass classification test set (around 1800 x 64; 10 classes)
    - Test points