CSE6242 / CX4242: Data and Visual Analytics, Spring 2014

Prepared by Robert Pienta, Long Tran, Polo Chau

Homework 4, Due: April 25, 2014, 11:55PM

Task 1: Decision Trees (50 points)

In this task, you will implement a very well-known classifier, decision tree and evaluate its classification performance on a provided dataset using cross validation. Decision trees and cross validation were covered in class (slides).

First, download the data here. The data provided are features extracted from spoken vowels. The data is stored in a *comma-separated-value* (CSV) file. Each line is a sample with the last value (a short string) being the class label which represents a vowel sound.

Your task is to implement a decision tree that correctly classifies as many data entries as possible. You will be implementing this classifier **from scratch by yourself** using a major programming language of your choice, without using an existing machine learning or decision tree library (more details in the "deliverables" section).

Note: The data in the last column are the ground-truth labels (the actual vowels from the study) and *are not input features*. Do not use them as inputs into your model.

In implementation your tree, you will address the following **main challenges**:

1. At each node in the tree, decide on which attribute to split on
2. After an attribute has been selected in (1), decide on which of its value to split on
3. Split stopping criteria at leaf nodes (i.e., when to stop splitting)
4. [Performance evaluation] You will evaluate your decision tree implementation using 10-fold cross-validation. You will first make a *stratified split* of the provided data into 10 parts such that for all classes C_i, the number of C_i samples in the 10 parts are roughly equal (i.e., approximately **10 percent** of C_i samples in each part). Then *hold out* 1 part as the *test set* and use the remaining 9 parts for *training*. Train your decision tree using the training set and use the trained decision tree to classify samples in the test set. For all pairs of classes C_i and C_j, record the number of **test samples** in class C_i classified as C_j. Repeat the processes so that all 10 parts are used as test set *ONCE* in order to accumulate the results into a **confusion matrix**.

In your decision tree implementation, you may freely choose how to proceed with (1), (2), and (3). However, you must explain your approaches and their effects on the classification

performance in a text file **dt_discussion.txt**. Report the best confusion matrix that your tree can achieve for (4), in your judgement (e.g., based on classification accuracy), in **dt_confusion.txt** (CSV format; the classes in the rows and columns follow this order **hid, hId, hEd, hAd, hYd, had, hOd, hod, hUd, hud, hed**).

For example, here is a confusion matrix in CSV format comparing the classification performance over 10 different classes:

```
1133,    1,    0,   0,     3,    0,    1,    0,    4,    1
   0, 1113,   19,   7,     1,    0,    0,    1,    0,    2
   0,   11, 1128,   0,     0,    0,    0,    5,    0,    0
   0,    4,    1, 1041,    1,    2,    0,    3,    0,    3
   0,    2,    0,    0, 1138,    0,    1,    0,    0,    3
   0,    0,    0,    6,    0, 1042,    0,    0,    3,    4
   2,    0,    1,    0,    1,    1, 1051,    0,    0,    0
   0,    9,    4,    2,    0,    0,    0, 1125,    2,    0
   4,    0,    0,    0,    0,    1,    0,    2, 1048,    0
   1,    3,    0,    1,    4,    2,    0,    2,    2, 1040
```

The confusion matrix above shows that most samples are classified correctly. That is, the classifier has good classification accuracy. Note that, the sum of all numbers in the confusion matrix should be equal to the number of samples in the dataset because each sample appears in a test set *only once*.

Deliverables

1. **(20 pt) decision_tree_<your GT username>.???** - a source file of your program in a major programming language of your choice (with comments at the beginning explaining how to compile/run the code on the provided data). Its output should be the confusion matrix (i.e. the accumulated classification results on test sets). You need to implement the classifier **from scratch by yourself**, without using an existing machine learning or decision tree library. Common utility libraries such as STL in C++ and the equivalent libraries in other language are allowed. You may submit the code using multiple source files, provided that the main source file *uses the specified name* and all source files are compressed into *ONE* archive file (e.g., a zip file).
2. **(10 pt) dt_confusion.txt** - the confusion matrix as a result of (4) (CSV format, classes in rows and columns following this order: **hid, hId, hEd, hAd, hYd, had, hOd, hod, hUd, hud, hed**).
3. **(20 pt) dt_discussion.txt** - discussion on your choices for (1), (2), and (3), how they affect the classification performance.

Task 2: Using WEKA (20 points)

In this task you will use several tools from WEKA to analyze additional multi-class data. Download and install WEKA.

The dataset we provide below examines several features extracted from a pen-based handwriting study. Each observation has 16 features extracted from writing a single handwritten digit. You will be provided the 16-feature dataset and the known labels. The data is provided for you here.

We want you to classify the results based on the 10 classes in the file.
Load the data into the WEKA Explorer (or double click the .arff file if you installed WEKA). Click the *classify* tab at the top. Under classifier you can select the different classifiers that WEKA offers. You can adjust the input parameters to many of the models by clicking on the text to the right of the *Choose* button in the *Classifier* section.

Report the formatted confusion matrix and the overall prediction accuracy of each of the following methods:

1. **Naive Bayes** - Under the *classifiers>bayes* folder select NaiveBayes.
    a. Add the confusion matrix to weka_output.txt
    b. Add the prediction accuracy to weka_output.txt

2. **Logistic Regression** - Under the *classifiers>functions* folder select Logistic.

3. **Random Forest** - Under the *classifiers>trees* select RandomForest.

4. **Discussion** - write your answers to the following questions (along with the question's letter next to its corresponding answer) in weka_discussion.txt.
    a. Which of the 3 models from Q1-3 performed the best (i.e., highest accuracy)?
    b. Which of the 3 models performed the worst?
    c. Did you change any of the parameters? If so what did you change and briefly explain why you think it improved prediction accuracy.

Deliverables

1. **(10 pt) weka_output.txt** - a text file containing the results you calculate in Task 2: Q1-3. This should contain 3 distinct confusion matrices and 3 distinct classifier accuracies.

2. **(10 pt) weka_discussion.txt** - a text file containing the discussion from Task 2: Q4.

Submission details

Submit the deliverables as individual files on the t-square submission site. 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. If you use slip days, please specify the number of days used in the box as well.

Please adhere to the 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 delayed resubmission will be counted as a late submission.