Final Lab Option #2 - K-Nearest Neighbor

Acknowledgment: This homework is adapted from **Adam Eck's** work.

This lab does not have interim due dates. I strongly suggest that you and your partner make a timeline and stick to it.

Before you start, you should retrieve `lab_e_knn_data` from the `~barbeda/cs365-share` directory. That contains the data you will test with.

Part 1 (The Only Part)

Data Sets

For this assignment, we will learn from four predefined data sets. Versions of these data sets are in the directory indicated above; you will not need to get them yourself:

1. monks1: A data set describing two classes of robots using all nominal attributes and a binary label. This data set has a simple rule set for determining the label: if head_shape = body_shape ∨ jacket_color = red, then yes, else no. Each of the attributes in the monks1 data set are nominal. Monks1 was one of the first machine learning challenge problems (http://www.mli.gmu.edu/papers/91-95/91-28.pdf). This data set comes from the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/MONK%27s+Problems
2. iris: A data set describing observed measurements of different flowers belonging to three species of Iris. (This data set should sound familiar.) The four attributes are each continuous measurements, and the label is the species of flower. The Iris data set has a long history in machine learning research, dating back to the statistical (and biological) research of Ronald Fisher from the 1930s (for more info, see https://en.wikipedia.org/wiki/Iris_flower_data_set). This data set comes from Weka 3.8: http://www.cs.waikato.ac.nz/ml/weka/ and is also on the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/iris
3. mnist_small.csv: A data set of optical character recognition of numeric digits from images. Each instance represents a different grayscale 28x28 pixel image of a handwritten numeric digit (from 0 through 9). The attributes are the intensity values of the 784 pixels. Each attribute is ordinal (treat them as continuous for the purpose of this assignment) and a nominal label. This version of MNIST contains approximately 50 instances of each handwritten numeric digit, sampled from the original training data for MNIST. The overall MNIST data set is one of the main benchmarks in machine learning: http://yann.lecun.com/exdb/mnist/.
4. mnist_big.csv: The same as mnist_small, except containing more instances of each handwritten numeric digit.

The file format for each of these data sets is as follows:

- The first row contains a comma-separated list of the names of the label and attributes
- Each successive row represents a single instance
- The first entry (before the first comma) of each instance is the label to be learned, and all other entries (following the commas) are attribute values.
- Some attributes are strings (representing nominal values), some are integers, and others are real numbers. Each label is a string.

Program

Your assignment is to write a program that behaves as follows:

1. It should take as input five parameters:
    a. The path to a file containing a data set
    b. The value of k to use in k-Nearest Neighbor
    c. The percentage of instances to use for a training set
    d. A random seed as an integer

    For example, I might run `python3 knn mnist_100.csv 1 0.75 12345` which will perform 1-Nearest Neighbor on mnist_100.csv using the Euclidean distance function and a random seed of 12345, where 75% of the data will be used for training. The remaining 25% will be used for testing.
2. To begin execution, the program should read in the data set as a set of instances
3. Second, the instances should be randomly split into training and test sets (using the percentage and random seed input to the program)
4. Next, predictions should be made for each instance in the test set created in Step 3 using the training set as the instances to compare to in k-Nearest Neighbor.
5. A confusion matrix should be created based on the predictions made during Step 4, then the confusion matrix should be output as a file with its name following the pattern: results_<DataSet>_<k>_<Seed>.csv (e.g., results_monks1_1_12345.csv).

Program Output

The file format for your output file should be as follows:

- The first row should be a comma-separated list of the possible labels in the data set, representing the list of possible predictions of the classifier. This row should end in a comma.
- The second row should be a comma-separated list of the counts for the instances predicted as the different labels whose true label is the first possible label, ending with the name of the first possible label (and not a final comma).

- The third row should be a comma-separated list of the counts for the instances predicted as the different labels whose true label is the second possible label, ending with the name of the second possible label (and not a final comma).
- Etc. for the remaining possible labels

For example, the confusion matrix:

```
Predicted Label
Yes   No
200   100 Yes    Actual
50    250 No     Label
```

would be output as:

```
Yes,No,
200,100,Yes
50,250,No
```

The output for your program should be consistent with the random seed. That is, if the same seed is input twice, your program should learn the exact same tree and output the exact same confusion matrix. You are free to also output other files, too, if you wish (e.g., a file describing the learned tree). These should be explained in your README.

Report Part 1

1. Pick a single random seed and a single (sensible) training set percentage and run k-Nearest Neighbor with a k = 1 on each of the four data sets. What is the accuracy you observed on each data set? Include the terminal commands you used to run each of these in your README.
2. How did your accuracy compare between the mnist_small and mnist_large data sets? Which had the higher average?
3. Pick one data set and three different values of k. Run the program with each value of k on that data set and compare the accuracy values observed. Did changing the value of k have much of an effect on your results? Speculate as to why or why not that observation occurred? Include the terminal commands you used to run each of these in your README.
4. Devise at least one interesting question you wish to know the answer to, distinct from the above, and run experiments to provide evidence to answer it. This might involve using a different distance function, investigating timing issues, or anything else. You will likely need to run additional code for this part of the lab; the code you write here should not interfere with what you've written for the earlier parts.

Deliverables

You should submit:

- Code file(s) that behave as specified above.
- A Report that contains the material from Report Part 1