Assignment #1: k-Nearest Neighbor
CSCI 374 Fall 2017 Oberlin College
Due: Monday September 18 at 11:59 PM

Background

Our first assignment this semester has two main goals:

1. Implement and evaluate our first machine learning algorithm, and
2. Practice using GitHub for source code management

Gitting Started

To begin this assignment, you will need to create a GitHub account if you don't already have one, as well as download the code for this assignment. Instructions can be found in the **Introduction to GitHub** document on the class webpage (under Handouts).

Once you have a GitHub account, you can get started on the assignment by following this link: https://classroom.github.com/a/_Yft2rE0

Data Sets

For this assignment, we will learn from four pre-defined data sets:

1. **monks1.csv**: 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.csv**: A data set describing observed measurements of different flowers belonging to three species of *Iris*. 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_100.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 100 instances of each handwritten numeric digit, randomly 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/. It was converted to CSV file using the python code provided at: https://quickgrid.blogspot.com/2017/05/Converting-MNIST-Handwritten-Digits-Dataset-into-CSV-with-Sorting-and-Extracting-Labels-and-Features-into-Different-CSV-using-Python.html

4. **mnist_1000.csv**: The same as mnist_100, except containing 1000 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 called **knn** that behaves as follows:

1) It should take as input five parameters:
   a. The path to a file containing a data set
   b. The name of the distance function to use, from the set {H, E} (where H stands for Hamming and E stands for Euclidian)
   c. The value of *k* to use in k-Nearest Neighbor
   d. The percentage of instances to use for a training set
   e. A random seed as an integer

   For example, if I wrote my program in Python 3, I might run

   ```
   python3 knn mnist_100.csv E 1 0.75 12345
   ```

   which will perform 1-Nearest Neighbor on mnist_100.csv using the Euclidian distance function and a random seed of 12345, where 75% of the data will be used for training (and 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 decision tree. **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).

Programming Languages

I would recommend using either the **Java** or **Python** programming languages to complete this assignment. If you have a different preferred language, please talk to me to make sure that I will be able to run your submission in that language.

Questions

Please use your program to answer these questions and record your answers in a README file:

1) Pick a single random seed and a single training set percentage (document both in your README) 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?
2) Calculate a 95% confidence interval for the accuracy on each data set.
3) How did your accuracy compare between the mnist_100 and mnist_1000 data sets? Which had the higher average? Why do you think you observed this result? Did their confidence intervals overlap? What conclusion can we draw based on their confidence intervals?
4) Pick one data set and three different values of k (document both in your README). 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?

Bonus Question (5 points)

5) Pick 10 random seeds (document them in your README file) and rerun k-Nearest Neighbor with a k = 1 on the iris.csv data. Record the average of the accuracy across the 10 runs.

   Next, rerun the program on the same 10 seeds for each pair of attributes in the data set (ignoring the other two attributes not in the pair) and record the average accuracy for each pair across the 10 seeds. Since there are four attributes, there are six possible pairs of attributes (e.g., sepallength-sepalwidth is one pair, so petallength and petalwidth would be ignored for this pair).

   Finally, compare the average accuracy results between all six pairs and the results using all four attributes. Did any pairs of attributes do as well (or better) than learning using all four attributes? Speculate why you observed your results.

README

Within a README file, you should include:

1) Your answers to the questions above,
2) A short paragraph describing your experience during the assignment (what did you enjoy, what was difficult, etc.)
3) An estimation of how much time you spent on the assignment, and
4) An affirmation that you adhered to the honor code

Please remember to commit your solution code, results files, and README file to your repository on GitHub. You do not need to wait to commit your code until you are done with the

assignment; it is good practice to do so not only after each coding session, but maybe after hitting important milestones or solving bugs during a coding session. ***Make sure to document your code***, explaining how you implemented the different components of the assignment.

Honor Code

Each student is to complete this assignment individually. However, students are encouraged to collaborate with one another to discuss the abstract design and processes of their implementations. For example, please feel free to discuss the pseudocode for the learning algorithm. You might also want to discuss the processes used to generate the training and test sets from the read in data set. Or, you might need to discuss how to work with the input and output files. At the same, since this is an individual assignment, no code can be shared between students, nor can students look at each other's code.