MSAI-349, Fall 2024
Homework #2: K-Nearest Neighbors and K-Means
Due Date: Tuesday, October 29th @ 11:59PM
Total Points: 10.0

In this assignment, you will work with your group to implement distance metrics, KNN and K-Means classifiers. You may discuss the homework with other groups but do not take any written record from the discussions. Also, do not copy any source code from the Web.

Part I

In this part of the assignment, you will be working with the MNIST handwritten digits dataset. In this dataset, the digits zero through nine are represented as a series of integers on a 28x28 grid, where the integers provide the grayscale intensity. From this data we constructed `mnist_train.csv`, `mnist_valid.csv` and `mnist_test.csv` splits with 2000, 200 and 200 observations, respectively. Each file contains one observation per row with a 'label' followed by 784 'grayscale' intensity values. A short program `starter.py` illustrates how to read these files and display the digit as 'bits' or 'intensity'.

Additional information on the MNIST dataset can be found here: http://yann.lecun.com/exdb/mnist/. This site includes several academic papers on performing classification with the full version of these datasets.

Steps to complete Part I of the assignment

1. (1.0 points) Implement functions to compute Euclidean Distance, Cosine Similarity, Pearson Correlation and Hamming Distance between two input vectors ***a*** and ***b***. These functions should return a scalar float value. To ensure your functions are implemented correctly, you may want to construct test cases and compare them against results packages like `numpy` or `sklearn`.
2. (2.0 points) Implement a k-nearest neighbors classifier for Euclidean distance and Cosine Similarity using the signature provided in `starter.py`. This algorithm may be computationally intensive. To address this, you must transform your data in some manner (*e.g.,* dimensionality reduction, mapping grayscale to binary, dimension scaling, etc.) -- the exact method is up to you. This is an opportunity to be creative with feature construction. Similarly, you can select your hyper-parameters (e.g., *K*, the number of observations to use, default labels, etc.). Please describe all of your design choices and hyper-parameter selections in a paragraph. Once you are satisfied with the performance on the validation set, run your classifier on the test set and summarize results in a 10x10 confusion matrix for each distance metric. Analyze your results in another paragraph.

3. (2.0 points) Implement a k-means classifier in the same manner as described above for the k-nearest neighbors classifier. The labels should be ignored when training your k-means classifier. Present a quantitative metric to measure how well your clusters align with the labels in `mnist_test.csv`. Describe your design choices and analyze your results in about one paragraph each.

Part II

In this part of the assignment, you will build a recommender system for the MovieLens dataset (see https://grouplens.org/datasets/movielens/100k/). This dataset contains approximately 100K observations, of which about 1,000 users rate 1,700 movies on a 1-to-5 scale. User demographic data is also available. The original dataset files for reviews, genre, titles, demographics and user occupations that we combined into a single training file `movielens.txt`. We also created separate training, validation and test sets for users a, b, and c (see `train_{a,b,c}.txt`, `valid_{a,b,c}.txt`, `test_{a,b,c}.txt`,).

Your recommender system should use collaborative filters. Collaborative filters are essentially how recommendation algorithms work on sites like Amazon ("people who bought *x* also bought *y*") and Netflix ("you watched *x*, so you might also like *y*"). They work by comparing distances between users. If two users (or sets of users) are similar, then items that one user has seen and liked but the other hasn't seen are recommended to the other user.

Steps to complete Part II of the assignment

4. (3.0 points) Using one (or more) of the distance metrics implemented in Question #1 above to build a collaborative filter using ***movie ratings only*** on `movielens.txt` to recommend movies for users a, b and c using the `train_{a,b,c}.txt`. The number of users to consider *K* and the number of movies to recommend *M* are hyper-parameters, among others, that you can tune on `valid_{a,b,c}.txt`. Please describe how your collaborative filter works, list the hyper-parameters and describe their role. Report *precision*, *recall*, and the *F1-score* on the validation and test sets for users a, b, and c. Discuss how *M* impacts your results.
5. (2.0 points) Try to improve your collaborative filter built in Question #5 by using movie `genre` or user demographic data (*e.g.*, `age`, `gender` and `occupation`). Report *precision*, *recall*, and the *F1-score* on the validation and test sets for users a, b, and c. Discuss your approach and whether or not considering additional features improved the performance of your collaborative filter.

Please note that several aspects of this assignment are deliberately vague. This is to give you some experience with the issues you will likely encounter when working on your Final Projects. You may need to make simplifying assumptions about your data, devise workarounds for computation bottlenecks or balance time spent on hyper-parameter tuning versus results.

Submission Instructions

Turn in your homework as a single zip file, in Canvas. Specifically:

1. Create a single pdf file hw2.pdf with the answers to the questions above and summaries of your results.
2. Create a single ZIP file containing:
    - `hw2.pdf`
    - All of your `.py` code files
3. Turn the zip file in under Homework #2 in Canvas.

Note: You may also complete the assignment and include all responses in a single Jupyter Notebook in a single ZIP file.

Good luck, and have fun!