Project 1

Due Date: Friday, October 11, 2019 at 11:59 PM.

Instructions:

1. Please use the directory name 'project1' while submitting the project via gsubmit. For details, refer to Piazza instructions.
2. The 2-page report on the project must be in `.pdf` format.
3. **DO NOT** submit the dataset along with your source code.
4. Source codes will be checked for plagiarism using automated tools.
5. Clarity and efficiency of source codes will be taken into consideration during grading.

Overview: In this project, we will explore clustering and experiment with high-dimensional data. We will be using a movies dataset available under the General Resources at `https://piazza.com/bu/fall2019/cs565/resources`. In this dataset, you will find data corresponding to 4,803 movies and you are asked to implement popular algorithms like *k-means* and *k-means++* clustering algorithms *efficiently* in order to discover the hidden structure (if any). You may use a programming language of your choice for the implementation. However, irrespective of the programming language used, your program must accept command line arguments in the specific format described below. The output of your program should be a `.csv` file with a specific format described below, for automated grading. Please refer to the 'Deliverables' section for further details.

Evaluation: There are two main components that shall be evaluated.

- A source code file containing your implementations. The source code should be self-sufficient (external library dependencies will be resolved by the grader). Source code that does not compile/execute as per the instructions below or contains unresolved bugs will lose all points for the implementation. There are points reserved for the readability of the code. **(80 points)**
- A 2-page project write-up in `.pdf` format to be submitted along with the code. Write-ups longer than 2 pages will not be graded. **(20 points)**

Budget: int
Genre: JSON (todo: parsing)
Homepage: url (string)
id: int (don't need in distance func?)
keywords: JSON (todo: parsing)
original_language: conv to 1-hot encoding
original_title: string
overview: string (future todo: consider hashing?)
popularity: double
production_companies: JSON (todo: parsing)
production_countries: JSON (todo: parsing)
release_date: DD-MM-YY -> convert into some n in range(0, 100)
revenue: int
runtime: int
spoken_languages: JSON (todo: parsing)
status: String (todo: 1 hot encoding)
tagline: string
title: string
vote_average: double
vote_count: int

:ontains information about 4. The attributes are as follows:

- budget (USD)
- genres (JSON string)
- homepage (URL)
- id (movie identifier)
- keywords (JSON string)
- original_language
- original_title
- overview
- popularity
- production_companies (JSON string)
- production_countries (JSON string)
- release_date
- revenue
- runtime
- spoken_languages (JSON string)
- status
- tagline
- title
- vote_average
- vote_count

Some of the attributes are real valued, while others are character strings. Since some of the attributes are in form of a JSON string, it would be advisable to convert such attributes into vectors of boolean attributes.

Exercise 1 (20 points): Implement *k-means* clustering algorithm. Your implementation should accept two arguments.

- X denoting the n data points, each represented as a real valued vector of attributes.
- k denoting the number of clusters into which the data points are to be clustered into.

The distance function should be L_2^2 norm. You may choose an appropriate constant (e.g. 1e-2) for the convergence condition.

Exercise 2 (20 points): Modify the implementation of *k-means* clustering algorithm to use the *k-means++* criterion of initialization. In order to do this, the implementation of Exercise 1 should accept an additional argument.

- $init$ - which can take string values "random" or "k-means++". When $init$=="random", the output should be that of *k-means* clustering. However, when $init$=="k-means++", the *k-means++* initialization criterion should be used for selecting the cluster centers.

Using the above implementation, cluster the movies from the provided dataset and assign appropriate cluster label to each movie. In your write-up, please provide and justify the appropriate value

of k that clusters the data points reasonably well. In order to find this value of k, you may plot the value of the cost function over a range of values of k, for both *k-means* and *k-means++* separately. Describe in the write-up any insights you may discover during this process.

Exercise 3 (10 points): In this exercise, we explore a popular dimensionality reduction technique known as *Principal Component Analysis*. You are not required to implement the PCA algorithm. For Python, the **scikit-learn** module provides an implementation of PCA - `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`. Now, we undertake a series of steps described below.

1. For each movie in the dataset, create a new attribute 'total_votes' as the product of the attributes 'vote_average' and 'vote_count'. Using this attribute, filter out a new dataset consisting of top 250 movies with highest values of 'total_votes'.

2. Use the appropriate clustering algorithms implemented in the previous exercise on this new dataset to assign each of these 250 movies with cluster labels.

3. Use the **fit_transform** method of the PCA implementation from the **scikit-learn** module and reduce the dimensions of each movie to 2.

Plot each movie on a 2D scatter plot using the values of 2 latent attributes found in the previous step. Furthermore, color each data point in the scatter plot with a color corresponding to the movie's cluster label from the step 2 above. Describe this plot in your write-up.

Exercise 4 (20 points): Implement 1-dimensional K-means clustering as covered in the lectures. In order to do so, we undertake the following steps:

1. Reduce the number of dimensions of each movie to 1 using the approach from the previous exercise.

2. Implement 1-dimensional k-means clustering algorithm as covered in the lectures.

Furthermore, we augment the implementation of Exercise 1 to accept an additional value of *init* argument. When, *init*=="1d", the 1-dimensional k-means clustering should be used for selecting the clusters. Using this implementation, cluster the movies from the provided dataset and assign appropriate cluster label to each movie. In your write-up, please provide and justify the appropriate value of k that clusters the data points reasonably well.

Excecise 5 (10 points): In this exercise, we compare the k-means++ clustering with 1-dimensional k-means clustering. In order to do so, we use the output cluster label assignments from Exercise 1 and Exercise 4, using the appropriate values of k found earlier for each of the two approaches. If we denote the k-means++ clustering by C_0 and 1-d k-means clustering by C_1, then we compute the disagreement distance between these two clusterings as follows.

For any two movies x and y,

$$I_{C_0,C_1}(x,y) = \begin{cases} 1 & \text{if } C_0(x) = C_0(y) \text{ and } C_1(x) \neq C_1(y) \\ & \text{or } C_0(x) \neq C_0(y) \text{ and } C_1(x) = C_1(y) \\ 0 & \text{otherwise} \end{cases}$$

Disagreement distance between the two clusterings is $D(C_0, C_1) = \sum_{x,y} I_{C_0,C_1}(x,y)$. Please provide the computed disagreement distance in your write-up.

Deliverables:

1. Strictly 2-page write-up submitted in `.pdf` format answering the questions from the exercises.

2. Source code of your implementations. Your code should be executable from command line with first argument as the path to the `movies.csv` file, second argument as number of clusters, and third argument as a string which can be either "random", "k-means++" or "1d". For example, a python source code should be executable with the command:

```
python source.py </path/to/movies.csv> <k> <init>
```

   On executing this command, your program should create a file named **output.csv** in the same directory. The format of output.csv is as follows:

   - The first line should be a header line with columns names– `id,label`.
   - Subsequent lines should contain actual movie id and the assigned cluster label. There should be a separate line for each movie in the dataset.

   A sample output file can be found under General Resources at `https://piazza.com/bu/fall2019/cs565/resources`.

We really encourage using Piazza forums for discussion and queries. Happy programming!