DM Models 2

Task 1 Algorithmic Analysis K-Means Clustering with Real World Dataset

First, download a simulated dataset: kmeans_data.zip from Modules->Datasets. Then, implement the K-means algorithm **from scratch**. K-means algorithm computes the distance of a given data point pair. Replace the distance computation function with Euclidean distance, 1-Cosine similarity, and 1 – the **Generalized** Jarcard similarity (refer to: https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/jaccard.htm).

Q1: Run K-means clustering with Euclidean, Cosine and Jarcard similarity. Specify K= the number of categorical values of y (the number of classifications). Compare the SSEs of Euclidean-K-means, Cosine-K-means, Jarcard-K-means. Which method is better? (10 points)

Distance Metric	SSE
Jaccard Distance	25417280944.387558
Cosine Distance	25419787879.166927
Euclidean Distance	25500254686.910610

Comparing the SSEs, we can see that the method using Jaccard distance is better as it gives the minimum SSE value.

Q2: Compare the accuracies of Euclidean-K-means Cosine-K-means, Jarcard-K-means. First, label each cluster using the majority vote label of the data points in that cluster. Later, compute the predictive accuracy of Euclidean-K-means, Cosine-K-means, Jarcard-K-means. Which metric is better? (10 points)

Distance Metric	Accuracy
Cosine Distance	61.34 %
Jaccard Distance	60.34 %
Euclidean Distance	58.74 %

Comparing the accuracy, we can see that Cosine distance has a better accuracy.

Q3: Set up the same stop criteria: “when there is no change in centroid position **OR** when the SSE value increases in the next iteration **OR** when the maximum preset value (e.g., 500, you can set the preset value by yourself) of iteration is complete”, for Euclidean-K-means, Cosine-K-means, Jarcard-K-means. Which method requires more iterations and times to converge? (10 points)

Stop Criteria for Euclidean	No. of Iterations	Time Taken (in sec)
Increase in SSE value	41	63.09
No change in centroid position	41	68.66
Maximum preset value	500	758.43

For Euclidean distance, setting the stop criteria as “Maximum preset value” requires more iterations and hence more time to converge.

Stop Criteria for Cosine	No. of Iterations	Time Taken (in sec)
Increase in SSE value	34	41.75
No change in centroid position	92	113.18
Maximum preset value	500	610.63

For Cosine distance, setting the stop criteria as “Maximum preset value” requires more iterations and hence more time to converge.

Stop Criteria for Jaccard	No. of Iterations	Time Taken (in sec)
Increase in SSE value	49	99.79
No change in centroid position	73	159.43
Maximum preset value	500	1088.74

For Jaccard distance, setting the stop criteria as “Maximum preset value” requires more iterations and hence more time to converge.

Q4: Compare the SSEs of Euclidean-K-means Cosine-K-means, Jarcard-K-means with respect to the following three terminating conditions: (10 points)

- when there is no change in centroid position
- when the SSE value increases in the next iteration
- when the maximum preset value (e.g., 100) of iteration is complete

Stop Criteria for Euclidean	SSE
No change in centroid position	25500254686.910610
Increase in SSE value	25500254686.910610
Maximum preset value	25500254686.910610

Stop Criteria for Cosine	SSE
No change in centroid position	25419787879.166927
Maximum preset value	25419787879.166927
Increase in SSE value	25435758379.335598

Stop Criteria for Jaccard	SSE
Increase in SSE value	25412820978.778763
No change in centroid position	25417280944.387558
Maximum preset value	25417280944.387558

Q5: What are your summary observations or takeaways based on your algorithmic analysis? (5 points)

Overall, Jaccard-K-means performs well compared to Euclidean-K-means and Cosine-K-means based on SSE minimization and accuracy although it takes a longer time to converge. This suggests that Jaccard-K-means is well suited for this dataset. However, the choice of a suitable distance measure and stop criteria may depend on specific characteristics of the dataset and the problem in hand.

Code: https://github.com/Viknesh-Rajaramon/Data-Mining/blob/master/HW3/HW3_Task1.ipynb

Task 2, Machine Learning with Matrix Data for Recommender Systems

1. Recommender systems are a hot topic. Recommendation systems can be formulated as a task of matrix completion in machine learning. Recommender systems aim to predict the rating that a user will give for an item (e.g., a restaurant, a movie, a product).
2. Download the movie rating dataset from: https://www.kaggle.com/rounakbanik/the-movies-dataset. These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages. This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.
3. Building a small recommender system with the matrix data: "ratings small.csv". You can use the recommender system library: Surprise (http://surpriselib.com), use other recommender system libraries, or implement from scratches.
    a. Read data from "ratings small.csv" with line format: 'userID movieID rating timestamp'.

    b. MAE and RMSE are two famous metrics for evaluating the performances of a recommender system. The definition of MAE can be found via: https://en.wikipedia.org/wiki/Mean_absolute_error. The definition of RMSE can be found via: https://en.wikipedia.org/wiki/Root-mean-square_deviation.

c. Compute the average MAE and RMSE of the Probabilistic Matrix Factorization (PMF), User based Collaborative Filtering, Item based Collaborative Filtering, under the 5-folds cross-validation (10 points)

d. Compare the **average (mean)** performances of User-based collaborative filtering, item-based collaborative filtering, PMF with respect to RMSE and MAE. Which ML model is the best in the movie rating data? (10 points)

	MAE	**RMSE**
Probabilistic Matrix Factorization (PMF)	0.689336	0.895424
Item based Collaborative Filtering	0.721213	0.935395
User based Collaborative Filtering	0.743733	0.966954

From the above table, we can see that Probabilistic Matrix Factorization (PMF) model is the best since it has the lowest average MAE and average RMSE.

e. Examine how the cosine, MSD (Mean Squared Difference), and Pearson similarities impact the performances of User based Collaborative Filtering and Item based Collaborative Filtering. Plot your results. Is the impact of the three metrics on User based Collaborative Filtering consistent with the impact of the three metrics on Item based Collaborative Filtering? (10 points)

f. Examine how the number of neighbors impacts the performances of User based Collaborative Filtering and Item based Collaborative Filtering? Plot your results. (10 points)

g. Identify the best number of neighbor (denoted by K) for User/Item based collaborative filtering in terms of RMSE. Is the best K of User based collaborative filtering the same with the best K of Item based collaborative filtering? (10 points)

The best number of neighbor for User based Collaborative Filtering does not match with the best number of neighbor for Item based Collaborative Filtering.

Code: https://github.com/Viknesh-Rajaramon/Data-Mining/blob/master/HW3/HW3_Task2.ipynb