Programming Assignment 2
(Advanced Data Analytics DS-503)

Instructions: This assignment can be done in pairs (group of 2 students). Each student in the group must understand the code and techniques used and be able to answer each question (don't divide the problems amongst yourselves). It is fine to use web-resources and help, provided you reference them and understand the code. Please don't copy blindly.

Due Date: Sep 26, 2021 (End of day)

Late Days: You have total of 5 late days for the semester for all homework and assignments, which you can use as you need them.

Submission: Submit your solution for each question as a separate Jupyter or Colab notebook. You can also submit a word or .pdf file that explains the solutions. Please use the headings and comments appropriately so we can understand your solution. Use of Matlab or other programming languages like C, C++, Java, R is also permitted, provided the code is well documented. We may ask you for a demo. Good solutions will be posted on the Course Git-hub.

Zip all your code and send to Aman along with your names and roll number in one zip file. You can also send him a Google drive link with appropriate permissions.

(40 marks) Problem 1: *Collaborative Filtering: Recommendation Systems*

DATASETS

You are given a dataset that contains the ratings of 100 items by 20,000 users who have rated 36 or more items, a matrix with dimensions 20000 X 101. The ratings are between -10 and 10. Not every item has been rated and we would like to use Collaborative Filtering to "guess" the rating of items which have not been rated yet. This problem is also known as matrix completion.

Format:

1. Data files are in .zip format, when unzipped, they are in Excel (.xls) format
2. Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").
3. One row per user
4. The first column gives the id of the user. The next 100 columns give the ratings for items 01 - 100.
5. The sub-matrix including only columns {5, 7, 8, 13, 15, 16, 17, 18, 19, 20} is dense. Almost all users have rated those items.

To evaluate the efficacy of your approach, we have prepared a testing set. In the training set, include the full ratings of the first 20,000 users that has been provided to you.

In the test set, we will give you 2000 users with 30 ratings per user and mark all the other ratings as unknown.

For your algorithm development, a validation set of 2000 users similar to the test set will also be provided to you. Validation set has 2 parts. 30 ratings are randomly revealed for each user. You must estimate the remaining 70 ratings. The ground truth contains the actual ratings on some of the items. Compute the RMSE (root mean square error) compared to the ground truth. Those ratings which are unknown (99) in the ground truth don't generate any loss.

Techniques you must implement:

1. Pseudo-Inverse Approach (10 marks)

Assume that we have to guess the rating of an item for a given user X. We know the rating of 30 items for this user (in our validation or test set). Now, we take all the users Y, who have rated all 100 items. We estimate that the ratings of user X are a linear weighted combination of the ratings of these users Y. So, we have an under-determined system with 30 equations and Y variables. Using the pseudo-inverse approach, we can compute the least-square estimate of the weights. Using these weights, we can estimate the ratings of all the 70 items for user X as weighted combinations of the ratings of users Y.

Nearest Neighbor approach: Item-Item Collaborative Filtering (10 marks)

In this approach, the rating of an item is estimated using the ratings of similar items. Assume that we have to guess the rating of an item "i" for a given user X. We first find K (e.g K=3) other items that have been rated by X and are most similar to item "i". Similarity can be computed using the Pearson Correlation Coefficient (take dot product after subtracting the mean). Using the correlation coefficients as weights, we can estimate the rating as the weighted average of the known ratings of items by that user. We also adjust for the deviation of the average ratings of the item “i” and user “X” from the overall average rating of all the items in the dataset.

You can read section 9.3 of the MMDS book or

https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d

https://towardsdatascience.com/item-based-collaborative-filtering-in-python-91f747200fab

Matrix Factorization approach (Latent Factors) (20 marks)

https://pypi.org/project/PyMF/

https://github.com/gbolmier/funk-svd

https://nimfa.biolab.si/nimfa.examples.recommendations.html

Use any of the above libraries to factorize the rating matrix R, into a product of 2 matrices; P and Q. P contains the latent user-interests and Q the item latent factors. You can use the training set to learn the parameters and apply it to the test set to make the predictions of ratings.

By limiting the size of P and Q, you can essentially impose the low-rank constraint. Compute the rank of the matrix PQ produced by different approaches. Plot the quality of your results vs. rank.

Note: We plan to have a leader-board for this problem after all the assignment submissions are completed. You can simply report your validation accuracy on the course WhatsApp group to keep other teams informed. The group with highest score will get full 10 marks for the class participation in the course.

(30 marks) Problem 2: *Network Flow Summaries*

Dataset: https://www.kaggle.com/jsrojas/labeled-network-traffic-flows-114-applications

The data presented here was collected in the network from Universidad Del Cauca, Popayán, Colombia by performing packet captures at different hours, during morning and afternoon, over different days between April and June of 2019. A total of 2,704,839 flows instances were collected and are currently stored in a CSV (Comma Separated Values) file.

Total Flows: 2,704,839

Each flow should update the following 3 types of sketches/summaries. You must process the data only once (only one pass is allowed).

Reservoir Sampling (10 marks)

Implement the reservoir sampling method taught in the lecture and maintain 10 reservoirs of size 1000 flows each. After processing all the flows, plot the histograms of pktTotalCount, avg_ps and flowDuration. Comment if the histograms are similar across the 10 reservoirs (samples).

If we use any one of these samples as a batch for SGD algorithm to build a regression model to estimate the data consumption by a given user, how good would it be? Can you think of better ways to make batches in the streaming model for SGD (where memory is constrained)?

Estimated Count of the distinct SRC-DST IP Pairs in the dataset using KMV sketch (10 marks)

Implement the KMV sketch taught in the class with k=1000. You can use the following implementation for the hash function provided by python. Make sure you finally map the results to interval (0,1) uniformly. For example, you can divide the hash value by the maximum value possible for the hash. You can also control the value of N by taking modulo on the hash.

https://docs.python.org/3.7/library/functions.html#hash

You can also use the min-heap in python to maintain k smallest values. The operations in min-heap are logarithmic in its size and hence the update operations are very fast.

https://www.geeksforgeeks.org/min-heap-in-python/

Compare the actual error with the error estimate guarantee provided by the sketch. (i.e. How many std. deviations away from the actual is the estimate?)

Extra credit (10 marks): To get more insight, you can maintain 10 different KMV sketches (with random seeds) and use the median as an estimate of the distinct count.

100 Most frequent SRC-DST IP pairs (use MG sketch)

Implement the MG sketch taught in the class so that the maximum error in the estimate is no more than 1%. What will be the size of the table? Print out the top 100 most frequent SRC-DST pairs sorted by their estimate frequency (count) after you have processed the whole stream.

For extra credit (10 marks), you can also implement Space Saving algorithm and compare the results with MG sketch. Do they agree on the top 90 pairs? What does this tell you?

(20 marks) Problem 3: *Spell Checker for Hotel reviews using Bloom Filters*

Spell Checker for hotel reviews using Bloom Filters.

Step 1: Use the list of words provided in NLTK using the following code:

```
import nltk
nltk.download('words')

from nltk.corpus import words
word_list = words.words()
print(f'Dictionary length: {len(word_list)}')
print(word_list[:15])
```

Step 2: Load the following hotel review dataset from Kaggle.

https://www.kaggle.com/ranjitha1/hotel-reviews-city-chennai

Step 3: Add each word in the dictionary to a BloomFilter using the following code. You are also welcome to add more words to the word list/filter based on the needs of your dataset.

```
from bloom_filter import BloomFilter

word_filter = BloomFilter(max_elements= <Fill the size here>)

for word in word_list:
  word_filter.add(word)

word_set = set(word_list)
```

Step 4: (2.5 marks) Compare the sizes of the following data-structures, we have created thus far, using the getsizeof() method!:

1. word_list, a Python list containing the English dictionary (in case insensitive order)
2. word_filter, a Bloom filter where we have already added all the words in the English dictionary
3. word_set, a Python set built from the same list of words in the English dictionary

Explain your observations.

Step 5: (2.5 marks) Compare the time to search for a word in each of the above data structures using the %timeit command.

Explain your observations.

Step 6: (10 marks) Go through each review in the dataset and print out the words which do not appear in the dictionary, along with their frequency in the dataset.

Step 7: (5 marks) Write your ideas of how to improve the spell checker. You don’t have to implement these ideas.

For extra credit (20 marks), you can collect your own dataset of at least 1000 additional reviews from any other city (apart from Chennai) of your choice using web-scraping from any popular travel website and perform a similar spell-check on them. Do you observe any difference in your results on the two datasets?

Example of popular travel websites:

https://www.goibibo.com/hotels/

https://www.trivago.in/