Mining Large Scale Datasets

2024 / 2025

Calendar updates

- Final written exam: **29th of May 2025 – in class**
- Assignment #3A due until **18th of May 2025 23h59**
- Assignment #3B
  - Code delivery date: **4th of June 2025 23h59**
  - Presentation date (5 minutes pitch): **5th of June 2025 – in class**
  - Will be available on the shared folder, under Assignment 3 by the end of the day

Recap

- Recommendation systems deal with users and items.

- A utility matrix offers known information about the degree to which a user likes an item.

- Normally, most entries are unknown, and the essential problem of recommending items to users is predicting the values of the unknown entries based on the values of the known entries.

Recap

- Two Classes of Recommendation Systems:
  - content-based; they measures similarity by looking for common features of the items.
  - collaborative filtering; these measure similarity of users by their item preferences and/or measure similarity of items by the users who like them.

Recap

Similarity of Rows and Columns of the Utility Matrix :

- Collaborative filtering algorithms must measure the similarity of rows and/or columns of the utility matrix.
- Jaccard distance is appropriate when the matrix consists only of 1's and blanks (for "not rated").
- Cosine distance works for more general values in the utility matrix. It is often useful to normalize the utility matrix by subtracting the average value (either by row, by column, or both) before measuring the cosine distance.

Recap

Clustering Users and Items:

- Since the utility matrix tends to be mostly blanks, distance measures such as Jaccard or cosine often have too little data with which to compare two rows or two columns.
- A preliminary step or steps, in which similarity is used to cluster users and/or items into small groups with strong similarity, can help provide more common components with which to compare rows or columns.

Recap

Root-Mean-Square Error:

- RMSE is computed by averaging the square of the differences between predicted ratings and the utility matrix, in those elements where the utility matrix is nonblank. The square root of this average is the RMSE.

Assignment 3B

- Implement a CF algorithm, using the item-item approach, to recommend new movies to users.
  - Use the MovieLens dataset, available from: https://grouplens.org/datasets/movielens/

- Start with the 100,000 ratings (Small) dataset, and afterwards try to apply your methods to the larger datasets (1M, 10M, 20M, 25M).

- You will need to implement an efficient approach for finding the near neighbors needed for predicting new rating (either LSH or clustering).

- Validate your method by leaving out 10% of the available ratings.

Hands-on

- Analyze Assignment 3B and structure the pseudocode that outlines the major steps and reasoning behind what you think might be your implementation

- Use pen and paper

- Alloted time: 20 minutes

1 - Data load

- Download and extract MovieLens 100k dataset
- Load the ratings.csv file into memory (?)
- Ratings Matrix
  - R[user_id][item_id] = rating

2 - Preprocessing

- Normalize ratings for each user
  - (optional, e.g., subtract mean rating per user)

- Create a train-test split
  - Randomly hold out 10% of ratings as TEST_SET
  - Use the remaining 90% as TRAIN_SET

3 - Build item-item similarity matrix

- Compute item profiles:
  - For each item I:
    - Construct vector V_I: all users' ratings for item I (sparse)
  - Option A: USE LSH for approximate nearest neighbors
    - ...
  - Option B: USE CLUSTERING (e.g., k-means)
    - Treat each item's rating vector as a feature vector
    - Cluster items into K clusters
    - For item I, similar items are those in the same cluster
- For each item I:
  - Find Top-N similar items using cosine similarity
    - (restricted to neighbors from LSH or clustering)
  - Store similarity scores: SIM[I][J] = similarity(I, J)

4 - Predict

- For each (user U, item I) in TEST_SET:
  - Compute predicted rating
    - Numerator = SUM over N: similarity(I, J) * rating(U, J)
    - Denominator = SUM over N: similarity(I, J)
  - Store predicted_rating

5 - Evaluate

- Compare predicted ratings with actual ratings in TEST_SET
  - Compute RMSE (Root Mean Squared Error)

6 - Scale

- Repeat steps 1–5 with MovieLens 1M, 10M, 20M, and 25M datasets

- Consider parallelization/distributed computation if needed