Project 4: Movie Recommender System

Fall 2024

Contents

MovieLens Dataset

The dataset comprises approximately 1 million anonymous ratings for 3,706 movies, provided by 6,040 MovieLens users who joined the platform in 2000.

You can find some insights from our exploratory data analysis: [Rcode_W13_Movie_EDA.html] [Python_W13_Movie_RS.html]

You can download a copy of the 6040-by-3706 rating matrix in CSV format, complete with column names ("m" + MovieID) and row names ("u" + UserID), from Coursera/Canvas.

Submission Requirements

To complete this assignment, please provide the following:

1. An R Markdown or Python Jupyter Notebook saved in **HTML** format, or a link to such a file. This file should contain all the necessary code to replicate the reported results. There is no page limit for this part.
2. A web link to your movie recommendation application built by your team. You may share the source code link or submit the code as a zip file on Coursera/Canvas.

It's important to note that you cannot utilize any recommender packages from R or Python. However, you are free to use other packages as needed.

HTML File (4 points)

The HTML file should contain two key components:

System I: Recommendation Based on Popularity

Recommend the **top ten** most popular movies. Please clearly define what you mean by "most popular." For example, are you considering movies with a high number of reviews as popular, or are you using additional criteria, such as counting only reviews above a specific threshold, or focusing on movies whose average or median rating exceeds a certain threshold (i.e., excluding movies with a significant number of low ratings from being classified as popular)?

There is no single correct answer. We are primarily interested in ensuring that your implementation aligns consistently with your definition.

> Provide the code for implementing your recommendation scheme and display the top ten movies, including their `MovieID` (or "m" + MovieID), `title`, and poster images.

System II: Recommendation Based on IBCF

For this system, follow these steps. Let R denote the 6040-by-3706 rating matrix.

1. Normalize the rating matrix by centering each row. This means subtracting row means from each row of the rating matrix R. Row means should be computed based on non-NA entries. For instance, the mean of a vector like `(2, 4, NA, NA)` should be 3.
2. Compute the (transformed) Cosine similarity among the 3,706 movies. For movies i and j, let $\mathcal{I}_{ij}$ denote the set of users who rated both movies i and j. We decide to ignore similarities computed based on less than three user ratings. Thus, define the similarity between movie i and movie j as follows, when the cardinality of $\mathcal{I}_{ij}$ is bigger than two,

   $$S_{ij} = \frac{1}{2} + \frac{1}{2}\frac{\sum_{l \in \mathcal{I}_{ij}} R_{li}R_{lj}}{\sqrt{\sum_{l \in \mathcal{I}_{ij}} R_{li}^2}\sqrt{\sum_{l \in \mathcal{I}_{ij}} R_{lj}^2}}$$

   This transformation (1 + cos)/2 ensures that similarity measures are between 0 and 1. NA values may occur when:

   1) the set $\mathcal{I}_{ij}$ has a cardinality less than or equal to two (i.e., this pair of movies have been rated by only zero, one, or two users);
   2) one of the denominators is zero.
3. Let S denote the 3706-by-3706 similarity matrix computed in previous step. For each row, sort the non-NA similarity measures and keep the top 30, setting the rest to NA. This new similarity matrix, still denoted as S, is no longer symmetric. Save this matrix. Note that some rows of the S matrix may contain fewer than 30 non-NA values.

   Display the pairwise similarity values from the S matrix (you obtained at Step 3) for the following specified movies: "m1", "m10", "m100", "m1510", "m260", "m3212". Please round the results to 7 decimal places.
4. Create a function named `myIBCF`:

- **Input**: `newuser`, a 3706-by-1 vector (denoted as w) containing ratings for the 3,706 movies from a new user. Many entries in this vector will be NA. Non-NA values should be integers 1, 2, 3, 4, or 5, as ratings are based on a 5-star scale (whole star ratings only). The order of the movies in this vector should match the rating matrix R. (Should we center w? For IBCF, centering the new user ratings is not necessary.)
- **Inside the function**: Upon receiving this input, your function should load the similarity matrix and use it to compute predictions for movies that have not been rated by this new user yet. Use the following formula to compute the prediction for movie i:

  $$\frac{1}{\sum_{j \in S(i) \cap \{l: w_l \neq NA\}} S_{ij}} \sum_{j \in S(i) \cap \{l: w_l \neq NA\}} S_{ij} w_j$$

  where $S(i) = \{l : S_{il} \neq NA\}$. The formula above is identical to the one on page 10 of [lec_W13_RecommenderSystem.pdf] where the rating for the j-th movie for this new user is denoted as w_j here, but as r_{aj} in lec_W13_RecommenderSystem.pdf. Note that NA values may arise when the denominator equals zero.

- **Output**: Based on your predictions, recommend the **top ten** movies to this new user, using the column names of the rating matrix R (i.e., "m" +`MovieID`).

  If fewer than 10 predictions are non-NA, select the remaining movies based on the popularity defined in System 1, prioritizing the most popular ones and excluding those already rated by the user. Save the ranking of all movies (based on popularity) as a separate file to avoid recomputing the ranking each time.

Test your function

For your function `myIBCF`, print the top 10 recommendations for the following two users:

- User "u1181" from the rating matrix R
- A hypothetical user who rates movie "m1613" with 5 and movie "m1755" with 4.

The App (3.5 points)

Build an application for System II. Here are the requirements:

- Present users with a set of sample movies and ask them to rate them.
- Use the ratings provided by the user as input for your `myIBCF` function.
- Display 10 movie recommendations returned by your `myIBCF` function.

To save space and/or memory, you can choose to display up to 100 movies. This means you can modify your `myIBCF` function slightly so that it only requires 100 columns of the S matrix. You can decide which 100 movies to display. Mention these additional implementation details at the end of your HTML file if applicable, but there's no need to include the modified version of the `myIBCF` function in the HTML file.

We developed an imitation app inspired by a [book recommendation system]. While it appears to gather user data, the 10 recommendations it provides are, in fact, fixed and unchanging, consistently featuring the same initial 10 movies.

https://fengliang.shinyapps.io/MovieRecommend/

Resources

You are welcome to use any existing code, provided you cite the source. For example, you can check how two packages implement IBCF:

- R code for package recommenderlab [https://github.com/mhahsler/recommenderlab/tree/master/R]
- Python code for package Surprise [https://github.com/NicolasHug/Surprise]
- The Github repository for the Book Recommender System mentioned above: [https://github.com/pspachtholz/BookRecommender].

For the App, you can use Shiny if using R. Python users can consider using frameworks like [Shiny], [Dash], or [Flask], and [Streamlit].