#ITCS-6162 #Data-Mining

Michael Gain

Movie Recommendation System Report

1. Introduction

Movie recommendation systems recommend to users what movies to watch based on similar user viewing habits, and by movies that are similar to those watched by the user.

These recommendation systems are important, especially for service providers such as Netflix, because good recommendations keep users on your platform for longer, increasing revenue. If the recommendation system is poor, it is less likely users will not be exposed to movies that they will enjoy. This will likely decrease their use of the service and possibly lead to cancellation of services, thereby reducing revenue.

There are three primary methods of forming recommendations, user-based, item-based, and random-walk.

- User-based
  - This method finds the similarity between two users using one of several different similarity functions such as Jaccard Similarity, Cosine Similarity, or Pearson Correlation Coefficient. The similarity is calculated based on ratings of movies that the two users share. Then the most similar users are used to estimate the rating the target user would give to a movie they have not yet rated.
- Item-based
  - This method compares items based on one or more features to find similar items. The similarity is calculated using the same functions as in User-based, Jaccard Similarity, Cosine Similarity, or Pearson Correlation Coefficient. Then items with the highest similarity are recommended to a user based on the items they have already interacted with/rated.
- Random Walk
  - Random walk works by associating items to users and users to items. Then from some given starting point, either a user or item, randomly select a connected node (user or item). Keep track of the number of visits to each item. Items that are visited more often are more popular, and are thus recommended more. In a user based random walk you start from a user and select an movie they have rated, then pick a user that has rated that movie and repeat for a certain number of steps, then report results. In item based

random walk, start from an item and pick a user that has rated it, then pick an item that user has rated, and repeat for some number of steps and report the results.

2. Dataset Description

- The MovieLens 100K Dataset includes
  - 61 unique users (based on unique user_id)
    - 936 if user_id is not unique
  - 1681 movies
  - 100000 ratings of movies by users
- The users table contains their user_id, age, gender, and occupation.
- The movies table contains the movie_id, title, and release date.
- The ratings table contains the user_id, movie_id, rating, and timestamp.
- To prepare the dataset empty rows were removed, and timestamps, which were given in the Unix epoch, were converted to human readable timestamps.

3. Methodology

3.1 Recommendation Systems Implemented

- User-based collaborative filtering (how user similarity was calculated).
  - Users and their ratings were first organized into a `user_movie_matrix` where users were rows and movies were columns.
  - Similarity was calculated using the cosine similarity using their ratings of movies. Movies they had not rated were set to `0.0`.
  - User-User similarity was organized into the `user_sim_df` Pandas DataFrame. To find the similarity for a user read the column `user_id` and the row corresponded to a different user and their computed similarity.
- Item-based collaborative filtering (how item similarity was determined).
  - Users and their ratings were first organized into a `user_movie_matrix` where users were rows and movies were columns.
  - Similarity was calculated using the transpose of that matrix. The cosine similarity was calculated using each different user's rating of that movies compared to another movie. Movies they had not rated were set to `0.0`.
  - Movie-Movie similarity was organized into the `item_sim_df` Pandas DataFrame. To find the similarity for a movie read the column `movie_id` and the row corresponded to a different movie and their computed similarity.
- Random-walk-based Pixie algorithm (why graph-based approaches are effective).
  - Graph based approaches are effective because they give relationship modeling. The Pixie algorithm, and inspired algorithms, use a bipartite graph associating items to users and user to items. This gives a natural structure to connections and similar items.

    - It also uses a deep search in the form of the random walk and walk limits. This can find non apparent connections between item-item and user-user.
    - They are also personalized to a user by their structure, making recommendations fast and easy.

3.2 Implementation Details

- First the adjacency matrix, `user_movie_matrix` was formed by creating a pivot table of user ratings and movies.
    - Each column is a movie, and each row is a user.
    - The intersection of a user and movie is that user's rating for that movie
    - If a user has not rated a movie, it is left as `NaN`
        - later functions will fill in all `NaN` with `0.0`
- User-based collaborative filtering
    - The cosine similarity of each row (user) was calculated and stored in a new `user_sim_df`
        - To find the similarity between two users, look up a column using a `user_id` and the row `user_id2`
    - Then to form a recommendation
        1. Retrieve the column `user_id` for which you want to generate recommendations
        2. Sort the column by similarity
        3. Remove all users that do not meet a similarity threshold (I used 0.45)
        4. Compute the average of those user's rating for all movies that the user has not not yet rated
        5. Reverse sort the computed expected ratings
        6. Return the first `N` movies as the recommendation
- Item-based collaborate filtering
    - First take the transpose of the `user_movie_matrix`
        - This puts each movie as a row, and each user as a column
        - Compute the cosine similarity of each movie (using the user ratings as feature)
        - The computed similarities are stored in the `item_sim_df` matrix
    - Then to form a recommendation
        1. Select a movie using the `movie_id` from the `item_sim_df`
        2. Reverse sort by similarity
        3. Return the first `N` movies as the recommendations
- Random Walk Based Recommendation
    - First a bipartite graph must be formed between users and movies
        - Create a graph that holds `users` and `movies`.
        - For each `movie_id` rated by a `user_id` add that `movie_id` to that user's rated list, and add that `user_id` to that `movie_id`'s list of raters
    - Now to form recommendations

1. Create an empty dictionary where `movie_id` is the key and `0` is the value. This will be the `visited` list
2. First select a user or movie to start from
3. Randomly select a movie or user (depends on which node type you are coming from)
    1. If you are visiting a movie, increase that movie's visited value by one in the `visited` list
4. Select a connected movie or user (depends on what node type you are coming from)
5. Repeat steps 3-4 for some `walk_length` number of steps
6. After reaching the `walk_length` sort the list by the number of times each movie had been visited
7. Return the first `N` items in the list as the recommendations

5. Results and Evaluation

- User-based Examples
    - `user_id` : 10
        -

Rankings	Movie Name
1	World of Apu, The (Apur Sansar) (1959)
2	Ghost in the Shell (Kokaku kidotai) (1995)
3	Bed of Roses (1996)
4	Year of the Horse (1997)
5	Golden Earrings (1947)

    - `user_id` : 100
        -

Rankings	Movie Name
1	Leading Man, The (1996)
2	Phantom, The (1996)
3	Star Kid (1997)
4	Long Kiss Goodnight, The (1996)
5	Star Trek: First Contact (1996)

- Item-based
    - `movie_name` : Jurassic Park (1993)

Rankings	Movie Name
1	Top Gun (1986)
2	Speed (1994)
3	Raiders of the Lost Ark (1981)
4	Empire Strikes Back, The (1980)
5	Indiana Jones and the Last Crusade (1989)

- `movie_name` : Toy Story (1995)

Rankings	Movie Name
1	Star Wars (1977)
2	Return of the Jedi (1983)
3	Independence Day (ID4) (1996)
4	Rock, The (1996)
5	Mission: Impossible (1996)

- Random-Walk
  - User Based: `user_id` : 10

Ranking	Movie Name
1	GoodFellas (1990)
2	Better Off Dead... (1985)
3	Game, The (1997)
4	Beauty and the Beast (1991)
5	Ref, The (1994)

  - Item-Based: `movie_name` : Jurassic Park (1993)

Ranking	Movie Name
1	Mystery Science Theater 3000: The Movie (1996)
2	Cinema Paradiso (1988)
3	Broken Arrow (1996)
4	Shawshank Redemption, The (1994)
5	Birdcage, The (1996)

- User and Item based both struggle from a cold start problem. If there are not enough users or items, recommendations may be poor or non-existent in the case of all possible recommendations do not meet the threshold.
- Currently User and Item based do not use a weighted average, implementing a weighted average will increase accuracy and usefulness.
- The random walk method is currently a purely random walk, so the stability of recommendations is low. It is difficult to judge the accuracy of recommendations. However the low stability may be a boon, giving fresh recommendations each time. The random walk should increase in accuracy with moderate `walk_length` values because values that are too large run the risk of reaching items that are not good recommendations and or breaking out of a community.

6. Conclusion

These different methods of recommendations provide different benefits. User-based is useful when there is a lot of users, however it has accuracy problems if a user is unique in their tastes. Item-based can still work with unique tastes however not with the features used in this project (the features are still based on user ratings). If different features were used for item-based it would have good accuracy, however gathering features and deciding what are good features can be difficult. The random walk pixie inspired algorithm is good at finding links between users and movies even when they may not be apparent. It is also very efficient and can be used for quick recommendation using multi-threading.
Real world applications for such recommender systems can vary from Netflix and YouTube, to Amazon Shopping, and Pinterest boards.