Collaborative Filtering

Armin Pousti
Dmitrii Vlasov
Mcgill Department of Computer Science

COMP 480
Professor: Joseph Vybihal
June 27, 2023

Collaborative filtering

Collaborative filtering, also known as social filtering, is a technique used in recommendation systems to predict a user's interests by collecting information from a large group of users. In collaborative filtering, the system gathers data on user behavior, such as online purchases, movie reviews, and books and articles read. This data is then used to find similarities or patterns among users and make recommendations on those similarities. Collaborative filtering is commonly used for online recommendations, and it will give an increasingly accurate and relevant suggestion with time. In collaborative filtering, which uses a user-matrix to discover similar users and terms, the main goal is to provide a personalized recommendation to users by making use of the collective wisdom of the community. Collaborative filtering can be divided into two main categories: user-based and item-based collaborative filtering, which can use different similarity measures.

User-based collaborative filtering

This approach finds users with similar preferences to a given user and recommends items that these similar users have liked or purchased. For example, if user A and user B rated several books similarly and user A has not read a particular book that user B rated highly, the system may recommend that book to user A based on user B's positive rating.

User-Based Nearest members (UBNM) and k-Nearest Neighbors (k-NN)

The variation between UBNM and k-NN algorithms is minor, therefore they can be discussed together.

1) Data Collection: The algorithm gathers information on user preferences such as ratings, reviews, purchases, or any other feedback. The data is organized in a user-item matrix, where each row represents a user, each column represents an item, and the cells contain the corresponding user-item interactions (for example ratings).

2) User Similarity Calculation: The algorithm calculates the similarity between users based on their preferences. One common similarity metric used in UBNM and k-NN is cosine similarity. Cosine similarity measures the cosine of the angle between two vectors. Each user's preferences are represented by an n-dimensional vector where n is the total number of items in the dataset. Each element of the vector represents the user's rating

or interactions with a specific item. For example, if there are 1000 items, each user's vector will have 1000 dimensions. In order to calculate the cosine similarity between two users, the algorithm computes the following:

$$cos(A, B) = \frac{A \cdot B}{||A||^* ||B||}$$

The resulting cosine similarity score ranges from -1 to 1. A score of 1 indicates that the two vectors have identical preferences (geometrically, their vectors coincide, pointing in the same direction). A score of -1 indicates completely opposite preferences (their vectors are pointing in opposite directions). A score of 0 suggests no similarity or independence between the user's preferences (their vectors are orthogonal).

3) Nearest Neighbors Selection: Based on the cosine similarity scores, the algorithm selects either k nearest neighbors for a given target user (k-NN). These nearest neighbors are users who have the highest similarity scores with the target user. Alternatively, the algorithm selects neighbors based on the similarity scores between users (UBNM), considering all users and their similarity to the target. The number of neighbors selected is not predetermined and can vary depending on the dataset.

4) Weighted Rating Aggregation: The algorithm combines the ratings of the nearest neighbors to generate recommendations for a targeted user. This aggregation process involves taking into account both the similarity between users and the ratings of the items. One common approach is to calculate a weighted average of the ratings, where the weights are determined by the similarity scores. The ratings of more similar users carry more weight in the aggregation. First, the algorithm retrieves the ratings of the neighbor for the items in consideration, then it multiplies the ratings by the similarity score between the target user and the neighbor. The result is the weighted rating which reflects the contribution of that neighbor's rating to the recommendation for the targeted user. After that, the weighted ratings across all nearest neighbors are summed up to collect aggregate weighted ratings. Based on aggregate ratings, the list of recommended items is then generated.

5) Recommendation Generation: Finally, the algorithm generates a list of recommended items for the targeted user based on the weighted ratings. These items are typically the

ones that have the highest ratings after the aggregate step. The number of recommendations presented to the user can be predetermined or based on a threshold.

UBNM is a simple and intuitive approach that is easy to implement and understand. It works well for small to medium-sized datasets and can capture user preferences based on similar users. k-NN on the other hand, while similar, offers flexibility by allowing the selection of the number of similar users considered for recommendation.

Adjusted Cosine Similarity (ACS)

This approach is functionally the same with minor variations to account for a potential user bias and, therefore, only differences are going to be mentioned in detail.

1) Data Collection: Similar to the previous algorithm
2) Rating Adjustment: For each user, the average rating for all the items are calculated. This average represents the user's rating bias or tendency. Then, the user's average rating is subtracted from the individual ratings in the matrix. This adjustment normalizes the ratings by removing the user's bias and focusing on the relative differences in the ratings.
3) Calculation of Adjusted Cosine Similarity: Similar to regular cosine similarity where the adjusted ratings are used instead.
4) Recommendation Generation: Similar to the previous algorithm

Users' ratings may vary in scale, with some being more lenient or strict compared to others. Additionally, users may exhibit bias towards popular or frequently rated items. The ACS algorithm addresses these factors by focusing on the relative differences in ratings, thereby accounting for biases and rating patterns. This adjustment enables the algorithm to capture the similarity between users based on their relative preferences rather than absolute ratings. However, there are potential issues associated with ACS. Sufficient ratings from users are necessary to accurately calculate average ratings and account for biases. Furthermore, while ACS aims to eliminate certain biases, it may inadvertently introduce new ones. The effectiveness of bias removal relies on the accuracy of average rating calculations and the assumption that adjusting ratings adequately captures biases.

Slope One (SO)

The SO algorithm is based on the concept of item-item collaborative filtering and utilizes the idea of slopes to make predictions. This algorithm involves:

1) Data Preparation: SO organizes the user-item interactions into a matrix, similarly to previous algorithms
2) Calculation of Deviations: For each pair of items, the deviation between ratings is calculated. The deviation represents the average difference in ratings givenby users who have rated both items. To calculate the deviation SO iterates through all pairs of items in the matrix and considers only those pairs for which users have rated both items. For each user who has rated both items A and B, the algorithm calculates the difference between their ratings for item A and item B. The differences are then summed up for all users who have rated both items. After that this sum is divided by the number of users who have rated both items to obtain average deviation. The deviations reflect relative preferences or relationships between items. Positive deviations indicate that users tend to rate item A higher than item B, while negative deviations indicate the opposite. These deviations capture the average differences in ratings between items based on historical user data. By calculating the deviations, the SO algorithms identifies the item-item relationships and preferences. These relationships are then used in the rating prediction step to make recommendations for the target user.
3) Rating Prediction: The calculated average deviations and the target user's ratings are
4) used to predict the rating for the target item. Finally, it aggregates the predictions from all items to obtain the final predicted rating for the target user-item pair.
5) Recommendation Generation: Based on the predicted ratings, the SO generates a list of recommended items for the target user. The recommendations are typically the items with the highest predicted ratings.

The SO algorithm requires minimal data preprocessing as it handles missing values easily and is suitable for large-scale datasets.

Pearson Correlation-Based Recommender Systems (PCRS)

The PCRS algorithm performs in the similar way as the k-NN and UBNN, except for the similarity calculation. Instead of cosine similarity it uses Pearson correlation coefficient. To

calculate the similarity between two users, the algorithm compares their ratings for items they have both rated using the following formula:

$$r = \frac{\sum(R_A - B)(R_B - Y)}{\sqrt{\sum(R_A - B)^2 \sum(R_B - Y)^2}}$$

Where

$r =$ correlation coefficient

R_A , R_B = ratings of User A and User B, respectively, for the common items they have both rated

A, B = means of the ratings of User A and User B, respectively, for the common items

Once the Pearson correlation coefficients are calculated for all pairs of users, the algorithm can select the nearest neighbors based on these similarity scores to make predictions and generate recommendations.

The PCRS measures linear relationships between users or items and provides both the measure of strength and direction of the relationships that can potentially be used outside recommender systems.

Co-Clustering

Co-Clustering is a matrix factorization-based algorithm that aims to simultaneously cluster both users and items based on their rating patterns to uncover underlying structures in the data. This approach can lead to more accurate and interpretable recommendations.

1) Data Preparation: user-item matrix is created
2) Co-clustering starts by applying a clustering algorithm (for example algorithm discussed in module 2 report). This clustering process groups similar users and items together based on their rating patterns. The number of clusters for users and items can be predefined or determined through optimization techniques.
3) Initialization of Cluster Representatives: After clusteringm the algorithm initializes the cluster representatives. These representatives serve as initial estimates for the values within each cluster.

4) Iterative Optimization: in each iteration, the algorithm alternates between optimizing the cluster assignments for users and items and updating the cluster representatives. The cluster representsatives are updated by recomputing the representative values based on the current assignments for users and items within each cluster. The process continues until convergence.
5) Rating prediction: Once the clustering process is complete, CoClustering can make rating predictions for users and items that have missing or unobserved ratings. To predict a missing rating for a user-item pair, the algorithm considers the cluster representatives of the user's cluster and the item's cluster. The prediction is computed based on the relationship between the user's cluster and the item's cluster, leveraging the representative values and the observed ratings within the clusters.
6) Recommendation Generation: Based on the predicted ratings, the algorithm generates recommendations for users by suggesting items with the highest predicted ratings that the user has not yet rated.

Co-clustering captures underlying structures in the user-item interaction data and provides more accurate and interpretable recommendations.

User- User Collaborative Filtering with Binary Feedback

In collaborative filtering, when explicit ratings are unavailable, one technique is collaborative filtering with binary feedback. In the binary feedback technique, feedback of binary nature, such as "liked" or "disliked," is used instead of ratings. This approach focuses on capturing users' positive and negative feedback patterns to recommend items.

1. User-Item Matrix: A user-item matrix is created similar to the previous algorithms
2. Similarity Calculation: Based on users' feedback patterns, a similarity between users is calculated using either Jaccard or cosine similarity.
    a. Cosine Similarity: We have extensively looked into how cosine similarity works
    b. Jaccord Similarity: Jaccord similarity is a measure used between two sets. Jaccord similarity is defined as the size of the intersection of two sets divided by the size of their union, and it works as follows:
        i. Set Representation: A set represents users or items, where elements of the set correspond to the items liked or rated by the user
        ii. Intersection and Union: To calculate the Jaccord similarity between the two sets, A and B, First, the intersection of A and B is calculated since the intersection of A and B contains the mutual elements between the sets.

Now we calculate the union of A and B to find all the unique elements from both sets.

iii. Calculations: The Jaccord similarity is calculated by dividing the size of the intersection by the size of the union:

$$Jaccard\ Similarity(A,\ B)\ = \frac{|A \cap B|}{|A \cup B|}$$

|A ∩ B| represents the number of elements in the intersection of A and B, and |A ∪ B| represents the number of elements in the union of A and B.

iv. Similarity Value: The resulting Jaccard similarity value ranges between 0 and 1. A value of 0 indicates no similarity between the sets, while 1 indicates complete similarity.

Jaccard similarity measures the similarity between users or items based on their preferences or characteristics. The higher the Jaccard similarity, the more significant the overlap in preferences or characteristics, indicating a higher degree of similarity. One limitation stemming from Jaccard's similarity is that it treats each element as equally essential and does not consider the intensity or weight associated with the element or the frequency of occurrence of the elements in the set.

3. Nearest Neighbors: The most similar users to the target user are identified based on the similarity measures. A fixed number of nearest neighbours, denoted as k, are chosen.
4. Aggregation of Feedback: To recommend the feedback from the k nearest neighbours aggregate.
5. Recommendation Generation: As a potential recommendation, the binary techniques suggest the item liked by the nearest neighbours to the target user based on the assumption that users with similar interests are likelier to like similar items.

User-user collaborative filtering with binary feedback leverages users' positive and negative feedback patterns to provide recommendations. It is used when explicit ratings are unavailable, or users' preferences can be adequately captured with binary feedback. However, this approach does not consider the varying levels of preference intensity, as it treats all positive feedback equally.

User Clustering

User clustering is a technique in collaborative filtering which groups similar users together based on their preferences and behaviours. By clustering users, it becomes possible to identify groups of users who exhibit similar patterns of item preferences, enabling more targeted and personalized recommendations.

1. User-Item Matrix: A user-item matrix is constructed similarly to the other algorithms. The matrix is populated with ratings, preferences, or interactions users provide for the items.
2. Similarity Calculation: Similarity for preference or behaviour between users is calculated using similarity measures such as cosine similarity, Pearson correlation coefficient, or Jaccard similarity, which we discussed earlier.
3. Clustering Algorithm: A clustering algorithm is applied to group similar users together. Various clustering algorithms, such as k-means, hierarchical clustering, or density-based clustering, are available, which group users based on the calculated similarity values.

K-means Clustering:

K-means clustering groups data points into K clusters based on their similarities. It is an iterative algorithm that aims to minimize the sum of squared distances between data points and their assigned cluster centroids.

1) Initialization: The algorithm randomly selects K initial cluster centroids which the algorithm can choose from the dataset or generate randomly: c_1, c_2,..., c_k

2) Assignment Step: The algorithm assigns each data point to the nearest centroid based on its similarity measure, typically the Euclidean distance, which calculates the distance between the data point and each centroid.

To calculate the distance of each data point to centroids(Euclidean distance):

$$D(x_i, c) = \sqrt{\sum (x_i - c)^2}$$

The we assign each data point to the nearest centroid:

$$argmin D(x_i, c) = assgined\ cluster\ for\ x_i$$

3. Update Step: After assigning each data point to a centroid, the centroids get updated by calculating the mean of the data points assigned to each cluster. The mean becomes the new centroid position.

Calculating the new centroid where N is the number of data points assigned to cluster j:

$$c = \frac{1}{N} \sum x_i$$

4. Iteration: The second and third steps are repeated until convergence occurs when the centroids no longer move significantly or when a maximum number of iterations is reached.

5. Final Clusters: Each data point is assigned to a final cluster based on the nearest centroid at convergence.
6. Cluster Representation: Centeroids represent each cluster, which can provide insights into the characteristics of the average behaviour of the data points in the cluster.
K-means clustering partitions the data into non-overlapping clusters by minimizing the distance between data points and the centroids of their assigned clusters, and it aims to find cluster centroids that minimize the total within-cluster variance.

Hierarchical Clustering:

Hierarchical clustering is a clustering algorithm that builds ahierarchy of clusters. Hierarchical clustering does not require the number of clusters to be predefined (Unlike k-means clustering), but rather it just recursively merges or splits clusters based on the similarity between data points.
1. Initialization: Each data point represents a separate cluster.
2. Similarity Calculation: A similarity measure, such as Euclidean distance or correlation, calculates the pairwise similarity between data points or clusters.
3. Merge or Split: The two most similar clusters or data points are merged into a single cluster based on the similarity measure. This process is repeated until all data points or clusters are merged into a single cluster, forming a tree-like structure called a dendrogram.
4. Dendrogram Construction: A dendrogram visually represents the hierarchy of clusters, with data points or clusters as leaf nodes and merged clusters as internal nodes. The height of each merging point on the dendrogram represents the similarity level at which the merge occurred.
5. Determining Cluster Membership: The dendrogram can be cut horizontally at a specific height to determine the number of clusters. The horizontal cut corresponds to a similarity threshold, where branches below the cut are considered individual clusters.
6. Cluster Representation: Each cluster is represented by its centroid, which can be the mean or median of the data points in the cluster.
Hierarchical clustering provides a hierarchical structure that allows for exploring clusters at different granularity levels.

Density Based Clustering:

Density-based clustering aims to discover clusters of arbitrary shape in a dataset based on the density of data points. It identifies dense regions separated by sparser regions in the data space.

1. Parameter Selection: The algorithm sets the parameters for the algorithm, including the minimum number of data points (MinPts) and the distance threshold (ε) for defining the neighbourhood of a point.

2. Core Points and Neighborhoods: For each data point, the algorithm calculates the number of data points within a distance ε of that point. If this count is greater than or equal to MinPts, the point is considered a core point. The set of points within the ε-distance of a core point forms its neighbourhood.

Calculating the distance between x_i and all other data points using Euclidean distance:

$$D(x_i, x□) = \sqrt{\sum(x_i - x□)^2}$$

Determine if x_i is a core point by checking if the number of data points within distance ε (excluding x_i) is greater than or equal to MinPts.

Define the neighbourhood of a core point x_i as the set of all data points within distance ε (including x_i).

3. Cluster Formation: The algorithm starts with an unvisited data point. If it is a core point, a new cluster is formed around it. The cluster includes the core point and all reachable data points within its neighbourhood. The cluster expands by iteratively adding new points to it. It is added if a point is within the ε-distance of an existing cluster point. The expansion continues until no new points can be added to the cluster.

4. Noise Points and Border Points: The algorithm considers data points not assigned to any cluster as noise points or outliers. However, points within the ε-distance of a core point but do not satisfy the MinPts criterion are considered border points.

Density-based clustering is advantageous for discovering clusters of arbitrary shape and handling datasets with varying densities. It is robust to noise and can identify clusters even in the presence of outliers. However, it relies heavily on adequately selecting parameters, such as MinPts and ε, which can influence cluster formation.

4. Cluster Formation: Users with high similarity scores are assigned to the same cluster. The number of clusters can be predetermined (k-means) or determined dynamically (hierarchical clustering).

5. Cluster Representation: Centroids represent each cluster, and each clustering technique which was discussed earlier, explains how to obtain the centroid

6. Recommendation Generation: The algorithm generates recommendations based on each cluster's user preferences. Users are recommended items that are highly rated or preferred by other users in their cluster but that have not been interacted with by the target user.

7. New User Clustering: When a new user joins the system, their preferences are compared to the existing user clusters and assigned to the most similar cluster, which allows for providing initial recommendations based on the behaviour of similar users.

User clustering helps to group users with similar preferences, which allows for a more targeted and personalized recommendation. Users have similar item preferences within the cluster, making the recommendations more relevant and accurate. It's important to note that the effectiveness of user clustering depends on the quality of similarity measures, the choice of the clustering algorithm, and the nature of the dataset.

Matrix Factorization

Matrix Factorization is used to uncover latent features or factors that explain the observed user-item interactions in a recommendation system. It decomposes the user-item interaction matrix into lower-dimensional matrices representing user and item features.

1. User-Item Interaction Matrix: The user-item interaction matrix is constructed similarly to the other algorithms.
2. Matrix Decomposition: Matrix factorization aims to decompose the user-item matrix. (A user matrix U and an item matrix V). The user matrix (U) captures the latent user features or preferences, while the item matrix (V) represents latent item features and characteristics. A smaller than U and V's original dimensions are chosen, resulting in a reduced-dimensional representation.
3. Factorization Optimization: User-item interaction matrix gets approximated by the product of the user matrix U and the item matrix V:

   $$R = U \times V^{\mathrm{T}}$$

   The factorization is optimized by minimizing the difference between the approximation R and the original user-item matrix.
4. Latent Feature Interpretation: The underlying user preferences and item characteristics of the factorized matrices U and V are discovered in the latent features.
5. Recommendation Generation: After the factorization, recommendations can be generated by predicting the missing or unobserved entries in the user-item interaction matrix. We calculate the predicted ratings or preference by multiplying the corresponding user and item vectors obtained from the factorized matrices.

Matrix Factorization can handle sparse data where the user-item matrix has missing entries. It can capture complex relationships between users and items by modelling them in a

low-dimensional space. Additionally, it can provide interpretable latent factors that aid in understanding user preferences and item characteristics.

Overall, the advantage of each algorithm depends on the specific characteristics of the dataset and the requirements of the recommendation system. Some algorithms, like UBNM and k-NN, are simple and easy to implement but may not capture complex relationships and may not account for biases. Adjusted Cosine Similarity and Pearson-Based Recommender Systems account for variations and biases in ratings. Slope One offers accuracy and efficiency, while co-clustering provides a way to uncover underlying structures in the data. Ultimately, it is important to consider the specific context, data characteristics, and performance requirements when selecting one of the above algorithms for a recommender system.

Work Cited

- Kumar, P. Pavan, et al. *Recommender Systems: Algorithms and Applications*. CRC Press, 2021.
- Robillard, Martin P., et al. An Introduction to Recommendation Systems in Software Engineering. Springer Science and Business, 2014.
- “An In-Depth Guide to How Recommender Systems Work.” *Built In*, builtin.com/data-science/recommender-systems. Accessed 20 May 2023.
- “What Is Collaborative Filtering: A Simple Introduction.” *Built In*, builtin.com/data-science/collaborative-filtering-recommender-system. Accessed 6 Aug. 2023.
- Business, Iterators Tech &, et al. “Collaborative Filtering in Recommender Systems: Learn All You Need to Know |.” *Iterators*, 15 July 2021, www.iteratorshq.com/blog/collaborative-filtering-in-recommender-systems/.