Algorithms for Data Science: Recommendation Algorithms

Pablo Mollá Chárlez

October 10, 2024

Contents

1 Introduction

Recommendation algorithms are a class of algorithms designed to suggest relevant items to users, typically in online settings such as e-commerce, streaming platforms, or social media. These algorithms aim to predict user preferences based on their historical behavior, preferences of similar users, or item characteristics. By doing so, they help users discover products, movies, books, or any content they might like but haven't encountered yet.

Recommendation systems are critical in various applications:

1. **E-commerce**: Suggest products to customers based on their past purchases.
2. **Media streaming**: Suggest movies, music, or TV shows based on previous views or listens.
3. **Social media**: Recommend posts or people to follow based on user activity.

There are three main approaches to recommendation systems: Content-Based Filtering, Collaborative Filtering and Latent Factor Models (often covered under matrix factorization methods, not discussed here).

2 Content-Based Filtering

Content-based filtering recommends items that are similar to the ones the user has previously liked or rated highly. This method focuses on the features of the items and matches them with the user's profile, which is created based on the user's historical interactions with items. It has many different applications:

1. Books, Movies, Music: Recommending items with similar characteristics such as genre, author, actors, or musicians.

2. E-commerce: Suggesting products that share attributes like brand, category, or style with items the user has purchased or viewed before.

2.1 Main Workflow

- **Item Profile Creation**: Each item is described using a set of features (referred to as the item profile). These features could be words in a document, genres of a movie, or the characteristics of a product. Various techniques like one-hot encoding for categorical variables (e.g., author names, actors) or embeddings for more complex representations can be used to represent these features. Features can be represented using various methods:

  - **One-hot encoding**: Each feature is treated as a binary vector, where a 1 indicates the presence of a feature (e.g., an actor or genre).
  - **Embeddings**: More sophisticated methods like embeddings can be used to represent items in continuous vector space based on feature similarity.

  A key technique often used in content-based filtering is TF-IDF (Term Frequency-Inverse Document Frequency), a measure from text mining that weighs features based on their frequency in an item relative to their frequency across all items. This allows the system to give more importance to distinctive features. Let's properly define TF-IDF:

  - Term Frequency (TF) measures how often a feature (e.g., a word, actor, or attribute) appears in an item compared to the maximum frequency of any feature in that item. It is calculated as:

    $$\text{TF}_{ij} = \frac{f_{ij}}{\max_k f_{kj}}$$

    where f_{ij} is the frequency of feature i in item j, and $\max_k f_{kj}$ is the maximum frequency of any feature in item j.
  - Inverse Document Frequency (IDF) measures how important a feature is by reducing the weight of features that appear in many items. It is calculated as:

    $$\text{IDF}_i = \log\left(\frac{N}{n_i}\right)$$

    where N is the total number of items, and n_i is the number of items that contain feature i.
  - The TF-IDF score for each feature-item pair is:

    $$w_{ij} = \text{TF}_{ij} \times \text{IDF}_i$$

    Items are then represented by the features with the highest TF-IDF scores, creating an item profile.

- **User Profile Creation**: The user profile is created by aggregating the profiles of items the user has interacted with. For example, if a user has rated or liked several movies, their user profile might be the weighted average of the TF-IDF profiles of these movies.

- **Matching Items**: Once the user profile is created, it is matched against other item profiles in the system. Items with profiles similar to the user profile are recommended. This similarity is often computed using methods like cosine similarity between the user profile and the item profile.

  Cosine Similarity is often used to compare user and item profiles. The similarity score $s(x, i)$ between user x and item i is given by:

  $$s(x, i) = \frac{\langle x, i \rangle}{\|x\| \cdot \|i\|}$$

  where $\langle x, i \rangle$ is the dot product between the user profile vector and the item profile vector, and $\|x\|$ and $\|i\|$ are the magnitudes of the respective vectors. Items are then recommended based on their similarity score to the user profile. The system might recommend the top-k most similar items or items with a similarity score above a certain threshold.

2.2 Advantages and Drawbacks of Content-Based Filtering

The advantages include:

- Not reliant on other users Content-based filtering does not require information about other users, making it effective for recommending items based solely on user preferences.
- Can handle niche users and new items Since the recommendations are based on item characteristics, the system can recommend items to users with unique preferences or suggest new items with similar characteristics to previously liked ones.
- Provides explanations The system can explain recommendations based on the item features (e.g., 'Recommended because you liked movies with this actor or genre').

The drawbacks considered are:

- Feature selection is challenging Identifying the right features to represent items can be difficult. Poor feature selection can lead to irrelevant recommendations.
- Cold start for new users The system struggles to recommend items to new users who haven't rated or interacted with many items.
- Limited discovery The system can only recommend items similar to those the user has already interacted with, making it harder to suggest items outside the user's established profile.

2.3 Short Example

Let's say we have a small dataset of **3 items** (e.g., movies), and each movie is described by the features **'action'**, **'romance'**, and **'comedy'**.

Movie	Action Count	Romance Count	Comedy Count
A	3	0	2
B	1	2	0
C	0	1	3

Table 1: Movie's Dataset

Let's determine the TF-IDF score.

- **Step 1: Item Profile Creation Using TF-IDF** We compute the TF, IDF and then the TF-IDF.
  - **Term Frequency (TF)**
    1. Movie A: 'TF_action = 3/3 = 1', 'TF_romance = 0/3 = 0', 'TF_comedy = 2/3 ≈ 0.67'
    2. Movie B: 'TF_action = 1/2 = 0.5', 'TF_romance = 2/2 = 1', 'TF_comedy = 0/2 = 0'
    3. Movie C: 'TF_action = 0/3 = 0', 'TF_romance = 1/3 ≈ 0.33', 'TF_comedy = 3/3 = 1'
  - **Inverse Document Frequency (IDF)**
    1. For action: Appears in 2 movies, so $IDF_{\text{action}} = \log(3/2) \approx 0.18$
    2. For romance: Appears in 2 movies, so $IDF_{\text{romance}} = \log(3/2) \approx 0.18$
    3. For comedy: Appears in 2 movies, so $IDF_{\text{comedy}} = \log(3/2) \approx 0.18$
  - **TF-IDF Scores**
    1. Movie A: $w_{\text{action}} = 1 \times 0.18 = 0.18$, $w_{\text{romance}} = 0 \times 0.18 = 0$, $w_{\text{comedy}} = 0.67 \times 0.18 \approx 0.12$
    2. Movie B: $w_{\text{action}} = 0.5 \times 0.18 = 0.09$, $w_{\text{romance}} = 1 \times 0.18 = 0.18$, $w_{\text{comedy}} = 0 \times 0.18 = 0$
    3. Movie C: $w_{\text{action}} = 0$, $w_{\text{romance}} = 0.33 \times 0.18 \approx 0.06$, $w_{\text{comedy}} = 1 \times 0.18 = 0.18$

- **Step 2: Create User Profile Based on Liked Items** Assume the user liked **Movie A** and **Movie C**. We create the user profile by aggregating their TF-IDF values (e.g., taking the average):
  - $w_{action} = \frac{0.18 + 0}{2} = 0.09$
  - $w_{romance} = \frac{0 + 0.06}{2} = 0.03$
  - $w_{comedy} = \frac{0.12 + 0.18}{2} = 0.15$
- **Step 3: Match Items to the User Profile** Now, we calculate the similarity between the user's profile and each movie using cosine similarity and the movie with the highest similarity score is recommended!
  - Movie A (cosine similarity with user profile):

$$s(A, \text{user}) = \frac{(0.09 \times 0.18) + (0.03 \times 0) + (0.15 \times 0.12)}{\sqrt{(0.09^2 + 0.03^2 + 0.15^2)} \times \sqrt{(0.18^2 + 0^2 + 0.12^2)}}$$

  - Repeat for **Movie B** and **Movie C**.

3 Collaborative Filtering

Collaborative Filtering (CF) is a popular recommendation technique that makes predictions about a user's interests by collecting preferences from many other users. The key assumption is that users who agreed on one item in the past are more likely to agree on others in the future.

For a given user *Pablo*, CF identifies other users who have similar tastes and recommends items that those users liked, which *Pablo* hasn't interacted with yet. There are two main types:

- User-User Collaborative Filtering Find users similar to a target user based on their past ratings.
- Item-Item Collaborative Filtering Find items similar to the ones a user has already liked and recommend them.

3.1 User-User Collaborative Filtering

- **Step 1: Finding Similar Users**

  For a given user *Pablo*, the system finds N other users who have rated similar items. The similarity between users is measured using different metrics:
  - **Jaccard Similarity**: This measures how much overlap there is in the items rated by two users, ignoring the actual ratings. It compares the sets of items rated by two users:

$$\text{sim}(x, y) = \frac{|\{1, 4, 5\} \cap \{1, 3, 4\}|}{|\{1, 4, 5\} \cup \{1, 3, 4\}|} = \frac{2}{4} = 0.5$$

  - **Cosine Similarity**: This measures the similarity of the angles between two users' rating vectors, treating ratings as points in space. The cosine of the angle between two vectors $r_x$and $r_y$gives their similarity:

$$\text{sim}(x, y) = \frac{\langle r_x, r_y \rangle}{\|r_x\| \cdot \|r_y\|} \approx 0.3$$

  - **Other options include Pearson correlation coefficient**
- **Step 2: Predicting Missing Ratings**

  Once similar users are found, the algorithm estimates the rating that user *Pablo* would give to an item i that they haven't rated yet. This is done using either:

– **Simple Average of Ratings**: The predicted rating for user *Pablo* is the average rating given to item i by similar users y:

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$

– **Weighted Average (using similarity scores)**: More weight is given to ratings from users more similar to *Pablo*:

$$r_{xi} = \frac{\sum_{y \in N} \text{sim}(x, y) \cdot r_{yi}}{\sum_{y \in N} \text{sim}(x, y)}$$

3.2 Item-Item Collaborative Filtering

Rather than comparing users, item-item CF compares items based on how users have rated them. This approach finds items that are similar to the ones a user has already rated and predicts the user's rating based on those similar items.

- **Step 1: Finding Similar Items**

  For a given item i, find other items j that have been rated similarly by users, using similarity metrics like cosine similarity.

- **Step 2: Predicting Ratings**

  Once similar items are found, the algorithm predicts the rating for user *Pablo* on item i by using the ratings the user gave to similar items j:

$$r_{xi} = \frac{\sum_{j \in N(i;x)} \text{sim}(i, j) \cdot r_{xj}}{\sum_{j \in N(i;x)} \text{sim}(i, j)}$$

  where $N(i; x)$is the set of items similar to $i$that user $x$has already rated.

3.3 In Practice

In real-world systems, collaborative filtering considers biases and averages to improve accuracy. For example, it considers:

- Global average rating μ,
- User-specific bias $b_x$and item-specific bias b_i.

A baseline estimator might look like this:

$$r_{xi} = \mu + b_x + b_i + \frac{\sum_{j \in N(i;x)} \text{sim}(i, j) \cdot (r_{xj} - b_x j)}{\sum_{j \in N(i;x)} \text{sim}(i, j)}$$

3.4 Advantages and Drawbacks of Content-Based Filtering

The advantages include:

- No feature selection CF relies only on user ratings, not on features of the items.
- Good for complex domains It can recommend based on user's hidden preferences without knowing much about the content.

The drawbacks considered are:

- Cold start problem It struggles to recommend items to new users or recommend new items (due to a lack of ratings).
- Sparsity problem Many users may not have rated the same items, making it hard to find similarities.
- Popularity bias CF tends to recommend popular items since they have more ratings.

3.5 Evaluation

To evaluate CF, common techniques include:

1. **Root Mean Square Error (RMSE)**: To measure how close predicted ratings are to actual ones.
2. **Precisionk**: The percentage of relevant items in the top-k recommendations.
3. **Spearman rank correlation**: Measures how well the algorithm ranks items compared to a user's actual preferences.

4 Latent Factor Models

Latent factor models (e.g., matrix factorization techniques like SVD or ALS) attempt to capture the underlying structure in user-item interactions by projecting both users and items into a shared latent space. These techniques reduce the dimensionality of the user-item interaction matrix, enabling the system to discover hidden patterns in user preferences that aren't directly observable through explicit features or behavior.