Recommender Systems

Somu Patil

May 2019

1 What is a Recommender System?

A recommender system/engine filters data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy.

2 How does it work?

We can recommend items in two different ways:

- Recommend items which are popular with most other users
- Recommend items tailored to their preferences and interests

Both methods have underlying flaws. When recommending the most popular items, every user will be recommended the same items. Tailoring to the preferences of one specific user becomes difficult as the number of users and items increase. Because of these shortcomings, companies use personalized methods to help them recommend more relevant items to users.

3 Filtering

The following algorithms establish relationships and connections between users and items. These established connections make it easier to come up with predictions and recommendations.

3.1 Content-Based Filtering

This algorithm uses an user's previous viewings and search history to recommend products. For example, if someone watched and liked a movie like Avengers: Infinity War, the algorithm would recommend movies in the superhero genre. It uses past likes as a basis for recommending new products.

3.1.1 The Algorithm

The user's past data is saved in a vector called the *profile vector*. This vector holds all the products the user has liked/disliked, and the ratings given to each one. Each product and its information is put in a vector called the *item vector*. The algorithm uses cosine similarity to find the cosine of the angle between the two vectors.

$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

The similarity scores of each product are then sorted and the products with the most similar characteristics to the user are recommended.

This process has a major drawback when recommending products. Unless the user has previous data/reviews of other products, the algorithm cannot determine products which would suit the user. This also leads to a narrow algorithm in general. If a user only has looked into horror movies, then the algorithm will only recommend horror movies.

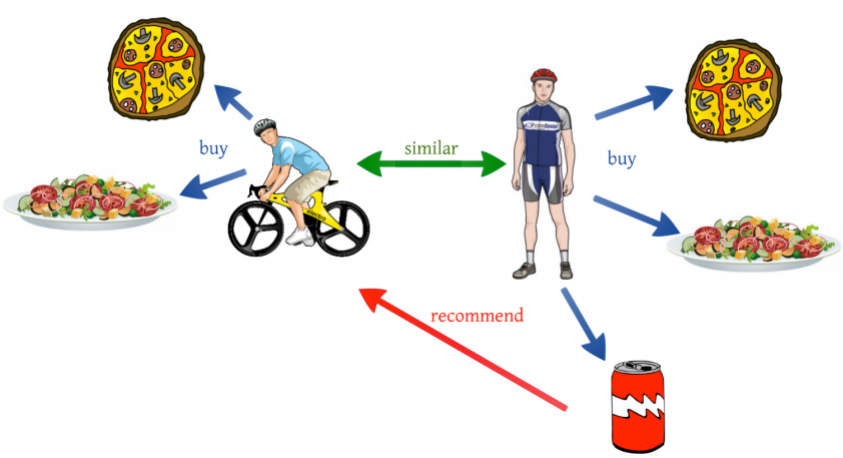
A more accurate algorithm would take in not just the user's previous history, but also their behavior as well.

3.2 Collaborative Filtering

Collaborative filtering is the use of the user's behavior exclusively to make predictions and recommendations for the user. This is one of the more popular algorithms in the field, because of lack of dependence on additional information. There are different ways to implement this idea, and two of them are explained below.

3.2.1 User-User Collaborative Filtering

This implementation uses similarities between other users to recommend products. The algorithm calculates similarity scores between each user pair, and determines which user is the most similar to the current user. Based on this, we can make predictions based on what this similar user had purchased/viewed.



The algorithm is as follows:

1. Compute similarity scores between every user present in the profile vector using the techniques listed above
2. For each pair of users, find the items that are common between them, and correlation is calculated based on these items and the users' behavior
3. Prediction values are then calculated for the user u in question using this formula:

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v s_{u,v}}$$

   - $P_{u,i}$ is the prediction of an item
   - $r_{v,i}$ is the rating user v gave for item i
   - $s_{u,v}$ is the similarity between user u and user v
4. Finally, based on these predictions, make recommendations for products that are most relevant with similar users

This algorithm is helpful when the number of users is less. However, this algorithm can become time-consuming as more and more users are added to the profile vector due to the pairwise calculation of similarity for each pair of users.

3.2.2 Item-Item Collaborative Filtering

This implementation uses similarities between items to recommend products. It is very similar to User-User Collaborative Filtering except the prediction formula which is:

$$P_{u,i} = \frac{\sum_N (s_{i,N} * R_{u,N})}{\sum_N (|s_{i,N}|)}$$

- $P_{u,i}$ is the prediction of an for user u
- $s_{i,N}$ is the similarity between item i and item N
- $R_{u,N}$ is the rating user u gives for item N

The similarities are between the items rather than the users. This is useful when there are more users than items.

3.3 Cold Starts

A cold start is a scenario where something new is introduced into the data-set. There are two types of cold starts, Visitor Cold Starts and Product Cold Starts.

3.3.1 Visitor Cold Start

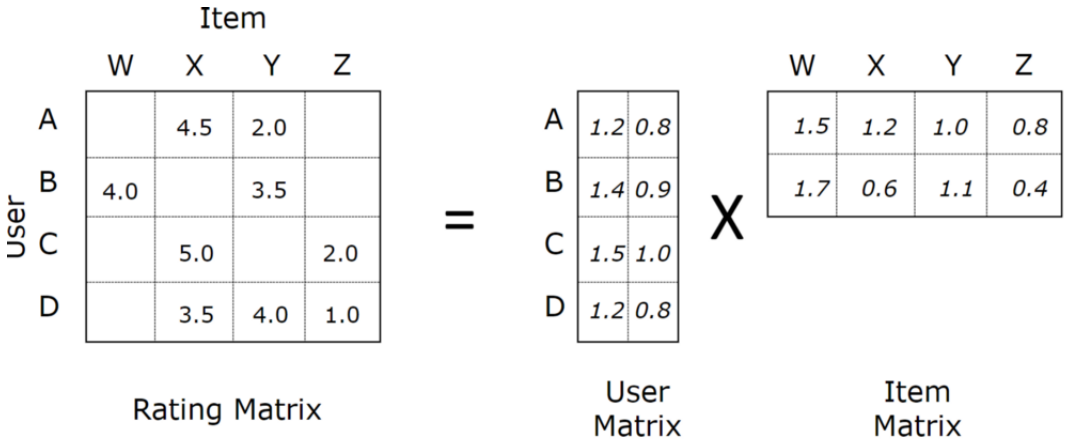
A visitor cold start is when a new user is introduced into the profile vector. Because the user has no past history, it is hard to recommend products to them at the jump. One way to bypass this is to recommend the most popular products to get a sense of their tastes and interests.

3.3.2 Product Cold Start

A product cold start is when a new product is introduced into the item vector. As the product gains more interaction, it becomes easier to recommend in the future. Initial runs would make use of Content-Based Filtering to ramp up the interactions, then switch to Collaborative Filtering after the product has become established in the item vector.

4 Matrix Factorization

Matrix factorization, a technique used in recommender systems, are used to solve a specific problem. Given a rating matrix with many empty values, predict the ratings of those empty values using the implicit information from the matrix.



As seen in the image above, the main goal of matrix factorization is to find matrixs p and q, which are the optimal divisions of the rating matrix. Using these optimal divisions, we can predict future recommendations from the past data.

To find the optimal division, we must use gradient descent to move our matrixes in the right direction. We have to take partials of the loss function below with respect to q and p.

$$e_{ui} \overset{def}{=} r_{ui} - q_i^T p_u.$$

After the partial derivatives, the gradients look like this:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

The model trains in order to reduce the error of the approximation of the dot-product between matrixs q and p. Once trained, one can predict the ratings for the empty cells in the rating matrix.

5 Coding Samples

Below is a link to a GitHub Repository with code samples of all the concepts discussed in the lecture:

https://github.com/somup27/RecommenderSystem

6 Acknowledgements

This lecture was developed through the resources from these websites:

- Analytics Vidhya
- Medium
- Towards Data Science
- Overleaf