Matrix Factorization

Open the browser, search for a product, scan the entire range, click, swipe, and smile!

Without batting an eyelid, we can purchase the latest model of mobile phones and electronic gadgets, or contemporary furniture or home accessories, just snapping our fingers. The e-commerce sites customize their range of offerings, as per our budget, needs, and tastes. And the happy customer will surely knock again in the future, and even recommend the site on his social circuit. So, every e-commerce site or content provider endeavors to enhance the online shopping experience, to attract and retain customers, bag higher ratings and positive reviews, and stay ahead of the competition. For this purpose, they need a diverse pool of offerings and a robust recommendation engine.

Matrix factorization is one of the most sought-after machine learning recommendation models. It acts as a catalyst, enabling the system to gauge the customer's exact purpose of the purchase, scan numerous pages, shortlist, and rank the right product or service, and recommend multiple options available. Once the output matches the requirement, the lead translates into a transaction and the deal clicks.

What is Matrix Factorization?

This mathematical model helps the system split an entity into multiple smaller entries, through an ordered rectangular array of numbers or functions, to discover the features or information underlying the interactions between users and items.

Where is Matrix Factorization used?

Once an individual raises a query on a search engine, the machine deploys uses matrix factorization to generate an output in the form of recommendations. The system uses two approaches– content-based filtering and collaborative filtering- to make recommendations.

Content-Based Filtering

This approach recommends items based on user preferences. It matches the requirement, considering the past actions of the user, patterns detected, or any explicit feedback provided by the user, and accordingly, makes a recommendation.

Example: If you prefer the chocolate flavor and purchase a chocolate ice cream, the next time you raise a query, the system shall scan for options related to chocolate, and then, recommend you to try a chocolate cake.

How does the System make recommendations?

Let us take an example. To purchase a car, in addition to the brand name, people check for features available in the car, most common ones being safety, mileage, or aesthetic value. Few buyers consider the automatic gearbox, while others opt for a combination of two or more features. To understand this concept, let us consider a two-dimensional vector with the features of safety and mileage.

Car features	Car A	Car B	Car C	Car D
Safety	4	1	2	1
Mileage	1	4	2	2

Individual preferences	*Safety*	*Mileage*	Car A	Car B	Car C	Car D
Person A	1	0	4	1	2	1
Person B	0	1	1	4	2	?
Person C	1	0	4	1	?	1
Person D	1	1	?	5	4	3

1. In the above graph, on the left-hand side, we have cited individual preferences, wherein 4 individuals have been asked to provide a rating on safety and mileage. If the individuals like a feature, then we assign the value 1, and if they do not like that particular feature, we assign 0. Now we can see that Persons A and C prefer safety, Person B chooses mileage and Person D opts for both Safety and Mileage.

Cars are been rated based on the number of features (items) they offer. A ranking of 4 implies high features, and 1 depicts fewer features.

- A ranks high on safety and low on mileage
- B is rated high on mileage and low on safety
- C has an average rating and offers both safety and mileage
- D is low on both mileage and safety

2. The blue colored? mark is the sparse value, wherein either person does not know about the car or is not part of the consideration list for buying the car or has forgotten to rate.

3. Let's understand how the matrix at the center has arrived. This matrix represents the overall rating of all 4 cars, given by the individuals. Person A has given an overall rating of 4 for Car A, 1 to Cars B and D, and 2 to Car C. This has been arrived, through the following calculations-

- For Car A = (1 of safety preference x 4 of safety features) +(0 of mileage preference x 1 of mileage feature) = 4;
- For Car B = (1 of safety preference x 1 of safety features) +(0 of mileage preference x 4 of mileage feature) = 1;
- For Car C = (1 of safety preference x 2 of safety features) +(0 of mileage preference x 2 of mileage feature) = 2;
- For Car D = (1 of safety preference x 1 of safety features) +(0 of mileage preference x 2 of mileage feature) = 1
- Now on the basis of the above calculations, we can predict the overall rating for each person and all the cars.

If person C is asking the search engine to recommend options available for cars, basis the content available, and his preference, the machine will compute the table below:

Individual preferences	*Safety*	*Mileage*	Car features	Car A	Car B	Car C	Car D
			Safety	4	1	2	1
			Mileage	1	4	2	2
Person A	1	0		4	1	2	1
Person B	0	1		1	4	2	2
Person C	1	0		4	1	2	1
Person D	1	1		5	5	4	3

- It will rank cars based on overall rating.
- It will recommend car A, followed by car C.

Mathematics behind the recommendations made using content-based filtering

In the above example, we had two matrices, that is, individual preferences and car features, and 4 observations to enable the comparison. Now, if there are ***'n'*** number of observations in both matrix ***a*** and ***b***, then-

- **Dot Product**
  The dot product of two length- n vectors a and b are defined as:

$$a.b = \sum_{i=1}^{n} aibi = a_1b_1+a_2b_2+\ldots\ldots\ldots\ldots\ldots+a_nb_n$$

n = number of observations

a and b= two separate matrices with n observations

The dot product is only defined for vectors of the same length, which means there should be equal number of 'n' observations.

- **Cosine**
  There are alternative approaches or algorithms available to solve a problem. Cosine is used as an approach to measure similarities. It is used to determine the nearest user to provide recommendations. In our chocolate example above, the search engine can make such a recommendation using cosine. It measures similarities between two non zero vectors. The similarity between two vectors a and b is, in fact, the ratio between their dot product and the product of their magnitudes.

$$\text{Similarity} = \cos\theta = \frac{a.b}{||a||.||b||} = \frac{\sum_{i=1}^{n} ab}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

By applying the definition of similarity, this will equal 1 if the two vectors are identical, and it will be 0, if the two are orthogonal. In other words, similarity is a number ranging from 0 to 1 and tells us the extent of similarity between the two vectors. Closer to 1 is highly similar and closer to 0 is dissimilar.

- **TF-IDF (Term frequency and Inverse document frequency)**
  Let us assume the matrix only has text or character elements, then the search engine uses TF- IDF algorithm to find similarities and make recommendations. Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF-IDF weight of that term.

$$\text{TF –IDF} = \text{TF(t)} \times \text{IDF(t)}$$

The TF (term frequency) of a word is the number of times it appears in a document. When you know it, you can find out if a term is being used too often or too rarely.

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}}$$

The IDF (inverse document frequency) of a word is the measure of how significant that term is in the whole corpus.

$$IDF(t) = \log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}\right)$$

The higher the TF-IDF score (weight), more important the term, and vice versa.

Advantages of content-based filtering

The model does not require any data about other users, since the recommendations are specific to one user. This makes it easier to scale it up to a large number of users. a) The model can capture specific interests of a user, and can recommend niche items that very few other users are interested in.

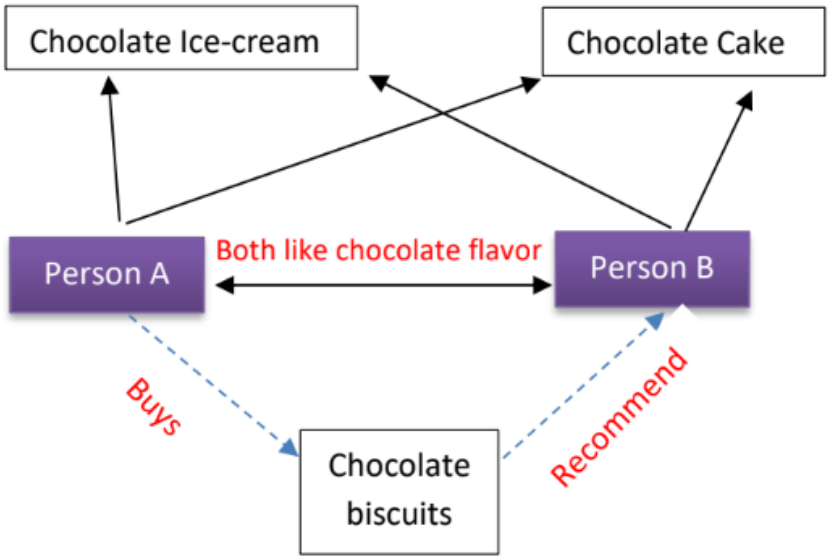
Disadvantages of content-based filtering

a) Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.

b) The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

Collaborative Filtering

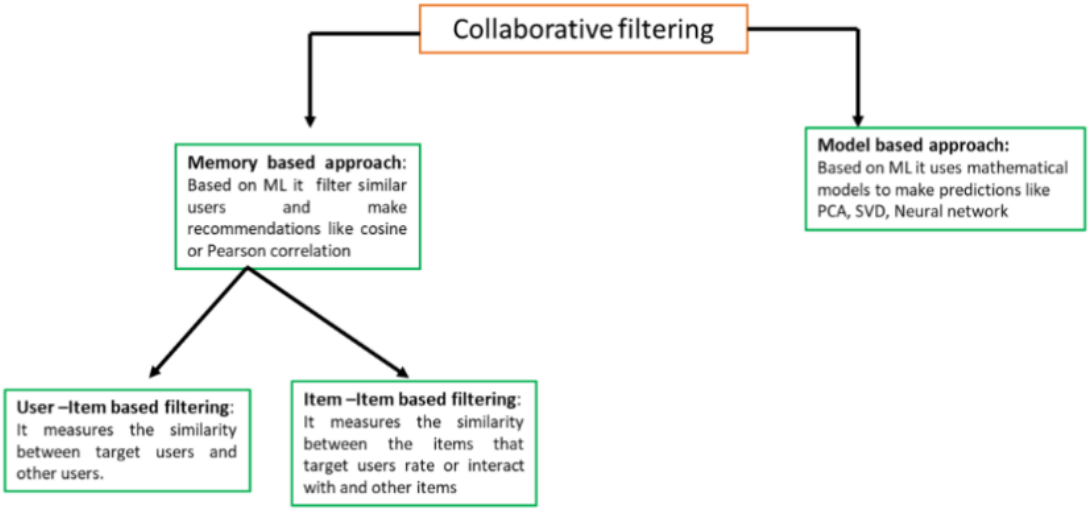
This approach uses similarities between users and items simultaneously, to provide recommendations. It is the idea of recommending an item or making a prediction, depending on other like-minded individuals. It could comprise a set of users, items, opinions about an item, ratings, reviews, or purchases.



Example: Suppose Persons A and B both like the chocolate flavor and have them have tried the ice-cream and cake, then if Person A buys chocolate biscuits, the system will recommend chocolate biscuits to Person B.

Types of collaborative filtering

Collaborative filtering is classified under the memory-based and model-based approaches:



How does system make recommendations in collaborative filtering?

In collaborative filtering, we do not have the rating of individual preferences and car preferences. We only have the overall rating, given by the individuals for each car. As usual, the data is sparse, implying that the person either does not know about the car or is not under the consideration list for buying the car, or has forgotten to give the rating.

Car features	Car A	Car B	Car C	Car D
Safety				
Mileage				

Individual preferences	*Safety*	*Mileage*	Car A	Car B	Car C	Car D
Person A			4	1	2	1
Person B			1	4	2	?
Person C			4	1	?	1
Person D			?	5	4	3

The task in hand is to predict the rating that Person C might assign to Car C (? marked in yellow) basis the similarities in ratings given by other individuals. There are three steps involved to arrive at a collaborative filtering recommendation.

Step-1

Normalization: Usually while assigning a rating, individuals tend to give either a high rating or a low rating, across all parameters. Normalization usually helps in balancing and evens out such measures. This is done by taking an average of rating available and subtracting it with the individual rating$(x- \bar{x})$

a) In case of Person A = $(4+1+2+1)/4 = 2 = \bar{x}$, In case of Person B = $(1+4+2)/3 = 2.3 = \bar{x}$

Similarly, we can do it for Persons C and D.

b) Then we subtract the average with individual rating

In case of Person A it is, 4-2, 1-2, 2-2, 1-2, In case for Person B = 1-2.3,4-2.3,2-2.3

Similarly, we can do it for Persons C and D.

You get the below table for all the individuals.

	Car A	Car B	Car C	Car D
Person A	2.0	-1.0	0.0	-1.0
Person B	-1.3	1.7	-0.3	?
Person C	2.0	-1.0	?	-1.0
Person D	?	1.0	0.0	-1.0

If you add all the numbers in each row, then it will add up-to zero. Actually, we have centered overall individual rating to zero. From zero, if individual rating for each car is positive, it means he likes the car, and if it is negative it means he does not like the car.

Step-2

Similarity measure: As discussed in content-based filtering, we find the similarities between two vectors a and b as the ratio between their dot product and the product of their magnitudes.

Car features	Car A	Car B	Car C	Car D
Individual preferences				
Person A	-0.42	0.54	1.00	0.54
Person B	0.24	-0.10	0.94	?
Person C	-0.42	0.54	?	0.54
Person D	?	0.54	1.00	0.54

Step-3

Neighborhood selection: Here in Car C column, we find maximum similarities between ratings assigned by Persons A and D. We use these similarities to arrive at the prediction.

$$\text{Prediction}_{\text{Person C-Car C}} = (1*4) + (1*2)/(1+1) = 3$$

Mathematics behind recommendations made using collaborative filtering

In this approach, similarities between pair of items are computed using cosine similarity metric. The rating for target item ***'i'*** for an active user ***'a'*** can be predicted by using a simple weighted average as:

$$p_{a,i} = \frac{\Sigma_{j \varepsilon k} r_{aj} W_{ij}}{\Sigma_{j \varepsilon k} W ij}$$

where k is the neighborhood of most similar items rated by active user a, and ***w(i,j)*** is the similarity between items ***i*** and ***j***.

Advantages of collaborative filtering

- There is no dependence on domain knowledge as embedding are automatically learned.
- The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.
- To some extent, the system needs only the feedback matrix to train a matrix factorization model. In particular, the system does not require contextual features.

Disadvantages of collaborative filtering

- The matrix cannot handle fresh items, for instance, if a new car is added to the matrix, it may have limited user interaction and thus, will rarely occur as a recommendation.
- The output of the recommendation could be biased, based on popularity, that is, if most user interaction is towards a particular car, then the recommendation will focus on that popular car only.