What is Matrix Factorization?

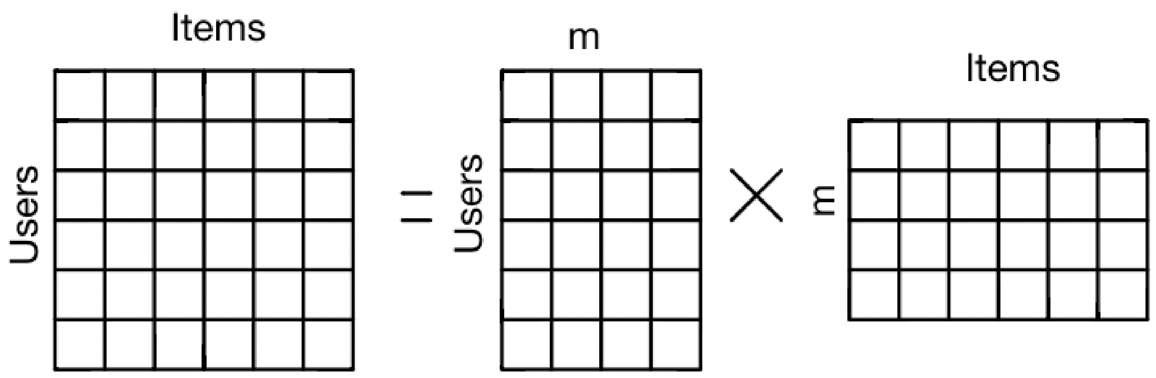
Without batting an eyelid, we can purchase the latest model of mobile phones and electronic gadgets, or contemporary furniture or home accessories, just snapping our fingers. The e-commerce sites customize their range of offerings, as per our budget, needs, and tastes. And the happy customer will surely 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.

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.

A recommendation engine is a subcategory of machine learning which aims to provide a rating for some user or item. Matrix factorization falls under the category of collaborative filtering in recommendation systems. Intuitively, collaborative filtering aims to identify items that a user A would like based on the interactions of other user(s) which are similar to user A. Hence this as an optimization problem where we aim to find a method which yields the best ratings for a given product / user.

The intuition behind matrix factorization is fairly simple, given some user-item matrix, you want to decompose that matrix such that you have a user matrix and an item matrix independently. This allows us to apply different regularization to each latent factor of the original matrix. For example, when products are songs, factors might measure the comparison between rap and country, count of times the song was played vs song duration, etc. The image below illustrates the decomposition of an input user-item (song) matrix M.



Why matrices?

Long before matrix factorization came … matrices. Lots of data we see and work with in data science can be represented as matrices. Here are few examples.

- A grayscale image is a matrix of pixels organized into rows(height) and columns(width)
- A video can be represented as a matrix with rows showing number of frames and columns being an image unwrapped to a 1D vector
- A corpus of documents can be represented as a matrix with rows being the documents and columns being the all the terms (i.e., vocabulary) present in the corpus
- Item ratings (such as movies or products) can be represented as a matrix with rows (one for each user) and columns (one for each item).

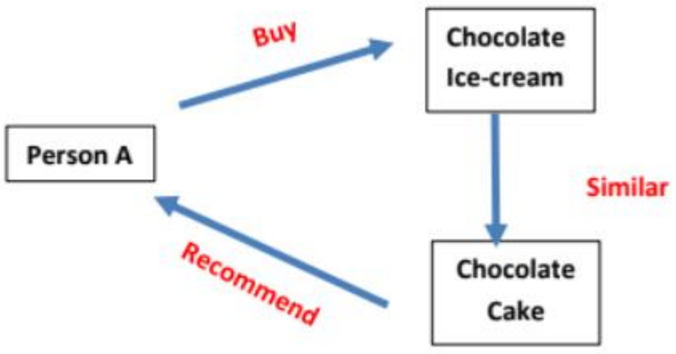
Why is Matrix Factorization used?

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.

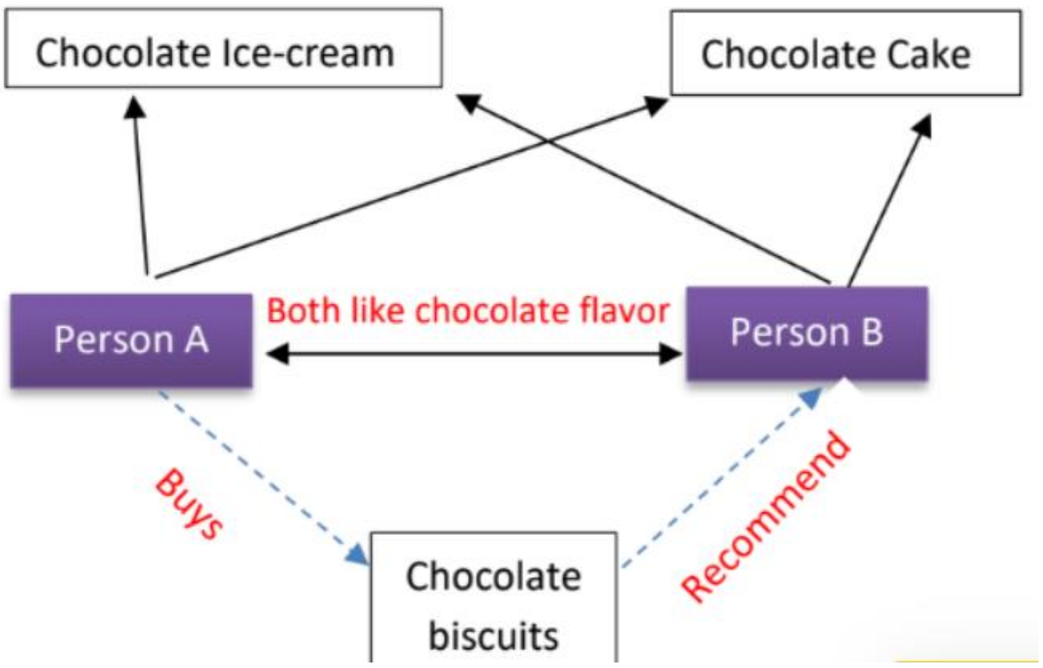


Advantages of content-based filtering

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.

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.

Disadvantages of collaborative filtering

1. There are a variety of challenges when working with collaborative filtering approaches in recommendation systems, the most common one is the cold start problem. The cold start problem refers to a recommendation engines inability to generate decent predictions for new items / users. Or users / items which have a low amount of interactions, essentially the user-item matrix it generates is very sparse. This is because of a lack of information for new items or product releases.
2. Another common problem with collaborative filtering which stems from the cold start problem is the diversity of the recommendations. This approach recommends products / users based on the historical interactions other users / products have had, essentially meaning that highly popular products will almost always get recommended over other products with a little too few interactions.
3. Scalability is a common issue for collaborative filtering algorithms, as the number of users, products and interactions increase, the size of the user-item matrix increases proportionally. This causes issues when data expands substantially and will yield an increase of computation time.

Where Matrix Factorization is used?

- Reduced computation time while training.
  This is one of the main reasons why one would want to use matrix factorization in highly dimensional datasets. It is faster and easier for the training algorithm (e.g. random forest, gradient boosting) to learn from compact information-dense features.

- Building a recommender system.
  By itself, the matrix factorization techniques such as singular value decomposition (SVD) can be used to build recommender systems.

- Dimensionality reduction.
  While dealing with high dimensional datasets, different transformation techniques can lead to different feature engineered datasets with huge dimensionality. We can first reduce the dimensionality of each technique and then concatenate the datasets into a smaller dataset to train on.

- As feature engineering.
  We can compute the latent vectors on the levels of original features or on the transformed features (e.g. log) and have different training models. We could use each model in an ensemble.
- Image compression-Matrix factorization can be used to store important content of an image needed to reconstruct it later (with a smaller memory footprint).