Recommendation Systems from Matrix Factorization

Mary Liu

Feb. 10, 2023

Contents

1 Introduction

As a company that shifted its main business focus from DVD retailing to online streaming, Netflix quickly found out that it needed to know customers' preferences to maximize its revenues. To do that, Netflix needs a recommendation system to be precise about those preferences. Here is a basic explanation of the recommendation system:

For a company focusing on online streaming, it has a set of users and a set of content (movies, shows, etc.). During or after users watching the content, they can rate the content that has been watched. Based on those existing ratings, recommendation systems estimate customers' ratings on content they have not made interaction with. The system lets the company know which content the company recommends to each customer.

In 2007, the winner of that year's Netflix Prize came up with the best-performing algorithm, which boosted the revenue by 10%. The algorithm is based on linear algebra,

mainly matrix factorization. After that, this algorithm has been used by many data-based companies. This paper will first introduce the basis of matrix factorization, the motivation for using it under a certain condition, and a common method of doing it. Then the paper will introduce the recommendation system, including its basic features. Overall, this paper will focus on how matrix multiplication and matrix factorization can be applied to the recommendation system.

2 Materials and Methods

2.1 Matrix Factorization

Definition 1. *A recommendation system is a system that supports users in the process of finding and selecting products (items) from a given assortment [1].*

In other words, a recommendation system works as it stores data and use certain algorithms to gather useful information about the data, hence make prediction among users.

2.1.1 Motivation of using Matrix Factorization

Theorem 2. *Let A be an $m \times n$ matrix with rank k. Then there exsit an $m \times k$ matrix X and an $k \times n$ matrix Y such that $A = XY$.*

Writing a matrix A in the form of a multiplication of two other matrices is called *Matrix Factorization*. We prefer using matrix factorization to store data in the recommendation system because when the number of entries of A becomes enormous, it takes less space in the computer storage to store entries of X and Y. Here is an example:

Let's say Netflix now has $10,000$ users and $10,000$ movies in its database. The table below is what the database needs to store. As M_i represents each movie, P_j represents each user, and $r_{i;j}$ represents each user's rating for each movie.

	M_1	M_2	$\dots$	$M_{10,000}$
P_1	$r_{1;1}$	$r_{1;2}$	$\dots$	$r_{1;10,000}$
P_2	$r_{2;1}$	$r_{2;2}$	$\dots$	$r_{2;10,000}$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$P_{10,000}$	$r_{10,000;1}$	$r_{10,000;2}$	$\dots$	$r_{10,000;10,000}$

And if we write the table out as a matrix A, it becomes:

$$\begin{bmatrix} r_{1;1} & r_{1;2} & \dots & r_{1;10,000} \\ r_{2;1} & r_{2;2} & \dots & r_{2;10,000} \\ \vdots & \vdots & \ddots & \vdots \\ r_{10,000;1} & r_{10,000;2} & \dots & r_{10,000;10,000} \end{bmatrix}$$

If we need to store every user's rating for each movie, $10,000 \times 10,000 = 100,000,000$ units would be stored in the database. But we may store fewer data if we use matrix

factorization to represent the matrix A in the other two matrices X and Y such that $A = XY$:

$$A = \begin{bmatrix} x_{1;1} & x_{1;2} & \cdots & x_{1;k} \\ x_{2;1} & x_{2;2} & \cdots & x_{2;k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{10,000;1} & x_{10,000;2} & \cdots & x_{10,000;k} \end{bmatrix} \begin{bmatrix} y_{1;1} & y_{1;2} & \cdots & y_{1;10,0000} \\ y_{2;1} & y_{2;2} & \cdots & y_{2;10,000} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k;1} & y_{k;2} & \cdots & y_{k;10,000} \end{bmatrix}$$

As $x_{i;j}$ represents each user's attitude toward a specific genre (action, comedy, horror, etc.) and $y_{i;j}$ represents the rating for each movie based on an independent genre.

If we store the data in the two new matrices, we must store $km + kn$ units. Notice that as long as $k < 5,000$, $km + kn < mn$. Hence it makes more sense to store the following two tables:

	F_1	F_2	...	F_k
P_1	$x_{1;1}$	$x_{1;2}$	$\cdots$	$x_{1;k}$
P_2	$x_{2;1}$	$x_{2;2}$	$\cdots$	$x_{2;k}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$P_{10,000}$	$x_{10,000;1}$	$x_{10,000;2}$	$\cdots$	$x_{10,000;k}$

	M_1	M_2	...	$M_{10,000}$
F_1	$y_{1;1}$	$y_{1;2}$	$\cdots$	$y_{1;10,000}$
F_2	$y_{2;1}$	$y_{2;2}$	$\cdots$	$y_{2;10,000}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
F_k	$y_{k;1}$	$y_{k;2}$	$\cdots$	$y_{k;10,000}$

As F_i represents each independent genre.

Companies like Netflix can have more than 10,000 users or 10,000 streaming content, and it can choose what other elements (like the genre in our example) to build up such matrices X and Y for matrix factorization.

2.1.2 Common Methods for Matrix Factorization

A common method for matrix factorization is called LU factorization. It works by factoring a matrix A into a lower triangular matrix L and an upper triangular matrix U [3]. We will demonstrate how LU factorization works with an example.

Suppose

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 5 & 7 & 1 \\ 3 & 2 & -3 \end{bmatrix}$$

Step 1: By EROs, changing A into its Row Echelon Form by using only type-three row operation, which is adding/subtracting one row to the multiple of another, i.e., replacing R_i by $R_i - kR_j$, $k \in \mathbb{F}$). We call the Row Echelon Form U.

$$\begin{bmatrix} 1 & 3 & 2 \\ 5 & 7 & 1 \\ 3 & 2 & -3 \end{bmatrix} \xrightarrow{R_2 - 5R_1} \begin{bmatrix} 1 & 3 & 2 \\ 0 & -8 & -9 \\ 3 & 2 & -3 \end{bmatrix} \xrightarrow{R_3 - 3R_1} \begin{bmatrix} 1 & 3 & 2 \\ 0 & -8 & -9 \\ 0 & -7 & -9 \end{bmatrix} \xrightarrow{R_3 - 7/8R_1} \begin{bmatrix} 1 & 3 & 2 \\ 0 & -8 & -9 \\ 0 & 0 & -9/8 \end{bmatrix} = U$$

To row operate A into U, notice that we first have $R_2 - 5R_1$ to make $A_{21} = 0$ (clear out second-row first column entry of A), so $k_{21} = 5$. Then we have $R_3 - 3R_1$ to make $A_{31} = 0$, so $k_{31} = 3$. In the same way, $k_{32} = 7/8$.

Step 2: Building the lower triangular matrix L, as all the diagonal entries are 1, and for each A_{ij} entries that have changed in A, the corresponding entries for L is $L_{ij} = k_{ij}$.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 0 \\ 3 & 7/8 & 1 \end{bmatrix}$$

Step 3: Check that $A = LU$

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 5 & 7 & 1 \\ 3 & 2 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 0 \\ 3 & 7/8 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 2 \\ 0 & -8 & -9 \\ 0 & 0 & -9/8 \end{bmatrix} = LU$$

There exist other methods for all types of matrices. Sometimes Gaussian Elimination leads to inaccurate results, and such problems can be avoided by using another common method, QR factorization: $A = QR$ where Q is unitary, i.e., $Q * Q = I$, and R is upper triangular [3].

2.2 Introduction to Recommendation Systems

2.2.1 Basic Features and Dependencies of Recommendation Systems

Let's start with a simple example to show how a recommendation system uses matrix factorization:

Let's say we have four people: A, B, C, D, and five movies: M1, M2, M3, M4, M5, and a rating scale from 1-5, with 1 being the worst and 5 being the best.

In reality, there will be some similarities in human preferences, i.e. the ratings would not be completely random. Also, it is not possible for all people to have exactly the same preferences, i.e., the rating for all people for all movies should not be the same number. Here is a reasonable rating table example from A, B, C, and D:

	M1	M2	M3	M4	M5
A	3	1	1	3	1
B	1	2	4	1	3
C	3	1	1	3	1
D	4	3	5	4	4

Notice some dependencies here:

1. We can see that rows 1 and 3 are the same, meaning A and C have a highly similar preference; we denote it as $A = C$.
2. Columns 1 and 4 are the same, meaning M1 and M4 have similar movie genres or contents; we denote it as $M1 = M4$.
3. Each column's rating number in row 2 plus the corresponding column's number in row 3 equals the corresponding column's number in row 4; we denote it as $B+C = D$.

It is reasonable when person D likes both what person B and C like. For example, let's assume B loves action movies and his rating is only based on how many action elements a movie contains, C loves comedy movies and her rating is only based on how many comedy elements a movie contains, and D loves action movies that contain comedy elements and his rating is only based on how much action and comedy elements a movie contains, then the rating that D gives for each movie would be the sum rating number that B and C gives.

4. Each row's rating in column 5 is the average of column 2's and column 3's; we denote it as $M5 = average(M2, M3)$. It is reasonable when elements in movies 2 and 3 are different, and movie 5 contains both elements in movies 2 and 3. For example, M2 is about sharks, M3 is about helicopters, and M5 is about saving sharks by using helicopters. Then from the table, we can tell that B does not like sharks but does like helicopters, so he may not particularly like nor dislike M5, and reasonably gives M5 a rating of 3.

Because of those dependencies, we can guess what people's ratings are even though they haven't watched the movies, and decide if we want to recommend the movies to them or not. For example, if C has not watched M5, i.e., row 3, column 5 is missing. Since $A = C$, we can conclude that the rating that C would give to M5 is 1. Hence the system should not recommend M5 to C.

2.2.2 Matrix Multiplication for Recommendation Systems

Before we dig into how to use matrix factorization when applying to a recommendation system, let's look at this in an opposite way to see why matrix factorization would indeed work for the recommendation system by having a look at matrix multiplication.

Using the same logic as the example in section 2.1.1 but in an opposite way, let's say we already know the two matrix factors, and we would like to know each user's rating for each movie.

Let's start with a small-size example: using an example from section 2.2.1, but instead of knowing the whole data, we know the user's attitude for each genre and the existing rating for each movie based on each independent genre. Here we introduce two genres: comedy and action. From people who have already watched it, we know that M1 has a rating of 3 in comedy and 1 in action. Let's assume people rate movies based only on how much comedy and action elements they contain. So if someone doesn't like one of the genres, the rating shown on that movie would be dropped, i.e., we would not count this rating.

Suppose the table below shows the attitudes toward comedy and action for each person (later on I'll introduce how we can come up with those attitudes but now let's just assume it's true) *Yes* means the person appreciates this genre in the movie, *No* means the opposite:

	Comedy	Action
A	Yes	No
B	No	Yes
C	Yes	No
D	Yes	Yes

And here is the table of the existing rating for each movie based on their comedy and action genres independently.

	M1	M2	M3	M4	M5
Comedy	3	1	1	3	1
Action	1	2	4	1	3

If we assign 1 to *Yes* and 0 to *No* for the first table, represent both tables as matrices and multiply, we get:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 & 3 & 1 \\ 1 & 2 & 4 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 1 & 3 & 1 \\ 1 & 2 & 4 & 1 & 3 \\ 3 & 1 & 1 & 3 & 1 \\ 4 & 3 & 5 & 4 & 4 \end{bmatrix}$$

and we denote the result matrix as matrix A. Notice that matrix A is equal to the table introduced on page 4. Also, we can check this equation and see if it makes sense:

For example, from the table, we know that person A likes comedy but hates action and hasn't watched M1 yet. So we say A would give $3 = 1 \times 3 + 0 \times 1$ to M1, and for a_{ij} as the ith row and jth column for A, $a_{11} = 3$. Person D likes both comedy and action, then we say D would give $4 = 1 + 3$ to M1, and it's true that $a_{41} = 4$.

Recall that we made many assumptions in the example: ratings based on only two genres; there are only 5 people and 5 movies. But in reality, all these elements can count as units based on bigger than thousands. Additionally, the attitude are natural number: 1 represents for a complete *Yes*, i.e., like the movie, and 0 represents for a complete *No*, i.e., dislike the movie for the movie. But people's attitude for each genre can be a range between $0 - 1$. So we can make it more general:

Suppose there are p people, P_1, P_2,..., P_p, m movies, M_1, M_2,...M_n and F genres, F_1, F_2,..., F_f, then there is a table of people's attitude for each genre, n_{ij} where it stands for the attitude of P_i for F_j:

	F_1	F_2	...	F_f
P_1	n_{11}	n_{12}	...	n_{1f}
P_2	n_{21}	n_{22}	...	n_{2f}
...	...	...	...	...
P_p	n_{p1}	n_{p2}	...	n_{pf}

And a table for each movie based on each independent feature, r_{ij} where it stands for the existing rating for M_i based on F_j:

	M_1	M_2	...	M_m
F_1	r_{11}	r_{12}	...	r_{1m}
F_2	r_{21}	r_{22}	...	r_{2m}
...	...	...	...	...
F_f	r_{f1}	r_{f2}	...	r_{fm}

Represent both tables as matrices and multiplying, we get a matrix, call it matrix G. G is a general database that a company like Netflix would like to store because of the motivation in section 2.1.1:

$$\begin{bmatrix} n_{11} & n_{12} & \dots & n_{1f} \\ n_{21} & n_{22} & \dots & n_{2f} \\ \vdots & \vdots & \ddots & \vdots \\ n_{p1} & n_{p2} & \dots & n_{pf} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{f1} & r_{f2} & \dots & r_{fm} \end{bmatrix}$$

$$= \begin{bmatrix} n_{11}r_{11} + ... + n_{1f}r_{f1} & \dots & n_{11}r_{1m} + ... + n_{1f}r_{fm} \\ \vdots & \ddots & \vdots \\ n_{p1}r_{11} + ... + n_{pf}r_{f1} & \dots & n_{p1}r_{1m} + ... + n_{pf}r_{fm} \end{bmatrix}$$

Again, notice that the equation looks pretty similar to the example in section 2.1.1, but in a more general size of both the number of users movies, and genres.

2.2.3 Matrix Factorization for Recommendation Systems

So, how to find the right factorization?

Training Models The first method is training models [2]:

Let's suppose we know some of the ratings of movies from some users, i.e., we know some entries of matrix A. The computer would randomly come up with two matrices and claim them as A's factors. Then by comparing the computer's results with our existing results, we give feedback to the computer. Then computer adjust the two matrices based on the feedback. By adjusting the matrices and continuing giving feedback after each adjustment, we can get accurate matrices as the factors. Using the example in 2.2.1 but with some missing ratings:

Suppose the table below is the rating information the company already has. The blank cells represent the rating the company needs to predict. And we write it as a matrix, denoting it as matrix A':

	M1	M2	M3	M4	M5
A	3	1	1		1
B	1		4	1	
C	3	1		3	1
D		3		4	4

$$\sim \begin{bmatrix} 3 & 1 & 1 & & 1 \\ 1 & & 4 & 1 & \\ 3 & 1 & & 3 & 1 \\ & 3 & & 4 & 4 \end{bmatrix}$$

Suppose the below is a factor that the computer randomly comes up with for the first time, representing people's appreciation level for comedy and action. And we write it as a matrix, denoting it as matrix X:

	Comedy	Action
A	0.2	0.5
B	0.3	0.4
C	0.7	0.8
D	0.4	0.5

$$\sim \begin{bmatrix} 0.2 & 0.5 \\ 0.3 & 0.4 \\ 0.7 & 0.8 \\ 0.4 & 0.5 \end{bmatrix}$$

And suppose the matrix below is another factor that the computer randomly comes up with, representing how many comedy and action features each movie contains. All numbers are between 0 and 1, as 0 represents fully dislike, and 1 represents fully like. And we transfer it into a matrix, denoting it as matrix Y:

	M1	M2	M3	M4	M5
Comedy	1.3	3.3	0.4	2.1	0.4
Action	2.6	3.2	2.4	1.3	0.7

$$\sim \begin{bmatrix} 1.3 & 3.3 & 0.4 & 2.1 & 0.4 \\ 2.6 & 3.2 & 2.4 & 1.3 & 0.7 \end{bmatrix}$$

Multipling X and Y, it tells us that the computer gives us a matrix A'' to approximate A':

$$XY = \begin{bmatrix} 0.2 & 0.5 \\ 0.3 & 0.4 \\ 0.7 & 0.8 \\ 0.4 & 0.5 \end{bmatrix} \begin{bmatrix} 1.3 & 3.3 & 0.4 & 2.1 & 0.4 \\ 2.6 & 3.2 & 2.4 & 1.3 & 0.7 \end{bmatrix}$$

$$= \begin{bmatrix} 1.56 & 2.26 & 1.28 & 1.07 & 0.43 \\ 1.43 & 2.27 & 1.08 & 1.15 & 0.4 \\ 2.99 & 4.87 & 2.2 & 2.51 & 0.84 \\ 1.82 & 2.92 & 1.36 & 1.49 & 0.51 \end{bmatrix}$$

Then we tell the computer to adjust all the entries in X and Y based on the difference between entries we already know in A' and the corresponding entries in A''. If $a''_{ij} < a'_{ij}$, we should increase the corresponding entries in X and Y that we used to obtain this specific a''_{ij}, and vice versa. For example, since we already know that $a'_{12} = 1$, from the assumption of computer, $a''_{12} = 2.26 > 1$, which means we need to tell the computer to decrease $x_{11}(0.2)$, $x_{12}(0.5)$,$y_{21}(3.3)$, and $y_{22}(3.2)$ (since $x_{11} \times y_{21} + x_{12} \times y_{22} = a''_{12}$). However, keep in mind that other a''_{ij} entries need to be modified since they are obtained based on x_{11}, x_{12}, y_{21}, and y_{22} too. We must list all the goals and modify each entry in X and Y to satisfy each condition.

This method is widely used, but the computer can develop a reasonable assumption for X and Y instead of random ones to accelerate the process. Firstly, the assumption should match with entries of A' we already knew. Secondly, the assumption should match the dependencies we mentioned in 2.2.1. For example, Based on the information we already had, A doesn't like M2, M3, and M5. Also, A and C have the same rating for M1, M2, and M5, the computer can come up with an assumption that matches the two observations. In this way, the adjusting process get shorten and the resulted matrices come out quicker. Additionally, some computer algorithms use *Error Function* to calculate how much it needs to change entries in X and x_{11} so the whole adjustment process gets accelerated [2].

3 Conclusion

In conclusion, this paper provides a comprehensive understanding of how a recommendation system can utilize matrix factorization to make personalized recommendations. The paper explores the concept of matrix factorization, highlights its importance in reducing the dimensions of large data sets, and explain the algorithm of improving the accuracy of the system. In the paper we saw some basic examples of how a recommendation system tied to matrix factorization. For each section, we saw limited examples and problem-solving ways. Still, there are other models and methods that can provide an accurate matrix factorization.

Having a good recommendation system is important as it can bring significant benefits to both the company and its users. From the firm's perspective, a good recommendation system can enhance user engagement, improve user retention, and increase sales. From the user's perspective, it enhances user experience by providing personalized and relevant recommendations, increasing satisfaction, and introducing users to new contents [1].

References

[1] B. Hallinan, T. Ttriphas, Recommended for you: The Netflix Prize and the production of algorithmic culture, *New Media and Society*, Vol. 18(1), 2016, 117-137.

[2] M. Nasiri, B. Minaei, Z. Sharifi, Adjusting data sparsity problem using linear algebra and machine learning algorithm, *Applied Soft Computing*, Vol. 61, 2017, 1153-1159.

[3] T. Lyche, G. Muntingh, Ø. Ryan, *Numerical Linear Algebra and Matrix Factorizations*, Springer, New York, 2020.

Reflection

1. Did you find the multiple drafts process useful?

   Yes it is useful. Multiple drafts process helps me plan things out evenly. I don't feel pressure out since I know at each deadline I just need to finish a certain thing.

2. Did you change your paper in response to your peer's comments or from your reading of their paper?

   For me, reading other's paper really helps improve mine. Since it's my first time writing a math paper, there are places where I'm not sure how to structure. But from reading others, I can learn from theirs.

3. What was the most important thing you learned writing this paper?

   How to construct an academic paper. I find out that when writing such a paper, I need to fully understand the materials that I'm going to write about and write it in a way that other people can understand.