# Recommendation Using Matrix Factorization

Paritosh Pantola · Follow

7 min read · Jun 10, 2018



In the era of digital world, we see recommendation in every area weather it is e-commerce website or entertainment website or social network sites. Recommendation not only gives user its recommended choice (based on past activity) but it also tells about user behavior (sentimental analysis or emotion AI).

So first, let's understand what recommendation is. It is basically recommending item to user based on its past search/activity.

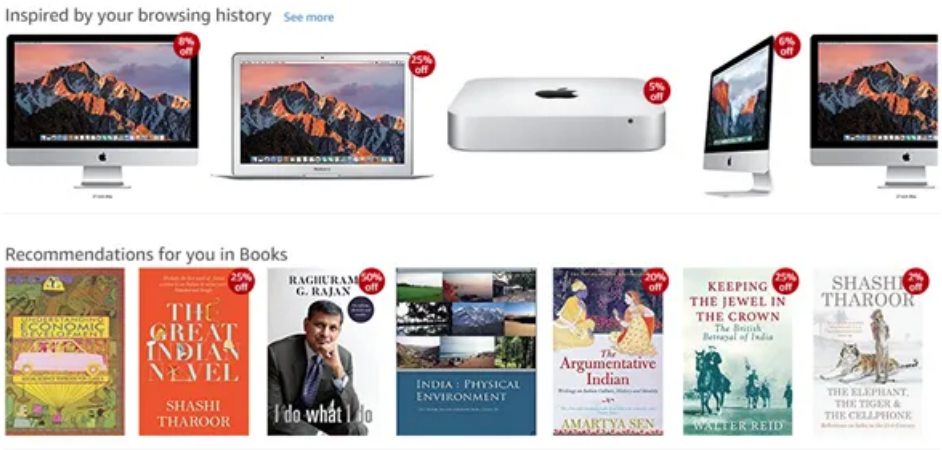



Figure 1- Amazon Recommendation

Figure 1 tells Amazon's recommendation based on the past browsing history and the past searches. So, we can say that recommendation is basically predicting future behavior based on past behavior. There are two type of approaches which is used in recommendation system

1- Content Based Filtering

2- Collaborative based filtering

**Content Based Filtering-**

It is based on the idea of recommending the item to user K which is similar to previous item highly rated by K. Basic concept in content based filtering is TF-IDF (Term frequency — inverse document frequency), which is used to determine the importance of document/word/movie etc. Content based filtering shows transparency in recommendation but unlike collaborative filtering it can't able to work efficiently for large data

**Collaborative based filtering**

It is based on the idea that people who share the same interest in certain kind of items will also share the same interest in some other kind of items unlike content based which basically rely on metadata while it deals with real life activity. This type of filtering is flexible to most of the domain (or we can say it is domain free) but because of cold start problem, data sparsity (which was handled by matrix factorization) these type of algorithm faces some setback in some scenario.

**Matrix factorization**

Matrix factorization comes in limelight after Netflix competition (2006) when Netflix announced a prize money of $1 million to those who will improve its root mean square performance by 10%. Netflix provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies.

Matrix factorization is the collaborative based filtering method where matrix m*n is decomposed into m*k and k*n . It is basically used for calculation of complex matrix operation. Division of matrix is such that if we multiply factorized matrix we will get original matrix as shown in Figure 2. It is used for discovering latent features between two entities (can be used for more than two entities but this will come under tensor factorization)

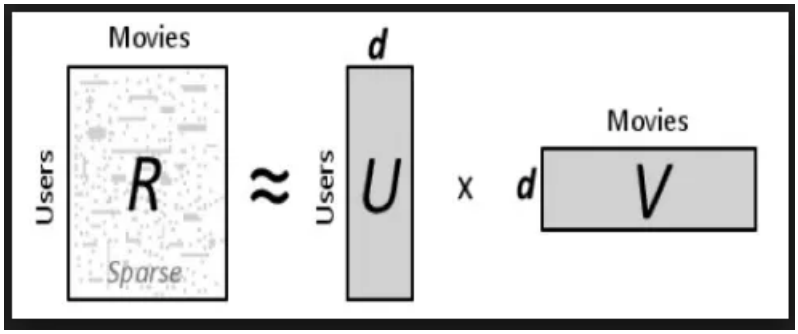



Figure 2: Matrix Factorization- Source

Matrix decomposition can be classified into three type-

**1- LU decomposition** — Decomposition of matrix into L and U matrix where L is lower triangular matrix and U is upper triangular matrix, generally used for finding the coefficient of linear regression. This decomposition failed if matrix can't have decomposed easily

**2- QR matrix decomposition**- Decomposition of matrix into Q and R where Q is square matrix and R is upper triangular matrix (not necessary square). Used for eigen system analysis

**3- Cholesky Decomposition**- This is the mostly used decomposition in machine learning. Used for calculating linear least square for linear regression

Matrix factorization can be used in various domain such as image recognition, Recommendation. Matrix used in this type of problem are generally sparse because there is chance that one user might rate only some movies. There are various applications for matrix factorization such as Dimensionality reduction (to know more about dimensionality reduction please refer to Curse Of Dimensionality), latent value decomposition

**Problem statement**

In Table 1 we have 5 users and 6 movies where each user can rate any movie. As we can see that Henry did not rate for Thor and Rocky similarly Jerry did not rate for Avatar. In real world scenario these types of matrix can be highly sparse matrix

Our problem statement is we have to find out ratings for unrated movies as shown in Table 1

| | Avenger | Thor | DeadPool | Avatar | Rocky | Titanic |
|---|---|---|---|---|---|---|
| Pumba | 4 | 5 | 3 | 3 | 1 | - |
| Henry | 5 | - | 3 | 2 | - | 5 |
| Jerry | 1 | 2 | 2 | - | 4 | 2 |
| Tom | 3 | 4 | - | 2 | 4 | 1 |
| Timon | 4 | 2 | 3 | 5 | 3 | - |

Table 1- Movie Dataset

As our aim to find rating of the user by matrix factorization but before that we have to go through Singular Value Decomposition (SVD) because matrix factorization and SVD are related to each other

**SVD and Matrix factorization**

Before going to depth of SVD lets first understand what is K and transpose(K). If we perform PCA on matrix K(Table 1) we get all the user vector. Later we can put these vectors into column of matrix U and if we perform PCA on transpose(K) we get all the movie vector which become column for matrix M.

So instead of performing PCA on K and transpose(K) separately, we can use SVD which can perform PCA on K and transpose(K) in a single shot. SVD is basically factorized matrix K into matrix U , matrix M and diagonal matrix:

$$K = M \sum U^T$$

where K is Original Matrix, U is user matrix,M is movie matrix and the middle one is diagonal matrix

For the sake of simplicity, we can remove diagonal matrix for a while so new equation becomes:

$$K = MU^T$$

Let r is rating defined for user u and item i, p is row of M for a user and q is the column of transpose(U) for a specific item i. So equation will become:

$$r_{ui} = p_u q_i$$

> *Note — If K is dense matrix value than M will be eigenvector of K*transpose(K), similarly value of U will be eigenvector of transpose(K)*K but our matrix is sparse matrix we cannot calculate U and M by this approach*

So here our task is to find matrix M and U . One way is to initialize random value to M and U and compare it with original Matrix K if value are close to K than stop the procedure otherwise minimize the value of U and M until they are close to K . The process of these type of optimization is called gradient descent

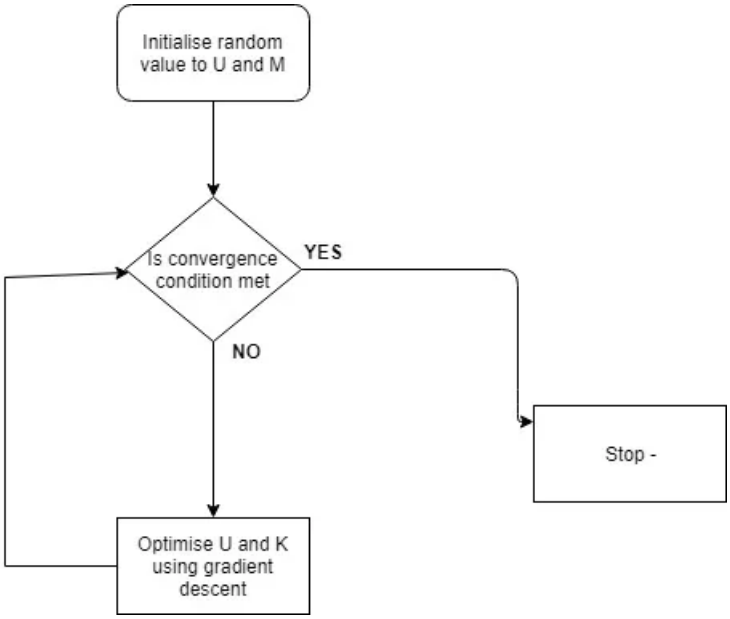



Figure 3: Gradient Descent Flow Chart

So our basic task is to minimize the mean square error function which can be represented as :

$$MSE = (Y - \widetilde{Y})^2 = (r_{ui} - \sum_{a}^{b} p_u q_i)^2$$

As our main task is to minimize the error i.e. we have to find in which direction we have to go for that we have to differentiate it, hence new equation will become

$$\frac{\partial MSE}{\partial q_i} = -2(Y - \widetilde{Y}) * p_u$$

$$\frac{\partial MSE}{\partial p_u} = -2(Y - \widetilde{Y}) * q_i$$

Hence the updated value for p and u will be now become:

$$\tilde{p}_u = p_u + \alpha \frac{\partial MSE}{\partial p_u}$$

$$\tilde{q}_i = q_i + \alpha \frac{\partial MSE}{\partial q_i}$$

Where alpha is learning rate generally its value is very small as higher alpha can overshoot the minimum value.

**Regularization in Recommendation**

When we fit the model to the training data it returns some decision line . Based on that decision line we can able to distinguish the distribution of data

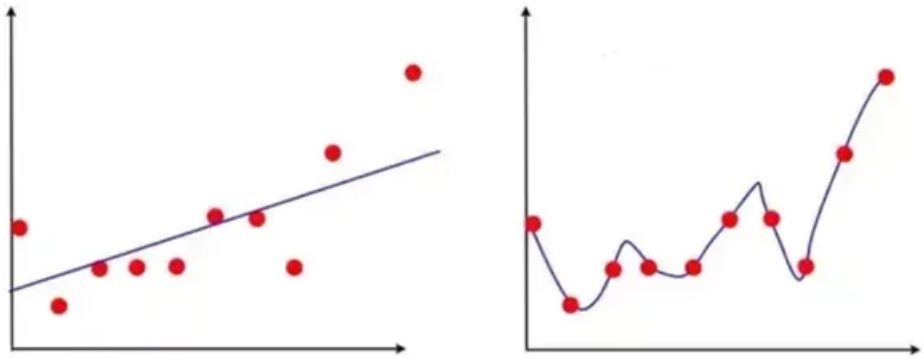


Figure 4: Underfit vs Overfit

In Figure 2 first plot is where model is fitted linearly while in second plot it is fitted with polynomial degree. At first look it looks like second plot is far better than first plot as it is giving 99% accuracy on training data but what if we introduce test data and it give 50% accuracy on test data . This types of problem is called overfitting and to overcome this problem we introduce a concept called regularization.

As overfitting problem is common in machine learning so there are two type of regularization approach which can be used to remove the overfitting

1- L1 regularization

2- L2 regularization

L1 regularization add linear magnitude of coefficient as penalty term while in L2 it add square magnitude of coefficient to the loss function/error function (as discussed above). L1 return sparse matrices while L2 does not. L1 regularization works well in feature selection in sparse matrix.

In recommendation dataset also, there are high chances of overfitting of data. So, to remove the overfitting problem we can add L2 regularization (as matrix is already sparse we do not need L1 regularization here)in loss function hence new equation of loss function will be:

$$MSE = (r_{ui} - \sum p_u q_i)^2 + \beta \sum ||P||^2 + ||Q||^2$$

Again we can find gradient on updated MSE and get updated points by following the same approach as discussed above

One important thing to be consider in above scenario the matrix P and Q are non-negative hence these types of factorization is also called non-negative factorization. As non-negative factorization automatically extracts information for non-negative set of vector. NMF is widely used in image processing ,text mining, analysis of high dimensional data

**Reference:**


1- Intro to Collaborative Filtering

2- Introduction to Content Filtering

3- Wikipedia

4- Understanding matrix factorization for recommendation

5- Matrix Factorization: A Simple Tutorial and Implementation in Python

6- A Gentle Introduction to Matrix Factorization for Machine Learning


Machine Learning | Matrix Factorization | Recommendations | Svd


## Written by Paritosh Pantola

108 Followers


Follow


More from Paritosh Pantola

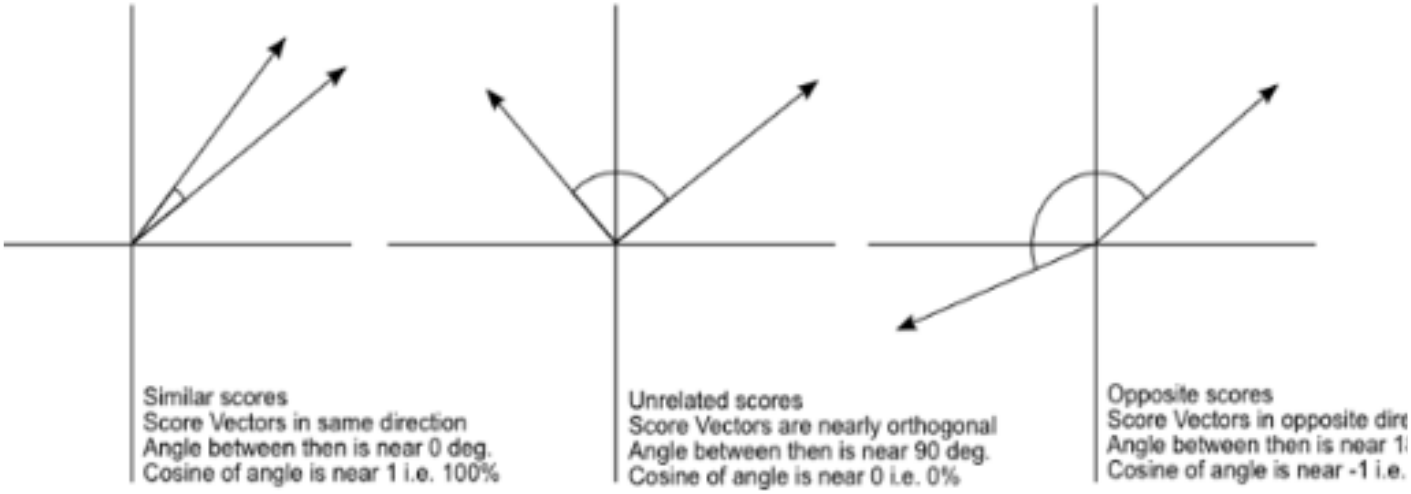



Figure 5- Cosine Similarity

Paritosh Pantola

## Natural Language Processing: Text Data Vectorization

Features in machine learning is basically numerical attributes from which anyone can perform some mathematical operation such as matrix...

9 min read · Jun 14, 2018

807 8

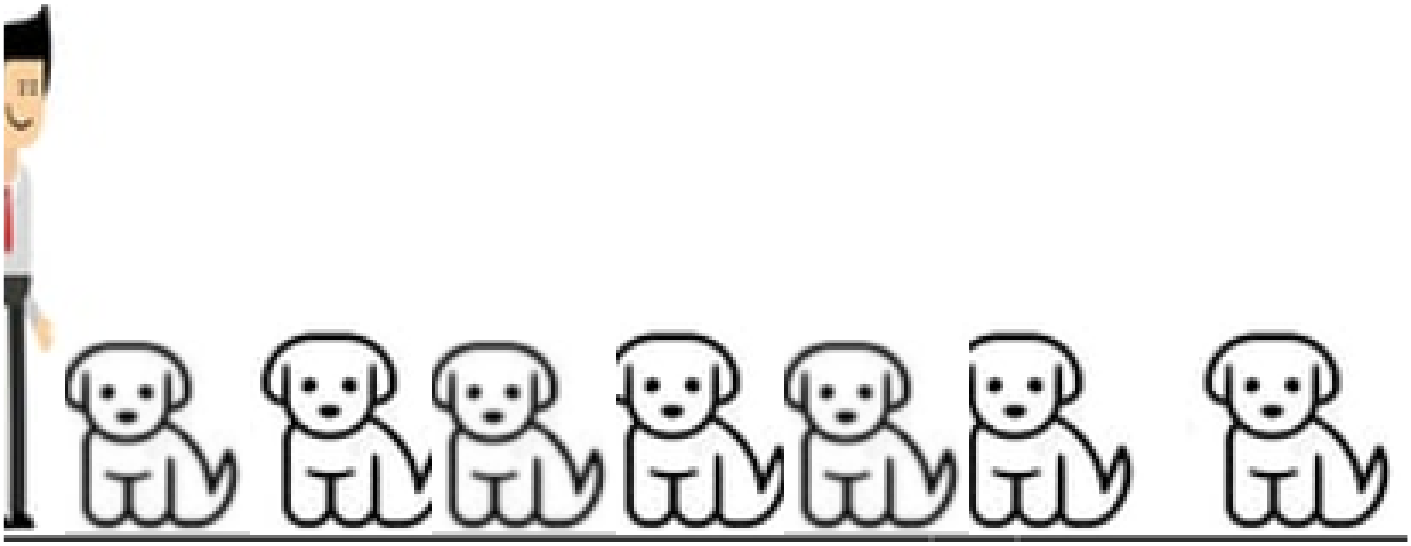


*Figure 1 - One-dimension scenario*

Paritosh Pantola

## Curse of dimensionality

Introduction

8 min read · May 10, 2018

46 1

Paritosh Pantola

## XAI(eXplainable AI)-Child of Deep Learning And Machine Learning

Due to the ambiguity in Deep Learning solutions, there has been a lot of talk about how to make explainability inclusive of an ML pipeline...

6 min read · May 15, 2020

41

See all from Paritosh Pantola

## Recommended from Medium


Konstantin Rink in Towards Data Science

## Mean Average Precision at K (MAP@K) clearly explained

One of the most popular evaluation metrics for recommender or ranking problems step by step explained

· 7 min read · Jan 18

50 7


Angel Das in Towards Data Science

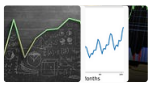
## Exploring Recommendation Systems: Review of Matrix Factorization & Deep Learning Models

Summary of Recommender Systems (Alternate Least Square, LightFM, Matrix Factorization with Neural Networks, and Neural Collaborative...

✦ · 9 min read · Nov 10, 2022

184

### Lists



**Predictive Modeling w/ Python**

18 stories · 106 saves

**Practical Guides to Machine Learning**

10 stories · 116 saves

**Natural Language Processing**

385 stories · 43 saves

**The New Chatbots: ChatGPT, Bard, and Beyond**

13 stories · 43 saves

Dr. Robert Kübler in Towards Data Science

## Introduction to Embedding-Based Recommender Systems

Learn to build a simple matrix factorization recommender in TensorFlow

✦ · 13 min read · Jan 25

393 4

Dr. Robert Kübler in Towards Data Science

# A Performant Recommender System Without Cold Start Problem

When collaboration and content-based recommenders merge

· 11 min read · Jan 31

51 4

Samuel Flender in Towards Data Science

## Biases in Recommender Systems: Top Challenges and Recent Breakthroughs

Behind the ongoing quest for building unbiased models from biased data

· 7 min read · Feb 23




Abhay Parashar in Level Up Coding

# Different Approaches For Building Recommender Systems Using Python

A Comprehensive Guide To Recommender Systems

· 8 min read · Mar 8