POLITECNICO
MILANO 1863

Numerical Analysis for Machine Learning
Academic Year 2021 - 2022

---

movieRecommendation

Sofia Martellozzo Matteo Nunziante

Professor

Edie Miglio

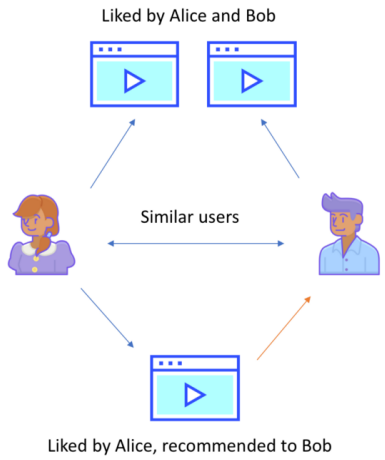
Contents

1 Introduction

With the advent of the internet today, we are witnessing an enormous information overload. This exponential growth in data results in difficulty organizing and analyzing this basic information but opens up new avenues on the paths of knowledge. The question is no longer to have the information but to find the relevant information simultaneously; from there, recommendation systems were born.

Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Recommender systems produce a list of recommendations in any of the two ways :

- **Collaborative filtering**: Collaborative filtering approaches build a model from the user's past behavior as well as similar decisions made by other users. This model is then used to predict ratings for items that users may have an interest in.



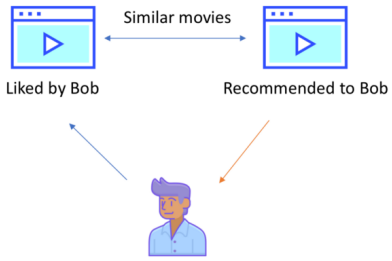
The advantages of this approach are:

  - Domain knowledge not required: The system does not required a domain knowledge because is based only on item ratings.
  - Serendipity: 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.

The limitations of this approach are:

- Cold start problem: The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item.
- Sparsity: The system may find problems on predicting evaluation because of a situation in which the users evaluate a little of the total number of items available in a dataset. This creates a sparse matrix with a high number of missing values.

- **Content-based filtering**: Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties. Content-based filtering methods are totally based on a description of the item and a profile of the user's preferences. It recommends items based on the user's past preferences.



The advantages of this approach are:

- User autonomy:The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
  The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.
- Immediate consideration of a new item: The model does not need the evaluation of all movie by a user, because it can be recommended without being evaluated.

The limitations of this approach are:

- Content too specific: 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.
- Big scale information: Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of

domain knowledge. The items must be enouth detailed descripted and a user must evaluate several items before the system can iterpret its preferences.

It is also possible to combine these two class of recommendation in order to overcome some limitations they faced. This type of approach is called **Hybrid Recommendation**.
In this project the items that has been avaluated are movies, and their score are the rating that the users gave them, is supposed after they whatched it.

2 Objectives

The purpose of this project is to develop a recommender system, which predicts the rating of a user towards a domain-specific item. In our case, this domain-specific item is a movie, and the main focus of our recommendation system is to filter and predict only those movies which a user would prefer. The approach chosen to develop this system is an hybrid one. At first we used a content-based filtering to predict some movie's ratings based on the similarity between movies. Then we adopted a collaborative filtering to make other predictions on movie's ratings deducted from similarity between users. The based knowledge of our system derives a dataset, described thoughly in the next section, while in the fourth section there is a step by step description of the algorithm, followed by an evaluation of it's performance and an example of output it generates.

3 Dataset description

The data provided are stored in the two CSV file named movies.csv and ratings.csv

3.1 Movies

In this dataset are stored information about the movies. The content is structured in 3 different columns :
the **id**, the **title** and a list of **genres** of one movie; each row represents a movie.
In it are stored 10329 movies with 20 different genres:
Western , Documentary, Children, Crime, Film-Noir, Comedy, Adventure, Fantasy, Horror, Thriller, Mystery, Sci-Fi, Musical, Romance, Action, Animation, IMAX, Drama, War, not-defined.
In the following image there is a representation of it:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
10324	146684	Cosmic Scrat-tastrophe (2015)	Animation|Children|Comedy
10325	146878	Le Grand Restaurant (1966)	Comedy
10326	148238	A Very Murray Christmas (2015)	Comedy
10327	148626	The Big Short (2015)	Drama
10328	149532	Marco Polo: One Hundred Eyes (2015)	(no genres listed)

10329 rows × 3 columns

3.2 Ratings

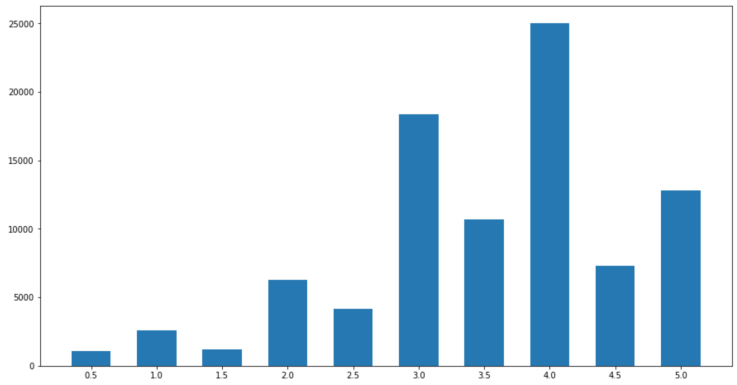
Here are stored all the ratings that the users gave to the movies from 0 to 5. The 4 columns are filled with respectively:

the id of a **user**, the id of a **movie**, the value of the **rate** of the user on the movie and the day and time on which the evaluation has been made. Counting the different userId we found that there are 668 users that overall made 105339 ratings.
To follow a representation of it:

	userId	movieId	rating	timestamp
0	1	16	4.0	1217897793
1	1	24	1.5	1217895807
2	1	32	4.0	1217896246
3	1	47	4.0	1217896556
4	1	50	4.0	1217896523
...	...	...	...	...
105334	668	142488	4.0	1451535844
105335	668	142507	3.5	1451535889
105336	668	143385	4.0	1446388585
105337	668	144976	2.5	1448656898
105338	668	148626	4.5	1451148148

105339 rows × 4 columns

We also plot the distribution of the ratings data:



4 Algorithm description

In this section a detailed step by step description of the algorithm adopted is provided.
At first we load the data with the *pandas* library, and then organize them in matrices.
And then we split these data:

- 80% as training set
- 20% as testing set

It is necessary to shuffle the data before, or in the test set would be only the data about lasts users; the analysis would not be correct.
The first matrix have all the 668 users id as rows, 10329 movies id as columns and filled with the rating that users gave on movies (the ones missing are set to 0.0).

	1	2	3	...	148626	149532
1	0.0	0.0	0.0		0.0	0.0
2	0.0	0.0	2.0		0.0	0.0
3	0.0	0.0	0.0		0.0	0.0
4	0.0	0.0	0.0		0.0	0.0
...						
668	0.0	3.0	0.0		4.5	0.0

4.1 Binary search

Binary search is an algorithm that find an element in an array with a complexity of O(log n). It search in a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the interval is empty. In this way we made the search of an element in the matrix in a more efficient way, also because of the big amount of indexes that compose it.

Built movie-gener matrix M :

$$M_{i,j} = \begin{cases} 1 & \text{if movie i is of genre j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

	Comedy	Drama	Horror	...	Fantasy	Sci-Fi
1	1	0	0		1	0
2	0	0	0		1	0
3	1	0	0		0	0
4	1	1	0		0	0
...						
149532	0	0	0		0	0

4.2 User-user similarity

Also calculate the similarity between 2 users, looking on their evaluation on common movies.

Get each user as vector of the ratings matrix and copare with the others, after just keeping the rating on the same movies:
user1= $\begin{pmatrix}1.0 & 3.0 & 2.0 & ... & 4.0 & 1.5\end{pmatrix}$
user2= $\begin{pmatrix}1.0 & 3.5 & 0.5 & ... & 4.0 & 3.0\end{pmatrix}$

4.3 Movie-movie similarity

Calculate the similarity between 2 movies and create a matrix with the results for all the pair of movies. Two movies are defined similar if are defined with almost the same genres.

At each iteration get two movie from the matrix M as vectors:
movie1= $\begin{pmatrix}1 & 0 & 0 & 1 & 0 & 1\end{pmatrix}$
movie2= $\begin{pmatrix}1 & 1 & 0 & 1 & 0 & 1\end{pmatrix}$

4.3.1 Jaccard similarity coefficient

[NOT USED AT THE END] It is a statistic index that is used for gauging the similarity and diversity of sample sets, defined by interaction divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

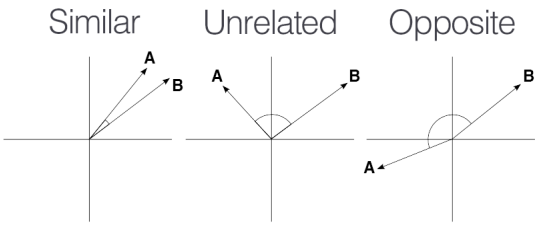
so it is a value between $0 \leq \text{J(A,B)} \leq 1$

4.4 Cosine similarity

It gaves a measure of similarity between two non-zero vectors of an inner product space.
It calculate the cosine of the angle(θ) between them, which is also the same as the inner product of the same vectors normalized to both have lenght 1.

Because of that it is bounded in the interval [-1,1] for any angle θ :

- two vectors with the same orientation have a cosine similarity of **1**



- two vectors oriented at right angle relative to each other have a similarity of **0**
- two vectors diametrically opposed have a similarity of **-1**

By using the Euclidean dot product formula:
$\mathrm{A} \cdot \mathrm{B} = \|A\| \, \|B\| \cos\theta$
We obtain the cosine similarity formula:

$$S_c(A, B) = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3}$$

where A_i anf B_i are components of vector A and B respectively.

Then create the clusters that, each, contains all the movies calculated before as similar one to another. Them are build as dictionary: the keys are the number of the corresponding group (equal to the order on wich are created), the values are list filled with the film of that group.
Create another dictionary for the cluster of user similars: the keys again the number of the cluster, the values aso a list but of user id.
To keep data organized and easilly to reach, sort the values in the dictionaries.

4.5 Collaborative filtering

Fill the matrix ratings, predicting the ones missing: for each user, get his similars from the cluster on which he belongs; get the avarage value of the rating gave the most by them.

4.6 Content-based filtering

For each user get all the movies he evaluate (with a ratings ≥ 0.5), and for each of them get all the movies similar to it from the movie-movie dictionary.The ones that haven't been evaluated yet, predict the rating with a value equal to the avarage of the one evaluated or predicted yet with the collaborative filtering (through the ones in the cluster).
Movie not evaluated= or anv of all the rating of similar movies or if predicted

with collaborative, the avg btw that predicted rating and the avg of all the similar

4.7 SVD

In linear algebra, the singular value decomposition (SVD) is a factorization of a real or complex matrix. It generalizes the eigendecomposition of a square normal matrix with an orthonormal eigenbasis to any m × n matrix.

$$S = U\Sigma V^T \tag{4}$$

Specifically, the singular value decomposition of an m × n complex matrix S is a factorization of the form $U\Sigma V^T$, where **U** is an m × m complex unitary matrix, Σ m × n rectangular diagonal matrix with non-negative real numbers on the diagonal, and **V** is an n × n complex unitary matrix.
The diagonal entries $\sigma_i = \Sigma_{ii}$ of Σ are known as the singular values of S. The number of non-zero singular values is equal to the rank of S. The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of S, respectively.
-Filter on the value just lower than the threshold
-reconstruct the matrix multiply the tre matrix U a diagonal one with the eigenvalues higher than the treshold and V transposed

5 Results

From the result obtained get an example: from one user, his ratings, the first 10 movie our system would reccommend

5.1 Error calculation

Compare the results obtained with the dataset part kept apart for the testing

5.1.1 RMSE

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

The formula is:

$$RMSE = \sqrt{\frac{1}{|\Omega_{test}|} \sum_{i,j \in \Omega_t est} (r_{i,j} - r_{i,j}^{pred})} \tag{5}$$

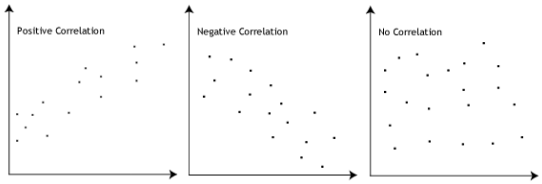
5.1.2 rho

Pearson correlation coefficient is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus it is essentially a normalised measurement of the covariance, such that the result always has a value between -1 and 1. As with covariance itself, the measure can only reflect a linear correlation of variables, and ignores many other types of relationship or correlation. It is calculated as follow:

$$\rho = \frac{\sum_{\Omega_{test}} (r_{ij} - \overline{r})(r_{ij}^{pred} - \overline{r^{pred}})}{\sqrt{\sum_{\Omega_{test}} (r_{ij} - \overline{r})^2} \sqrt{\sum_{\Omega_{test}} (r_{ij}^{pred} - \overline{r^{pred}})^2}} \tag{6}$$

The result is a value between [-1,1]:

- 1 = perfect match-strong positive relation; all data points lie on a line for which both variable r and r_{pred} are increasing
- 0 = random; there is no linear dependency between the variables
- -1 = strong negative relation; also here a linear equation describes the relationship between the variables r and r_{pred} and the points lie on a line in which are decreasing



5.2 Precision

Precision tries to answer the following question:
'What proportion of positive identification was actually correct?'

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

5.3 Recall

Recall tries to answer the following question:
'What proportion of actual positives was identified correctly?'

$$Recall = \frac{TP}{TP + FN} \tag{8}$$