Movie Recommender System

A project by

Abdul Qadir (1MS16EC004)

Anudeep Kumar (1MS16EC016)

Aparna Karnik (1MS16EC019)

Gaurav Chaudhary (1MS16EC034)

Movie Recommender System

1.Problem Statement

We have all received suggestions on Amazon on what to buy next? Or suggestions on what websites you may like on Facebook? Recommender systems are so prevalently used in the net these days that we all have come across them in one form or another. Such systems are helpful in navigating us into the right direction.

We are aiming to build a simple recommender system for a movie database which will be able to: suggest top N movies similar to a given movie title to users, and predict user votes for the movies they have not voted for.

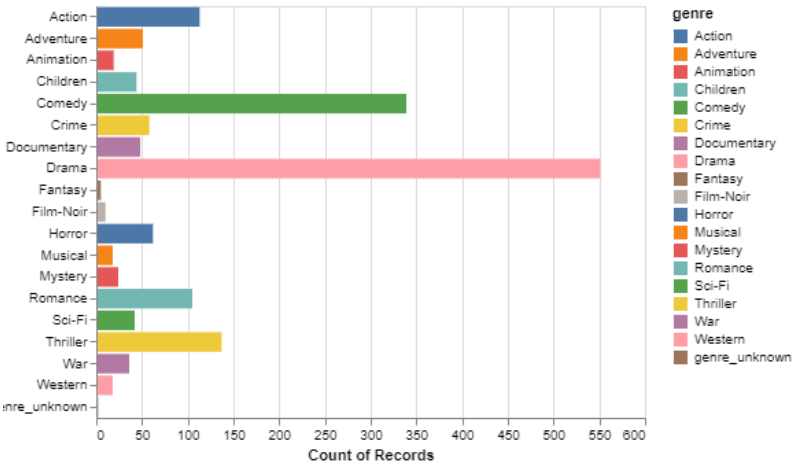
2.Dataset

2.1 Movie Lens

Movie Lens is a web-based recommender system and virtual community that recommends movies for its users to watch, based on their film preferences using collaborative filtering of members' movie ratings and ovie reviews. It contains about 11 million ratings for about 8500 movies. Movie Lens dataset was created in 1997 by GroupLens Research, a research lab in the Dept of Computer Science and Engineering at the University of Minnesota, in order to gather research data on personalised recommendations. The full data set contains 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. It also

includes tag genome data with 12 million relevance scores across 1,100 tags .There are many types of research conducted based on the MovieLens data sets. Liu et al. used MovieLens data sets to test the efficiency of an improved random walk algorithm by depressing the influence of large-degree objects.

Fig1: Number movies in each genre graphs for MovieLens Dataset



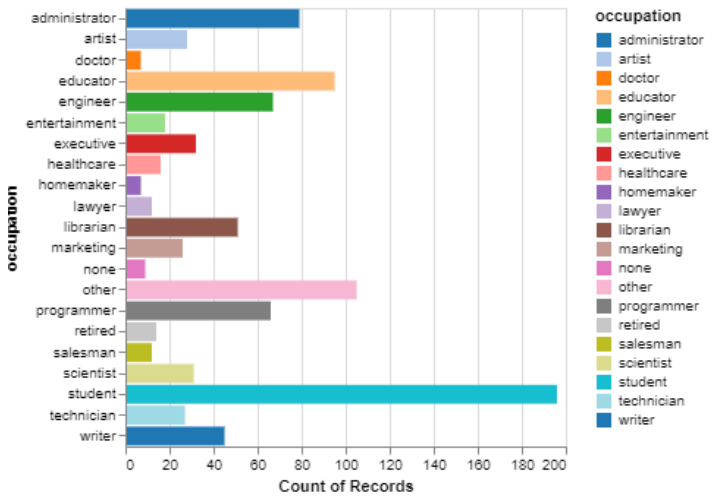


Fig2: Occupations of users graphs in MovieLens Dataset.

2.2 Indian Regional Movie Dataset

Indian regional movie dataset is the first database of regional Indian movies, users and their ratings. It consists of movies belonging to 18 different Indian regional languages and metadata of users with varying demographics. Through this dataset, the diversity of Indian regional cinema and its huge viewership is captured. We analyze the dataset that contains roughly 10K ratings of 919 users and 2,851 movies using supervised and unsupervised collaborative filtering techniques like Probabilistic Matrix Factorization, Matrix Completion, Blind Compressed Sensing etc. The dataset consists of metadata information of users like age, occupation, home state and known languages. It also consists of metadata of movies like genre, language, release year and cast. India has a wide base of viewers which is evident by the large number of movies released every year and

the huge box-office revenue. This dataset can be used for designing recommendation systems for Indian users and regional movies

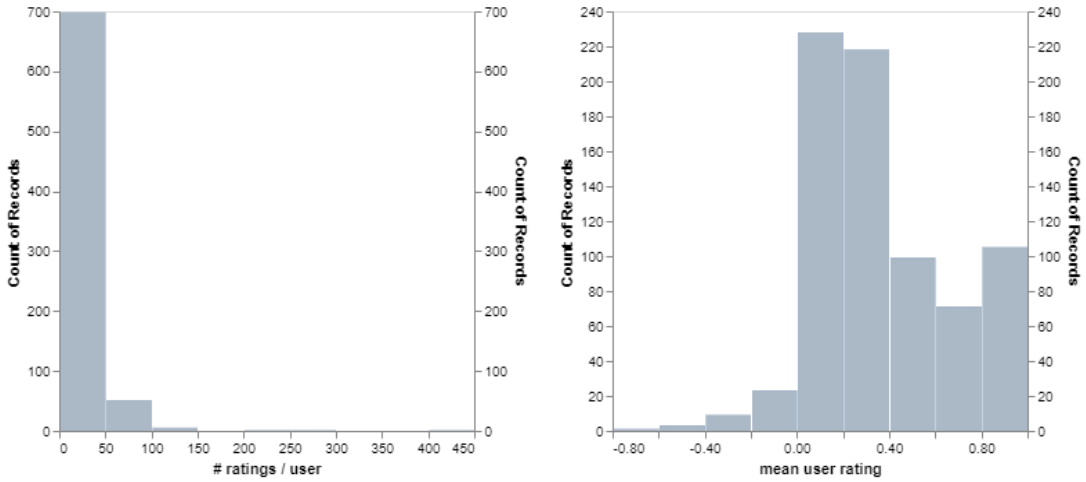


Fig3: number of ratings per user, mean user rating

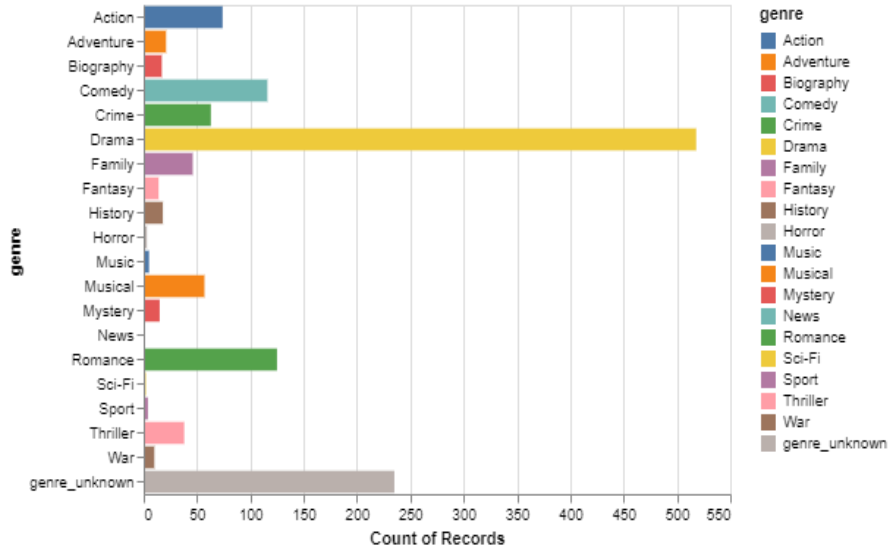


Fig4: number of ratings per movie, mean movie rating and number of movies in different genre graphs for Indian Regional Movie Dataset.

3.Approach

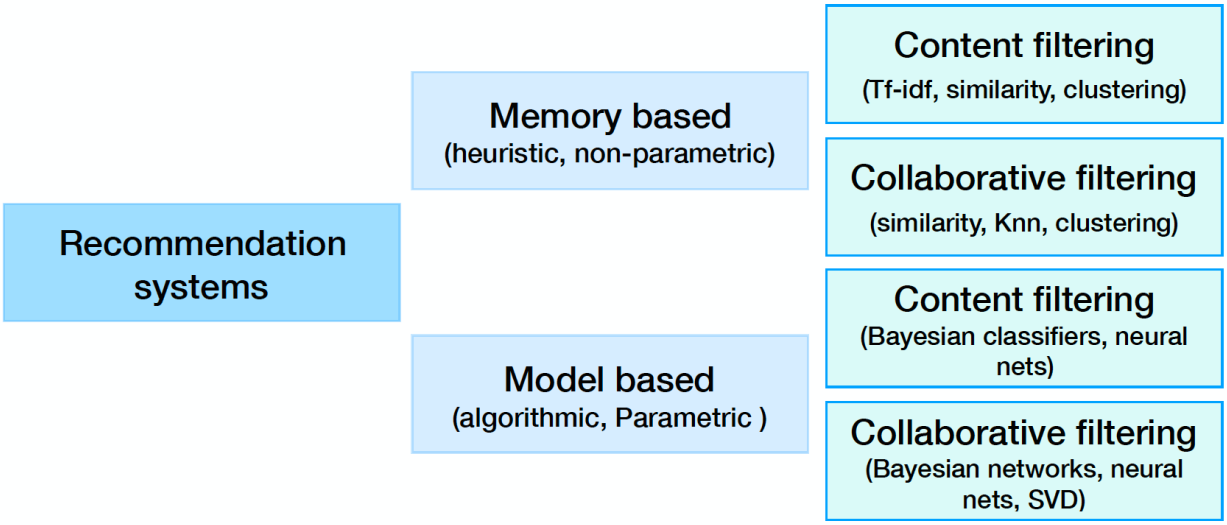


Fig 5 Different methodologies for building a recommender system

3.1 Machine Learning Approach

We will be computing an estimate to SGD in an iterative learning process. For this purpose, we only use the known ratings and try to minimise the error of computing the known rates via gradient descent. SGD was chosen because it produces a comparable accuracy to neural nets with a simpler training procedure. This algorithm was popularised during the Netflix prize for the best recommender system.

Matrix factorization is a simple embedding model. Given the feedback matrix $A \in \mathbb{R}^{m \times n}$, where m is the number of users (or queries) and n is the number of items, the model learns:

- A user embedding matrix $U \in \mathbb{R}^{m \times d}$, where row i is the embedding for user i.
- An item embedding matrix $V \in \mathbb{R}^{n \times d}$, where row j is the embedding for item j.

The embeddings are learned such that the product UV^T is a good approximation of the feedback matrix A. Observe that the (i,j) entry of $U.V^T$ is simply the dot product $\langle U_i, V_j \rangle$ of the embeddings of user i and item j, which you want to be close to $A_{i,j}$

One intuitive objective function is the squared distance. To do this, minimize the sum of squared errors over all pairs of observed entries:

$$\min_{U \in \mathbb{R}^{m \times d},\ V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2.$$

In this objective function, you only sum over observed pairs (i, j), that is, over non-zero values in the feedback matrix. However, only summing over values of one is not a good idea—a matrix of all ones will have a minimal loss and produce a model that can't make effective recommendations and that generalizes poorly.

Perhaps you could treat the unobserved values as zero, and sum over all entries in the matrix. This corresponds to minimizing the squared Frobenius distance between A and its approximation UVT

$$\min_{U \in \mathbb{R}^{m \times d},\ V \in \mathbb{R}^{n \times d}} \|A - UV^T\|_F^2.$$

You can solve this quadratic problem through **Singular Value Decomposition** (**SVD**) of the matrix. However, SVD is not a great solution either, because in real applications, the matrix A may be very sparse. For example, think of all

the videos on YouTube compared to all the videos a particular user has viewed. The solution UVT (which corresponds to the model's approximation of the input matrix) will likely be close to zero, leading to poor generalization performance.

In contrast, **Weighted Matrix Factorization** decomposes the objective into the following two sums:

- A sum over observed entries.
- A sum over unobserved entries (treated as zeroes).

$$\min_{U \in \mathbb{R}^{m \times d},\, V \in \mathbb{R}^{n \times d}} \sum_{(i,j) \in \text{obs}} (A_{ij} - \langle U_i, V_j \rangle)^2 + w_0 \sum_{(i,j) \notin \text{obs}} (\langle U_i, V_j \rangle)^2.$$

3.2. Deep Learning Approach

3.2.1 Model 1

The first model we implement is a simple linear model where we learn a dense representation for each movie and each user in our dataset. This model does an element-wise multiplication of the movie vector and the user vector then applies a linear layer in order to produce the predicted score.

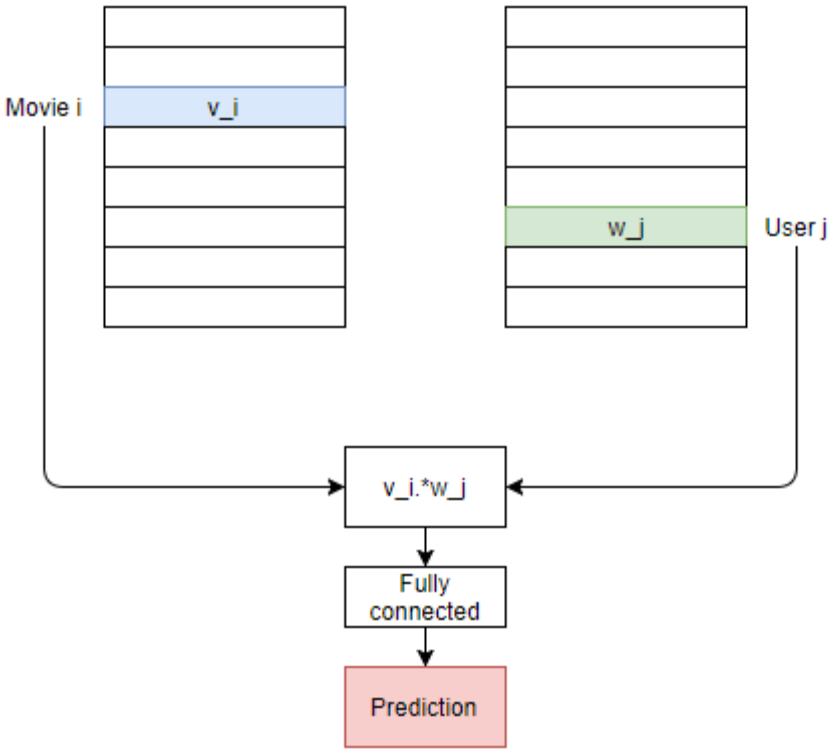


Fig 6 Block Diagram representing model1

3.2.2 Model 2

The first model does not explicitly take into account the bias that a user might have in giving consistently high scores to every movie he watches or a movie having consistently bad scores for all users. This is why we introduce a bias for each of the users and for each movie.

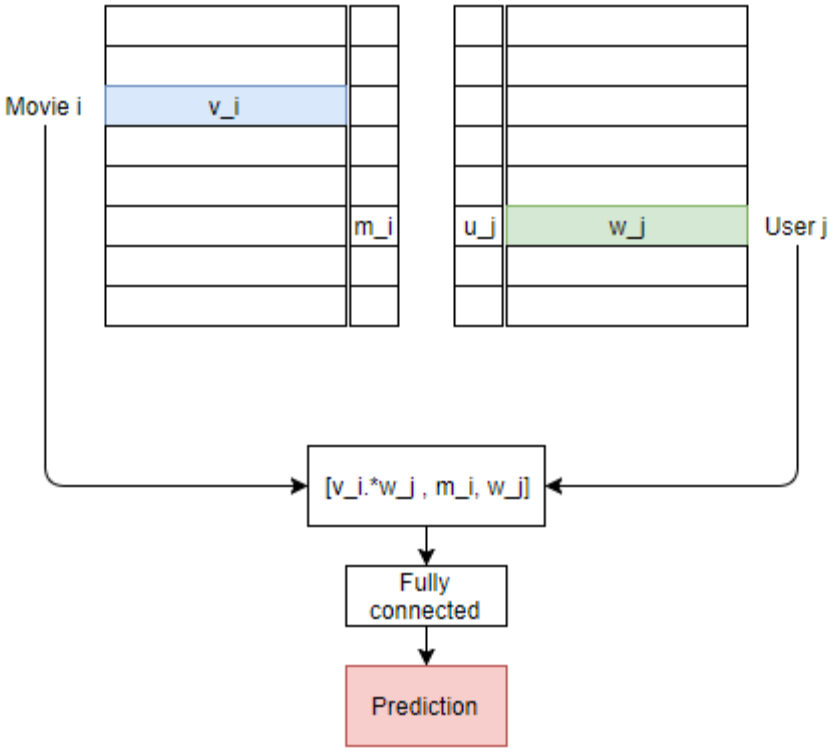


Fig 7 Block Diagram representing model 2

3.2.3 Model 3

We add a nonlinear fully connected layer in order to improve the expressivity and complexity of our network created in model 2.

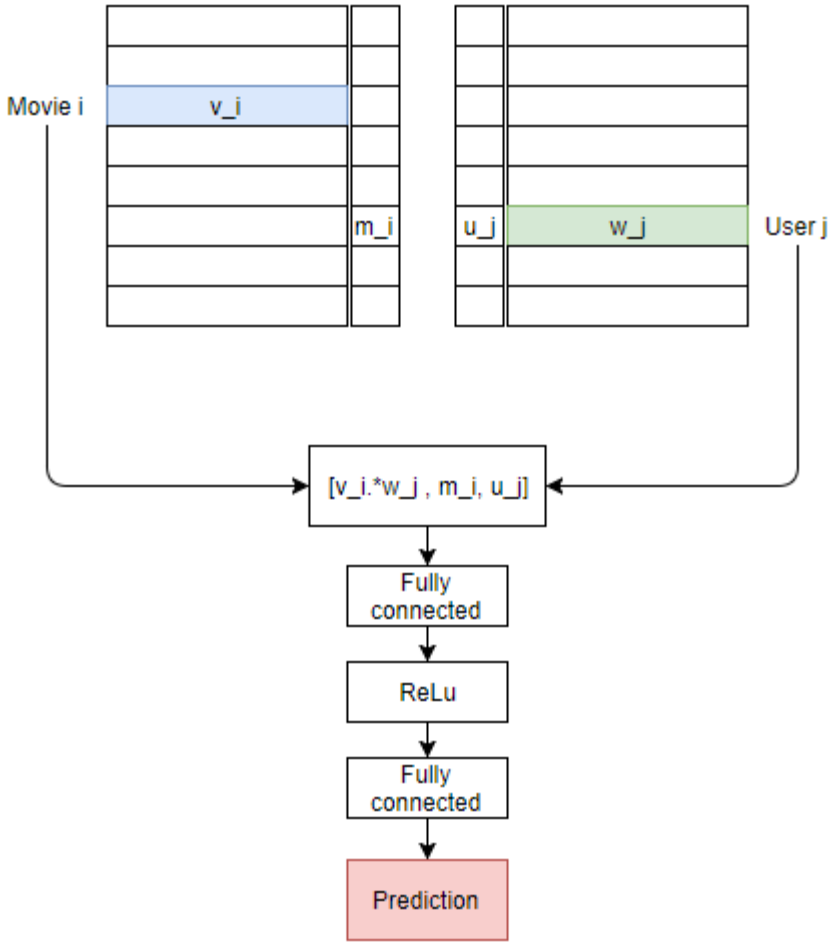


Fig 8 Block Diagram to represent model 3

One other improvement we can make is to include contextual information like the category of the movie or the description.

4.Calculation of Error

Users give a rating between 1 and 5 to some of the movies and the objective is to predict the score they are going to give to other movies in the future in order to recommend to them the ones they are most likely going to want to see.

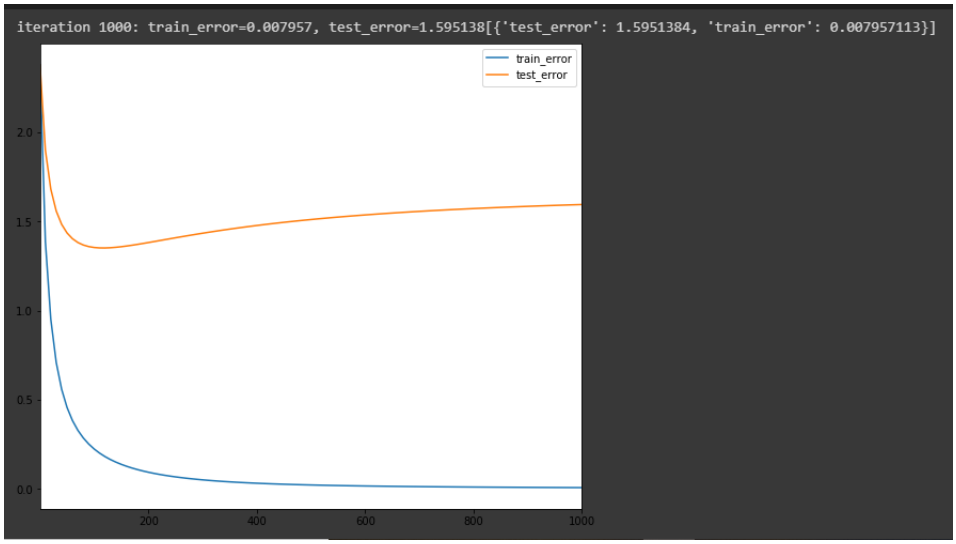
$$mean\_absolute\_error = |predicted\_scores - actual\_scores|$$

(1)

5.RESULTS

The Matrix factorization approach:

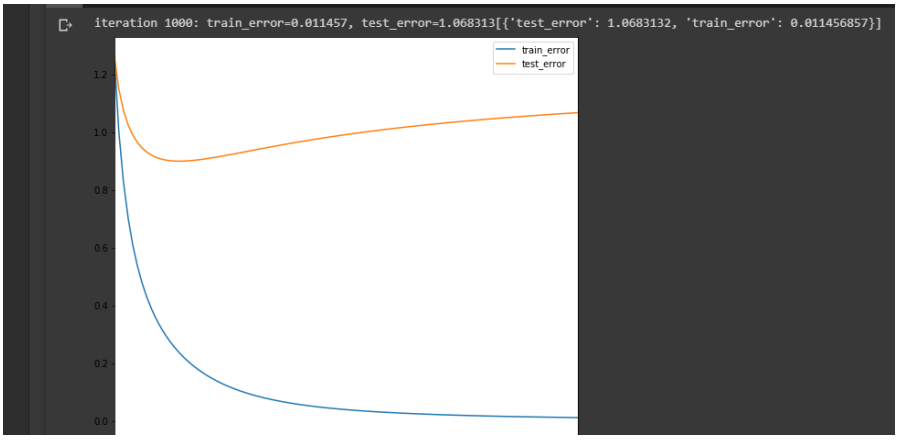
With the first approach keeping embeddings 30 and initial standard deviation 0.5

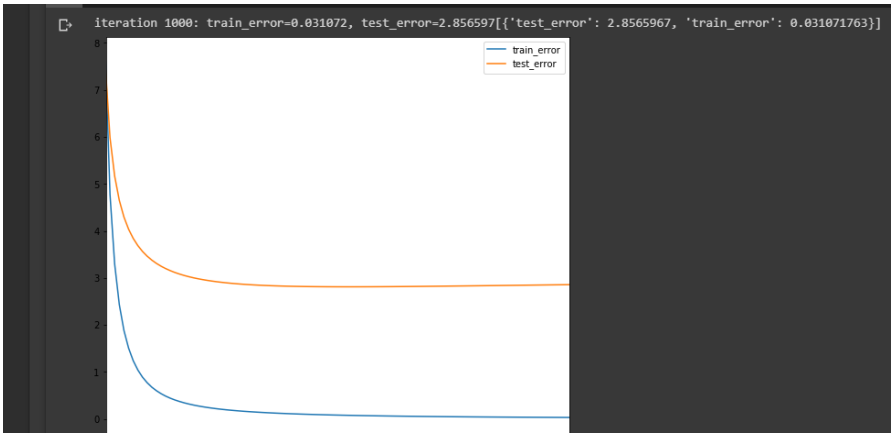


,

learning rate 10 and for 1000 epochs

Tweaking parameters still did not help in improving r





The recommendations were unpopular and did not make sense to some extent.

```
[58] 1 user_recommendations(model, measure=DOT, k=5)
```

	dot score	movie_id	titles	genres
1054	3.106	1054	Khodar Pore Ma	Action-Drama
1275	2.899	1275	Nalaik	Action-Comedy-Crime
2220	2.705	2220	Manchi Manasulu	Drama
697	2.635	697	Pujarini	genre_unknown
2586	2.601	2586	Badavara Bandhu	genre_unknown

The recommendation is also tested by finding closest neighbours to the movie via dot and cosine measures which also were not fitting well

Nearest neighbors of : Bajrangi Bhaijaan.

	dot score	titles	genres
60	5.612	Bajrangi Bhaijaan	Action-Comedy-Drama
1543	3.535	Action Hero Biju	Action-Comedy-Thriller
59	3.338	Kuch Kuch Hota Hai	Comedy-Drama-Musical
2234	3.252	Chantabbai	Comedy
1172	3.048	Long Da Lishkara	Drama-Romance
63	3.041	Guru	Drama-Biography-Musical

Nearest neighbors of : Barfi!.

	cosine score	titles	genres
28	1.000	Barfi!	Adventure-Comedy-Drama
2432	0.668	Bommarillu	Comedy-Drama-Romance
1942	0.664	Nari Akhire Nian	Drama
1659	0.618	Vitthal Vitthal	genre_unknown
997	0.587	Pora Mon	Romance
1240	0.556	Til Til Dalekha	genre_unknown

Depending on how the model is initialized, we observed that some niche movies (ones with few ratings) have a high norm, leading to spurious recommendations. This can happen if the embedding of that movie happens to be initialized with a high norm. Then, because the movie has few ratings, it is infrequently updated, and can keep its high norm. This will be alleviated by using regularization.

So we tried changing the value of the hyper-parameter init_stddev. One quantity that can be helpful is that the expected norm of a d -dimensional vector with entries $\sim N(0,\sigma 2)$ is approximately $\sigma\sqrt{d}$

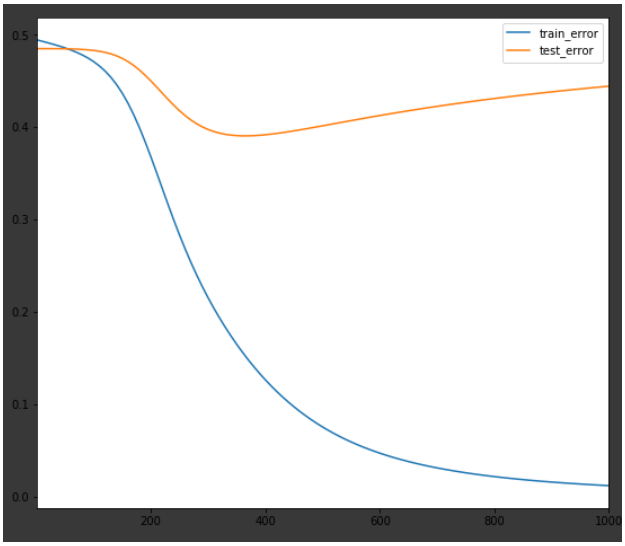
```
iteration 1000: train_error=0.011665, test_error=0.444375Nearest neighbors of : Barfi!.
```

	dot score	titles	genres
28	4.819	Barfi!	Adventure-Comedy-Drama
62	2.783	Munna Bhai M.B.B.S.	Comedy-Drama
103	2.380	English Vinglish	Comedy-Drama-Family
15	2.361	3 Idiots	Comedy-Drama
44	2.353	Jodhaa Akbar	Action-Adventure-Biography
50	2.342	Saala Khadoos	Action-Drama-Sport

Nearest neighbors of : Bajrangi Bhaijaan.

	cosine score	titles	genres
60	1.000	Bajrangi Bhaijaan	Action-Comedy-Drama
586	0.563	Ek Se Badhkar Ek	Action-Crime-Drama
277	0.547	Bin Roye	Drama-Romance
62	0.527	Munna Bhai M.B.B.S.	Comedy-Drama

Now the closest neighbours are befitting



Our loss was defined as the mean squared error on the observed part of the rating matrix. This can be problematic as the model does not learn how to place the embeddings of irrelevant movies. This phenomenon is known as folding.

We added regularization terms that will address this issue. We used two types of regularization:

Regularization of the model parameters. This is a common ℓ_2 regularization term on the embedding matrices, given by

$$r(U,V) = \frac{1}{N}\sum_i \|U_i\|^2 + \frac{1}{M}\sum_j \|V_j\|^2.$$

A global prior that pushes the prediction of any pair towards zero, called the gravity term. This is given by

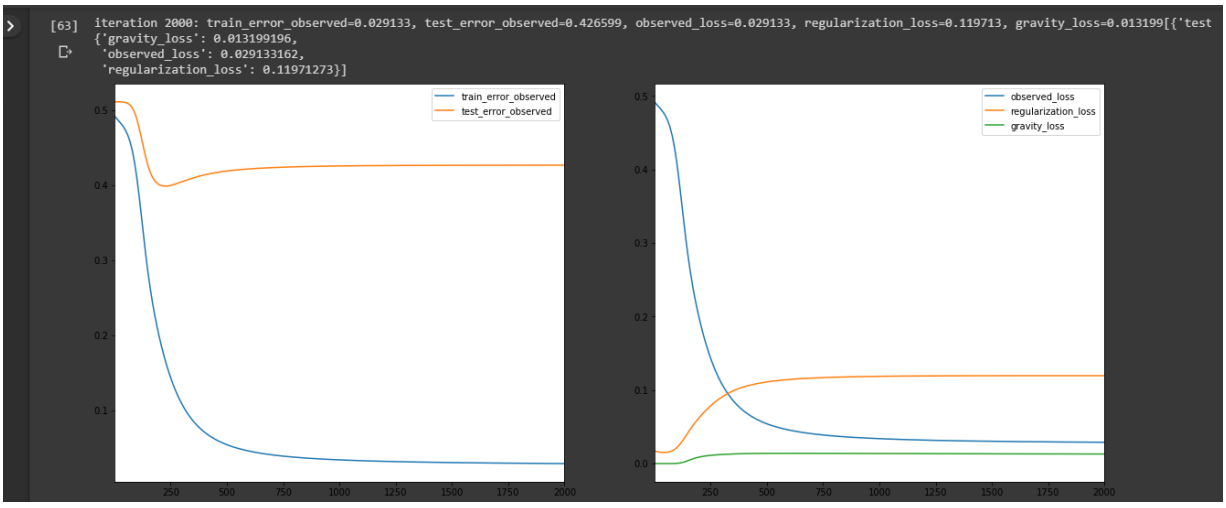
$$g(U,V) = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\langle U_i, V_j\rangle^2.$$

The total loss is then given by

$$\frac{1}{|\Omega|}\sum_{(i,j)\in\Omega}(A_{ij} - \langle U_i, V_j\rangle)^2 + \lambda_r r(U,V) + \lambda_g g(U,V)$$

where λr and λg are two regularization coefficients (hyper-parameters).

Training had results



The recommendations improved drastically

[64]

	dot score	movie_id	titles	genres
117	1.131	117	Madras Cafe	Action-Drama-History
562	1.122	562	Deewaar	Action-Crime-Drama
93	1.103	93	Vicky Donor	Comedy-Romance
20	1.091	20	Pink	Drama-Thriller
16	1.033	16	M.S. Dhoni: The Untold Story	Drama-Biography-Sport
23	0.990	23	Dil Chahta Hai	Comedy-Drama-Romance
62	0.990	62	Munna Bhai M.B.B.S.	Comedy-Drama
22	0.988	22	Swades	Drama
60	0.983	60	Bajrangi Bhaijaan	Action-Comedy-Drama
34	0.976	34	Rang De Basanti	Comedy-Drama-History

So did the Nearest neighbours

Nearest neighbors of : 3 Idiots.

	dot score	titles	genres
15	5.830	3 Idiots	Comedy-Drama
34	3.982	Rang De Basanti	Comedy-Drama-History
28	3.352	Barfi!	Adventure-Comedy-Drama
51	3.305	Jab We Met	Comedy-Drama-Romance
293	3.266	Don	Action-Crime-Thriller
162	3.219	Baghban	Drama-Romance

Nearest neighbors of : A Wednesday.

	cosine score	titles	genres
24	1.000	A Wednesday	Crime-Drama-Mystery
13	0.675	Airlift	Action-Drama-History
98	0.673	OMG: Oh My God!	Comedy-Drama-Family
66	0.666	Queen	Adventure-Comedy-Drama
7	0.636	PK	Comedy-Drama-Romance
582	0.594	Sholay 3D	Action

Deep Learning Approach:

We just started the deep learning approach and the errors obtained from each model discussed above are

Model 1: Simple linear model achieved a mean absolute error of .98.

Model 2: Introducing bias for each of the users and for each movie

achieved a mean absolute error of .92.

Model 3: Adding nonlinear fully connected layer achieved mean absolute

error of 0.84.

We will further delve into the deep learning approach and run the Regional dataset on the network.

6.Conclusion

We experimented with various versions and appendages to the matrix factorization method and drew a number of inferences to land finally to regularisation of the model and obtaining a fine recommendation system.

The machine learning approach has a less complex and faster processing approach but is highly restricted because of inability to incorporate more number of features namely, language, actor, genre into account.

References

- Vozalis, Manolis G., and Konstantinos G. Margaritis. "Applying SVD on item-based filtering." 5th International Conference on Intelligent Systems Design and Applications (ISDA'05). IEEE, 2005.

- Harper F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems, 5(4), Article 19, 19 pages. DOI=http://dx.doi.org/10.1145/2827872.
- Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. arXiv:1703.04247 [cs.IR].