Movie Recommendation using Neural Collaborative Filtering

Parth Hasmukh Jain
Rutgers University
parth.jain@rutgers.edu

Jenil Ashwin Jain
Rutgers University
jenil.jain@rutgers.edu

Aditya Jain
Rutgers University
aditya.jain03@rutgers.edu

ABSTRACT

Recommender systems, which have been widely implemented by many online businesses, social media sites, play a critical role in easing information overload in the age of information explosion. Speech recognition, computer vision, and natural language processing have all benefited greatly from deep neural networks. The use of deep neural networks in recommender systems, on the other hand, has gotten less attention. For this, we strive to develop a movie recommendation system on 'MovieLens' dataset using techniques based on neural networks i.e Neural Collaborative Filtering. We have developed a couple of baseline models such as SVDpp and KNN too along with the Deep Neural network model inorder to test our main model. Each model's hyperparameter tuning, testing accuracy, evaluation on different parameters and review of recommendations are meticulously carried out. The study includes a detailed analysis of each model.

KEYWORDS

deep neural networks, movie-recommendation, Neural Collaborative Filtering, SVDpp, KNN

1 INTRODUCTION

With the amount of data growth we have seen in the recent years, recommender systems play a pivotal role in providing services that are user specific and modelling user's preferences.

Matrix factorization (MF) is the most widely used collaborative filtering technique, which projects users and items into a common latent space using a vector of latent features to represent a user or an item. Why would we have cause to expect that adopting a neural network will increase the performance of our model on a theoretical level? The explanation is that a regular matrix factorization approach's linearity restricts a model's expressiveness, or how complex of a relationship it can model between all of our users and products.

The user-item inner product is replaced with a neural architecture in Neural Collaborative Filtering (NCF). Under this framework, NCF attempts to express and generalize MF and uses a multi-layer perceptron to learn user-item interactions. It learns user-item interactions using a Multi-Layer Perceptron (MLP). This is an improvement over MF since MLP can learn any continuous function (theoretically) and has a high level of non-linearities (due to several layers), making it well suited to learn user-item interaction functions.

We have evaluated the models on different evaluation parameters like Root Mean Square Error(RMSE), Precision, Recall and Normalized Discounted Cummulative Gain(NDCG).

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

2.1 Collaborative based Filtering

The recommendations in collaborative filtering are based on previous interactions between users and movies. A collaborative filtering system's input is historical data from user interactions with the films they see. Collaborative filtering employs ratings to create a neighborhood N for a user, say X, whose ratings (likes and dislikes) are comparable to those of his or her neighbor Y. With a notion of similarity defined, the neighborhood of X is defined using the nearest neighbor approach. If user A watches M1, M2, and M3, and user B watches M1, M3, and M4, we propose M1 and M3 to a user C who is comparable to user to A and B. Collaborative filtering can be divided into two types: user-to-user and item-to-item. This method is different from content based as the neighborhood or similarity is defined only on the basis of rating behaviour.

2.2 Matrix Factorization based recommendation models

Matrix factorization is the most widely used collaborative filtering technique, which projects persons and stuff into a shared latent space using a vector of latent features to represent a user or an item. The inner product of a user's latent vectors is then used to model their interaction with an object.

MF assigns a real-valued vector of latent features to each user and item. Let's call the latent vectors for user u and item i p_u and q_i, respectively. MF calculates an interaction $y_u i$ as the inner product of p_u and q_i.

$$\hat{y}_u i = f(u, i|p_u, q_i) = p_u^T q_i = \sum_{K=1}^{k} p_{uk} q_{ik}$$

where K is the latent space dimension.

2.3 Types of Matrix Fatorization based models

2.3.1 ***Singular Value Decomposition (SVD).*** The SVD algorithm uses a lower-dimensional representation of movies to map people who like similar films together. It finds hidden latent characteristics that allow movies to be mapped into the same space as the user. To minimize the RMSE for unseen test data, we can use the SVD model to build an optimization problem. As a result, the use of

Model	RMSE	RECALL	PRECISION	NDCG
SVD	0.8736	0.239	0.6300	0.72
SVD plus plus	0.8676	0.2512	0.648	0.72
KNN	0.9296	0.244511	0.6411	0.71

Table 1

the gradient descent approach and regularization allows for a more complex model structure. We can use a baseline predictor to model biases and interactions, which incorporates the user bias bx, the item or movie bias bi, and the overall mean rating into its rating estimation. The results are improved by including these biases.

2.3.2 ***SVD plus plus***. This model is an extension of SVD that incorporates implicit ratings. The fact that a user u rated an item j, regardless of the rating value, is referred to as an implicit rating. This combination of implicit rating along with explicit ratings gives the best outcomes. However, SVDpp takes the longest time to train and fit and is much slower than other simpler models

2.4 Types of Collaborative filtering models

Although the two approaches are mathematically similar, there is a conceptual distinction between them. Here's how they compare:

2.4.1 ***User-based***. :
For a user U, with a set of similar users determined based on rating vectors consisting of given item ratings, the rating for an item I, which hasn't been rated, is found by picking out N users from the similarity list who have rated the item I and calculating the rating based on these N ratings.

2.4.2 ***Item-based***. :
For an item I, with a set of similar items determined based on rating vectors consisting of received user ratings, the rating by a user U, who hasn't rated it, is found by picking out N items from the similarity list that have been rated by U and calculating the rating based on these N ratings.

2.4.3 ***K-nearest neighbour Algorithm***. KNN is the go-to model for item-based collaborative filtering and a great starting point for recommender system development. KNN is a lazy learning algorithm that is non-parametric. It makes inferences for new samples using a database in which the data points are divided into many clusters.
KNN depends on item feature similarity rather than making any assumptions about the underlying data distribution. When KNN produces a movie inference, it calculates the "distance" between the target movie and every other movie in its database, ranks the distances, and returns the top K closest neighbor movies as the most comparable movie recommendations.

To compare our proposed model based on Neural Collaborative Filtering we evaluate them based on different evaluation metrics that are RMSE, Recall, Precision and NDCG with different Collaborative filtering models that are KNN, SVD and SVDpp. Appendix explains what different evaluation metrics mean. Table 1 is the result of the NDCG, RMSE, Precision and recall on the base line models.

From the above results in table 1 we can see that vanilla SVD performs the worst in almost all the evaluation metrics and the KNN which is a item based collaborative filtering method performs better than vanilla SVD but its not comparable to SVD plus plus.Even though SVD plus plus takes a lot of time to execute but it dopes give better results than the other base line models

3 PROBLEM FORMULATION

3.1 Dataset Description

The dataset for this is taken from the *grouplens movie review* dataset. The dataset consists of 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 *MovieLens* users who joined *MovieLens* in 2000. Each rating is a value between 0-5.

3.2 Data Preprocessing

In order to avoid overfitting at a later stage, we randomly shuffled the data. The advantage of this is it helps the training to converge quickly, reduces bias during training, and prevents the model from memorizing the training order.

Converted the *userid* and *movieid* to integers in order to feed it to the embedding layers of the model. The embedding layer is essentially a matrix dot product of the embedding weights and the user/movie.

We performed a chronological 80/20 *train/test* split of the data. This split is on the basis of *userid* such that 80% of the reviews by every single user is used for training purposes and the remaining 20% for testing. Out of the training set, 10% of the data was used for validation.

3.3 Exploratory Analysis

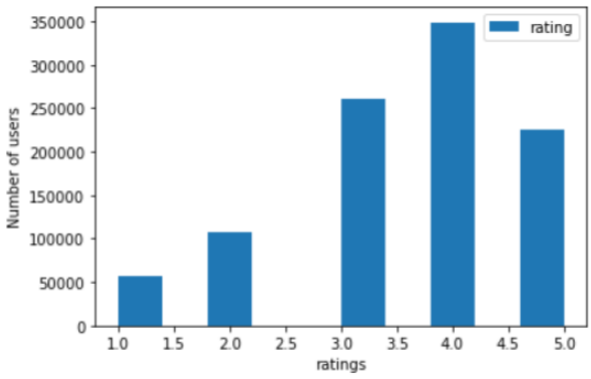

Figure 1: Histogram for ratings

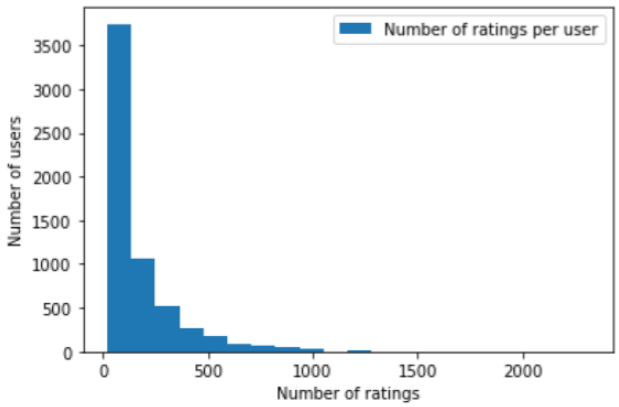

Figure 2: User-rating distribution

4 THE PROPOSED MODEL

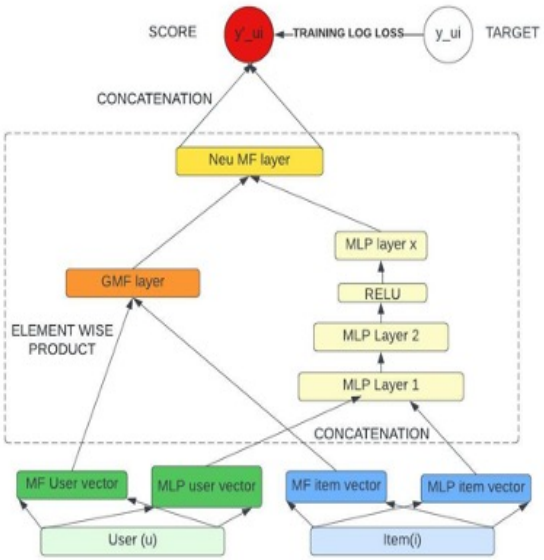

Figure 3: NCF Architecture

Despite the effectiveness of matrix factorization for collaborative filtering, it's performance is hindered by the simple choice of interaction function - inner product. Its performance can be improved by incorporating user-item bias terms into the interaction function. This proves that the simple multiplication of latent features (inner product), may not be sufficient to capture the complex structure of user interaction data. This calls for designing a better, dedicated interaction function for modeling the latent feature interaction between users and items. Neural Collaborative Filtering (NCF) aims to solve this by:-

1. Modeling user-item feature interaction through neural network architecture. It utilizes a Multi-Layer Perceptron(MLP) to learn user-item interactions. This is an upgrade over MF as MLP can (theoretically) learn any continuous function and has high level of nonlinearities(due to multiple layers) making it well endowed to learn user-item interaction function.

	edge weights	activation function	Variation
1	1	Identity function	Matrix Factorisation
2	learnable	Identity function	MF with varying importance of latent dimensions
3	learnable	non linear function	Non-linear MF which is more expressive

Table 2

2. Generalizing and expressing MF as a special case of NCF. As MF is highly successful in the recommendation domain, doing this will give more credence to NCF.

Because the Neural collaborative Filtering aims to solve the problem faced by the baseline models we aim to implement the Neural Collaborative Filtering Recommendation model.

Our NCF model as seen in Figure 3 has 2 components GMF and MLP with the following benefits:- 1. Generalized Matrix Factorization that applies the linear kernel to model user-item interactions like vanilla Matrix Factorization 2. Multi-layer Perceptron(MLP) that uses multiple neural layers to layer nonlinear interactions

4.1 Generalized Matrix Factorization

The predicted output of the NCF can be expressed as

$$\hat{y}_{ui} = a_{out}(h^T(p_u \odot q_i))$$

where
a-out: activation function
h: edge weights of the output layer
where $\odot$ denotes the element-wise product of vectors.

We can play with a-out and h to create multiple variations of GMF.

As you can see from table 2 that GMF with identity activation function and edge weights as 1 is indeed MF. The other 2 variations are expansions on the generic MF. The last variation of GMF with sigmoid as activation is used in NCF.

In this work, we implement a generalized version of MF under NCF that uses the sigmoid function $\sigma(x) = 1/(1+ex)$ as aout and learns h from data with the log loss .This is known as the generalized matrix factorization.

4.2 Multi-layer Perceptron

NCF is an example of multimodal deep learning since it contains data from two paths: user and item. Concatenation is the most natural approach to join them. However, a simple vector concatenation is insufficient to describe the collaborative filtering effect since it does not account for user-item interactions. In order to learn user-item interactions, NCF adds hidden layers on top of concatenated user-item vectors (MLP framework). This gives the model a lot of flexibility and non-linearity when it comes to learning user-item interactions. This is an improvement above MF, which employs a fixed element-by-element product. To be more specific, the MLP modifies Equation 1 as follows:

$$z1 = \phi 1\left(pu, qi\right) = \begin{bmatrix} pu \\ qi \end{bmatrix}$$

$$\phi_2\left(z1\right) = a2\left(W2^T z1 + b2\right)$$

$$\ldots\ldots$$

$$\phi L\left(zL-1\right) = aL\left(WL^T zL - 1 + bL\right)$$

$$\hat{y}ui = \sigma\left(h^T \phi_L\left(z_{L-1}\right)\right)$$

where: W(x): Weight matrix
b(x): bias vector
a(x): activation function for the x-th layer's perceptron
p: latent vector for the user
q: latent vector for an item
NCF uses ReLU as an activation function for its MLP part. Due to multiple hidden layers, the model has sufficient complexity to learn user-item interactions as compared to the fixed element-wise product of their latent vectors(MF way).

4.3 Fusion of Multi-layer Perceptron and Generalized Matrix Factorization (Proposed Model)

Figure 3 is the architecture of our proposed model. NCF combines these models together to superimpose their desirable characteristics. NCF concatenates the output of GMF and MLP before feeding them into NeuMF layer.

Some important points to note about our hybrid model:
1. User and object embeddings are independent in GMF and MLP. This is to ensure that they both learn optimal embeddings on their own.
2. GMF replicates the vanilla MF by element-wise product of the user-item vector.
3. The input to MLP is a concatenation of user-item latent vectors.
4. The outputs of GMF and MLP are concatenated in the final NeuMF(Neural Matrix Factorisation) layer. The equation is modeled as:

$$\phi^{GMF} = pu^G \odot qi^G$$

$$\phi^{MLP} = a_L\left(W_L^T\left(aL-1\left(\ldots a_2\left(W2^T\begin{bmatrix} p_u^M \\ q_i^M \end{bmatrix} + b_2\right)\ldots\right)\right) + b_L\right)$$

$$\hat{y}_{ui} = \sigma\left(h^T\begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}\right)$$

G: GMF
M: MLP
p: User embedding
q: Item embedding
This model combines the linearity of MF and non-linearity of DNNs for modeling user-item latent structures through the NeuMF (Neural Matrix Factorisation) layer. Due to the non-convex objective function of NeuMF,gradient-based optimization methods can only find locally-optimal solutions. This could be solved by good weight initializations. To solve this NCF initializes GMF and MLP with pre-trained models by Random Initialisation.
1. Train GMF+MLP with random initializations until convergence.
2. Use model parameters of 1 to initialize NCF.

	no of factors	no of hidden layers	RMSE	Precision	Recall	NDCG
0	8	1	0.85722948	0.61361032	0.35473116	0.89311203
1	8	2	0.86369382	0.64042902	0.39390078	0.89381565
2	8	3	0.85706142	0.64041371	0.38342682	0.89436839
3	16	1	0.85967642	0.61494613	0.35408325	0.89363721
4	16	2	0.85169369	0.58466914	0.31761629	0.89537664
5	16	3	0.85207007	0.59376314	0.32608606	0.8941957
6	32	1	0.86002881	0.61063367	0.35317673	0.89338116
7	**32**	**2**	**0.856293**	**0.634037**	**0.37366**	**0.894659**
8	32	3	0.85395078	0.60217249	0.34024898	0.89476213
9	64	1	0.85899143	0.61368601	0.35286318	0.89272187
10	64	2	0.85484338	0.60478023	0.3496602	0.8945359
11	64	3	0.85738993	0.61959904	0.36343007	0.89448366
12	100	1	0.8583463	0.60732642	0.34815741	0.89250146
13	100	2	0.85472379	0.61417803	0.35527467	0.89368871
14	100	3	0.85120168	0.61522719	0.34747834	0.89517455

Table 3

Model	**RMSE**	**RECALL**	**PRECISION**	**NDCG**
SVD	0.8736	0.239	0.6300	0.72
SVD plus plus	0.8676	0.2512	0.648	0.72
KNN	0.9296	0.244511	0.6411	0.71
NCF	0.8562	0.37366	0.6340	0.8946

Table 4

3. The weights of the two models are concatenated for the output layer as

$$h \leftarrow \begin{bmatrix} \alpha h^{GMF} \\ (1-\alpha) h^{MLP} \end{bmatrix}$$

where
h(GMF): h vector of the pre-trained GMF
h(MLP): h vector of the pre-trained MLP
alpha: Hyper-parameter determining the trade-off between the 2 pre-trained models

5 EXPERIMENTS

To properly see what hyper parameters gives us the best results we tuned our model based on different hyper parameters that are number of factors and number of hidden layers and Table 3 shows the result .

As we can see that model number 7(which is in bold) with number of factors as 8 and number of hidden layers as 2 gives us the best values for evaluation metrics . Now we will compare these values with our baseline models .

From Table 4 we can see that our proposed model performs better than other baseline models.

6 CONCLUSIONS

We could see that we trained and generated recommendation based on different collaborative filtering models and amongst the different

baseline model SVD plus plus performed the best that is a variation of SVD which takes into account users implicit ratings. NCF learns a probabilistic model that emphasizes the binary property of implicit data. We discussed how MF can be expressed and generalized under NCF (Using General Matrix Factorisation GMF). NCF explores the use of DNNs for collaborative filtering, by using a multi-layer perceptron (MLP) to learn the user-item interaction function. Lastly, we discussed a new neural matrix factorization model called NeuMF, which ensembles MF and MLP under the NCF framework; it unifies the strengths of linearity of MF and non-linearity of MLP for modeling the user-item latent structures. Then we implemented our proposed model and found out that our model performs better than other baseline models.

7 FUTURE WORK

A possible extension to improve the model's accuracy, is using non-numerical features like movie reviews. To combat the issue of cold start, using user segmentation and classify user in one of the segments and recommend products in that segment could be also done. Using deep learning to predict time-aware recommendations and learning to rank. Varying hyper parameters such as learning rate and adding regularization tricks like dropout to avoid overfitting.

8 ACKNOWLEDGEMENT

We would like to thank the Professor Mr. Yongfeng Zhang and the teaching assistant Mr. Zhiqing Hong for their guidance as well as for providing necessary information regarding the project and also for their support in completing the project.

9 APPENDIX

Rating evaluation parameters

9.1 Root Mean Square Error

RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data, that is, how close the observed data points are to the model's predicted values. It is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed. The smaller an RMSE value, the closer the predicted and observed values are or the closer you are to finding the line of best fit. RMSE is the square root of the average of squared errors. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Formula to calculate RMSE:

$$\textbf{RMSE} = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$$

Where,
n is the number of data points,
y_i is the predicted value,
xi is the actual/observed value

Recommendation Evaluation parameters

9.2 Precision

The top-k set is a collection of k items that are recommended to each user. Precision at k is the proportion of recommended items in the top-k set that are relevant. It assesses the prediction model's ability to create accurate predictions. For example, if a top-5 recommendation model's Precision at 5 is calculated to be 80%, it signifies that 80% of the recommendations given by this model are relevant to the user. As a result, we strive to improve the correctness of our model at all times. Precision is calculated using the following formula:

$$Precision = \frac{|Recommended - Items - that - are - relevant|}{Recommend - items}$$

Recommended Items that are relevant are the items with both the original and predicted ratings greater than or equal to t, Recommended items are those items which have predicted rating greater than or equal to t.

9.3 Recall

The proportion of relevant items identified in the collection of top-k recommendations is known as recall at k. For example, if our prediction model's recall at 10 is found to be 35%, this means that 35% of the total number of relevant items appear in the top-10 recommendations. So, we always try to maximize the recall value for the model.Below is teh formula for recall

$$Precision = \frac{|Recommended - Items - that - are - relevant|}{Relevant - items}$$

Recommended Items that are relevant are those items with both the original and predicted ratings greater than or equal to t, Relevant items are those items which have original rating greater than or equal to t

9.4 Normalized Discounted Cumulative Gain (NDCG):

NDCG is a ranking metric which allows relevance scores in form of real numbers. It is used to evaluate the goodness of the recommendation list returned by the model. It is based on the concept that more relevant items should be placed at the beginning of the recommended list as items placed at later positions in the list tend to be overlooked by the users. Each item in the recommendation list has a relevance score which is the original rating in our case. This is called as gain Gi. For items which don't have a relevance score (or original rating), gain is usually set to zero. To see the most relevant items at the top of the list, we divide each by a growing number (usually a logarithm of the item position) which is called as discounting. We then add these discounted items to get a DCG

To make DCGs directly comparable between users, we need to normalize them. We arrange all the items in the ideal order using the actual ratings and compute DCG for them which is called as Ideal DCG (IDCG). We then divide the raw DCG by this ideal DCG to get NDCG@K, a number between 0 and 1.

$$NDCG = \frac{DCG}{IDCG}$$

The worst possible DCG when using non- negative relevance scores is zero. Therefore, good recommendation lists have NDCG scores closer to 1.

REFERENCES

[1] X. He ,L,Nie ,L. Liao ,X Hu, H. Zhang , T. Chua .Neural Collaborative Filtering.In https://arxiv.org/abs/1708.05031

[2] I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. In WWW, 2017.

[3] T. Chen, X. He, and M.-Y. Kan. Context-aware image tweet modelling and recommendation. In MM, pages 1018–1027, 2016.

[4] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In WWW, pages 278–288, 2015.

[5] Y. Zheng, B. Tang, W. Ding, and H. Zhou. A neural autoregressive approach to collaborative filtering. In ICML, pages 764–773, 2016.

[6] https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401