Recommendation System - Matrix Factorization

Ashwini Muralidharan (200483207)
Purushothaman Saravanan (200483316)
Shahil Manoj Dhotre (200475496)

1 Introduction

While the initial motivation may seem to be derived from the popular "Netflix prize problem" [1], recommendation systems find their use and application in many other fields such as "Business-to-customer" in E-commerce, sports game results prediction, gene expression prediction, online learning recommender systems, etc.

The types of recommendation systems approaches include [2]

- Collaborative filtering: Collaborative filtering predicts suggestions for a particular by observing and processing the preferences of many users (collaborating). It works on the assumption that if person A has the same preference as person B on a product, A is more likely to have B's opinion on a different product as well
- Content-based filtering: It recommends other items similar to what the user has previously liked/preferred. Future predictions are based on their previous actions or explicit feedback
- Hybrid approaches: This approach combines collaborative filtering, content-based filtering, and other approaches

Collaborative filtering is the basis of the Recommender System used in this project, and it predicts suggestions for a particular user by observing and processing the preferences of many users (collaborating). It can be achieved by applying Matrix Factorization [3], where a sparse matrix contains the user ratings for movies. Predicting the unknown matrix entry, which relates to user-item ratings, would result in a new suggestion of a movie, item, or product to the user.

The usage of recent technology in recommendation systems has been documented in [4, 5]. [4] provides a thorough review of some highly notable works on deep learning-based recommender systems. This paper served as motivation to delve into Neural Architecture and Deep Learning. Deep learning effectively captures the non-linear and non-trivial user-item relationships, and enables the codification of more complex abstractions as data representations in the higher layers, hence gaining popularity over conventional recommendation systems.

Matrix completion techniques are explored in this project to build a recommendation system. The method was implemented on the MovieLens dataset [6] using Keras to create a flexible model that can predict the possible ratings the user can give to unwatched movies and suggest the best among them.

2 Recommender System

2.1 Dataset

The data for the Deep learning model is extracted from Grouplens's MovieLens 25M dataset [6] which contains: 25 million ratings and one million tag applications applied to 62,000 movies by 162,000 users. It consists of ratings assigned to movies by users.

2.2 Model Architecture

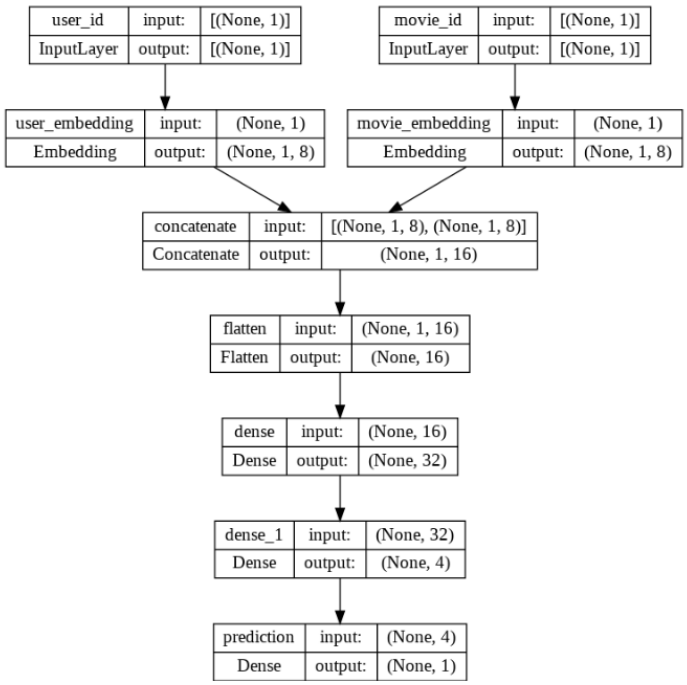

Figure 1: Model Architecture

A network of different layers of artificial "neurons" that processes inputs to produce desired output is the essence of Neural Networks. Neural networks tend to have multiple "hidden" layers as part of deep learning algorithms, where each hidden layer has its activation function. They apply transformations on the input data via weighted interconnected 'nodes' containing activation functions. Outputs from one hidden layer serve as inputs to the next one. Lastly, there is an output layer, that returns values that correspond to the prediction value.

An interesting detail to note here is that neither the User IDs and Movie IDs were passed as numerical inputs nor were they treated as categorical data and One-Hot Encoded as inputs. Algorithms can misinterpret these numerical values as some hierarchy/order and prioritize the higher numbers. One-Hot Encoding would generate a large sparse matrix to represent a single movie. Hence, a special layer called the Embedding layer [7] that maps each element of a sparse input vector to a dense vector of real numbers was implemented. It is generally the first layer of a network. It is also trained along with the prediction model as it not only acts as a "look-up table" but also encompasses some useful latent properties of that object.

Fig. 1 represents the flowchart of the deep learning model used for matrix factorization. It was designed using the Keras.Model class. User IDs and Movie IDs were embedded into a vector space of size eight i.e., the size of the output vectors from the embedding layers for each value of User IDs and Movie IDs was 8. The Concatenate layer returned a single tensor that is the concatenation of all inputs and the Flatten layer reshaped the tensor to have a shape that is equal to the number of elements along a particular single dimension. Further, it was passed to the Dense layer.

The Dense layer performs dot product between the input tensor and weight kernel matrix, and an element-wise activation of the output values is returned. In this experiment, two dense layers having the ReLU activation function were used. The final Dense layer used the Linear activation function and generated a single output value. The model was compiled using the Adam Optimizer to minimize Mean Square Error (MSE) and Mean Absolute Error (MAE) was also reported since the output is ranging between 0 and 5.

3 Results and Discussion

Once the architecture described was constructed using Keras, the model had 2,974,393 trainable parameters. The model was fit for a batch size of 5000 for 20 epochs. The training loss was observed to be 0.5604, with an MAE of 0.5661. Fig. 2 shows the MAE after every training epoch. The orange curve, representing training MAE, reduces as the number of epochs increases. And the blue curve representing validation MAE fluctuates around 0.61, thereby optimizing the model.

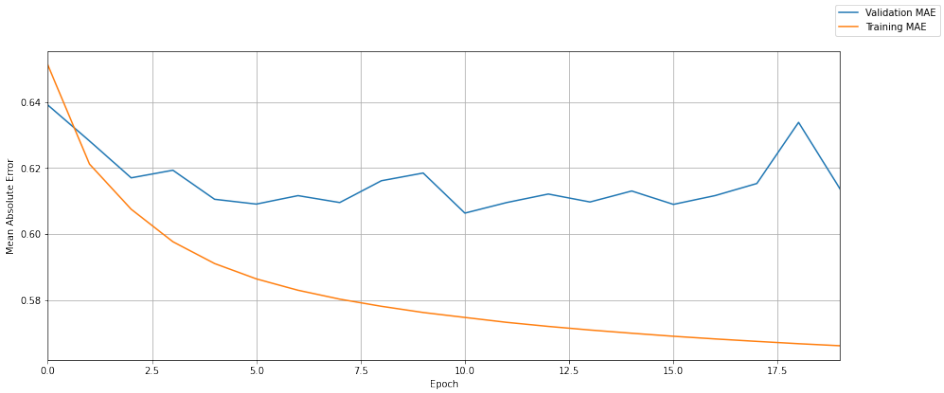

Figure 2: Mean Absolute Error vs Epoch

3.1 Prediction output for an existing user

A random user from the dataset was picked and their ratings for unwatched movies were predicted. Top five movies with the highest predicted ratings were recommended to that user. For example, an arbitrary user was picked and the above strategy was implemented, the results of which are tabulated in Table 1 and Table 2.

It is observed that the user has given high ratings to comedy and drama movies (Table 1) and our trained Keras model was also able to capture this pattern and predict movies in similar genres as shown in Table 2.

Title	Rating	Genre
Forrest Gump	5.0	Comedy, Drama, Romance, War
I Want You	5.0	Drama, Romance
Hachiko	5.0	Drama
Requiem for a Dream	5.0	Drama
The Dark Knight Rises	5.0	Action, Adventure, Crime

Table 1: Top five rated movies given by a particular user in dataset

Title	Predicted Rating	Genre
Hoaxed	4.1	Drama, Documentary
Truth and Justice	4.0	Drama
Diamond Dust	3.9	Drama, Mystery, Thriller
Boys Over Flowers	3.9	Comedy, Drama, Romance
Burn the Stage: The Movie	3.9	Documentary

Table 2: Top five predicted rating for the above user

3.2 Prediction output for a new user

MovieId	Movie Title	Genre	genre
59315	Iron Man	Action	5
88140	Captain America: The First Avenger (2011)	Action	4
106072	Thor: The Dark World (2013)	Action	5
122892	Avengers: Age of Ultron (2015)	Action	4
122904	Deadpool (2016)	Action	5

Table 3: New user's ratings

For a new user who is not in the database, retraining the whole model is cumbersome and inefficient. Instead, the new user is mapped to existing users based on the movie-rating input the new user gives. A similar, existing user from the dataset is picked and the model predicts ratings for the new user. The model then returns the top 5 movies for the similar user. This is then presented to the new user as suggestions.

This mechanism captures the essence of collaborative filtering, which assumes that if person A and person B share similar opinions on a particular issue, they will share similar opinions on a different issue. In the example depicted in Table 3, the new user has a preference for "Action" movies and is mapped to existing user number 73, who also prefers "Action" movies and has given similar ratings as the new user. The model predicts movies based on the preferences of user 73 and returns them as recommendations for the new user (Table 4 & Table 5).

4 Conclusion

Recommendation systems increase the exposure of the service by promoting user satisfaction and enhancing customer experience. Deep learning-based recommendation systems have proven to be useful due to their ability to handle large data and process

UserId	MovieId	Rating
73	260	2
73	59315	4
73	88140	4.5
73	106072	5
73	122892	5
73	122904	4.5
73	140110	4

Table 4: New user mapped to user 73

Title	Predicted Rating	Genre
White Squal	5	Action, Adventure, Drama
Trainspottin	5	Comedy, Crime, Drama
Dead Man Walking	5	Crime, Drama
Courage Under Fire	5	Action, Crime, Drama, War
Star Wars: Episode IV - A new Hope	5	Action, Adventure, Sci-Fi

Table 5: New user mapped to user 73

non-linear data. This project successfully developed a Recommendation System by applying Matrix Factorization and the trained model was tested for both an existing user and a new user.

References

[1] The netflix prize: How a $1 million contest changed binge-watching forever. [Online]. Available: https://www.thrillist.com/entertainment/nation/the-netflix-prize

[2] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, p. 331–370, nov 2002. [Online]. Available: https://doi.org/10.1023/A:1021240730564

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug 2009.

[4] S. Zhang, L. Yao, and A. Sun, "Deep learning based recommender system: A survey and new perspectives," *CoRR*, vol. abs/1707.07435, 2017. [Online]. Available: http://arxiv.org/abs/1707.07435

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," *CoRR*, vol. abs/1708.05031, 2017. [Online]. Available: http://arxiv.org/abs/1708.05031

[6] Movielens 25m dataset. [Online]. Available: https://grouplens.org/datasets/movielens/25m/

[7] Understanding embedding layer in keras. [Online]. Available: https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce