Recommendation Systems Course - Final Project Report

By Idan Kashtan

Deep Neural Networks for Recommendations

The anchor paper is: **Deep Neural Networks for YouTube Recommendations (2016)**

YouTube is the world's largest platform for creating, sharing, and discovering video content. Their recommendations are responsible for helping more than a billion users discover personalized content from an ever-growing corpus of videos. The main objective of the paper was to present the impact that deep learning has had on their recommendations system compared to SOTA from three major perspectives:

1. **Scale:** Highly specialized and efficient serving systems are essential for handling a massive user base and corpus.
2. **Freshness:** The recommendation system should be responsive enough to model newly uploaded content as well as the latest actions taken by the user.
3. **Noise:** Historical user behavior is difficult to predict due to sparsity and a variety of unobservable external factors.

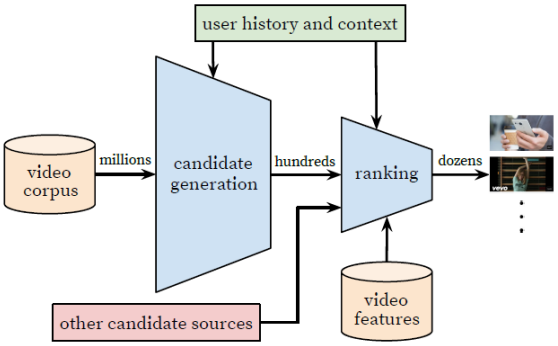

Figure 1: Recommendation system architecture. Starting from millions of videos and ending up scoring only dozens.

The system is comprised of two neural networks: one for candidate generation and one for Ranking. The Candidate generation retrieves a small subset (hundreds) of videos from a large corpus given the user's activity and features. This network only provides broad personalization via collaborative filtering. The ranking network uses a rich set of features describing only the candidate videos and users to predict the expected watch time.

Both Candidate generation and Ranking networks have the same architecture. They both use Embedding to learn a dense vector representation and use three fully connected layers with ReLu between them.

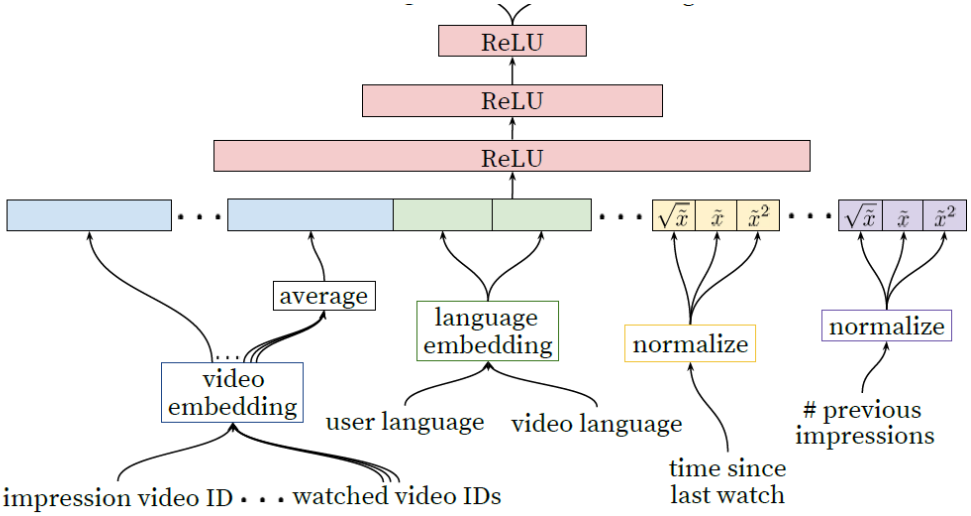

Figure 2: Deep neural network for both *Candidate generation and Ranking. Using three layers and Relu: 1024 ReLu -> 512 ReLu -> 256 ReLu.*

Innovation

At first, I considered the problem of cold start for users when we don't know the relationship between the user and the videos. I tried using KNN with Pearson Correlation Coefficient to find the top k users that are closest to the current user and feed those users' features to the network. After playing with the idea, I didn't get any improvement. I'm guessing it's because the model is getting the user feature (sex, demographic, preferred genres, etc...), and he has already overcome this issue. Another Idea I had was training an Autoencoder on the users' features. Maybe the model will learn some new connections. I left this idea for the same reason as before. In the end, I decided to read the paper: **Beyond Parity: Fairness Objectives for Collaborative Filtering**. Then, I got the idea of using the fairness objective to help the model when we have less representation of one class in our dataset and prevent bias in the model .

In the paper, they describe that biased data can lead CF methods to make unfair predictions for minority groups. Because recommender systems make predictions based on observed data, they can easily inherit bias that may already exist. They talked about four new unfairness metrics:

1. *Value unfairness*: measures inconsistency in signed estimation error across the user types. Value unfairness occurs when one class of users is consistently given higher or lower predictions than their true preferences.
2. *Absolute unfairness*: measures inconsistency in absolute estimation error across user types. If one user type has a small reconstruction error and the other user type has a large reconstruction error, one type of user has the unfair advantage of a good recommendation.
3. *underestimation unfairness:* measures inconsistency in how much the predictions underestimate the true ratings. Underestimation of unfairness is important in settings where missing recommendations are more critical than extra recommendations.
4. *overestimation unfairness:* measures inconsistency in how much the predictions overestimate the true ratings. Overestimation unfairness may be important in settings where users may be overwhelmed by recommendations.

These fairness metrics can be optimized by adding fairness terms to the learning objective, each adding unfairness penalties as regularizers. Each has a straightforward subgradient and can be optimized by various subgradient optimization techniques.

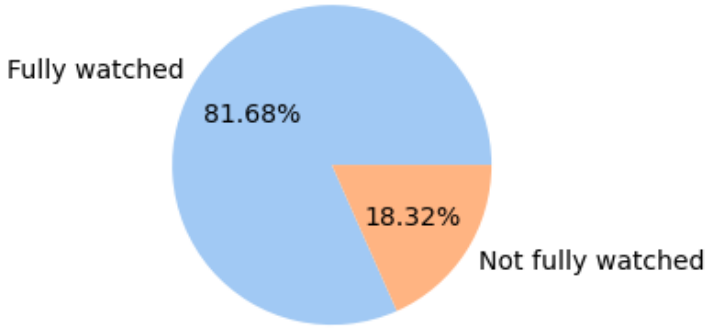

Figure 3: Shows that 81.68% of our data watched the movies till the end, and 18.32% didn't watch them till the end.

Due to time limitations, I didn't implement the metrics in the paper nor used some sophisticated metrics. Instead, I added a simple penalty to the Binary Cross Entropy loss. We can see in figure 3 that most of our examples did watch the whole movie, and I don't want the model to be biased in this direction. I could have easily added more data examples to overcome this issue, but I wanted to show what we can do if we can’t create more data.

The ranking model with fairness objective

In the paper, they describe the goal of the ranking model. The goal is to predict the expected watch time given training examples. In Movielens, we don't know the user's watch time for the movie, so I decided to use a binary value. 0 if the user rates the movie as less than 3 and 1 otherwise. This will indicate if a user watched the whole movie or not. To achieve that, I used the Sigmoid activation function at the last layer to get a number between 0 and 1 and used Binary Cross Entropy as the loss. Now that we have our objective, we can add our fairness metric and help the model overcome the biased data. As described before, I used a simple penalty when the model predicted wrong the underrepresented class .

In the paper, they talked about offline accuracy, and I tried to improve the model accuracy when predicting the underrepresented data. We'll see that choosing a good fairness objective is important to balance the predictions and prevent making bias in the other direction.

Baseline algorithm

As a baseline algorithm, I choose Matrix Factorization, implemented as a simple NN with two embedding layers, one for the users and one for the items. The algorithm tries to learn underlying patterns or latent factors that can explain the observed data. The idea is to represent the original matrix as a product of these smaller embedding vectors.

Dataset

The dataset used in the original paper is not public, so I used Movielens 100k and preprocessed the data to try to match their dataset. It contains 100,000 ratings (on a scale from 1 to 5) given by 943 users and 1682 movies (Having 93.7% sparsity), along with demographic information about the users (age, gender, occupation) and genre information about the movies. After the preprocessing, I split the data into three parts: train/validation/test

The preprocess I did to get as close as possible to the original dataset:

1. I took all the categorical features and converted each one into one hot vector.
2. I normalized the continuous values.
3. I created an example age for each movie.
4. I created a history watch for each user based on the rating timestamp.
5. I created positive/negative examples.

user_id	item_id	ratings	genres_categorical_vector	positives	negatives	next_id	next	next_label	example_age	user_features
160	[405, 832, 864, 288, 109, 589, 1019, 461, 474,...	[1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[5]	[405, 288, 589, 1019, 461, 474, 719, 11, 55, 1...	[832, 864, 109]	153	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	1	[0.13732731986221103]	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
399	[238, 204, 176, 746, 174, 1401, 64, 399, 172, ...	[0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, ...	[5, 14]	[204, 176, 746, 174, 1401, 64, 399, 172, 320, ...	[238, 710, 195, 55]	66	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	1	[0.04879434639014169]	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
896	[275, 237, 168, 28, 39, 478, 420, 200, 69, 136...	[1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[8]	[275, 237, 168, 478, 420, 200, 69, 136, 705, 2...	[28, 39, 274, 468]	1045	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	1	[0.07670726943428388]	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
56	[559, 144, 568, 720, 578, 550, 797, 62, 161, 8...	[1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, ...	[5]	[559, 144, 568, 720, 578, 550, 797, 62, 161, 2...	[849, 154, 715, 235, 122]	158	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	1	[0.13093504740935402]	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
889	[833, 763, 252, 473, 818, 411, 405, 455, 1079,...	[1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, ...	[1, 6, 14]	[833, 763, 252, 473, 818, 455, 235, 3, 1152, 5...	[411, 405, 1079, 819, 1014, 687]	92	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...	1	[0.07383074683049824]	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...

Figure 4: The training dataset after preprocessing. We can see we have item id (history), the rating for each movie in history list, current movie genres, positive/negative examples, example age, and user features.

Hyperparameter optimization and training

I've tried many permutations for the following hyperparameters, and most of them got around the same results:

1. Learning rate: tried [0.1, 0.01, 0.001, etc…]
2. Epochs: tried [10, 20, 100, 200]
3. regularization: tried [0.1, 0.01, 0.001]
4. Optimizer: used Adam optimizer
5. Hidden layers: 1024 ReLu -> 512 ReLu -> 256 ReLu

Evaluation

In the original paper, they talked about offline accuracy, but they didn't mention a specific number. After training each model, here are the results when predicting if a user will not fully watch a video:

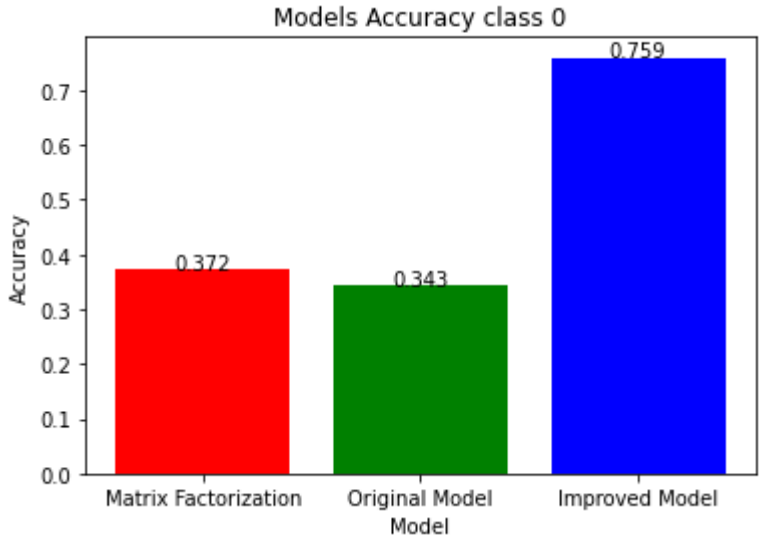

Figure 5: MF got 0.372 acc, the original model 0.343 acc and the improved model got 0.759.

We can see in figure 5 that the fairness objective helped the model overcome the bias problem we had in the dataset. It improved the accuracy by 221 percent. If I used more sophisticated fairness objectives, as they mentioned in the Beyond Parity paper, I could get better results. One drawback I saw when using a simple fairness objective was that the total accuracy of the model dropped, as we see in figure 6. It's crucial to keep in mind the trade-offs involved, as too simple or complex fairness objectives may lead to a drop in overall model accuracy. Therefore, it's essential to carefully evaluate and select the right fairness objective based on the specific problem at hand to achieve optimal performance and fairness.

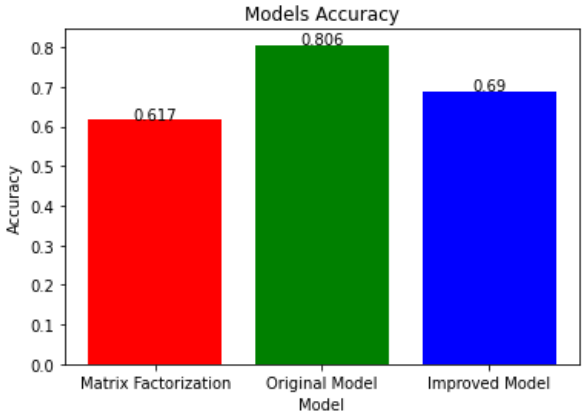

Figure 6: We can see that the overall accuracy of the model is worst than the original model by 16 percent.

Conclusions

This project aimed to test how we can take a DNN model on a biased dataset and improve the accuracy for the underrepresented data by using the fairness objective. Due to a time limitation, I used a simple objective. It was a good example of the trade-off using fairness and the importance of choosing a good objective .

We first described a deep neural network architecture for recommending YouTube videos, split into two distinct problems: candidate generation and ranking. We saw how we handled the problem with millions of items and ranking only dozens. After that, we introduce the fairness objective and how we use it to help the model overcome biased datasets. We showed the baseline algorithm (MF) and described the preprocess that I did. In the end, we saw the result and improved accuracy for the biased class.

References

- “Deep Neural Networks for YouTube Recommendations” (2016)
- “Beyond Parity: Fairness Objectives for Collaborative Filtering” (2017)