Recommender Systems with DeepFM and FinalMLP

Wei Hang Peh, Yew Chian Kng

***Abstract*—DeepFM and FinalMLP are two click-through rate prediction models that make use of embeddings and deep neural networks. DeepFM leverages factorization machines and a deep neural network to learn both high and low order relationships between features. FinalMLP implements a feature selection gate in a two-stream multi-layer perceptron to better train its weights. In this paper, we alter the models for regression to investigate their effectiveness in prediction of ratings on the Google Restaurants Dataset.**

***Keywords*—Deep neural networks, recommender systems, factorization machines**

I. Introduction

Recommender systems assist in making choices when there is insufficient personal information about the alternatives [1]. In a basic recommender system such as collaborative filtering, users that are similar are assumed to have similar interests in products [2]. The model learns the embedding representations of users and products through matrix multiplication and backpropagation.

Both DeepFM and FinalMLP are click-through rate prediction models that predict a final binary output, which may not be suitable for example in the context of restaurants where customer satisfaction is more important. The concepts used in both models are useful in capturing relationships between features and are definitely useful in rating prediction.

The models DeepFM and FinalMLP both build upon the concepts of matrix factorization in collaborative filtering, as well as other concepts mentioned below.

Matrix Factorization

Matrix factorization is a common approach in collaborative filtering. Each user and item in such an approach is represented by embeddings. These embeddings are trained such that the dot product of the embeddings of a user-item pair would give an accurate representation of a measure of interest within that pair.

The embeddings can be a measure of similarity between users and also between items. Items that are rated in a similar way by a user would have similar embeddings [3].

On top of using ratings or other forms of explicit feedback to infer embeddings, matrix factorization methods can utilize implicit feedback, such as number of mouse clicks, to infer the user's preferences relative to each item.

Adding Neural Networks

A common challenge in basic matrix factorization approaches is the inability to include the biases of users and products into their recommendations. For example, such models are unable to account for natural product pairings such as masks and hand sanitizers. By adding a neural network to collaborative filtering, the model achieves better recommendation performance. The use of a non-linear approach in the neural network improves the model's ability to learn latent feature interactions [4].

Factorization Machines

Another drawback with the basic matrix factorization approach lies in its use of only user and item embeddings. There exists other information such as demographic information that would improve the models ability to train more accurate and meaningful embeddings.

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j \qquad (1)$$

Factorization machines also take in user and item indicators in a one-hot encoded form as an input. On top of performing the dot-product between the user and item indicators to find embeddings, features such as user race or product price influence the embeddings with their respective weights [5].

II. DeepFM

DeepFM is a model with architecture to model both high and low-level feature interactions. In order to predict click-through rates (CTR), the model uses embeddings of user and item data to calculate a probability of a user clicking on a specific item.

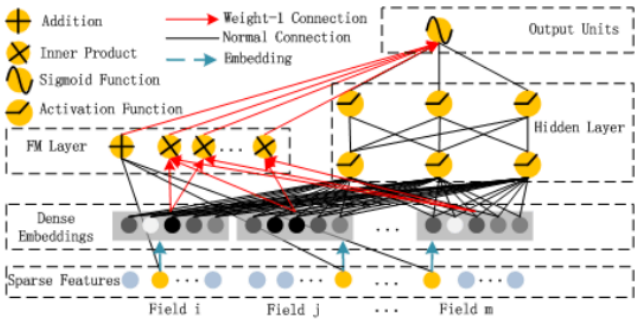

Fig. 1. DeepFM architecture

"Wide and Deep"

DeepFM's architecture supports its goal of modeling high and low order relationships between features accurately.

The "wide" part refers to the Factorization Machine (FM) component, or *FM component* of the model. As mentioned in the previous section, FM allows the model to learn low-order feature interactions well by taking user or item features as an input to learn embeddings.

The "deep" part refers to the *deep component* of the model. It takes in the same inputs as the wide part in the form of embeddings, and utilizes a feed-forward neural network to learn higher-order relationships between features [6].

Components share embeddings

Both the *FM component* and the *deep component* share the same input vectors in the form of dense embeddings. Again, this helps the model learn the high and low order interactions between features. Contrasting to the other recommender models of that time, this meant that there was no need for manual feature-selection, which often required some extra knowledge in the topic of the recommender system to improve the model.

III. FinalMLP

FinalMLP is another model designed to predict CTR. The premise is that, Multi-layer perceptrons (MLP), are very powerful tools in machine learning, but work inefficiently with feature-interactions that are multiplicative such as the dot product [7]. Normally, these feature-interactions are better modeled with approaches such as factorization machines, but they lack the expressiveness of MLP in modeling high-order feature-interactions well. [8].

Although there are models such as DeepFM that cover both weaknesses by implementing both an MLP and some approach to track low-order feature interactions, FinalMLP attempts to tackle the problem in a simpler manner.

Two-stream MLP model

By using two MLP in parallel, each network can learn feature interactions from the same inputs, but learn patterns in different ways. In the ideal scenario, one network could learn lower-order feature interactions (in place of an approach like factorization machines), and the other could learn higher-level feature interactions which feed-forward neural networks excel at.

To encourage even more differentiation between the two networks, FinalMLP adopts the following approaches.

Feature Gating

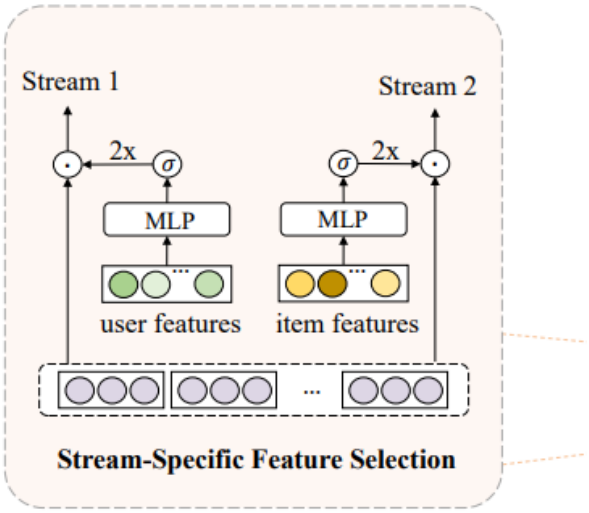

Fig. 2. Feature Gating

By training weights on the features before they are fed into each of the two networks, the models are able to increase the impact from important features and lessen the impact of less important features in predicting CTR. From Fig. 2, each stream receives a different representation of the inputs, encouraging each stream to learn different types of feature interactions.

Different Parameters for each stream

By altering parameters such as number of hidden layers, each stream is more likely to learn different order feature interactions.

Stream-Level Fusion

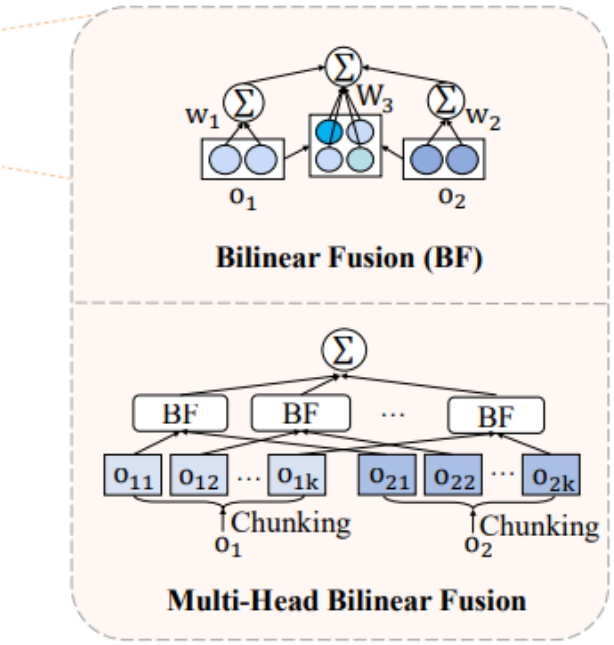

Fig. 3. Multi-head bilinear fusion

The outputs of each of the two networks are combined in a linear function, and possibly an activation function. This allows the model to better understand the importance of each stream in predicting CTR [8].

FinalMLP also learns the embeddings of its users, items and features. We note that the study seems to be focusing on how well this model performs despite its simplicity. In its evaluation, it is seen to perform better than DeepFM on the MovieLens Dataset.

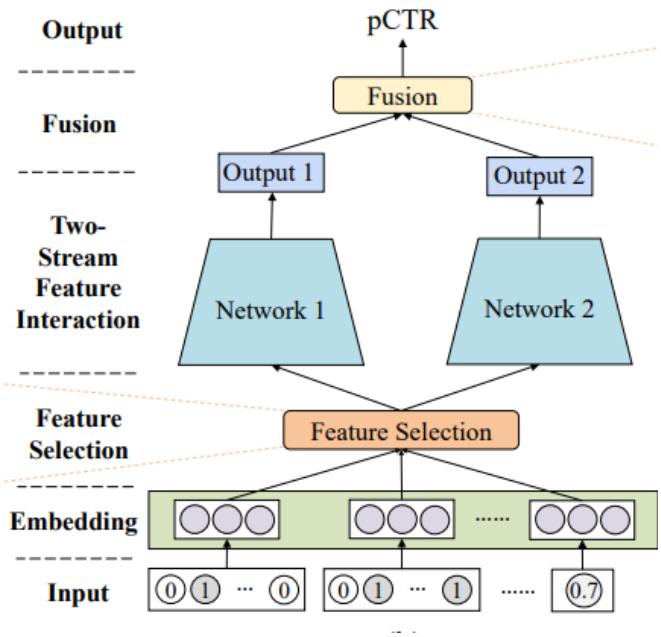

Fig. 4. FinalMLP architecture

IV. Experiment

We aim to compare the effectiveness of each model, DeepFM, and FinalMLP, on the task of predicting restaurant ratings, based on the Google Restaurants Dataset.

Dataset

Each row Google Restaurants Dataset contains the following features:

1. User ID
2. Restaurant ID

Each row contains a rating of either 1.0, 2.0, 3.0, 4.0 or 5.0 stars.

Data Preparation

In our experiment, the ratings have been scaled from to 0 to 1.0.

Note that in the original ratings, a rating of 0 is not possible.

All users and restaurants with less than 7 ratings were omitted..

Approach

Both models were altered for regression. This included adding a final hidden layer of size 1 at the end of both models changing the loss function.

One random rating from each user is used as the test set.

Both models were trained with the parameters below:

TABLE I. Model Parameters

Epochs	*Batch Size*	*Optimizer*	*Loss*
100	8	Adam, lr = 1e-4	RMSE

V. Results

TABLE II. Best losses from each model

Model	*Test Loss (RMSE)*
DeepFM	**0.1656**
FinalMLP	0.1949

TABLE III. DeepFM results

Deep Layers	*Embedding Size*	*Dropout*	*Test Loss (RMSE)*
[32, 32]	10	[0.5, 0.5]	0.1812
[32, 32]	5	[0.5, 0.5]	0.1659
[32, 32, 32]	10	[0.5, 0.5, 0.5]	0.1764
[32, 32, 32]	5	[0.5, 0.5, 0.5]	**0.1656**

TABLE IV. FinalMLP results

Feature Selection Layers	*MLP1 Layers*	*MLP2 Layers*	*Embedding Size*	*Dropout*	*Test Loss (RMSE)*
800	[400]	[100]	10	[0.4]	0.2024

Feature Selection Layers	*MLP1 Layers*	*MLP2 Layers*	*Embedding Size*	*Dropout*	*Test Loss (RMSE)*
800	[400]	[100]	5	[0.4]	0.2024
800	[400, 200]	[100, 50]	10	[0.4, 0.4]	0.1972
800	[400, 200]	[100, 50]	5	[0.4, 0.4]	**0.1949**

Firstly, DeepFM was able to predict restaurant ratings better than FinalMLP. The specific use of the *fm component* in DeepFM might have been stronger at learning the lower-level feature interactions. Although the two-streams in FinalMLP ideally can learn both high and low order feature interactions, it might have struggled due to having sub-optimal parameters to promote differentiation between the two networks.

The value of 0.1656 (Test loss in RMSE) can be understood as, the model was on average 0.1656 / 0.25 = 0.6624 stars away from the actual rating on the test set. Unfortunately, this dataset has not been used for regression problems in other papers, and hence a good benchmark for the model's performance is hard to obtain.

DeepFM Hyper-Parameters

The DeepFM model performed best with an embedding dimension of 5, and with 3 deep layers, as compared to 2 deep layers in its deep component.

Theoretically, a larger embedding size provides more space for the model to learn its lower-order interactions. It is possible that the model needs to train for more epochs to capitalize on a larger embedding dimension.

Having more deep layers in the deep component of DeepFM seems to have improved its test loss, possibly because it is better able to model higher-level feature interactions with a more complex neural network.

FinalMLP Hyper-Parameters

In this experiment, we attempted to have different MLP1 and MLP2 hidden layer sizes to promote each network to differentiate and learn different orders of feature interactions. The embedding sizes were also varied.

However, there did not seem to be any large differences to the test losses of FinalMLP with the different hyper-parameters.

VI. Conclusion

It seems that both models performed well enough when altered for regression, although it seems that FinalMLP is unable to capture the low-order feature interactions between the users and restaurants. In this experiment, the hyper-parameters did not seem to impact the model performance as much as expected with both models not arriving at a significantly better solution after increasing the sizes of the embeddings or the MLP in the models.

References

[1] P. Resnick and H. R. Varian, "Recommender systems," Communications of the ACM, vol. 40, no. 3, pp. 56–58, Mar. 1997, doi: 10.1145/245108.245121.

[2] J. Bobadilla, F. Ortega, A. Hernando, and J. L. Alcalá, "Improving collaborative filtering recommender system results and performance using genetic algorithms," Knowledge Based Systems, vol. 24, no. 8, pp. 1310–1316, Dec. 2011, doi: 10.1016/j.knosys.2011.06.005.

[3] "Matrix Factorization Techniques for Recommender Systems," IEEE Journals & Magazine | IEEE Xplore, Aug. 01, 2009. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5197422

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, Neural Collaborative Filtering. 2017. doi: 10.1145/3038912.3052569.

[5] "Factorization Machines," IEEE Conference Publication | IEEE Xplore, Dec. 01, 2010. https://ieeexplore.ieee.org/abstract/document/5694074?casa_token=zu7bZvWLAs4AAAAA:jjwpHNEoLcxOZ2V-k9ReK5NBEdB7YzEvObq7U03pQsP8xg4mOHJGzvegwjUEBSJs1EA1VOmZokIXtQ

[6] H. Guo, "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction," arXiv.org, Mar. 13, 2017. https://arxiv.org/abs/1703.04247

[7] R. Wang, "Deep & Cross Network for Ad Click Predictions," arXiv.org, Aug. 17, 2017. https://arxiv.org/abs/1708.05123

[8] K. Mao, "FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction," arXiv.org, Apr. 03, 2023. https://arxiv.org/abs/2304.00902