CS420 Course Project Report
ESENFFM Model for CTR Prediction
Lower Variance without Regularization Bias

Li Jiasen
ACM Honor Class, Shanghai Jiao Tong University
lijiasen0921@sjtu.edu.cn

Abstract

This paper introduces a new method to predict the click through rate of the advertisement in order to get a higher AUC. My approach is called Early-Stopped Ensemble of Field-aware Neural Factorization Machine (ESENFFM). And I compare it with the XGBoost after data pre-processing and the NFFM model. It works much better than XGBoost and has lower variance and lower bias than NFFM does.

1. Introduction

Click through rate accuracy is very important nowadays. Click through rate is the most important part in the advertisement system. Click through rate is highly related to the profit of an advertisement system, because advertisements are often paid by clicks. It means that every click of an advertisement, but not every sight of the advertisement, is important to be calculated.

$$Total\ profit \approx CTR \times BID \times AD\ shown\ times$$

With click through rate predicted, a company can also push the advertisement to those who are more likely to click through. [WNZhang 2019]

$$Total\ Profit \approx \int Max_{ad}\left(CTR_{ad,user} \times BID_{ad,user}\right) \cdot d(user)d(times)$$

Not only is the CTR prediction important for the company, CTR prediction also makes it more convenient for customers to get what they need.

Nowadays smart devices are getting smaller and smaller, from computer to personal computer to smart phone to smart watch. In such a small screen, it is very inconvenient for the clients to search keywords to get what they need. The fewer steps there are to get the resource, the more likely they are going to like the application.

Here, in this project, our task is to develop a click through rate model that can predict click through rate.

2. Form of Data

The data is already provided. The form of data is a sparse matrix.

Feature id		Category	Feature id		Category	Feature id		Category
from	to		from	to		from	to	
0	101448	User id	634573	637707	Site id	642098	642105	F4
101449	101455	Day of the week	637708	637731	Site type	642106	642112	F5
101456	101479	Hour of the day	637732	637738	Slot position	642113	642538	F6
101480	101484	Device type	637739	641740	Item id	642539	642547	F7
101485	107409	Device model	641741	641768	F0	642548	642713	F8
107410	631081	Ip	641769	642020	F1	642714	642717	F9
631082	634568	Domain	642021	642030	F2	642718	642777	F10
634569	634572	Connection type	642031	642097	F3	642778	645194	F11

There are many features in the matrix.

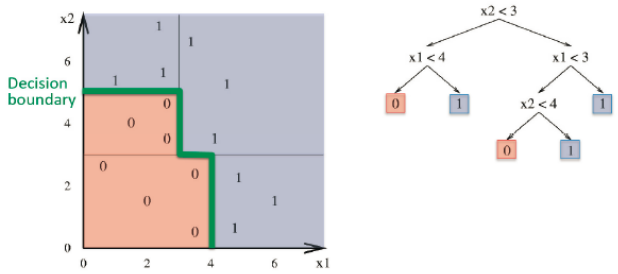
All the features are presented in a discrete way.

3. Pre-Processing

User id and IP address are methods to distinguish the users. However, it is not necessary to keep all of them. IP may hint whether he is at work or at home. It is the User ID that really distinguishes the users. And the IP can then be projected into a low dimension, meaning the status of a user.

4. First Approach: XGBoost

This problem is seemed very likely to be solved by a tree model. Embedding the items and users using KNN, the xgboost model may recognize that some kinds of users may likely choose some kinds of items under some conditions. [T Chen *et al* 2016]



However, there is no need to have the embedding continuous, so it can just be a one-hot low-dimension vector, built mainly by human intelligence clustering.

We can define distance from one user to another as the sine average function.

$$s^{\text{cosine}}(u_a, u_b) \coloneqq \frac{u_a^T u_b}{\|u_a\|_F \cdot \|u_b\|_F}$$

$$s^{sine}(u_a, u_b) \coloneqq s^{cosine}(u_a - mean(u_a), u_b - mean(u_b))$$

And we can project the users to a linear space such that the distances of the users are the given sine function. Here u can store the items that a user clicks. And the same can be done to the items. After the projection, clustering is much easier. [Blum, Hopcroft and Kannan 2015]

However it cannot work very well, because other features cannot be projected and clustered easily. And those high-dimension one-hot vectors may ruin the xgboost.

In this part xgboost has AUC 0.63133 at first, and it is certainly a bad case.

5. Second Approach: NFFM

NFFM or we call it Field-aware Neural Factorization Machine, is a deep CTR model.

It is a click-through prediction model which obtains the advantages of FFM in second order feature interaction and improves NFM's ability on higher order feature interaction. [Z.Li 2019]

This model can be described as the following definition: see[Google 2016]

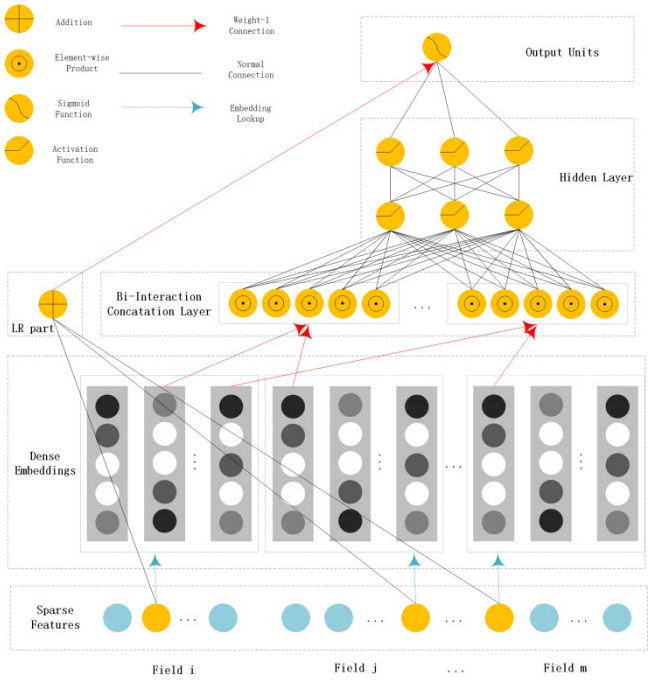
$$\hat{y}_{NFFM}(x) \coloneqq w_0 + \sum_1^n w_i x_i + DNN\big(f_{BI}(V_x)\big)$$

Here, $w_0 + \sum_1^n w_i x_i$ is a regression part of the model. The regression part is important because it is very stable and can pull back some very risky predictions.

x_i is the i^{th} feature. Because the features are discrete, x_i is a one hot vector and $w_i x_i$ is an inner product of w_i and x_i.

We can rewrite the formula into the following format:

$$\hat{y}_{NFFM}(x) = w \cdot x + DNN(f_{BI}(V_x))$$
$$w \coloneqq (w0; w1; \dots; wn) \quad x \coloneqq (1; x1; \dots; xn)$$



5.1. Embedding Layer

As we all know there is a traditional embedding method for each x_i:

$$e_i \coloneqq V_i x_i$$

The V_i here is a vector of shape $D_i \times K_i$. K_i is the length of x_i and D_i is the target dimension of the embedded vector.

We can have the traditional way used for n times, and have the D_i independent on the index i.

$$V \coloneqq (0; V_1; V_2; \dots; V_K)_{D \times (K+1)}$$

Here, $V_i \in \mathbb{R}^D$ is selected if the number of x in the corresponding dimension is one. After the step, output e can be shaped $D \times (N+1)$. And we write the columns of V which are in set x as $V_x \in \mathbb{R}^{D \times (N+1)}$. [Google 2016]

5.2. Bi-Interaction Connection Layer

The columns of V should contact with each other to form a connection, such that some conditions can form a new relationship.

For example, both conditions "Sunday" and "12:00" may lead to prediction that the user is going out for lunch, and restaurant Ads are likely to be clicked.

NFFM uses the Bi-Interaction Connection Layer to handle these kinds of situations by pairwise element product and then the external direct sum of the element products.

$$a_{i,j} \coloneqq (V_x)_i \odot (V_x)_j$$

$$f_{BI}(V_x) \coloneqq \oplus_{1 \le i \le j \le N} a_{i,j}$$

Here, $\odot$ is the element-to-element product. $\oplus$ is the external direct sum.

There will be $\binom{N+1}{2}$ elements of $a_{i,j}$, totally $\binom{N+1}{2} \times D$ dimensions in $f_{BI}(V_x)$.

Because $i \neq j$ is not required, this can still have the direct information from V_x, and this may work better than traditional NFFM.

5.3. Multi-Layer Perceptron (MLP)

MLP can be used to extract high-ordered features and for prediction, with the embedded BI-Interaction Connected data as its input.

With MLP, it is easy to figure out the co-relationship from one feature to another, to give more accurate prediction. Each layer except the last one in MLP is using activation function RELU as usual. [ZHZhou 2016]

After the last layer of the Multi-Layer Perceptron, there is an average of linear regression and the MLP result.

Here, we can use three layers of Multi-Layer Perceptron.

5.4. Prediction Generating

A soft-max layer is added after the last layer of Multi-Layer Perceptron, in order to complete the probability prediction. After the soft-max layer a positive real number less than one is generated. [HLi 2012]

To be convenient we write the result of the probability or the predicted click through rate as

$p(x) \coloneqq softmax(\hat{y}_{NFFM})$.

5.4.1. Loss Function

The well-known loss function, cross-entropy with Logits, is used here to evaluate the loss.

$$L_{x,y} = -[y \cdot ln(p(x)) + (1-y) \cdot ln(1-p(x))]$$

This is the loss function for a certain data point. We can get the global loss function by averaging the result for the data points in the data set.

$$L = mean_{(x,y)\in S}(L_{x,y})$$

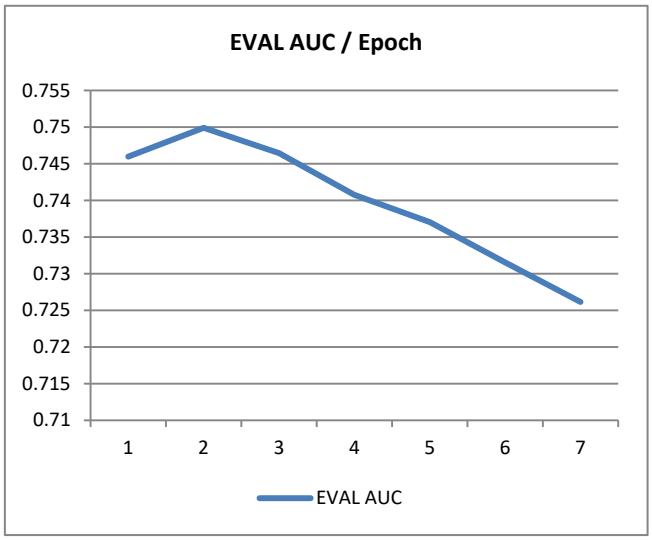
S is the training or evaluation data set. x is the input, describing the features, and y is the prediction output, describing the predicted click through rate.

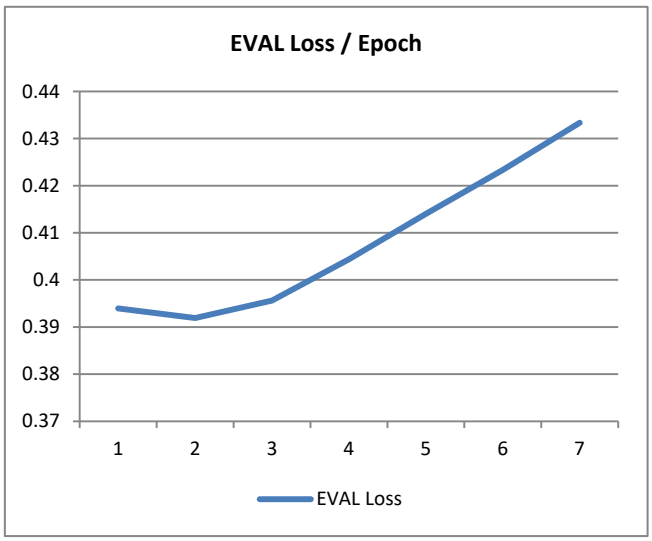
5.4.2. Optimizer

ADAM Optimizer is used here, with learning rate 0.001

5.5. Experiments and Results

The traditional NFFM does not work very well. It over-fits seriously after the second epoch.





Dealing with this, a simple idea is early stopping. Using this strategy, I do not have to do the tradeoff between bias caused by regularization and over-fitting.

Dropout can also be applied here to get better performance.

However, even with early stopping and dropout [Hinton *et.al* 2014] in each layer, it can only get to an AUC about 0.7532, which is not acceptable in this project. So, I am going to give a better solution based on this.

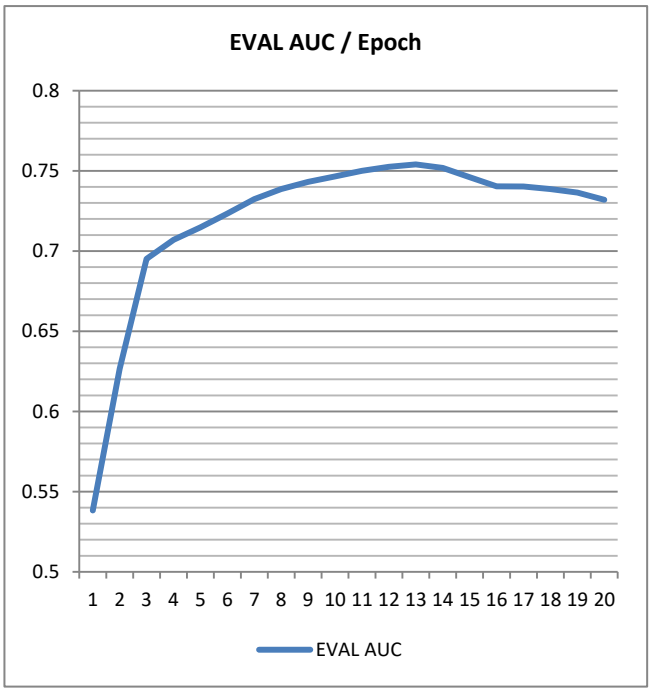
6. The Last Approach: ESENFFM

This approach is call the Early-Stopped Ensemble of Field-aware Neural Factorization Machine.

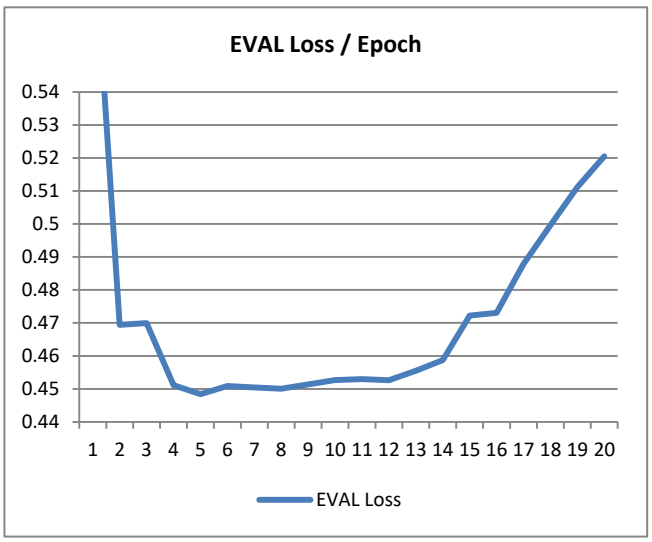
As we know, the Early Stopping still retains a huge amount of variance, and this is not good for prediction. However, if we add regularization, there will be a bias caused by the

penalty of the regularization, and waiting for the point-wise convergence costs more epochs when regularization is added.

What I do first is to get the early-stopping results of the NFFM.



Here, I change the learning rate and the batch size in order to select the best early-stopping point more precisely.



I collected eight outputs of the NFFM. And make a projection.

$$Prediction(x) \coloneqq softmax\left(mean_{1\le i\le n}(\hat{y}_{NFFM_i}(x))\right)$$

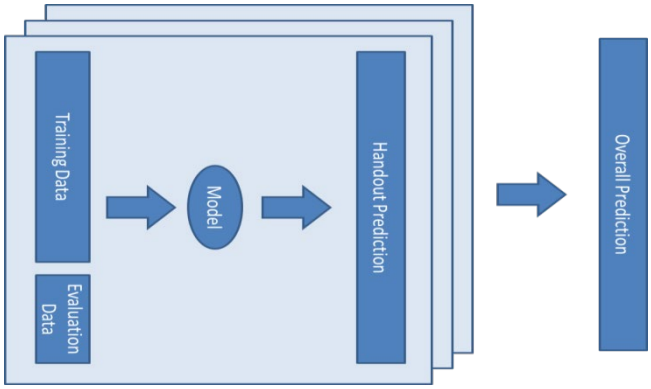
n is the ensemble size

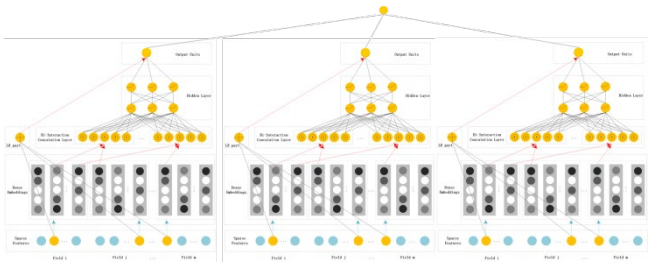
According to the Law of Large Number, [Blum, Hopcroft and Kannan 2015]

$$Prob\left(\left|\frac{x_1+x_2+\cdots+x_n}{n}-E(x)\right|\ge\epsilon\right)\le\frac{Var(x)}{n\epsilon^2}$$

For each point, for a fixed error that is acceptable before soft-max, the probability that its error is too much falls quickly. n is the ensemble size.

$E(x)$ does not mean the Ground Truth. It is just an expectation of estimation of the Ground Truth given by the early-stopping model. This method to decrease the variance of the model does not take in additional bias, such as the regularization bias.





Here we have the result.

Model	Method	AUC
Tree Model	XGBoost	0.63133
NFFM		0.73749
NFFM	Early-Stopping & Dropout	0.75253
ESENFFM		0.76065
ESENFFM	Precise Early-Stopping Point	0.76822

And it doesn't need to run until the point-wise convergence, which saves a lot of epochs. It can go to a good point in three epochs if the learning rate is not adjusted.

7. Conclusion

Training a recommendation system, low bias and low variance are both important. Our model ESENFFM, Early-Stopped Ensemble Field-aware Neural Factorization Machine, can have lower bias by not taking in the regularization bias, and can have lower variance by ensemble. We present the ESENFFM to combine the both strength. And it trains faster than some regularization model. Kaggle off-line experiments show that it can work better than the previous models at least in this case.

Reference

[Google 2016] Google. *Wide & Deep Learning for Recommender Systems.* 2016.

[Blum, Hopcroft and Kannan 2015] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science.* Cornel 2015

[ZHZhou 2016] ZHZhou. *Machine Learning.* Tsinghua University Press 2016.

[WNZhang 2019] WNZhang. *2019 CS420 Machine Learning Lecture.* SJTU 2019

[Hinton *et.al* 2014] N Srivastava, G Hinton, A Krizhevsky. *Dropout: a simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research. 2014.

[HLi 2012] HLi. *Statistical Learning Methods.* Tsinghua University Press 2012.

[Z.Li 2019] Li Zhang, Weichen Shen, Shijian Li, Gang Pan. *Field-aware Neural Factorization Machine for Click-Through Rate Prediction.* Zhejiang University. 2019.

[T Chen *et al* 2016] T Chen, C Guestrin *Xgboost: A scalable tree boosting system* KDD '16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016.