CTR PREDICTION SYSTEM BASED ON WIDE & DEEP LEARNING (COMBINED WITH GBDT)

XUANJUN CHEN, ZHENG WEI

National Taiwan University of Science and Technology
Department of computing science and Information Engineering
Taipei, Taiwan
E-mail:achenxuanjun@gmail.com

ABSTRACT

Click-through rate (CTR) prediction is an essential task in in industrial applications, such online advertising. Recently deep learning based models have been proposed, which can strengthen the generalization ability of the model. Wide & Deep Learning[1] by replacing the transformation function with complex MLP network, which enhances the model capability greatly. However, it is hard for the Wide component to deal with high dimensions and chose the most important features in the wide component training. To solve this problem, we use Gradient Boosting Decision Trees Algorithms (GBDT), which result is used to chose the key features for the wide component training. Offline experiment results show that our approach significantly increased the capabilities of Wide & Deep model in small scale dataset. Besides, it can save a lot time for the jointly training and keep the benefits of memorization and generalization.

Index Terms— CTR Prediction System, GBDT, Wide & Deep Learning

1. INTRODUCTION

In this paper, I implemented the **LDG model** on CTR prediction task, which is combined **Wide & Deep** learning model with **GBDT** . Moreover, I present the experiment result between the Wide & Deep model and our approach. There are three parts, dataset processing, model training and evaluation. This paper is organized as follows. In section 2, I introduce some related work. In section 3, I explain the whole model. In section 4, we have a discussion on experiment result. In section 5, I have a few suggestions for future improvements. Finally, in section 6, I propose my conclusion of this project. We have also open-sourced our implementation in GitHub at VictorChan1,

2. RELATED WORK

2.1. Decisions trees plus logistic regression

The hybrid model[2], which combines decision trees with logistic regression, help us to pick out the key features in dataset. In prediction system, the most important thing is to have the right features. Once we have the right features and the right model, other factors play small roles. However, this hybrid model also has only memorization.

2.2. Wide Deep learning

Wide & Deep learning[1] framework can achieve both memorization and generalization in one model, by jointly training a linear model component and a neural network component. This kind of CTR prediction model reduces the manual feature engineering jobs greatly. Our base model follows this kind of of model structure.

3. PREDICTION SYSTEM STRUCTURE

In this section we present a hybrid model structure: the combination of Deep Neural Network and the concatenation of boosted decision trees and of a probabilistic sparse linear classifier, illustrated in Fig.1..

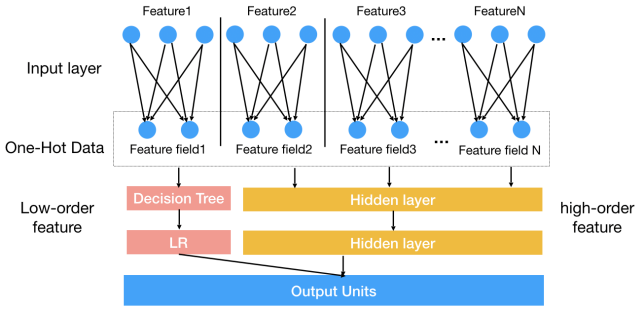

Fig. 1. Hybrid model structure, which is combined Wide Deep learning model with decisions trees.

3.1. The Wide Component

The wide component, consists mainly of a generalized linear model and decisions trees, as illustrated in Fig.1.(left). Let us denote the generalized linear model as $y = W^T X + b$. y is the prediction, $X = [x_1, x_2, x_3, x_4..., x_d]$ is a vector of d features, $W = [w_1, w_2, w_3..., w_d]$ are the parameters and b is the bias. Before training, we need to implement non-linear and tuple transformations by boosted decision trees, which are a powerful way. The GBDT algorithm can be seen as an addition model consisting of K trees, it can denote as:

$$\hat{y}_i = \sum_{K=1}^{K} f_k(x_i), f_k \in F$$

3.2. The Deep Component

The deep component is a feed-forward neural network, as shown in Fig.1. (right). Firstly, the original inputs are need to convert into a low-dimensional and dense real-valued vector. Each hidden layer performs the following computation:

$$a^{l+1} = f(W^{(l)} a^{(l)} + + b^{(l)}$$

where l is the layer number and f is the activation function, often rectified linear units (ReLUs). However, we adopted softsign in practice, which performs better in our project. We can denote Softsign functions as:

$$y = \frac{x}{1 + |x|}$$

$a^{(l)}$,$b^{(l)}$, and $W^{(l)}$ are the activations, bias, and model weights at l-th layer.

3.3. Joint Training of Wide Deep Model

In this subsection, the wide component and deep component are combined using a weighted sum of their output log odds as the prediction, which is then fed to one common logistic loss function for joint training.

4. EXPERIMENT

4.1. Dataset and Experimental Setup

4.1.1. Feature Engineering

Avazu Dataset contains click-through data, ordered chronolsogicasly.Non-clicks and clicks are subsampled according to different strategies. Because the original features contain insufficient information, we will generate more meaningful features of the heart based on the original features, so that the neural network behind us can better learn the click rate.After basic feature engineering, features include click, id, C1, app_category, app_domain, app_id, banner_pos, device_conn_type, device_id, device_ip, device_model, device_type, hour, site_category, site_domain, site_id and C14-C21.

4.1.2. Experimental Setup

For the models, we use SGD as the optimizer with momentum, in which both learning rate and momentum are set to 0.2. The mini-batch size is set to be 20. In order to obtain better training performance, I adopted early stopping to avoid over-fitting neural networks.We take accuracy as our evaluation metirc. Finally, more detail of the **LDG model** are shown in Fig.2.

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	(None, 929)	0	
dense_2 (Dense)	(None, 20)	18600	input_2[0][0]
batch_normalization_1 (BatchNor	(None, 20)	80	dense_2[0][0]
activation_1 (Activation)	(None, 20)	0	batch_normalization_1[0][0]
dense_3 (Dense)	(None, 20)	420	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 20)	80	dense_3[0][0]
activation_2 (Activation)	(None, 20)	0	batch_normalization_2[0][0]
dense_4 (Dense)	(None, 20)	420	activation_2[0][0]
batch_normalization_3 (BatchNor	(None, 20)	80	dense_4[0][0]
input_1 (InputLayer)	(None, 519)	0	
activation_3 (Activation)	(None, 20)	0	batch_normalization_3[0][0]
dense_1 (Dense)	(None, 1)	520	input_1[0][0]
dense_5 (Dense)	(None, 1)	21	activation_3[0][0]
add_1 (Add)	(None, 1)	0	dense_1[0][0] dense_5[0][0]
activation_4 (Activation)	(None, 1)	0	add_1[0][0]

Total params: 20,221
Trainable params: 20,101
Non-trainable params: 120

Fig. 2. Hybrid model detail

4.2. Performance

In this subsection, I compared the accuracy between **LDG model** with different max depth of decision tree and the Wide & Deep model. The maximum decision tree depth to compare is in the range of 1 to 9. At the same time, I also compared validation accuracy between different models with different training data sizes. The data sizes to be compared are 80k, 70k, 60k, 50k, 40k, 30k, 20k, 10k. Finally, I recorded the time spent in each epoch with different data size during the training of the two models.

4.2.1. Training Time

Because I set early stopping to get a better training model. So the number of training epochs for the two models will be different. But the mini-batch size of each model training for each epoch is the same, so we can compare the efficiency of training between the two models by comparing their training

time for each epoch. From Fig.5., we can see that although the training time of both models increases as the data size increases, LGD model takes relatively less time to train than wide & deep model.

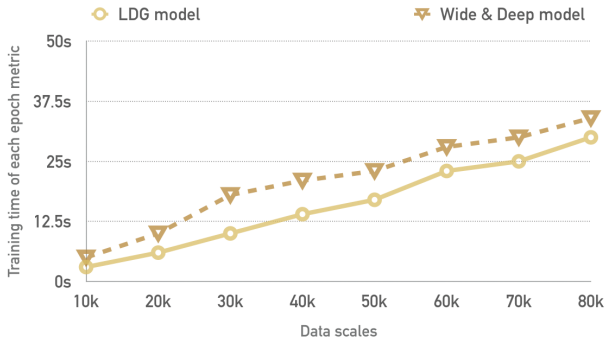

Fig. 3. Comparison of training time

4.2.2. *Effect of Data Size*

From Fig.3., we can find that although the training data scales increases, the training accuracy of the LDG model has always been better than that of the wide & deep model. Especially when the data size is 10k, LGD model accuracy is close to 2% higher than wide & deep model accuracy. I tried to implement lager data scales with LDG model, such 500k, 1000k and so on. However, it doesn't work. Because when the amount of data is too large, the computational complexity of the GBDT algorithm will greatly increase.

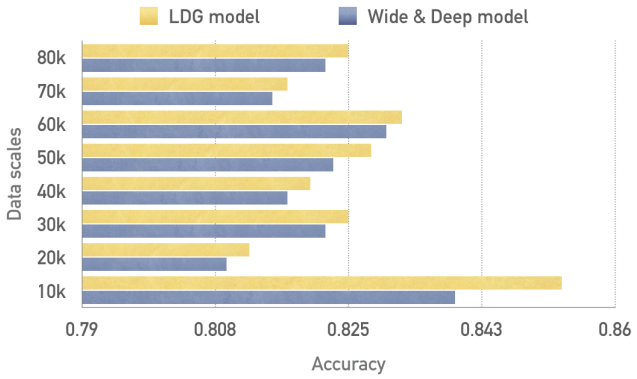

Fig. 4. Comparison of different data sizes

4.2.3. *Effect of Decision Tree Maximum Depth*

In this subsection, it's going to compare decision trees with different maximum depths, which will affect the LDG model accuracy. As shown in Fig.5., when the maximum depth is greater than 4, it performs better than when it is less than 4. But it only affect the accuracy of LDG model slightly.

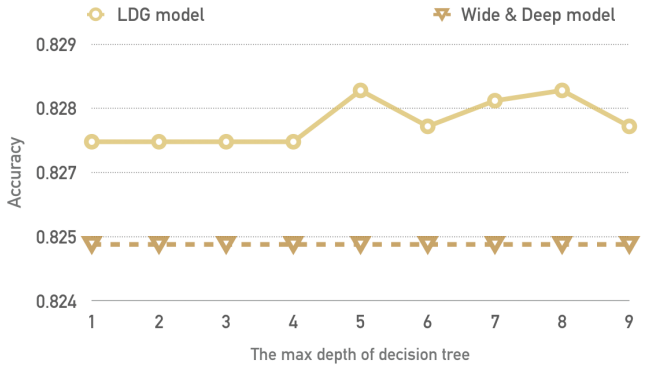

Fig. 5. Comparison of decision trees with different maximum depths

5. FURTHER IMPROVEMENT

In CTR prediction model, time series features are critical part of model training. During the training, time series features will show a regular trend of some things happening in big data, which contributes to dig more high-value information. Due to algorithm and hardware limitations, we only trained a small data set in the experiment. This situation causes the time series features to be treated as category features, which greatly affects the model training effect.

6. CONCLUSION

According to our experiment, we find that Gradient Boosting decision tree definitely improves the capabilities of Wide Deep model by reduce the input dimensions of wide linear model. Summary, the main contributions of this paper are as follow:

(1) LDG model performs better than Wide Deep learning model in small-scale dataset.

(2) LDG model spend less time than Wide Deep learning model.

(3) The maximum depth of the decision tree will only affect the accuracy of LDG model slightly.

7. REFERENCES

[1] Cheng H. et al., "Wide deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016.

[2] Xinran He. et al., "Practical lessons from predicting clicks on ads at facebook," in *International Workshop on Data Mining for Online Advertising (ADKDD)*. Facebook, 2014.