DeepFM: An End-to-End Wide Deep Learning Framework for CTR Prediction

1 Citation

Wang, Qianqian, Shuning Xing, and Xiaohui Zhao. ”Research on CTR prediction based on stacked autoencoder.” Applied Intelligence: 1-12.

`https://arxiv.org/pdf/1804.04950.pdf`

2 Abstract

Our DeepFM model combines factorization machines with neural nets to make it easy to learn lower and higher order feature interactions. Like Google’s Wide and Deep model, we have a wide part and deep (either PNN or DNN) part. We A/B test our model in Huawei’s App Market.

3 Introduction

Click Through Rate (CTR) is the probability that a user will click on an ad. In a recommendation system, we aim to maximize the number of clicks or the total CTR $\times$ bid.

Features in your model interact. For example, people order food apps during meal time (2nd order interaction between app category and time of day). Teenage boys like shooter games (3rd order interaction between app category, age, and gender).

Traditional methods like generalized linear models require you to manually add generalization terms. Factorization machines can model feature interactions with dot products, but they typically only consider pairwise interactions. People have tried CNNs and RNNs for this problem, but they aren’t great. Wide and Deep models are the best approach so far.

Our DeepFM learns both low and high order feature interactions with little feature engineering. Unlike Wide and Deep, it’s end-to-end trainable. Also, you can plug in whatever you want for the deep part. We use either PNN or DNN to as our deep part, calling the resulting models DeepFM-P and DeepFM-D, respectively.

4 Related Work

Collaborative filtering (CF) is a popular approach. We assume users with similar past behaviors will like similar items. Memory based methods directly learn (user, item) similarity scores, so they struggle when data is sparse. Model based methods use a model (typically matrix factorization) to predict the similarity.

Content based models use content features about the user and item to compute a similarity score. They are useful for mitigating the cold start problem. Hybrid methods combine content based models and CF models. They struggle to scale though, so we do not use them for CTR prediction.

Linear models were also used for CTR prediction, and factorization machines improve them by incorporating feature interaction.

We can split deep learning models into CTR based and rating based models. We use a CTR based model here. A rating based model is either hybrid of CF based. They aim to predict a rating that a user will give an item. Hybrid models are more sophisticated (e.g. using recurrence to take into account ordered user history).

5 Our Approach

Our training consists of (x, y) pairs where x represents the features about the user, item, and context and where $y \in \{0, 1\}$ indicates whether the user clicked on the item. Features can be continuous or categorical (in which case we assume they are one-hot encoded). We aim to predict the probablity that the user will click on the item (i.e. CTR).

For each feature i, it's order 1 importance is w_i. We learn its order 2 interactions with the wide (i.e. FM) part. The higher order interactions come from the deep part. The prediction is:

$$\hat{y} = \text{sigmoid}(y_{FM}(x) + y_{Deep}(x)) \tag{1}$$

The FM model computes (1) the weighted sum of the features and (2) all pairwise inner products of the embeddings of each feature. Notice that we need to convert categorical features into embeddings with an embedding matrix or they will be too high dimensional to deal with. Unlike Wide and Deep, we don't pretrain the FM. Instead, it's part of the overall model and gets end-to-end trained. It shares its categorical feature embedding matrices with the deep part.

$$y_{FM}(x) = w \cdot x + \sum_{i=1}^{d} \sum_{j=i+1}^{d} (V_i \cdot V_j) x_i x_j \tag{2}$$

The deep component uses a feedforward neural net to compute higher order feature interactions. Its input is the embeddings (assume continuous features have a trivial embedding $V_{field_i} = 1$).

$$a^{(0)} = [e_1, ..., e_m] \tag{3}$$

$$e_i = V_{field_i} x_{field_i} \tag{4}$$

Now we need to pick the architecture for the deep part. If we use DNN, we have H hidden layers with activation function σ where:

$$a^{(l+1)} = \sigma(W^{(l)} a^{(l)} + b^{(l)}) \tag{5}$$

$$y_{DNN}(x) = W^{(|H|+1)} a^{(|H|)} + b^{(|H|+1)} \tag{6}$$

If we use PNN (product based neural network), we add a product layer between the embedding layer and first hidden layer. The product layer concatenates all the embedding vectors (called the linear neurons), just like in DNN. It also computes pairwise products (either inner product, outer product, or both) of the embedding vectors (called the quadratic neurons) and feeds both the linear and quadratic neurons into the hidden layer. That is, we have:

$$a^{(1)} = \sigma(W^{(0)}_{linear} z + W^{(0)}_{quadratic} p + b^{(0)}) \tag{7}$$

$$z = a^{(0)} \tag{8}$$

$$p = [g(e_i, e_j) \forall i \in \{1...m\}, j \in \{i+1, ..., m\}] \tag{9}$$

$$g = \text{inner product, outer product, or both} \tag{10}$$

Our objective function is cross entropy: $-y \log \hat{y}(x) - (1 - y) \log (1 - \hat{y}(x))$. We use L2 regularization and dropout.

We train in a distributed environment with multiple GPUs. Data reading happens asynchronously to the model training.

Unlike previous work, we don't require any pretraining or feature engineering.

6 Experiments

We do both offline experiments and online (A/B test) experiments. We use the Criteo recommendation dataset and an internal App Market dataset. AUC and log loss are our evaluation metrics for offline evaluation. They are preferable to precision and recall because we do not need to pick a threshold.

While our DeepFM-D and DeepFM-P models are not the fastest to train, they outperform other traditional techniques and the Wide and Deep model.

We use ReLU activation, dropout between 0.6 to 0.9, 200 to 400 neurons per layer, 3 hidden layers, and a constant network shape (i.e. all hidden layers have the same size).

Our DeepFM models get higher click through and conversion rates during the A/B testing.

We do simulation to find that DeepFM also gives a good level of diversity and personalization in its recommendations.