DeepFM: A Factorization-Machine based Neural Network for CTR Prediction

Jiho Kang
2022.04.30

Abstract

- CTR (Click through rate)

  A prediction to estimate the probability of click rate.

  CTR = Total measured clicks / Total Measured Impressions * 100

- Learning **complex feature interactions** in user behavior is an important key point to maximize CTR.
- In this paper, introduces a model that proceeds with **end-to-end learning without feature engineering** while simultaneously using **low- and high-dimensional feature interactions**.
- DeepFM is a model that **combines factorization-machine (FM) and deep neural network (DNN) into neural network structures**.
- Unlike the Wide & Deep (2016) model, which is a similar idea presented by Google, DeepFM **does not require feature engineering because it shares input values** in the wide part(FM) and the deep part(DNN).

Introduction

- Implicit feature interactions behind the user’s click behavior.
  - People install food delivery apps at mealtimes.

    → Meaning order-2 interactions between 'app category' and 'time-stamp'.
  - Boys like RPG games.

    → Meaning order-3 interactions between ‘gender’, ‘age’and 'app category'.

- Comparison of other models
  - FNN: Factorization-machine supported Neural Network
  - PNN: Product-based Neural Network
  - Wide & Deep

Table 1: Comparison of deep models for CTR prediction

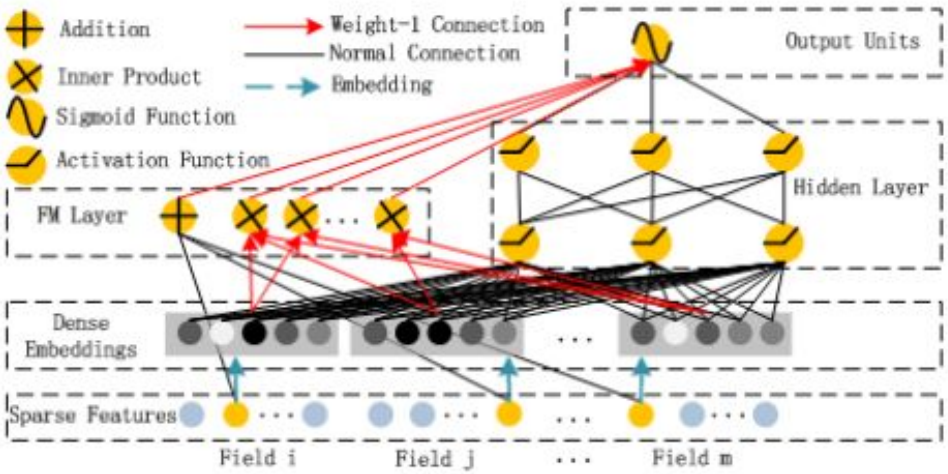
	No Pre-training	High-order Features	Low-order Features	No Feature Engineering
FNN	×	√	×	√
PNN	√	√	×	√
Wide & Deep	√	√	√	×
DeepFM	√	√	√	√

Our Approach

- DeepFM is a neural network model that combines FM structure and neural network structure DNN. This model captures **low-dimensional interactions (in FM / linear / wide part)** and **high-dimensional interactions (in DNN / deep part)**.

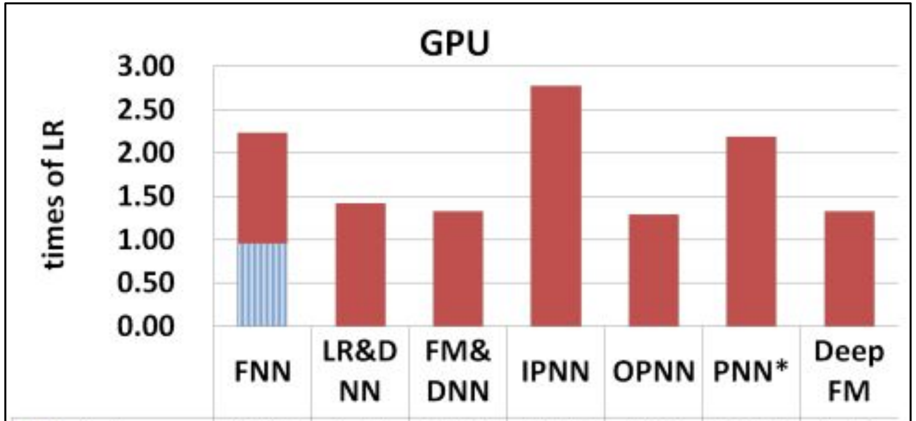
- The Wide & deep model receives a large-sized vector in the wide part for pairwise feature interactions, which greatly increases the complexity. However, deepFM is effectively learned because it receives the **same embedding vector in wide part (FM) and deep part (DNN).**

- DeepFM performs better than the models existing for CTR prediction on benchmark datasets and commercial data.



Experiments

- Efficiency Comparison
  - Pre-training of FNN makes it less efficient.
  - Although the speed up of IPNN and PNN∗ on GPU is higher than the other models, they are still computationally expensive because of the inefficient inner product operations.
  - The DeepFM achieves almost the most efficient in tests.



	FNN	LR&DNN	FM&DNN	IPNN	OPNN	PNN*	DeepFM
training	1.28	1.41	1.32	2.77	1.29	2.19	1.32
pre-training	0.95	0.00	0.00	0.00	0.00	0.00	0.00

- ❖ LR & DNN : Original wide and deep model
- ❖ FM & DNN : Wide and deep model that changes wide part from linear to FM like DeepFM
- ❖ IPNN : inner product
- ❖ OPNN : outer product
- ❖ PNN* : both

Experiments

- Effectiveness Comparison
  - Learning **feature interactions** improves the performance.
    → LR(does not consider feature interactions) performs worse than the other models.
  - Learning high- and low-order feature interactions **simultaneously** and properly improves the performance.
    → DeepFM outperforms FM(low-order feature) and FNN and PNNs (high-order feature)
  - Learning high- and low-order feature interactions simultaneously **while sharing the same input**(feature embedding) improves the performance.
    → DeepFM outperforms LR&DNN and FM&DNN (different input)

Table 2: Performance on CTR prediction.

	Company*		Criteo	
	AUC	LogLoss	AUC	LogLoss
LR	0.8640	0.02648	0.7686	0.47762
FM	0.8678	0.02633	0.7892	0.46077
FNN	0.8683	0.02629	0.7963	0.45738
IPNN	0.8664	0.02637	0.7972	0.45323
OPNN	0.8658	0.02641	0.7982	0.45256
PNN*	0.8672	0.02636	0.7987	0.45214
LR & DNN	0.8673	0.02634	0.7981	0.46772
FM & DNN	0.8661	0.02640	0.7850	0.45382
DeepFM	**0.8715**	**0.02618**	**0.8007**	**0.45083**

Conclusion

- Overcoming the shortcomings of the SOTA model of the existing CTR prediction, it showed better performance.
- Time is reduced because pre-training is not required.
- Learn both high and low order feature interactions.
- There is no need for feature engineering.

References

https://greeksharifa.github.io/machine_learning/2019/12/21/FM/

https://hongl.tistory.com/242

https://jae-eun-ai.tistory.com/21

https://hyunlee103.tistory.com/69

https://cleancode-ws.tistory.com/160