Order-aware Embedding Neural Network for CTR Prediction

Wei Guo*
Wuhan University
weige@whu.edu.cn

Ruiming Tang†
Noah's Ark Lab, Huawei
tangruiming@huawei.com

Huifeng Guo
Noah's Ark Lab, Huawei
huifeng.guo@huawei.com

Jianhua Han
Noah's Ark Lab, Huawei
hanjianhua4@huawei.com

Wen Yang
Wuhan University
yangwen@whu.edu.cn

Yuzhou Zhang
Noah's Ark Lab, Huawei
zhangyuzhou3@huawei.com

ABSTRACT

Product based models, which represent multi-field categorical data as embedding vectors of features, then model feature interactions in terms of vector product of **shared embedding**, have been extensively studied and have become one of the most popular techniques for CTR prediction. However, if the shared embedding is applied: (1) the angles of feature interactions of different orders may conflict with each other, (2) the gradients of feature interactions of high-orders may vanish, which result in learned feature interactions less effective. To solve these problems, we propose a novel technique named **Order-aware Embedding** (i.e., multi-embeddings are learned for each feature, and different embeddings are applied for feature interactions of different orders), which can be applied to various models and generates feature interactions more effectively. We further propose a novel order-aware embedding neural network (**OENN**) based on this embedding technique for CTR prediction. Extensive experiments on three publicly available datasets demonstrate the effectiveness of **Order-aware Embedding** and show that our **OENN** outperforms the state-of-the-art models.

CCS CONCEPTS

• **Information systems** → **Recommender systems**;

KEYWORDS

CTR Prediction; Feature interactions; Order-aware Embedding

ACM Reference Format:
Wei Guo, Ruiming Tang, Huifeng Guo, Jianhua Han, Wen Yang, and Yuzhou Zhang. 2019. Order-aware Embedding Neural Network for CTR Prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331332

*This work was done when Wei Guo was an intern at Noah's Ark Lab, Huawei.
†Corresponding author.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331332

Table 1: An example for explaining the limitation of shared embedding.

smokes	sedentary life	unhealthy diet	life expectancy
0	0	0	10
1	0	0	8
0	1	0	8
0	0	1	8
1	1	0	2
1	0	1	2

1 INTRODUCTION

The prediction of click-through rate (CTR) plays a crucial role in online advertising, which is a multi-billion dollars business nowadays. Factorization-machine supported Neural Network (FNN) [12] pre-trains Factorization-machine (FM) [10] to obtain feature embeddings before applying deep neural networks (DNN) for prediction. Neural Factorization-Machine (NFM) [3] and Product-based Neural Network (PNN) [8] further introduce a bilinear interaction pooling layer and a product layer between the embedding layer and fully-connected layers separately to model the feature interactions, and does not rely on pre-trained FM. It is noticed that FNN, NFM and PNN focus more on high-order feature interactions while ignore low-order feature interactions, which are also essential. Therefore, the Wide & Deep [1] and FM based neural network (DeepFM) [2] models are proposed to overcome this problem by combining a shallow model to learn low-order feature interactions explicitly and a DNN model to learn high-order feature interactions implicitly. However, as the final function learned by DNN can be arbitrary, there is no theoretical guarantee on whether each order of feature interaction is modeled or not. Deep & Cross Network (DCN) [11] solves this problem by explicitly applying feature crossing at each layer. Thus the orders increase at each layer and are determined by layer depth. Inspired by DCN, Compressed Interaction Network (CIN) [6], a more effective model, is proposed to capture the feature interactions of bounded orders. In order to learn feature interactions better, Kernel Product Neural Network (KPNN) [9] and Product-network In Network (PIN) [9] utilize a kernel product and a micro-net architecture respectively to model feature interactions.

Observe that all the existing deep models leverage vector product of shared embedding (i.e., only one embedding is learned for each feature) for producing 2-order or higher-order feature interactions. Despite their success, there are two problems. Here we follow an example from [7], as shown in Table 1. Assuming the

one-hot encoding representations and embedding vectors of features "smokes", "sedentary life" and "unhealthy diet" as x_1,x_2,x_3 and u_1,u_2,u_3, we use the sum of 1-order feature (embedding vector) and 2-order feature interactions (inner product of embedding vector) for prediction. Assume embedding size as 1, the model equation is:

$$y(x) = b_0 + u_1x_1 + u_2x_2 + u_3x_3 + u_1u_2x_1x_2 + u_1u_3x_1x_3 + u_2u_3x_2x_3 \quad (1)$$

Fit the model with the first four samples in Table 1, we can get

$$\{b_0, u_1, u_2, u_3\} = \{10, -2, -2, -2\} \quad (2)$$

The corresponding weights for 2-order feature interactions are:

$$\{u_1u_2, u_1u_3, u_2u_3\} = \{4, 4, 4\} \quad (3)$$

It means that smoking and having a unhealthy diet should have a positive impact on life expectancy, which is obviously ridiculous. If we fit u_1,u_2,u_3 to all the samples, the model will not be satisfied. When the size of embedding is larger than 1, the inner product of some pairs of negative impact embeddings may be non-negative [7]. The conflict between 1-order feature and 2-order feature interactions still exists. When shared embedding is applied for high-order feature interactions generation, due to the continuous multiplication, (1) the angles of feature interactions of different orders may *conflict* with each other, (2) the gradients of feature interactions of high-orders may *vanish*. These problems make it difficult to learn the feature interactions of different orders. However, if we use order-aware embedding for generating feature interactions, where multi-embeddings are learned for each feature, and different embeddings are applied for feature interactions of different orders, as follows

$$y(x) = b_0 + u_1x_1 + u_2x_2 + u_3x_3 + u'_1u'_2x_1x_2 + u'_1u'_3x_1x_3 + u'_2u'_3x_2x_3 \quad (4)$$

so that the corresponding weights for 1-order feature and 2-order feature interactions are:

$$(b_0, u_1, u_2, u_3, u'_1u'_2, u'_1u'_3) = (10, -2, -2, -2, -4, -4) \quad (5)$$

We can easily solve the aforementioned two problems.

The main contributions of this work are summarized as follows: (1) We propose a novel embedding technique: order-aware embedding, in which multi-embeddings are learned and applied for feature interactions of different orders. (2) Based on the order-aware embedding, we propose Order-aware Embedding Neural Network (OENN). (3) We conduct extensive experiments on three publicly available datasets. The results demonstrate the effectiveness of order-aware embedding and the superiority of OENN compared to several state-of-the-art models.

2 OUR APPROACH

From a bottom-up perspective, OENN consists of three components: order-aware embedding component, feature representation component and deep network component, as shown in Figure 1. In the following sections, we will present these components in detail.

2.1 Order-aware Embedding Component

A feature has multi-embeddings in OENN, each of which is applied for a specific order of feature interactions. The embedding vectors of field i for different orders can be formulated as:

$$\mathbf{e}_i^1 = \mathbf{W}_i^{0,1}\mathbf{x}[start_i : end_i] \quad (6)$$

$$\cdots$$

$$\mathbf{e}_i^j = \mathbf{W}_i^{0,j}\mathbf{x}[start_i : end_i] \quad (7)$$

where $\mathbf{e}_i^j$ is the embedding vector of field i for generating j-order feature interactions ($1 \le j \le O$, O is the highest order of feature interactions to be generated). $\mathbf{x}$ is the input one-hot vector containing m different fields, $\mathbf{x}[start_i : end_i]$ represents the one-hot vector for field i. $\mathbf{W}^0$ represents the parameters of the embedding layer. $\mathbf{W}_i^{0,j}$ represents the parameters used for learning the $\mathbf{e}_i^j$. $\mathbf{W}_i^{0,j} \in R^{D\times(end_i - start_i+1)}$ is fully connected with $\mathbf{x}[start_i : end_i]$ and $\mathbf{e}_i^j$, D is the dimension of embedding vectors.

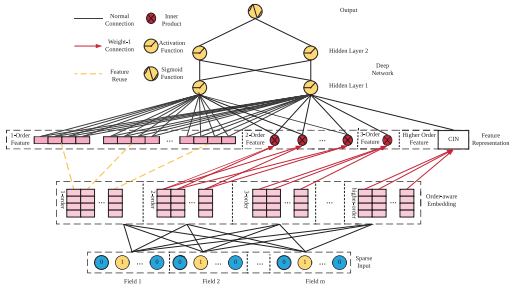


Figure 1: The architecture of OENN.

2.2 Feature Representation Component

The feature representation component in OENN is to model the feature interactions of different orders in an explicit fashion. We use embedding vectors as 1-order features directly:

$$\mathbf{f}^1 = \mathbf{e}^1 = [\mathbf{e}_1^1, \mathbf{e}_2^1, \cdots, \mathbf{e}_m^1] \quad (8)$$

Due to its superior performance, we use inner product for generating 2-order feature interactions explicitly:

$$\mathbf{f}^2 = [<\mathbf{e}_1^2, \mathbf{e}_2^2>, <\mathbf{e}_1^2, \mathbf{e}_3^2>, \cdots, <\mathbf{e}_{m-1}^2, \mathbf{e}_m^2>] \quad (9)$$

where $<\mathbf{e}_1^2, \mathbf{e}_2^2> = \sum_{i=1}^{D}\prod_{j=1}^{2}\mathbf{e}_j^2[i]$ denotes the inner product of embedding vector $\mathbf{e}_1^2$ and $\mathbf{e}_2^2$. $\mathbf{e}_j^2[i]$ is the i-th element of the vector $\mathbf{e}_j^2$. For convenience, we define $<\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n> = \sum_{i=1}^{D}\prod_{j=1}^{n}\mathbf{e}_j[i]$, and we abuse *n-order inner product* to represent this vector product operation in the rest of this paper. Then the calculation of 3-order feature interactions can be formulated as:

$$\mathbf{f}^3 = [<\mathbf{e}_1^3, \mathbf{e}_2^3, \mathbf{e}_3^3>, \cdots, <\mathbf{e}_{m-2}^3, \mathbf{e}_{m-1}^3, \mathbf{e}_m^3>] \quad (10)$$

Due to the exhaustive calculation of inter-fields feature interactions, the complexity of n-order inner product grows exponentially with the order of interactions. It becomes very complex and time-consuming for n-order inner product when $n > 3$. CIN solves this problem by compressing the high-order interaction vectors to a fixed value. It learns feature interactions explicitly, and the order of feature interactions grows with the network depth. So we use CIN for generating higher-order feature interactions. For a clearer explanation, we take the generation of 4-order feature interactions as an example. Formulate the embedding vectors as a matrix:

$$\mathbf{X}^{0,4} = [\mathbf{e}_1^4; \mathbf{e}_2^4; ...; \mathbf{e}_m^4] \quad (11)$$

The h-th feature vector of the k-th layer in CIN is:

$$\mathbf{X}_{h,*}^{k,4} = \sum_{i=1}^{H_{k-1}}\sum_{j=1}^{m}\mathbf{W}_{ij}^{k,h,4}(\mathbf{X}_{i,*}^{k-1,4} \circ \mathbf{X}_{j,*}^{0,4}) \quad (12)$$

where $\mathbf{X}_{j,*}^{0,4} = \mathbf{e}_j^4$, $\mathbf{X}^{k,4} \in R^{H_k\times D}$, $1 \le h \le H_k$, H_k denotes the number of feature vectors in the k-th layer, and $H_0 = m$. $\mathbf{W}^{k,h,4} \in R^{H_{k-1}\times m}$ is the parameter matrix for the h-th feature vector. $\circ$ denotes the Hadamard product, where $\mathbf{e}_1 \circ \mathbf{e}_2 = [\mathbf{e}_1[1]\times\mathbf{e}_2[1], \mathbf{e}_1[2]\times\mathbf{e}_2[2], \cdots, \mathbf{e}_1[D]\times\mathbf{e}_2[D]]$. As $\mathbf{X}^{k,4}$ is derived via the Hadamard product of $\mathbf{X}^{k-1,4}$ and $\mathbf{X}^{0,4}$, thus the order of

interactions grows with the layer depth of the CIN. We can get that the 4-order feature interactions are the output vector of the third layer in the CIN, so we use it as the 4-order feature interactions:

$$\mathbf{f}^4 = [\sum_{i=1}^{D} \mathbf{X}_{1,i}^{3,4}, \sum_{i=1}^{D} \mathbf{X}_{2,i}^{3,4}, \cdots, \sum_{i=1}^{D} \mathbf{X}_{H_4,i}^{3,4}] \tag{13}$$

In the same way, we can get the higher-order feature interactions:

$$\mathbf{f}^j = [\sum_{i=1}^{D} \mathbf{X}_{1,i}^{j-1,j}, \sum_{i=1}^{D} \mathbf{X}_{2,i}^{j-1,j}, \cdots, \sum_{i=1}^{D} \mathbf{X}_{H_j,i}^{j-1,j}] \tag{14}$$

where $4 \leq j \leq O$. The output of feature representation component is the concatenation of feature interactions with different orders:

$$\mathbf{F} = [\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3, \cdots, \mathbf{f}^O] \tag{15}$$

2.3 Deep Network Component

The deep network component is a deep neural network, which is used as a classifier and for implicit feature learning. The output of feature representation component is fed into the neural network.

$$\mathbf{a}^{(1)} = \sigma(\mathbf{W}^{(1)}\mathbf{F} + \mathbf{b}^{(1)}) \tag{16}$$

$$\mathbf{a}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}) \tag{17}$$

where l is the layer depth and σ is the activation function. $\mathbf{a}^{(l-1)}$, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the input, model parameters and bias of the l-th layer. The output of OENN is a real number as the predicted CTR:

$$y_{(oenn)} = Sigmoid(\mathbf{W}^{(l+1)}\mathbf{a}^{(l)} + \mathbf{b}^{(l+1)}) \tag{18}$$

3 EXPERIMENTS

We conduct experiments to answer the following questions: **(Q1)** How does our proposed order-aware embedding perform for feature interactions learning? **(Q2)** How does our proposed OENN model perform compared to the state-of-the-art models? **(Q3)** How do the key hyper-parameters of OENN impact its performance?

3.1 Experiment Setup

3.1.1 Datasets. We evaluate our proposed models on three large scale real-world datasets: Criteo[1], Avazu[2] and iPinYou[3]. We use the pre-processed train/validation/test set as [9] exactly.

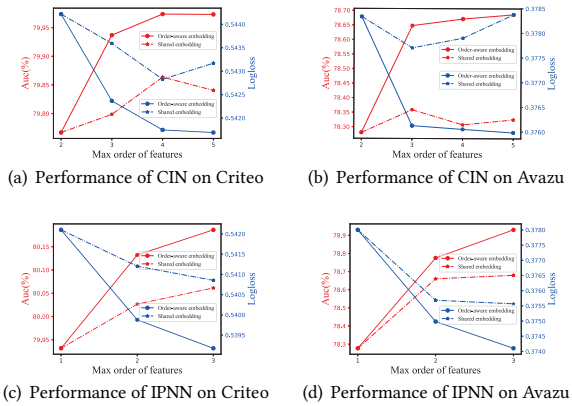
3.1.2 Baseline Methods and Evaluation Metrics. We compare OENN with 7 baseline models, including LR [5], FM, FFM [4], DeepFM, xDeepFM [6], IPNN [9] and PIN.

The evaluation metrics are **AUC** (Area Under ROC), and **LogLoss** (cross entropy).

3.1.3 Parameter Settings. We implement our model using Tensorflow[4], and we will release our source code upon the acceptance of this work. To evaluate the models fairly, we follow the parameter settings in [9] in all the following experiments. Specifically, we use embedding size of 10 for Criteo dataset to reduce memory footprint, which is the same as the parameter settings in [2].

3.2 Study of Order-aware Embedding(Q1)

We choose CIN and IPNN as cases to study the performance of order-aware embedding compared with shared embedding. Figure 2 shows the performance of CIN and IPNN on Criteo and Avazu dataset. It is noticed that the max order of CIN grows with the depth of the network. The max order equals 2 when the depth of CIN is 1. As for IPNN, max order being 1 means that only feature embeddings are used as feature representation, max order being 3 means that 3-order inner product of feature embeddings are also employed for feature representation. We can see that order-aware embedding consistently improves the performance of CIN and IPNN in terms of AUC and Losloss when different orders of feature interactions are applied. The improvement suggests that order-aware embedding based models are more plausible to model feature interactions of different orders than shared embedding.



(a) Performance of CIN on Criteo (b) Performance of CIN on Avazu

(c) Performance of IPNN on Criteo (d) Performance of IPNN on Avazu

Figure 2: Performance comparison of CIN and IPNN with shared embedding and order-aware embedding.

3.3 Performance Comparison(Q2)

Table 2: Overall performance of different models on Criteo, Avazu and iPinYou datasets.

	Criteo		Avazu		iPinYou	
Model	AUC(%)	Logloss	AUC(%)	Logloss	AUC(%)	Logloss
LR	78.02*	0.5631*	76.76	0.3868	76.38	0.005691
FM	78.80*	0.5560*	77.93	0.3805	77.17	0.005595
FFM	79.80	0.5438	78.31	0.3781	76.18	0.005695
DeepFM	79.75*	0.5444*	78.36	0.3777	77.92	0.005588
xDeepFM	80.00*	0.5420*	78.55*	0.3766*	78.04*	0.005555*
IPNN	80.03*	0.5412*	78.68	0.3757	78.17	0.005549
PIN	80.09*	0.5402*	78.72	0.3755	78.22	0.005547
OENN	**80.23**	**0.5388**	**78.99**	**0.3741**	**78.37**	**0.005543**

Results of compared models are reported from [9], except the ones marked by (∗), which are ontained by our re-implemention. Table 2 summarizes the best performance of these models on Criteo, Avazu and iPinYou datasets, and bold numbers are best results among these models. OENN achieves the best performance among all models in three datasets. Specifically, OENN outperforms the best baseline model PIN with a 0.17%, 0.34% and 0.19% improvement in terms of AUC (0.26%, 0.37% and 0.07% improvement in terms of Logloss). This demonstrates the effectiveness of OENN in CTR prediction. In fact, a small increase in offline test may lead to a significant improvement in online test. As presented in [1], 0.275% offline AUC improvement results in 3.9% online CTR improvement.

[1] http://labs.criteo.com/downloads/download-terabyte-click-logs/.
[2] http://www.kaggle.com/c/avazu-ctr-prediction.
[3] http://data.computational-advertising.org.
[4] https://www.tensorflow.org

3.4 Hyper-parameter Investigation(Q3)

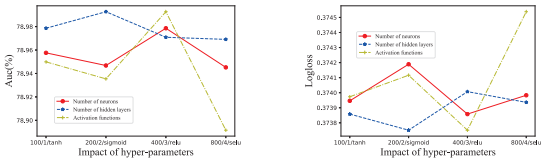
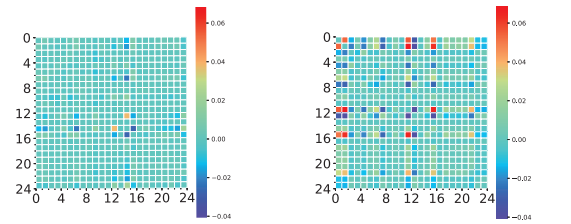
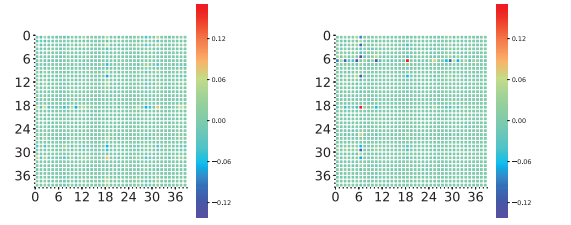


Figure 3: Impact of network hyper-parameters on performance of OENN on Avazu dataset. The x-axis represents different settings of hyper-parameters.

We explore the impact of different hyper-parameters on the performance of OENN based on Avazu dataset in this section, including (1) number of neurons per layer, (2) number of hidden layers, and (3) activation function. As we can see from Figure 3, increasing the number of neurons doesn't always bring benefit. This is because an over-complicated model is easy to overfit. And we can see that the network with 400 neurons per layer achieves the best performance on Avazu dataset. We can observe that the performance of OENN increases with the number of hidden layers at the beginning. However, model performance degrades if the number of hidden layers keeps growing. This is also because of overfitting. And the network with 2 hidden layers achieves the best performance on Avazu dataset. We can also find that relu has the best performance. Besides, selu has worse performance than sigmoid. Possible reasons may be that relu induces sparse activation and efficient gradient propagation, but selu makes computation complicated.



(a) Shared embedding on Avazu (b) Order-aware embedding on Avazu



(c) Shared embedding on Criteo (d) Order-aware embedding on Criteo

Figure 4: Heatmaps of 2-order feature interactions. The x-axis and y-axis both represent fields in Avazu and Criteo.

4 INTERACTIONS VISUALIZATION

In this section, we take CIN on Avazu and Criteo datasets as cases to visualize and analyze the feature interactions. Because the number of features in each field ranges from tens (e.g., country) to millions (e.g., user ID), it is hard to analyze from category level, so we turn to field level. We use mean embedding vectors within a batch of 2000 samples to represent the feature vector of each field, as the embedding vectors within a field are semantically close to each other. Figure 4 presents the heatmaps of 2-order feature interactions with respect to CIN in the cases of shared embedding and order-aware embedding, respectively. Because of rare strong response of feature interactions in practice, as reported in [9], the well-learned heatmap of feature interactions should be sparse and more isolated bright red (strong positive response) or deep blue (strong negative response) points are expected as a bright red or deep blue point means that this feature interaction is likely to provide more information. It can be seen from Figure 4, 2-order feature interactions with order-aware embedding has more bright red and deep blue points, which is more beneficial for prediction. Besides, as there are more fields in Criteo dataset, higher-order feature interactions may be more useful than 2-order feature interactions. So the heatmap of 2-order feature interactions in Criteo dataset are more sparse than in Avazu dataset. Nevertheless, there are still more bright red or deep blue points with order-aware embedding than shared embedding in Criteo dataset.

5 CONCLUSION

In this paper, we propose OENN, an order-aware embedding neural network to solve the issues of feature interactions generation with shared embedding (i.e., angles confliction and gradients vanishing). With expressive order-aware embedding, OENN outperforms the state-of-the-art models on three real-world datasets. There are two directions for future study. One is exploring a unified method for generating feature interactions with different orders. The other is to develop a distributed version of OENN to make it suitable for large-scale problems.

REFERENCES

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proc. of the 1st Workshop on DLRS*. ACM, 7–10.
[2] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proc. of the 26th IJCAI*. 1725–1731.
[3] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.
[4] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proc. of the 10th RecSys*. ACM, 43–50.
[5] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past erformance data. In *Proc. of the 18th SIGKDD*. ACM, 768–776.
[6] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proc. of the 24nd SIGKDD*. 1754–1763.
[7] Sebastian Prillo. 2017. An Elementary View on Factorization Machines. In *Proc. of the 10th RecSys*. ACM, 179–183.
[8] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2017. Product-Based Neural Networks for User Response Prediction. In *IEEE International Conference on Data Mining*. 1149–1154.
[9] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 5.
[10] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.
[11] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proc. of the ADKDD'17*. ACM, 12.
[12] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.