Representation Learning-Assisted Click-Through Rate Prediction

Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Chao Qi, Zhaojie Liu and **Yanlong Du**
Alibaba Group
{maiwei.oywt, xiuwu.zxw, shukui.rsk, qichao.qc, zhaojie.lzj, yanlong.dyl}@alibaba-inc.com

Abstract

Click-through rate (CTR) prediction is a critical task in online advertising systems. Most existing methods mainly model the feature-CTR relationship and suffer from the data sparsity issue. In this paper, we propose DeepMCP, which models other types of relationships in order to learn more informative and statistically reliable feature representations, and in consequence to improve the performance of CTR prediction. In particular, DeepMCP contains three parts: a matching subnet, a correlation subnet and a prediction subnet. These subnets model the user-ad, ad-ad and feature-CTR relationship respectively. When these subnets are jointly optimized under the supervision of the target labels, the learned feature representations have both good prediction powers and good representation abilities. Experiments on two large-scale datasets demonstrate that DeepMCP outperforms several state-of-the-art models for CTR prediction.

1 Introduction

Click-through rate (CTR) prediction is to predict the probability that a user will click on an item. It plays an important role in online advertising systems. For example, the ad ranking strategy generally depends on CTR $\times$ bid, where bid is the benefit the system receives if an ad is clicked by a user. Moreover, according to the common cost-per-click charging model, advertisers are only charged once their ads are clicked by users. Therefore, in order to maximize the revenue and to maintain a desirable user experience, it is crucial to estimate the CTR of ads accurately.

CTR prediction has attracted lots of attention from both academia and industry [He *et al.*, 2014; Shan *et al.*, 2016; Guo *et al.*, 2017]. For example, the Logistic Regression (LR) model [Richardson *et al.*, 2007] considers linear feature importance and models the predicted CTR as $\hat{y} = \sigma(w_0 + \sum_i w_i x_i)$, where $\sigma(\cdot)$ is the sigmoid function, x_i is the ith feature and w_0, w_i are model weights. The Factorization Machine (FM) [Rendle, 2010] is proposed to further model pairwise feature interactions. It models the predicted CTR as $\hat{y} = \sigma(w_0 + \sum_i w_i x_i + \sum_i \sum_j \mathbf{v}_i^T \mathbf{v}_j x_i x_j)$, where $\mathbf{v}_i$ is the latent embedding vector of the ith feature. In recent years, Deep Neural Networks (DNNs) [LeCun *et al.*, 2015] are exploited for CTR prediction and item recommendation in order to automatically learn feature representations and high-order feature interactions [Van den Oord *et al.*, 2013; Zhang *et al.*, 2016b; Qu *et al.*, 2016; Covington *et al.*, 2016]. To take advantage of both shallow and deep models, hybrid models are also proposed. For example, Wide&Deep [Cheng *et al.*, 2016] combines LR and DNN, in order to improve both the memorization and generalization abilities of the model. DeepFM [Guo *et al.*, 2017] combines FM and DNN, which further improves the model ability of learning feature interactions. Neural Factorization Machine [He and Chua, 2017] combines the linearity of FM and the non-linearity of DNN.

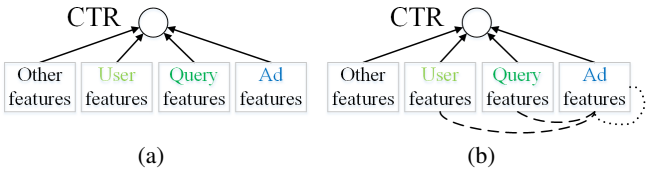


Figure 1: (a) Classical CTR prediction methods model the feature-CTR relationship. (b) DeepMCP further models feature-feature relationships, such as the user-ad relationship (dashed curve) and the ad-ad relationship (dotted curve).

Nevertheless, these models only consider the **feature-CTR** relationship. In contrast, the DeepMCP model proposed in this paper additionally considers **feature-feature** relationships, such as the **user-ad** relationship and the **ad-ad** relationship. We illustrate their difference in Figure 1. Note that the feature interaction in FM still models the feature-CTR relationship. It can be considered as two_features-CTR, because it models how the feature interaction $\mathbf{v}_i^T \mathbf{v}_j x_i x_j$ relates to the CTR $\hat{y}$, but does not model whether the two feature representations $\mathbf{v}_i$ and $\mathbf{v}_j$ should be similar to each other.

In particular, our proposed DeepMCP model contains three parts: a matching subnet, a correlation subnet and a prediction subnet. They share the same embedding matrix. The matching subnet models the user-ad relationship (i.e., whether an ad matches a user's interest) and aims to learn useful user and ad representations. The correlation subnet models the ad-ad relationship (i.e., which ads are within a time window in a user's click sequence) and aims to learn useful ad representations. The prediction subnet models the feature-CTR relationship and aims to predict the CTR given all the fea-

Label	User ID	User Age	Ad Title
1	2135147	24	Beijing flower delivery
0	3467291	31	Nike shoes, sporting shoes
0	1739086	45	Female clothing and jeans

Table 1: Each row is an instance for CTR prediction. The first column is the label (1 - clicked, 0 - unclicked). Each of the other columns is a field. The instantiation of a field is a feature.

tures. When these subnets are jointly optimized under the supervision of the target labels, the feature representations are learned in such a way that they have both good prediction powers and good representation abilities. Moreover, as the same feature appears in different subnets in different ways, the learned representations are more statistically reliable.

In summary, the main contributions of this paper are

- We propose a new model DeepMCP for CTR prediction. Unlike classical CTR prediction models that mainly consider the feature-CTR relationship, DeepMCP further considers user-ad and ad-ad relationships.
- We conduct extensive experiments on two large-scale datasets to compare the performance of DeepMCP with several state-of-the-art models. We make the implementation code of DeepMCP publicly available[1].

2 Deep Matching, Correlation and Prediction (DeepMCP) Model

In this section, we present the DeepMCP model in detail.

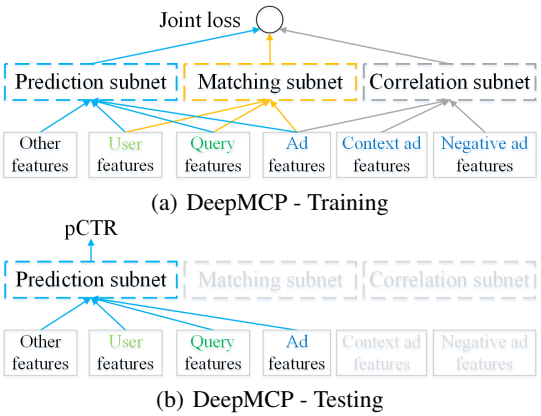
2.1 Model Overview

The task of CTR prediction in online advertising is to estimate the probability of a user clicking on a specific ad. Table 1 shows some example instances. Each instance can be described by multiple *fields* such as user information (user ID, city, etc.) and ad information (creative ID, title, etc.). The instantiation of a field is a *feature*.

Unlike most existing CTR prediction models that mainly consider the feature-CTR relationship, our proposed DeepMCP model additionally considers the user-ad and ad-ad relationships. DeepMCP contains three parts: a matching subnet, a correlation subnet and a prediction subnet (cf. Figure 2(a)). When these subnets are jointly optimized under the supervision of the target labels, the learned feature representations have both good prediction powers and good representation abilities. Another property of DeepMCP is that although all the subnets are active during training, only the prediction subnet is active during testing (cf. Figure 2(b)). This makes the testing phase rather simple and efficient.

We segregate the features into four groups: *user* (e.g., user ID, age), *query* (e.g., query, query category), *ad* (e.g., creative ID, ad title) and *other* features (e.g., hour of day, day of week). Each subnet uses a different set of features. In particular, the prediction subnet uses all the four groups of features, the matching subnet uses the user, query and ad features, and the correlation subnet uses only the ad features. All the subnets share *the same* embedding matrix.

[1]https://github.com/oywtece/deepmcp



(a) DeepMCP - Training

(b) DeepMCP - Testing

Figure 2: Sketch view of the DeepMCP model: (a) Training, (b) Testing. All the subnets are active during training, but only the prediction subnet is active during testing. pCTR is predicted CTR.

2.2 Prediction Subnet

The prediction subnet presented here is a typical DNN model. It models the feature-CTR relationship (where explicit or implicit feature interactions are modeled). It aims to predict the CTR given all the features, supervised by the target labels. Nevertheless, the DeepMCP model is flexible that the prediction subnet can be replaced by any other CTR prediction model, such as Wide&Deep [Cheng *et al.*, 2016] and DeepFM [Guo *et al.*, 2017].

First, a feature $x_i \in \mathbb{R}$ (e.g., a user ID) goes through an embedding layer and is mapped to its embedding vector $\mathbf{e}_i \in \mathbb{R}^K$, where K is the vector dimension and $\mathbf{e}_i$ is to be learned. The collection of all the feature embeddings is an embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times K}$, where N is the number of unique features. For multivalent categorical features such as the bi-grams in the ad title, we first map each bi-gram to an embedding vector and then perform sum pooling to generate the aggregated embedding vector of the ad title.

We then concatenate the embedding vectors from all the features as a long vector $\mathbf{m}$. The vector $\mathbf{m}$ then goes through several fully connected (FC) layers with the ReLU activation function ($\mathrm{ReLU}(x) = \max(0, x)$), in order to exploit high-order nonlinear feature interactions [He and Chua, 2017]. Nair and Hinton [2010] show that ReLU has significant benefits over sigmoid and tanh activation functions in terms of the convergence rate and the quality of obtained results.

Finally, the output $\mathbf{z}$ of the last FC layer goes through a sigmoid function to generate the predicted CTR as

$$\hat{y} = \frac{1}{1 + \exp[-(\mathbf{w}^T \mathbf{z} + b)]},$$

where $\mathbf{w}$ and b are model parameters to be learned. To avoid model overfitting, we apply dropout [Srivastava *et al.*, 2014] after each FC layer. Dropout prevents feature co-adaptation by setting to zero a portion of hidden units during training.

All the model parameters are learned by minimizing the average logistic loss on a training set as

$$\mathrm{loss}_p = -\frac{1}{|\mathbb{Y}|} \sum_{y \in \mathbb{Y}} [y \log \hat{y} + (1 - y) \log(1 - \hat{y})], \quad (1)$$

where $y \in \{0, 1\}$ is the true label of the target ad corresponding to $\hat{y}$ and $\mathbb{Y}$ is the collection of labels.

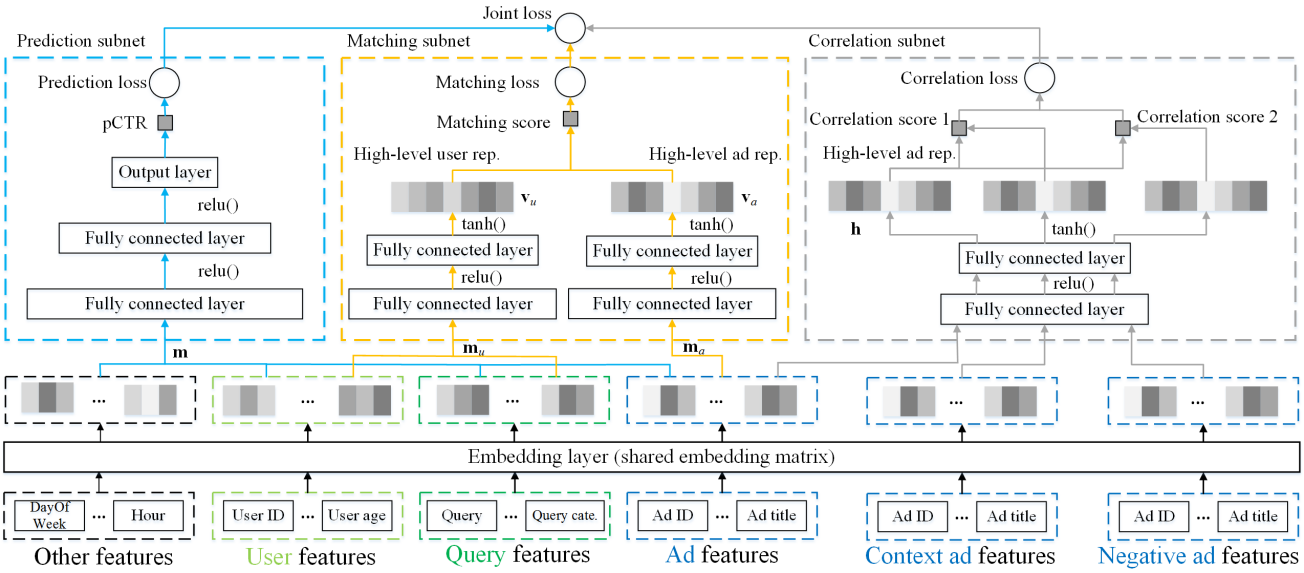


Figure 3: Detailed view of the DeepMCP model. The prediction, matching and correlation subnets share *the same* embedding matrix.

2.3 Matching Subnet

The matching subnet models the user-ad relationship (i.e., whether an ad matches a user's interest) and aims to learn useful user and ad representations. It is inspired by semantic matching models for web search [Huang *et al.*, 2013].

In classical matrix factorization for recommendation [Koren *et al.*, 2009], the rating score is approximated as the inner product of the latent vectors of the user ID and the item ID. In our problem, instead of directly matching the user ID and the ad ID, we perform matching at a *higher level*, incorporating all the features related to the user and the ad. When a user clicks an ad, we assume that the clicked ad is relevant, at least partially, to the user's need (given the query submitted by the user, if any). In consequence, we would like the representation of the user features (and the query features) and the representation of the ad features to match well.

In particular, the matching subnet contains two parts: "user part" and "ad part". The input to the "user part" is the user features (e.g., user ID, age) and query features (e.g., query, query category). As in the prediction subnet, a feature $x_i \in \mathbb{R}$ first goes through an embedding layer and is mapped to its embedding vector $\mathbf{e}_i \in \mathbb{R}^K$. We then concatenate all the feature embeddings as a long vector $\mathbf{m}_u \in \mathbb{R}^{N_u}$ (N_u is the vector dimension). The vector $\mathbf{m}_u$ then goes through several FC layers in order to learn more abstractive, high-level representations. We use $\tanh$ (rather than ReLU) as the activation function of the last FC layer, which is defined as $\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}$. We will explain the reason later. The output of the "user part" is a high-level user representation vector $\mathbf{v}_u \in \mathbb{R}^M$ (M is the vector dimension).

The input to the "ad part" is the ad features (e.g., creative ID, ad title). Similarly, we first map each ad feature to its embedding vector and then concatenate them as a long embedding vector $\mathbf{m}_a \in \mathbb{R}^{N_a}$ (N_a is the vector dimension). The vector $\mathbf{m}_a$ then goes through several FC layers and results in a high-level ad representation vector $\mathbf{v}_a \in \mathbb{R}^M$. Note that, the inputs to the "user" and "ad" parts usually have different sizes, i.e., $N_u \neq N_a$ (because the number of user features and the number of ad features may not necessarily be the same). However, after the matching subnet, $\mathbf{v}_u$ and $\mathbf{v}_a$ have the same size M. In other words, we project two different sets of features into a common low-dimensional space.

We then compute the matching score s as

$$s(\mathbf{v}_u, \mathbf{v}_a) = \frac{1}{1 + \exp(-\mathbf{v}_u^T \mathbf{v}_a)}.$$

We do not use ReLU as the activation function of the last FC layer because the output after ReLU will contain lots of zeros, which makes $\mathbf{v}_u^T \mathbf{v}_a \to 0$. There are at least two choices to model the matching score: point-wise and pair-wise [Liu and others, 2009]. In a point-wise model, we could model $s(\mathbf{v}_u, \mathbf{v}_a) \to 1$ if user u clicks ad a and model $s(\mathbf{v}_u, \mathbf{v}_a) \to 0$ otherwise. In a pair-wise model, we could model $s(\mathbf{v}_u, \mathbf{v}_{a_i}) > s(\mathbf{v}_u, \mathbf{v}_{a_j}) + \delta$ where $\delta > 0$ is a margin, if user u clicks ad a_i but not ad a_j.

We choose the point-wise model because it can directly reuse the training dataset for the prediction subnet. Formally, we minimize the following loss for the matching subnet

$$\begin{aligned} \text{loss}_m = -\frac{1}{|\mathbb{Y}|} \sum_{y \in \mathbb{Y}} \big[& y(u,a) \log s(\mathbf{v}_u, \mathbf{v}_a) \\ & + (1 - y(u,a)) \log(1 - s(\mathbf{v}_u, \mathbf{v}_a)) \big], \end{aligned} \tag{2}$$

where $y(u,a) = 1$ if user u clicks ad a and it is 0 otherwise.

2.4 Correlation Subnet

The correlation subnet models the ad-ad relationship (i.e., which ads are within a time window in a user's click sequence) and aims to learn useful ad representations. The skip-gram model is proposed in [Mikolov *et al.*, 2013] to learn useful representations of words in a sequence, where words within a context window have certain correlation. It has been widely applied in many tasks to learn useful low-dimensional representations [Zhao *et al.*, 2018; Zhou *et al.*, 2018].

In our problem, we apply the skip-gram model to learn useful ad representations, because the clicked ads of a user also

form a sequence with certain correlation over time. Formally, given a sequence of ads $\{a_1, a_2, \cdots, a_L\}$ clicked by a user, we would like to maximize the average log likelihood as

$$ll = \frac{1}{L}\sum_{i=1}^{L}\sum_{-C\le j\le C}^{1\le i+j\le L, j\neq 0} \log p(a_{i+j}|a_i),$$

where L is the number of ads in the sequence and C is a context window size.

The probability $p(a_{i+j}|a_i)$ can be defined in different ways such as softmax, hierarchical softmax and negative sampling [Mikolov *et al.*, 2013]. We choose the negative sampling technique due to its efficiency. $p(a_{i+j}|a_i)$ is then defined as

$$p(a_{i+j}|a_i) = \sigma(\mathbf{h}_{a_{i+j}}^T \mathbf{h}_{a_i}) \prod_{q=1}^{Q} \sigma(-\mathbf{h}_{a_q}^T \mathbf{h}_{a_i}),$$

where Q is the number of sampled negative ads and $\sigma(\mathbf{h}_{a_{i+j}}^T \mathbf{h}_{a_i}) = \frac{1}{1+\exp(-\mathbf{h}_{a_{i+j}}^T \mathbf{h}_{a_i})}$. $\mathbf{h}_{a_i}$ is a high-level representation vector that involves all the features related to ad a_i and that goes through several FC layers (cf. Figure 3).

The loss function of the correlation subnet is then given by minimizing the negative average log likelihood as

$$\text{loss}_c = \frac{1}{L}\sum_{i=1}^{L}\sum_{-C\le j\le C}^{1\le i+j\le L, j\neq 0} \Big[-\log\Big[\sigma(\mathbf{h}_{a_{i+j}}^T \mathbf{h}_{a_i})\Big] - \sum_{q=1}^{Q} \log\Big[\sigma(-\mathbf{h}_{a_q}^T \mathbf{h}_{a_i})\Big]\Big]. \quad (3)$$

2.5 Offline Training Procedure

The final joint loss function of DeepMCP is given by

$$\text{loss} = \text{loss}_p + \alpha \text{loss}_m + \beta \text{loss}_c, \quad (4)$$

where loss_p is the prediction loss in Eq. (1), loss_m is the matching loss in Eq. (2), loss_c is the correlation loss in Eq. (3), and α and β are tunable hyperparameters for balancing the importance of different subnets.

The DeepMCP model is trained by minimizing the joint loss function on a training dataset. Since our aim is to maximize the CTR prediction performance, we evaluate the model on a separate validation dataset and record the validation AUC (an evaluation metric, which will be explained in §3.4) during the training procedure. The optimal model parameters are obtained at the highest validation AUC.

2.6 Online Procedure

As we have illustrated in Figure 2(b), in the online testing phase, the DeepMCP model only needs to compute the predicted CTR (pCTR). Therefore, only the features from the target ad are needed and only the prediction subnet is active. This makes the online phase of DeepMCP rather simple and efficient.

3 Experiments

In this section, we conduct experiments on two large-scale datasets to evaluate the performance of DeepMCP and several state-of-the-art methods for CTR prediction.

Dataset	# Instances	# Fields	# Features
Avito	11,211,794	27	42,301,586
Company	60,724,276	28	29,997,247

Table 2: Statistics of experimental large-scale datasets.

3.1 Datasets

Table 2 lists the statistics of two large-scale datasets.

1) **Avito advertising dataset**[2]**.** This dataset contains a random sample of ad logs from avito.ru, the largest general classified website in Russia. We use the ad logs from 2015-04-28 to 2015-05-18 for training, those on 2015-05-19 for validation, and those on 2015-05-20 for testing. In CTR prediction, testing is usually the next-day prediction. The test set contains 2.3×10^6 instances. The features used include 1) user features such as user ID, IP ID, user agent and user device, 2) query features such as search query, search category and search parameters, 3) ad features such as ad ID, ad title and ad category, and 4) other features such as hour of day and day of week.

2) **Company advertising dataset.** This dataset contains a random sample of ad impression and click logs from a commercial advertising system in Alibaba. We use ad logs of 30 consecutive days during Aug.-Sep. 2018 for training, logs of the next day for validation, and logs of the day after the next day for testing. The test set contains 1.9×10^6 instances. The features used also include user, query, ad and other features.

3.2 Methods Compared

We compare the following methods for CTR prediction.

1. **LR**. Logistic Regression [Richardson *et al.*, 2007]. It models linear feature importance.
2. **FM**. Factorization Machine [Rendle, 2010]. It models both first-order feature importance and second-order feature interactions.
3. **DNN**. Deep Neural Network. It contains an embedding layer, several fully connected layers and an output layer.
4. **Wide&Deep**. The Wide&Deep model in [Cheng *et al.*, 2016]. It combines LR (wide part) and DNN (deep part).
5. **PNN**. The Product-based Neural Network in [Qu *et al.*, 2016]. It introduces a production layer between the embedding layer and fully connected layers of DNN.
6. **DeepFM**. The DeepFM model in [Guo *et al.*, 2017]. It combines FM (wide part) and DNN (deep part).
7. **DeepCP**. A variant of the DeepMCP model, which contains only the correlation and the prediction subnets. It is equivalent to setting $\alpha = 0$ in Eq. (4).
8. **DeepMP**. A variant of the DeepMCP model, which contains only the matching and the prediction subnets. It is equivalent to setting $\beta = 0$ in Eq. (4).
9. **DeepMCP**. The DeepMCP model (§2) which contains the matching, correlation and prediction subnets.

3.3 Parameter Settings

We set the embedding dimension of each feature as $K = 10$, because the number of distinct features is huge. We set the

[2]https://www.kaggle.com/c/avito-context-ad-clicks/data

	Avito		Company	
Algorithm	AUC	Logloss	AUC	Logloss
LR	0.7556	0.05918	0.7404	0.2404
FM	0.7802	0.06094	0.7557	0.2365
DNN	0.7816	0.05655	0.7579	0.2360
PNN	0.7817	0.05634	0.7593	0.2357
Wide&Deep	0.7817	0.05595	0.7594	0.2355
DeepFM	0.7819	0.05611	0.7592	0.2358
DeepCP	0.7844	0.05546	0.7610	0.2354
DeepMP	0.7917	0.05526	0.7663	0.2345
DeepMCP	**0.7927**	**0.05518**	**0.7674**	**0.2341**

Table 3: Test AUC and Logloss on two large-scale datasets. DNN = Pred, DeepCP = Pred+Corr, DeepMP = Pred+Match, DeepMCP = Pred+Match+Corr.

number of fully connected layers in neural network-based models as 2, with dimensions 512 and 256. We set the batch size as 128, the context window size as $C = 2$ and the number of negative ads as $Q = 4$. The dropout ratio is set to 0.5. All the methods are implemented in Tensorflow and optimized by the Adagrad algorithm [Duchi *et al.*, 2011].

3.4 Evaluation Metrics

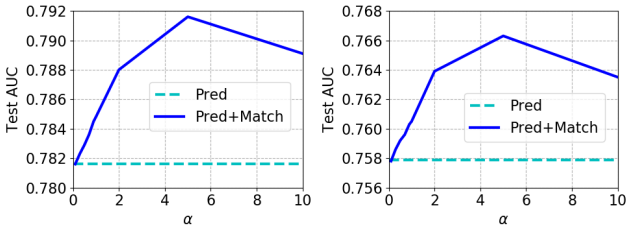
We use the following evaluation metrics.

1. **AUC**: the Area Under the ROC Curve over the test set. The larger the better. It reflects the probability that a model ranks a randomly chosen positive instance higher than a randomly chosen negative instance. A small improvement in AUC is likely to lead to a significant increase in online CTR [Cheng *et al.*, 2016].
2. **Logloss**: the value of Eq. (1) over the test set. The smaller the better.

3.5 Effectiveness

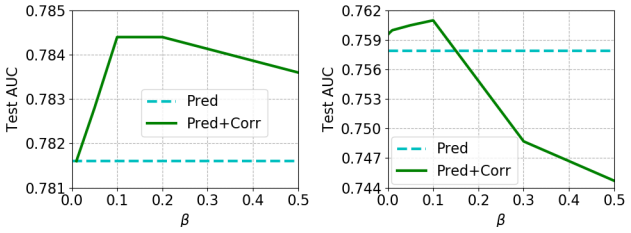
Table 3 lists the AUC and Logloss values of different methods. It is observed that FM performs much better than LR, because FM models second-order feature interactions while LR models linear feature importance. DNN further outperforms FM, because it can learn high-order nonlinear feature interactions [He and Chua, 2017]. PNN outperforms DNN because it further introduces a production layer. Wide&Deep further outperforms PNN, because it combines LR and DNN, which improves both the memorization and generalization abilities of the model. DeepFM combines FM and DNN. It performs slightly better than Wide&Deep on the Avito dataset, but slightly worse on the Company dataset.

It is observed that both DeepCP and DeepMP outperform the best-performing baseline on the two datasets. As the baseline methods only consider the prediction task, these observations show that additionally consider representation learning tasks can aid the performance of CTR prediction. It is also observed that DeepMP performs much better than DeepCP. It indicates that the matching subnet brings more benefits than the correlation subnet. This makes sense because the matching subnet considers both the users and the ads, while the correlation subnet considers only the ads. It is observed that DeepMCP that contains the matching, correlation and prediction subnets performs best on both datasets. These observations demonstrate the effectiveness of DeepMCP.



(a) Avito (b) Company

Figure 4: AUC vs. α in DeepMCP ($\beta = 0$): effect of the matching subnet, in addition to the prediction subnet. DNN = Pred, DeepMP = Pred+Match.



(a) Avito (b) Company

Figure 5: AUC vs. β in DeepMCP ($\alpha = 0$): effect of the correlation subnet, in addition to the prediction subnet. DNN = Pred, DeepCP = Pred+Corr.

3.6 Effect of the Balancing Parameters

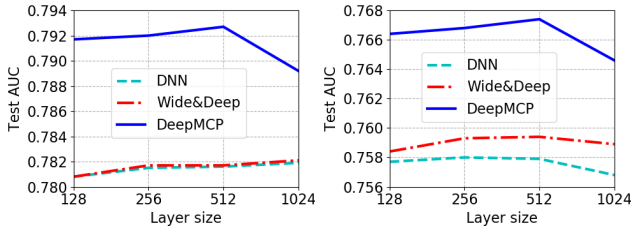
In this section, we examine the effect of tuning balancing hyperparameters of DeepMCP. Figure 4 and Figure 5 examine α (matching subnet) and β (correlation subnet) respectively. It is observed that the AUCs increase when one hyperparameter enlarges at the beginning, but then decrease when it further enlarges. On the Company dataset, large β can lead to very bad performance that is even worse than the prediction subnet only. Overall, the matching subnet leads to larger AUC improvement than the correlation subnet.

3.7 Effect of the Hidden Layer Size

In order not to make the figure cluttered, we only show the results of DNN, Wide&Deep and DeepMCP. Figure 6 plots the AUCs vs. the hidden layer sizes when the number of hidden layers is 2. We use a shrinking structure, where the second layer dimension is only half of the first. It is observed that when the first layer dimension increases from 128 to 512, AUCs generally increase. But when the dimension further enlarges, the performance may degrade. This is possibly because it is more difficult to train a more complex model.

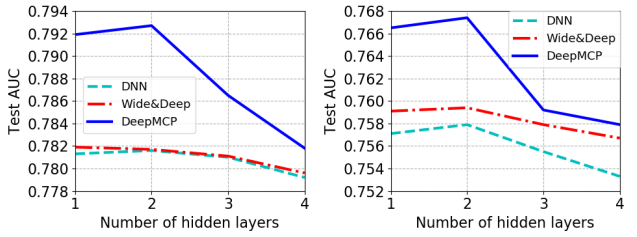
3.8 Effect of the Number of Hidden Layers

The dimension settings are: 1 layer - [256], 2 layers - [512, 256], 3 layers - [1024, 512, 256], and 4 layers - [2048, 1024, 512, 256]. It is observed in Figure 7 that when the number of hidden layers increases from 1 to 2, the performance generally increases. This is because more hidden layers have better



(a) Avito (b) Company

Figure 6: AUC vs. hidden layer size. The number of hidden layers is 2. The x-axis is the dimension of the first hidden layer. The dimension settings are: [128, 64], [256, 128], [512, 256], [1024, 512].



(a) Avito (b) Company

Figure 7: AUC vs. the number of hidden layers. The dimension settings are: 1 layer - [256], 2 layers - [512, 256], 3 layers - [1024, 512, 256], and 4 layers - [2048, 1024, 512, 256].

expressive abilities [He and Chua, 2017]. But when the number of hidden layers further increases, the performance then decreases. This is because it is more difficult to train deeper neural networks.

4 Related Work

CTR Prediction

CTR prediction has attracted lots of attention from both academia and industry [He *et al.*, 2014; Cheng *et al.*, 2016; He and Chua, 2017; Zhou *et al.*, 2018]. Generalized linear models, such as Logistic Regression (LR) [Richardson *et al.*, 2007] and Follow-The-Regularized-Leader (FTRL) [McMahan *et al.*, 2013], have shown decent performance in practice. However, a linear model lacks the ability to learn sophisticated feature interactions [Chapelle *et al.*, 2015]. Factorization Machines (FMs) [Rendle, 2010; Rendle, 2012] are proposed to model pairwise feature interactions in terms of the latent vectors corresponding to the involved features. Field-aware FM [Juan *et al.*, 2016] and Field-weighted FM [Pan *et al.*, 2018] further consider the impact of the field that a feature belongs to in order to improve the performance of FM.

In recent years, Deep Neural Networks (DNNs) are exploited for CTR prediction and item recommendation in order to automatically learn feature representations and high-order feature interactions [Van den Oord *et al.*, 2013; Covington *et al.*, 2016; Wang *et al.*, 2017; He and Chua, 2017]. Zhang et al. [2016b] propose Factorization-machine supported Neural Network (FNN), which pre-trains an FM before applying a DNN. Qu et al. [2016] propose the Product-based Neural Network (PNN) where a product layer is introduced between the embedding layer and the fully connected layer. Cheng et al. [2016] propose Wide&Deep, which combines LR and DNN in order to improve both the memorization and generalization abilities of the model. Guo et al. [2017] propose DeepFM, which models low-order feature interactions like FM and models high-order feature interactions like DNN. He et al. [2017] propose the Neural Factorization Machine which combines the linearity of FM and the non-linearity of neural networks. Nevertheless, these methods mainly model the feature-CTR relationship. Our proposed DeepMCP model further considers user-ad and ad-ad relationships.

Multi-modal / Multi-task Learning

Our work is also closely related to multi-modal / multi-task learning, where multiple kinds of information or auxiliary tasks are introduced to help improve the performance of the main task. For example, Zhang et al. [2016a] leverage heterogeneous information (i.e., structural content, textual content and visual content) in a knowledge base to improve the quality of recommender systems. Gao et al. [2018] utilize textual content and social tag information, in addition to classical item structure information, for improved recommendation. Huang et al. [2018] introduce context-aware ranking as an auxiliary task in order to better model the semantics of queries in entity recommendation. Gong et al. [2019] propose a multi-task model which additionally learns segment tagging and named entity tagging for slot filling in online shopping assistant. In our work, we address a different problem and we introduce two auxiliary but related tasks (i.e., matching and correlation with shared embeddings) to improve the performance of CTR prediction.

5 Conclusion

In this paper, we propose DeepMCP, which contains a matching subnet, a correlation subnet and a prediction subnet for CTR prediction. These subnets model the user-ad, ad-ad and feature-CTR relationship respectively. Compared with classical CTR prediction models that mainly consider the feature-CTR relationship, DeepMCP has better prediction power and representation ability. Experimental results on two large-scale datasets demonstrate the effectiveness of DeepMCP in CTR prediction. It is observed that the matching subnet leads to higher performance improvement than the correlation subnet. This is possibly because the former considers both users and ads, while the latter considers ads only.

References

[Chapelle *et al.*, 2015] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology*, 5(4):61, 2015.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10. ACM, 2016.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198. ACM, 2016.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[Gao *et al.*, 2018] Li Gao, Hong Yang, Jia Wu, Chuan Zhou, Weixue Lu, and Yue Hu. Recommendation with multi-source heterogeneous information. In *IJCAI*, 2018.

[Gong *et al.*, 2019] Yu Gong, Xusheng Luo, Yu Zhu, Wenwu Ou, Zhao Li, Muhua Zhu, Kenny Q Zhu, Lu Duan, and Xi Chen. Deep cascade multi-task learning for slot filling in online shopping assistant. In *AAAI*, 2019.

[Guo *et al.*, 2017] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*, pages 1725–1731, 2017.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364. ACM, 2017.

[He *et al.*, 2014] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*, pages 1–9. ACM, 2014.

[Huang *et al.*, 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, pages 2333–2338. ACM, 2013.

[Huang *et al.*, 2018] Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. Improving entity recommendation with search log and multi-task learning. In *IJCAI*, pages 4107–4114, 2018.

[Juan *et al.*, 2016] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *RecSys*, pages 43–50. ACM, 2016.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[LeCun *et al.*, 2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[Liu and others, 2009] Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[McMahan *et al.*, 2013] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *KDD*, pages 1222–1230. ACM, 2013.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Nair and Hinton, 2010] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[Pan *et al.*, 2018] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. Field-weighted factorization machines for click-through rate prediction in display advertising. In *WWW*, pages 1349–1357. International World Wide Web Conferences Steering Committee, 2018.

[Qu *et al.*, 2016] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *ICDM*, pages 1149–1154. IEEE, 2016.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.

[Rendle, 2012] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57, 2012.

[Richardson *et al.*, 2007] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, pages 521–530. ACM, 2007.

[Shan *et al.*, 2016] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *KDD*, pages 255–262. ACM, 2016.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Van den Oord *et al.*, 2013] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *NIPS*, pages 2643–2651, 2013.

[Wang *et al.*, 2017] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *ADKDD*, page 12. ACM, 2017.

[Zhang *et al.*, 2016a] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362. ACM, 2016.

[Zhang *et al.*, 2016b] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data. In *ECIR*, pages 45–57. Springer, 2016.

[Zhao *et al.*, 2018] Kui Zhao, Yuechuan Li, Zhaoqian Shuai, and Cheng Yang. Learning and transferring ids representation in e-commerce. In *KDD*, pages 1031–1039. ACM, 2018.

[Zhou *et al.*, 2018] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068. ACM, 2018.