Convolutional Neural Networks based Click-Through Rate Prediction with Multiple Feature Sequences

Patrick P. K. Chan[1], **Xian Hu**[1,2], **Lili Zhao**[2], **Daniel S. Yeung**[1], **Dapeng Liu**[2], **Lei Xiao**[2]
[1] School of Computer Science and Engineering, South China University of Technology, China
[2] Tencent, China
patrickchan@ieee.org, huxian.scut@gmail.com, danyeung@ieee.org
{karenzhao, rocliu, shawnxiao}@tencent.com

Abstract

Convolutional Neural Network (CNN) achieved satisfying performance in click-through rate (CTR) prediction in recent studies. Since features used in CTR prediction have no meaningful sequence in nature, the features can be arranged in any order. As CNN learns the local information of a sample, the feature sequence may influence its performance significantly. However, this problem has not been fully investigated. This paper firstly investigates whether and how the feature sequence affects the performance of the CNN-based CTR prediction method. As the data distribution of CTR prediction changes with time, the best current sequence may not be suitable for future data. Two multi-sequence models are proposed to learn the information provided by different sequences. The first model learns all sequences using a single feature learning module, while each sequence is learnt individually by a feature learning module in the second one. Moreover, a method of generating a set of embedding sequences which aims to consider the combined influence of all feature pairs on feature learning is also introduced. The experiments are conducted to demonstrate the effectiveness and stability of our proposed models in the offline and online environment on both the benchmark Avazu dataset and a real commercial dataset.

1 Introduction

Displaying suitable ads to users not only enhances their experience but also increases the profit of advertisement publishers. Click-through rate (CTR), which measures the ratio of users who click on a recommended ad to the number of total users who view a page with this ad, is one of popular measures for evaluating online advertising [McMahan *et al.*, 2013; He *et al.*, 2014]. In some online advertising systems, ads are displayed according to the probability of user click and bid price in order to maximize the profit. As a result, accurate CTR prediction plays an important role in the success of online advertising.

Many recent studies apply CNN to CTR prediction [Liu *et al.*, 2015; Li *et al.*, 2015; Chen *et al.*, 2016; Edizel *et al.*, 2017] due to its success in many complex applications, *e.g.,* computer vision [He *et al.*, 2016; Yang *et al.*, 2017] and natural language processing [Zhang *et al.*, 2015; Wang *et al.*, 2016; 2017a]. The CNN-based CTR prediction model [Liu *et al.*, 2015], commonly used in the study of CTR prediction, is the focus of this study. Its architecture is illustrated in Fig. 1. The model maps the information of a user, ad, and context to estimate CTR of the user on the ad. A sample contains a number of fields represented by a vector of one-hot encoding. Therefore, the input space of CTR prediction is usually sparse and has a high dimension [Guo *et al.*, 2017; Shan *et al.*, 2016; Wang *et al.*, 2017b].

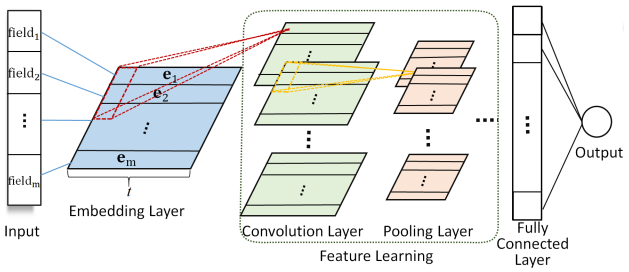


Figure 1: The architecture of a common CNN-based CTR prediction model.

The embedding layer [Rendle, 2010; Juan *et al.*, 2016] is applied to map the fields into a structural and dense input space, *i.e.,* the ith field is mapped into $\mathbf{e}_i$, where $\mathbf{e}_i$ denotes the ith embedding feature vector of length t. $\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, ...\mathbf{e}_m]$, where $i = 1, 2, ..., m$ and m is the number of fields. The embedding feature vectors are fed into feature learning layers including convolution and pooling. All learned latent features are then processed by the fully connected layer to predict the CTR in the last stage of the model.

Research of CNN-based CTR prediction mainly focuses on the design of the embedding layer [Edizel *et al.*, 2017] and the architecture of the CNN model [Liu *et al.*, 2015]. Different from applications like image or natural language processing in which a sample has a natural sequence, embedding feature vectors ($\mathbf{e}$) of a CTR prediction can be arranged in any order.

However, the sequence of embedding feature vectors affects the local information learnt by CNN since its convolution and pooling layers capture information in local receptive fields. This issue has not been investigated in depth.

This paper firstly discusses whether and how the sequence of embedding feature vectors influences the CNN-based CTR prediction method in Sect. 2. Experiments are also carried out to illustrate the AUC (Area Under ROC Curve) [Calders and Jaroszewicz, 2007] obtained by the model using different sequences of embedding feature vectors in the benchmark dataset (*i.e.,* Avazu dataset) and a real commercial dataset (*i.e.,* the internal advertising dataset in Tencent). The fluctuation of the performance of a model with a sequence in different periods of time is also investigated since CTR prediction is a non-stationary problem due to the daily change on ads.

Choosing the best sequence is one of intuitive solutions. However, the process may be inefficient and the potential information provided by other sequences is ignored. Moreover, the best sequence for current data may perform differently in future in a non-stationary problem. As a result, two models considering multi-sequence of embedding feature vectors are proposed in Sect. 3. The information provided by embedding feature vectors with multi-sequence is firstly combined and is learnt by one feature learning module in the first model, Multi-Sequence Model with Single Feature Learning Module (MSS). Although the time complexity of the model is low, using a large number of embedding feature sequences may downgrade the model's performance since the feature learning module cannot learn the information provided by all sequences efficiently. To overcome this drawback, we introduce another model, Multi-Sequence Model with Multiple Feature Learning Modules (MSM), in which embedding feature vectors with each sequence are learnt by one feature learning module, and the learned representation is combined in the fully connected layer. The better performance is expected to be achieved by *MSM* than *MSS* since the information provided by multi-sequence is learnt more efficiently in *MSM*.

In addition, a sequence generation method is also proposed in Sect. 3.1. Instead of generating embedding feature sequences randomly, the generated sequences aim to guarantee that the combined information of every pair of embedding feature vectors can be learnt in the feature learning. The proposed models are evaluated in offline and online environment using the Avazu and Tencent dataset in Sect. 4. Finally, the conclusion and future work are discussed in Sect. 5.

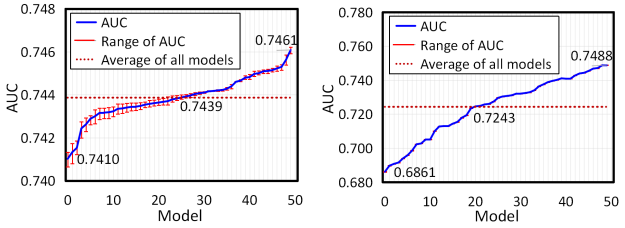
The main contributions of this work can be summarized as follows:

- This study investigates whether and how the order of the embedding feature vectors influences the performance of the CNN-based CTR prediction model. Besides the conceptual discussion, experimental study is also carried out to evaluate the accuracy of the models using different feature sequences. The performance fluctuation of a model with the same sequence in different periods of time is also investigated.
- Two models with multi-sequence (*i.e., MSS* and *MSM*) are proposed. Both models consider multiple feature sequences to provide additional local information for learning. *MSS* simply learns the feature with sequences by using one feature learning module, while a feature learning module is used for each sequence in *MSM*.
- The method of generating a set of feature sequences which aims to consider the combined influence of every feature pair on the output of the feature learning is presented.
- The proposed models are evaluated by using the benchmark dataset (Avazu) and a commercial dataset collected by Tencent. The impact of parameters of the models is also investigated. Experimental results consistently suggest stability and effectiveness of our models.

2 Embedding Feature Sequence Analysis

The features used in CTR prediction has no naturally meaningful sequence, which is different from the samples in some applications, *e.g.,* image and audio. The embedding features with different orders contain various information for learning of CNN since CNN captures the local information in convolution and pooling layers. As a result, a CNN-based CTR prediction model may perform differently depending on the choice of the sequence of embedding feature vectors.

An experiment is carried out to illustrate the influence of a model using different feature sequences on the accuracy of CTR prediction. Avazu dataset and Tencent dataset are used. Avazu dataset includes samples with 22 fields collected in ten days, while samples with 35 fields are collected in 17 days in the Tencent dataset. 50 CNN-based CTR prediction models [Liu *et al.*, 2015] with different sequences of embedding feature vectors generated randomly are trained. The detailed description on the datasets and the parameter settings are introduced in Sect. 4.1 and 4.2. For the Avazu (Tencent) dataset, the training set contains the samples in the first nine (seven) days, while samples in the tenth (eighth) day are used for evaluation. Each experiment is carried out three times independently. AUC is used as the evaluation metric.



(a) Avazu dataset (b) Tencent dataset

Figure 2: Mean and range of AUC of the CTR prediction model with different feature sequences sorted in ascending order by AUC.

Fig. 2 shows the mean and range of AUC of the models with different feature sequences. The models are sorted in ascending order by AUC from left to right. The average AUC of all models is represented by the horizontal dotted line in red. The results suggest that the models with different sequences yield different AUCs, *i.e.,* the best (worst) model achieves 0.7461 (0.7410) for the Avazu dataset and 0.7488 (0.6861)

for the Tencent dataset. It should be noted that the AUC difference is small but significant in practice. Moreover, the variance of the models due to the random initialization on the training parameters is much smaller than the one due to using different sequences, *i.e.,* the AUC range of three independent runs, shown in red solid line, is much smaller than the AUC difference of the models using different sequences, especially in the Tencent dataset.

For the Avazu dataset, the AUC of more than 75% of the models is between 0.743 and 0.745, but only a few models achieve an AUC close to the best value (0.7461). It means searching for a sequence yielding good performance is difficult. On the other hand, the performance of the models in the Tencent dataset is more evenly distributed than the one in the Avazu dataset, *i.e.,* the line of AUC is close to a linear function in the Tencent dataset. As a result, the performance of the models is more sensitive to different sequences in the Tencent dataset.

Data distribution of CTR prediction changes frequently as ads are updated every day. We evaluate the stability of a model with the same sequence at different periods of time. Data in the Tencent dataset in three periods of time are considered, including: 1st to 8th day, 4th to 11th day, and 7th to 14th day. For each period, the samples in the first seven days are used in training and the samples in the last day are reserved for evaluation. 50 CNN-based models with different feature sequences generated randomly are trained by using the data in these three periods separately. The average range of AUC of these models on three periods is 0.0020, which is significantly large in CTR prediction. This indicates that the performance of the model using a single sequence fluctuates, *i.e.,* there is no best sequence all the time.

These experimental results suggest that the sequence of embedding features affect the accuracy of CTR prediction significantly. Moreover, choosing the best sequence may not be realistic since CTR prediction is a non-stationary problem. A sequence which works well for the current data may not be suitable for future data.

3 Proposed Methods

The influence of the sequence of embedding feature vectors in CTR prediction should be addressed since it affects the prediction accuracy dramatically. However, choosing the best current sequence may be time-consuming, and the sequence may be not suitable for future data. Moreover, the potential information provided by other sequences is also ignored. As a result, this study proposes two models using multi-sequence to consider additional information provided by embedding feature vectors with different sequences.

A sequence generation method aiming to provide a set of sequences of embedding features to enable the feature learning module of a CTR prediction model learn the combined influence of all feature pairs is firstly discussed in Sect. 3.1. The generated sequences can be applied to *MSS* and *MSM* considering multi-sequence of embedding feature vectors proposed in Sect. 3.2 and 3.3. The information provided by multi-sequence is learnt by one feature learning module in *MSS*, while a number of feature learning modules are applied to learn from multi-sequence separately in *MSM*. The pros and cons of the models are also discussed.

3.1 Sequence Generation Method

A method to generate a set of sequences of embedding feature vectors is introduced in this section. The set of sequences aims to minimize the variance of the combined influence of all feature pairs on the output of the feature learning module in order to provide different local information for learning.

Given the set of parameters θ of the feature learning in the CNN-based CTR prediction model and a sequence of embedding feature vectors $\mathbf{s}$, where $\mathbf{s} = \{s_1, s_2, ..., s_m\}$ is the order list and the embedding features are listed in the order of $\mathbf{e}_{s_1}, \mathbf{e}_{s_2}, ..., \mathbf{e}_{s_m}$, we firstly define the relation $r \in \mathscr{R}$ between $\mathbf{e}_i$ and $\mathbf{e}_j$ as the consideration level on their combined influence in the feature learning. Both the architecture of feature learning module (θ) and the feature sequence ($\mathbf{s}$) affect $r_\mathbf{s}$. The definition of $r_\mathbf{s}(\mathbf{e}_i, \mathbf{e}_j)$ is given as follows:

$$r_\mathbf{s}(\mathbf{e}_i, \mathbf{e}_j) = g(d_\mathbf{s}(\mathbf{e}_i, \mathbf{e}_j), C_\theta(l)) \tag{1}$$

where $d_\mathbf{s}(\mathbf{e}_i, \mathbf{e}_j) \in Z^+$ measures the distance between $\mathbf{e}_i$ and $\mathbf{e}_j$ in $\mathbf{s}$. L1-norm is used in this study. $C_\theta(l)$ is a function which outputs the number of embedding feature vectors $\mathbf{e}$ affecting the output of a neuron in the lth pooling layer in the feature learning. $r_\mathbf{s}(\mathbf{e}_i, \mathbf{e}_j) = 0$ means that the combined influence of two embedding feature vectors are not considered by the feature learning. On the other hand, larger $r_\mathbf{s}$ indicates that the feature learning learns more information from the feature pair due to the local receptive fields of CNN. For the same C_θ, smaller $d_\mathbf{s}$ yields larger $r_\mathbf{s}$, *i.e.,* more local information of features is captured when features are closer to each other.

Smaller $C_\theta(l)$ means less number of embedding feature vectors are considered in the lth pooling layer in the feature learning, and versa vice. C_θ can be calculated as:

$$C_\theta(l) = \sum_{i=1}^{l} [(t_\theta(i) - 1) \cdot \prod_{j=1}^{i} h_\theta(j-1)] + h_\theta(0) \tag{2}$$

where l denotes the lth pooling layer in the feature learning, and $t_\theta(i)$ is the number of features being extracted from the previous convolution layer. $t_\theta(i) = k_h(i) + k_g(i) \times (p_h(i) - 1)$, where $k_h(i)$ and $k_g(i)$ denote the height of the filter and the height of convolution stride in the ith convolution layer, and $p_h(i)$ is the height of the pooling in the ith pooling layer. $h_\theta(i)$ influences the outputs of two adjacent neurons in the ith pooling layer. $h_\theta(i) = k_g(i) \times p_g(i)$, where $k_g(i)$ and $p_g(i)$ are the height of the stride in the ith convolution and pooling layer. In the special case when $l = 0$, $C_\theta(0) = h_\theta(0) = 1$, which means only one feature affects the output of the input layer.

g is a decreasing function with $d_\mathbf{s}$ when C_θ is fixed, *i.e.,* $g(d_i, C_\theta) \geq g(d_j, C_\theta)$ when $d_i < d_j$. In this study, g is defined as a staircase function, *i.e.,* $g = 1 - (i-1)/L$ when $C_\theta(i-1) < d_\mathbf{s} \leq C_\theta(i)$, where L denotes the total pairs of convolution and pooling layers in the feature learning, and $1 \leq i \leq L$. When the pair of embedding feature vectors are considered in a shallower layer, g tends to 1; otherwise, g tends to 0. When $d_\mathbf{s} > C_\theta(L)$, $g = 0$, which indicates that the feature learning module (θ) does not consider the combined

influence of two embedding feature vectors with $d_{\mathbf{s}}$. On the other hand, $g = 1$ when $d_{\mathbf{s}} = 0$, which means two embedding feature vectors are the same.

The sequence generation can be formulated as the optimization problem shown in (3). The model aims to consider the combined influence on the feature learning for every feature pair. The optimal sequence set $\mathbf{S}^{\star}$ with n number of sequences are determined by minimizing the variance of $r_{\mathbf{s}}$ of all feature pairs.

$$\begin{aligned} \min_{\mathbf{S}} \quad & var([r|r = \textstyle\sum_{\mathbf{s}\in\mathbf{S}} r_{\mathbf{s}}(\mathbf{e}_i, \mathbf{e}_j); 1 \le i < j \le m)]) \\ \text{s.t.} \quad & |\mathbf{S}| = n \end{aligned} \tag{3}$$

The optimization problem in (3) is a NP-hard problem. Therefore, a practical method with greedy search is proposed to find a sub-optimal solution. Let $\mathbf{G}$ be a set containing all the possible sequences of embedding feature vectors. A sequence $\mathbf{s}$ selected randomly from $\mathbf{G}$ is added to $\mathbf{S}$ and $\mathbf{G} = \mathbf{G} - \mathbf{s}$ in the initialization. For each iteration, $\mathbf{s} \in \mathbf{G}$ is added into $\mathbf{S}$ and $\mathbf{G} = \mathbf{G} - \mathbf{s}$ when $\mathbf{S} \bigcup \mathbf{s}$ yields the smallest $var(\mathbf{r})$, and $\mathbf{r} = [r|r = \sum_{\mathbf{s}\in\mathbf{S}} r_{\mathbf{s}}(\mathbf{e}_i, \mathbf{e}_j); 1 \le i < j \le m)]$. The iteration is terminated when $|\mathbf{S}| = n$. However, the running time of the algorithm may still be long since $|\mathbf{G}| = m!$ is a huge number. $\mathbf{G}$ is replaced by $\tilde{\mathbf{G}}$ which contains a number of elements randomly selected from $\mathbf{G}$, *i.e.,* $\tilde{\mathbf{G}} \subset \mathbf{G}$ and $|\tilde{\mathbf{G}}| \ll |\mathbf{G}|$. The size of $\tilde{\mathbf{G}}$ is the tradeoff between the time complexity and the preciseness of the algorithm.

3.2 Multi-Sequence Model with Single Feature Learning Module (MSS)

A model learning the multiple embedding feature sequences is introduced in this section. The multi-sequence (MS) layer is added after the embedding layer in the CNN-based CTR prediction model, shown in Fig. 3. The MS layer aims to arrange the embedding feature $[\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_m]$ into n different orders, *i.e.,* $[\mathbf{e}_{s_{i1}}, \mathbf{e}_{s_{i2}}, ..., \mathbf{e}_{s_{im}}]$, according to the sequence $\mathbf{S}$, where $\mathbf{s}_i \in \mathbf{S}$, $\mathbf{s}_i = [s_{i1}, s_{i2}, ..., s_{im}]$ and $i = 1, 2, ..., n$. $\mathbf{S}$ can be obtained by using sequence generation methods mentioned in the previous section.

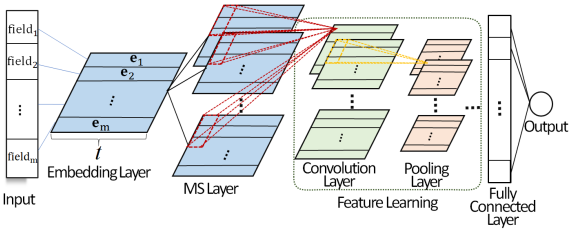


Figure 3: Architecture of *MSS*. MS layer which aims to arrange the embedding features into different orders is added between the embedding layer and the first convolution layer. The outputs of MS layer are fed into a feature learning module.

All feature maps of the MS layer are then treated as inputs of the first convolution layer, *i.e.,* $c_i^0 = [\mathbf{e}_{s_{i1}}, \mathbf{e}_{s_{i2}}, ..., \mathbf{e}_{s_{im}}]$. In general, the ith output of the lth pair of convolution and pooling layer, c_i^l, can be defined as:

$$\mathbf{c}_i^l = q(f(\sum_{j=1}^{t_{l-1}} conv(\mathbf{c}_j^{l-1}, \mathbf{w}_{ij}^l) + \mathbf{b}_{ij}^l)) \tag{4}$$

where q and f are the pooling function and the activation function respectively, $\mathbf{w}_{ij}$ represents the weights of the ith filter with the jth input, $\mathbf{b}$ is the bias term, and t_l is the number of feature maps in the ith layer. The cross-entropy loss function is applied:

$$\mathbf{L} = -\frac{1}{u}\sum_{i=1}^{u}[y_i log\hat{y}_i + (1 - y_i) log(1 - \hat{y}_i)] \tag{5}$$

where u is the number of training samples, and y_i and $\hat{y}_i$ denote the true and the predicted value on the ith sample.

One advantage of the model is that the additional time complexity of training is insignificant in comparison with the model with single sequence since only one feature learning module is applied to learn the information provided by multi-sequence. However, the information provided by different sequences may exert influence on each other, which increases the learning difficulty to the feature learning module. As a result, the performance of *MSS* may drop when many sequences of embedding features (*i.e., n* is large) are considered.

3.3 Multi-Sequence Model with Multiple Feature Learning Modules (MSM)

Another multi-sequence model is proposed to overcome the problem of *MSS*. The embedding feature vectors with different sequences are learnt by different feature learning modules individually. The outputs of all feature learning modules are then concatenated before the fully connected layer. The architecture of *MSM* is illustrated in Fig. 4

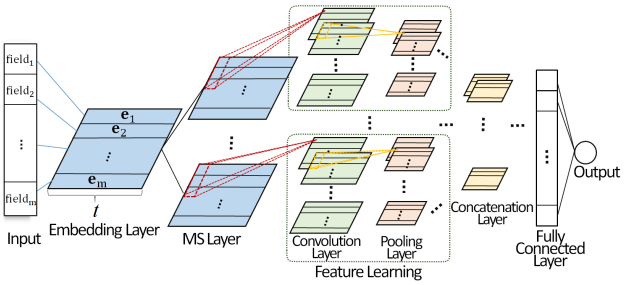


Figure 4: Architecture of *MSM*. Each feature map in MS layer is learnt by a feature learning module.

Different from *MSS*, each feature mapping in the MS layer is learnt by a feature learning module separately. The output of the first pair of convolution and pooling layer is defined as:

$$\mathbf{u}_{ij}^1 = q(f(conv(\mathbf{c}_j^0, \mathbf{w}_{ij}^1) + \mathbf{b}_{ij}^1)), j = 1, 2, ..., n \tag{6}$$

where $\mathbf{c}_j^0$ denotes the jth feature map of the MS layer. Finally, the concatenated latent representations are fed into the fully connected layer to obtain the estimated CTR.

As multiple sequences are processed separately before concatenation in this model, the information of each sequence can be learnt more effectively in order to improve the accuracy of the prediction. However, the time complexity is a concern since the training time of *MSM* is approximately equal to n times of the model using a single sequence, where n is the number of sequences used in *MSM*.

4 Experiments

The proposed models are evaluated in this section experimentally. The datasets and experimental setting are firstly introduced. The performance of our models and the model using one sequence of embedding feature vectors is compared in terms of AUC in the offline and online environment in Sect. 4.3 and 4.4 respectively.

4.1 Datasets

Ad platform generates billions of dollars income for Tencent every year. We focus on an advertising space in which ads are recommended from hundreds of thousands of candidates to hundred millions of users every day. A subset with hundreds of ad candidates is firstly selected by the pre-screening procedure. 35 fields of features separated into three categories (*i.e.,* user profile, ad information and context information) are selected for each sample in our experiments. In practice, the ratio of the positive (click) samples are relatively small, *i.e.,* CTR prediction is an imbalance problem [Deng *et al.*, 2017]. A subsampling process is carried out on the negative samples while all positive samples are used. 2 billion samples for 17 consecutive days are contained in the *Tencent dataset*. On the other hand, the *Avazu dataset*[1] is provided in the competition of Kaggle in 2014. The dataset contains 40 million samples with 22 fields for ten consecutive days. All fields are used in the experiments.

4.2 Settings

The model using only one feature sequence is used as the benchmark model (*base*). The feature sequence is selected according to AUC in a preliminary experiment for *base*. Our proposed sequence generation (*SG*) is compared with the random sequence generation (*RD*). *RD* generates sequences randomly except the first sequence which is provided by *base*. *MSS* and *MSM* with proposed sequence generation method (*MSS-SG*, *MSM-SG*) and also with sequence generated randomly (*MSS-RD*, *MSM-RD*) are included in the experiment.

The common CNN-based CTR prediction model [Liu *et al.*, 2015] is considered in our experiment. Each field is mapped to an embedding feature vector of $t = 30$. All models are implemented with two pairs of convolution and max-pooling layer. The filter sizes are 4×4 (5×5) and 3×3 (4×4) in the first and second convolution layer respectively for the Avazu (Tencent) dataset, and the convolution stride is set as 1. Pooling size of both layers is 2×2, and the pooling stride is 2 for both datasets. ReLU is used as the activation function. A fully connected layer with a single neuron outputs estimated CTR. For our models, $|\tilde{\mathbf{G}}|$ is set as 1000. All experimental settings for *base*, *MSS* and *MSM* are the same. Average AUC on test sets are used as evaluation criterion.

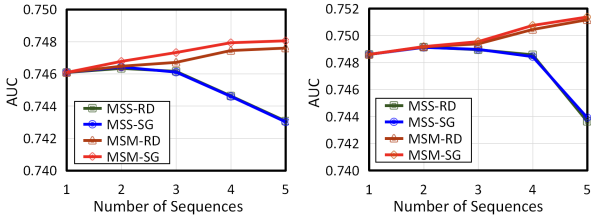
4.3 Evaluation of the Offline System

This experiment evaluates the models offline setting in which a model is not updated after training. The samples in the first nine (seven) days are reserved for training and the samples of the tenth (eighth) day is used for evaluation in the Avazu (Tencent) dataset. Each experiment has been repeated three times independently. We apply mini-batch stochastic optimization with Adam optimizer [Kingma and Ba, 2014].

[1] https://www.kaggle.com/c/avazu-ctr-prediction

Analysis on Feature Sequence Number

The influence of the number of feature sequences on the performance of the models is discussed. AUC of the models with the sequence number (n) from 1 to 5 is shown in Fig. 5.



(a) Avazu dataset (b) Tencent dataset

Figure 5: AUC of the models with $n = 1, 2, ..., 5$

When $n = 1$, the model is identical to *base*, in which only a single sequence is considered. It explains why AUC of all models is the same. The similar performance of all methods can be observed in both datasets. For *MSS-SG* and *MSS-RD*, AUC reaches the peak at $n = 2$ (*i.e.,* 0.7464 for the Avazu and 0.7491 for the Tencent dataset), and decreases with the increase of n. These results are consistent with our expectation that *MSS-SG* and *MSS-RD* may not perform well when n is large since one feature learning module may not be able to learn the information provided by multi-sequence efficiently. Therefore, AUC of *MSS-SG* and *MSS-RD* slightly increases comparing to the one for *base*, but the feature learning module cannot learn well when the sequence number increases.

Different form *MSS-SG* and *MSS-RD*, AUC of *MSM-SG* and *MSM-RD* rises with the increase of n in general. It indicates that various local information provided by different feature sequences can be learnt. The results show that using more sequences yields a better result. However, the increase of AUC of *MSM-SG* and *MSM-RD* with $n = 4$ and 5 is small in the Avazu dataset, *i.e.,* AUC becomes stable when no additional information is provided by a new sequence. As the Tencent dataset is more complicated than the Avazu dataset, we expect that similar results will be observed in the Tencent dataset when n is larger.

The advantage of the proposed sequence generation method only can be observed in *MSM* but not in *MSS*. This is because the information of multi-sequence cannot be learnt efficiently in *MSS*. The difference of sequences generated randomly or by our method is not significant. However, from the difference on the performance of *MSM-RD* and *MSM-SG*, we can conclude that a set of sequences generated by our method provides more useful information for learning.

Accuracy of the Proposed Multi-Sequence Models

This section focuses on AUC obtained by the CNN-based CTR prediction models. n are set as 2 and 5 for *MSS* and *MSM* according to the experimental results in last section. Average AUC and relative improvement (RI) defined as $(\frac{AUC(m)-0.5}{AUC(base)-0.5} - 1) \times 100\%$ [Yan *et al.*, 2014] of the model m on the Avazu and Tencent dataset are listed in Table **??**.

Method	Avazu		Tencent	
	AUC	RI (%)	AUC	RI (%)
Base	0.7461	/	0.7485	/
MSS-RD	0.7463	0.0813	0.7490	0.2012
MSS-SG	0.7464	0.1219	0.7491	0.2414
MSM-RD	0.7476	0.6095	0.7512	1.0903
MSM-SG	**0.7480**	**0.7720**	**0.7514**	**1.1771**
AdPredictor	0.7375*	-3.4945	/	/
FTRL	0.7357*	-4.2259	/	/
MatchBox	0.7426*	-1.4222	/	/

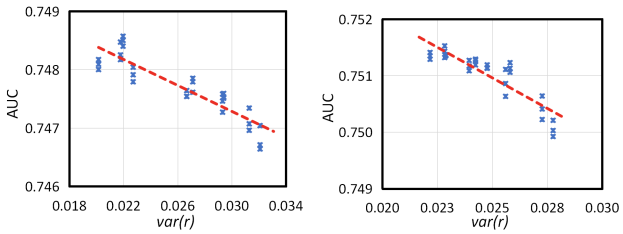
* AUC is obtained from [Liu *et al.*, 2017]

Table 1: AUC and RI of the CTR prediction models.

All proposed multi-sequence models outperform *base*, *i.e.,* all RIs of our models are positive. Compared to *base*, *MSM* achieves more than 0.60% (1.00%) on the Avazu (Tencent) dataset. Besides, AUC of *MSM-SG* is 0.16% (0.09%) higher than the one of *MSM-RD* on the Avazu (Tencent) dataset. The average AUC of *MSS* obtains more than 0.10% (0.22%) improvement of *base* on the Avazu (Tencent) dataset. The improvement is small but significant in practice [Zhou *et al.*, 2017]. For the Avazu dataset, we compare our results with the one obtained in the same experimental settings [Liu *et al.*, 2017]. AdPredictor, FTRL, and MatchBox, which are not CNN-based methods, but are commonly used in CTR prediction. *base* performs significantly better than these methods in terms of AUC, which shows the advantage of the generation ability of CNN-based models.

Analysis of Sequence Generation Method

This section discusses the relation between the performance of a model and $var(\mathbf{r})$ of the sequences used by the model. The last section shows that there is no significant difference on the performance of *MSS-RD* and *MSS-SG*, *i.e.,* the feature sequences do not affect the performance of *MSS*. As a result, only *MSM* is considered in this section. Ten sets of sequences are generated randomly. *MSM* with each sequence set is trained three times independently to obtain three results. AUC of 30 models are illustrated in Fig. 6.



(a) Avazu dataset (b) Tencent dataset

Figure 6: Relation of AUC of *MSM* and $var(\mathbf{r})$ of its sequences.

Each cross in the graphs denotes a model. X-axis and Y-axis represent $var(\mathbf{r})$ and AUC. The red dotted line represents the least squares regression trained by using the 30 points. The clear and strong relation between $var(\mathbf{r})$ and AUC is observed in both datasets, *i.e.,* $\mathbf{S}$ with smaller $var(\mathbf{r})$ yields higher AUC. These results explain the reason why the performance of *MSM-SG* is significantly better than *MSM-RD*.

4.4 Evaluation of the Online System

CTR prediction is a non-stationary problem due to the daily change of ads and users. For example, the chosen advertising space in Tencent advertising system has 10% to 20% records being new ads in the repository each day. Since the sample distribution is not stable, the prediction performance may change significantly. Therefore, the adaptation of the models on the change of the sample distribution is evaluated.

The Tencent dataset is applied in this section to evaluate the performance of the models under the online environment. The training set with the samples in the first seven days is used to train the initial model. The model is then updated every two hours during the rest ten days. Each update uses the training set containing samples collected between the last hour and last 25 hours, while the samples in the last hour are used for model validation. The performances of all models with online learning are shown in Fig. 7.

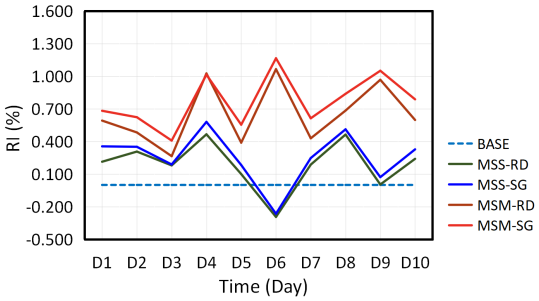


Figure 7: RI (%) of all models in the online environment.

The performance of *base* is the worst among all methods in the online environment because the best sequence may not always be good for future data. A number of sequences provides more information than the single one which leads to better result in general. The results of the online environment are consistent with the offline ones, *i.e.,* the results suggest that models with multi-sequence achieve more accurate CTR prediction in practice. Moreover, our models provide prediction in 5ms or less, which is less than the 10ms latency budget for CTR predictions per request for the Tencent advertising system.

5 Conclusions

This study shows that embedding feature vectors with different sequences provide useful information for CNN-based CTR prediction. The current methods which consider a single sequence cannot perform well consistently in periods of time due to the change of data distribution. As a result, two multi-sequence models are proposed to capture the information provided by different feature sequences. *MSS* learns multi-sequence by a feature learning module, while each sequence is learnt by a feature learning module separately in *MSM*. The experimental results confirm that both models achieve significantly higher accuracy than the model using only one sequence in terms of AUC for the benchmark Avazu dataset and the the internal advertising dataset in Tencent. Sequence generation method is also proposed to provide a set of

feature sequence which considers the combined influence of every feature pair on the output of feature learning. Its superiority in comparison with random generation is also confirmed experimentally.

One drawback of *MSM* is its high time complexity in training. One possible solution is to simplify the architecture of the feature learning module for each feature sequence. In addition, some feature pairs may provide more useful information than others. One future work may focus on investigating the contribution of a feature pair to the learning, and arrange the feature sequences according to their contributions.

References

[Calders and Jaroszewicz, 2007] Toon Calders and Szymon Jaroszewicz. Efficient auc optimization for classification. In *European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 42–53, 2007.

[Chen *et al.*, 2016] Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, and Xian-Sheng Hua. Deep ctr prediction in display advertising. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 811–820. ACM, 2016.

[Deng *et al.*, 2017] Yue Deng, Yilin Shen, Hongxia Jin, Yue Deng, Yilin Shen, Hongxia Jin, Yue Deng, Yilin Shen, and Hongxia Jin. Disguise adversarial networks for click-through rate prediction. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1589–1595, 2017.

[Edizel *et al.*, 2017] Bora Edizel, Amin Mantrach, and Xiao Bai. Deep character-level click-through rate prediction for sponsored search. *arXiv preprint arXiv:1707.02158*, 2017.

[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. *arXiv preprint arXiv:1703.04247*, 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 *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Juan *et al.*, 2016] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014.

[Li *et al.*, 2015] Siqin Li, Lei Lin, and Chengjie Sun. Click-through rate prediction for search advertising based on convolution neural network. *Intelligent Computer and Applications*, 5:007, 2015.

[Liu *et al.*, 2015] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. A convolutional click prediction model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1743–1746. ACM, 2015.

[Liu *et al.*, 2017] Xun Liu, Wei Xue, Lei Xiao, and Bo Zhang. Pbodl: Parallel bayesian online deep learning for click-through rate prediction in tencent advertising system. *arXiv preprint arXiv:1707.00802*, 2017.

[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 *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[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 *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 255–262. ACM, 2016.

[Wang *et al.*, 2016] Peng Wang, Bo Xu, Jiaming Xu, Guanhua Tian, Cheng Lin Liu, and Hongwei Hao. Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. *Neurocomputing*, 174(PB):806–814, 2016.

[Wang *et al.*, 2017a] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2915–2921, 2017.

[Wang *et al.*, 2017b] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. *arXiv preprint arXiv:1708.05123*, 2017.

[Yan *et al.*, 2014] Ling Yan, Wujun Li, Guirong Xue, and Dingyi Han. Coupled group lasso for web-scale ctr prediction in display advertising. In *International Conference on Machine Learning*, pages 802–810, 2014.

[Yang *et al.*, 2017] Jufeng Yang, Dongyu She, and Ming Sun. Joint image emotion classification and distribution learning via deep convolutional neural network. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3266–3272, 2017.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann Lecun. Character-level convolutional networks for text classification. pages 649–657, 2015.

[Zhou *et al.*, 2017] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. 2017.