Personalized Recommendation Systems with User-dependent Gating

Zhiyu Yuan[1,2], Kai Ren[1*], Chao Zhang[3], Hongbo Ao[1], Wanhong Xu[1], Xin Miao[2*]

[1]Kuaishou Technology
[2]School of Software, Tsinghua University
[3]Machine Intelligence Laboratory, Engineering Department, University of Cambridge
yzy18@tsinghua.org.cn, kair@alumni.cmu.edu, cz277@eng.cam.ac.uk, miaoxin@tsinghua.edu.cn
aohongbo@kuaishou.com, wanhong.xu@gmail.com

Abstract

Making personalized recommendations is an important research topic, often achieved by leveraging user embeddings as extra input features to neural network (NN) models. However, given a model with fixed number of parameters, a recommendation system often struggles to model the diversified personal interests and behaviors given an increasingly large number of users. In this paper, we propose a user-dependent gating (UDG) mechanism, which can improve the representational ability towards user-specific interests of an NN-based recommendation system. Specifically, a small gating network is introduced to generate gating vectors based on the current user, item, and the user's click history. Such gating vectors are used as dynamic factors applied to scale the static NN model parameters, creating a set of personalized model parameters for each click of a user towards an item. Regarding new users unseen during training, a meta-learning-based method is developed to provide better initial values for the user embeddings. Experiments were conducted using public datasets as well as data collected from a commercial mobile video app with over 100 million daily active users, which shows that UDG can provide considerable and consistent improvements for personalized recommendation.

1 Introduction

In recent years, recommendation systems have drawn much attention from both academia and industry (Resnick and Varian 1997; Lu et al. 2015; Zhang et al. 2019; Richardson, Dominowska, and Ragno 2007; Rendle 2010; Juan et al. 2016; Xiao et al. 2017; He et al. 2014; McMahan et al. 2013; Gai et al. 2017). A recommendation system aims to provide users with the most relevant items to maximize their engagement. For example, in Internet video applications (Davidson et al. 2010; Covington, Adams, and Sargin 2016), the recommendation system selects the most attractive videos from thousands of different categories based on users' historical behaviors. The performance of the recommendation system is critical to service providers since it has a significant impact on user experience and marketing metrics.

With the advent of deep learning, neural network (NN) models are widely used for recommendation systems for click-through rate (CTR) prediction (Covington, Adams, and Sargin 2016; Cheng et al. 2016; Shan et al. 2016; Zhai et al. 2016; Zhou et al. 2018b, 2019). Although these NN-based methods are effective, it is still challenging to make a good balance between the model representational ability, which means the ability to model the diversified user interests and behaviors, and its number of parameters, which determines the test-time computational cost that is critical for large-scale commercial systems with millions of users (Gupta et al. 2020).

In this paper, we propose an user-dependent gating (UDG) mechanism to improve the model representational ability of an NN recommendation system given a certain amount of model parameters. Instead of using the same parameters for all users and items as in conventional methods, UDG creates personalized sets of model parameters specific for each user, item, and the relevant click history by transforming the static model parameters in a dynamic way. This is achieved by dynamically scaling the hidden activation function output values of a *base model* using gating vectors produced by a small gating network, whose input include user features, item features, and behavior features and is thus referred to as an *UDG network*. UDG serves as an auxiliary network that can be easily applied to existing base models for CTR prediction, such as PNN (Qu et al. 2016) and DIN (Zhou et al. 2018b). Regarding real-world large-scale recommendation systems, a large number of new users can inflow continuously during the model test-time, which creates a challenge for the conventional methods that uses a fixed amount of static model parameters. In contrast, UDG estimates a distinct model for each user, item, and click, and thus enables the use of an unlimited number of dynamic parameters to handle the unknown number of new users. Furthermore, UDG suffers from a *cold-start problem*, which refers to the problem that the CTR prediction accuracy for new users and new items encountered at test-time without sufficient past clicks can be worse than average. To resolve this issue, we also propose a meta-learning-based method that can estimate better initial values for the embeddings of new users using another small network. Offline recommendation experiments on three public datasets were conducted to evaluate our methods. Results show that UDG improves CTR prediction accuracy. We also tested UDG on a private dataset collected from a mobile video app with over 100 million daily active users, and the engagement metrics are considerably improved by 1.8% and 2.0%.

*These authors contributed equally.

The rest of the paper is organized as follows. The related work is discussed in Sec. 2. Our proposed method is presented in Sec. 3. Experimental setup and results are given in Sec. 4. We conclude in Sec. 5.

2 Related Work

NN-based models have been widely used for CTR prediction (Zhang et al. 2014; Hidasi et al. 2015; Yu et al. 2016; Song, Elkahky, and He 2016; Zhou et al. 2018a; Parsana et al. 2018; Zhang et al. 2014; Hidasi et al. 2015; Yu et al. 2016; He and Chua 2017; Yang et al. 2020). Most of methods use a feedforward or recurrent NN structure, such as multi-layer perceptron (MLP) and gated recurrent unit (GRU) (Cho et al. 2014), with user and item embedding as input features, such as FM (Rendle 2012), PNN (Qu et al. 2016), W&D (Cheng et al. 2016), DeepFM (Guo et al. 2017), DIN (Zhou et al. 2018b) and DIEN (Zhou et al. 2019).

The gating technique has been widely used in a number of well-known recurrent (Hochreiter and Schmidhuber 1997; Cho et al. 2014) and feedforward (Srivastava, Greff, and Schmidhuber 2015; Bradbury et al. 2017) model structures, which allows a target vector to be dynamically scaled by the gating vector in an element-by-element way. In recommendation systems, gating is used in multi-task training (Ma et al. 2018) and representation learning (Xia et al. 2019). DIEN modifies the GRU structure with an extra gating unit to strengthen the influences from relevant user interests which is referred to as AUGRU (Zhou et al. 2019).

3 Our Approach

Our base model structure and the proposed UDG method are first presented. Then the proposed meta-learning-based technique to avoid the cold-start problem is also introduced.

Base Model

The base model is presented from three aspects: input features, model structure, and loss function.

Input Features: The input features used in our approach include user feature, item feature and behavior feature:

- **user feature** is a concatenation of the one-hot vector of the user identity (UID) and user profile (if exists);
- **item feature** is a concatenation of two one-hot vectors of the item identity and its relevant category;
- **behavior feature** is a multi-hot vector of the items that an user clicked in the past. The elements relevant to the clicked item and their categories are set to one. This feature is the key evidence to reveal the user interests.

These one-hot or multi-hot vectors are used as the input values to the base model embedding layer to derive different types of the embeddings, as shown in Fig. 1.

Model Structure: Our base model is shown in the left part of Fig. 1, which consists of an embedding layer, a middle layer, and a final MLP serving as the prediction network.

- **Embedding Layer:** Let $\boldsymbol{x}$ be an multi-hot vector, $\boldsymbol{E}$ be the embedding table related to $\boldsymbol{x}$. The feature embedding $\boldsymbol{e}$ can be obtained by

$$\boldsymbol{e} = \boldsymbol{E}\,\boldsymbol{x}, \tag{1}$$

  which is the table lookup mechanism to extract the columns corresponding to the one values in $\boldsymbol{x}$ from $\boldsymbol{E}$ using the matrix-vector product. When $\boldsymbol{x}$ is the user feature, item feature, and behavior feature, the resulted user embedding, item embedding, and behavior embedding are denoted as $\boldsymbol{e}_{u,t}^{\text{user}}$, $\boldsymbol{e}_{i}^{\text{item}}$, and $\boldsymbol{e}_{t}^{\text{behv}}$ respectively, where u, i and t are the indexes of the current user, item, and click.
- **Middle Layer:** Base model like PNN and DeepFM do not have the middle layer. DIN and DIEN use the middle layer to improve the model representational ability. The output of the middle layer, $\boldsymbol{e}_{i,t}^{\text{mid}}$, is obtained by

$$\boldsymbol{e}_{i,t}^{\text{mid}} = f_{\text{mid}}\left(\text{Concat}\left(\boldsymbol{e}_{i}^{\text{item}}, \boldsymbol{e}_{t}^{\text{behv}}\right)\right), \tag{2}$$

  where $f_{\text{mid}}(\cdot)$ is the mapping function implemented by the middle layer and $\text{Concat}(\cdot)$ refers to concatenate multiple vectors into one.
- **Final MLP:** The input vector to the first layer of the MLP at time t for user u and item i, $\boldsymbol{h}_{u,i,t}^{(0)}$, is formed by

$$\boldsymbol{h}_{u,i,t}^{(0)} = \text{Concat}\left(\boldsymbol{e}_{i,t}^{\text{mid}}, \boldsymbol{e}_{i}^{\text{item}}, \boldsymbol{e}_{u,t}^{\text{user}}\right). \tag{3}$$

  $\boldsymbol{h}_{u,i,t}^{(0)}$ is then transformed by multiple fully-connected (FC) layers in the MLP. Each FC layer has a hidden activation function, such as the ReLU function. The output vector of the l th FC layer, $\boldsymbol{h}_{u,i,t}^{(l)}$, can be derived by

$$\boldsymbol{h}_{u,i,t}^{(l)} = \text{ReLU}\left(\boldsymbol{W}^{(l)}\,\boldsymbol{h}_{u,i,t}^{(l-1)} + \boldsymbol{b}^{(l)}\right), \tag{4}$$

  where $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are the weight matrix and bias vector associated with the l th FC layer. The final layer of the MLP uses the softmax activation function with two output targets to estimate, $p_{u,i,t}$ and $1 - p_{u,i,t}$, which are the probabilities of user u clicking item i at time t or not.

Loss Function: The cross-entropy (CE) loss function is used to train a CTR prediction model, which is defined as

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log p_n + (1 - y_n)\log(1 - p_n)\right], \tag{5}$$

where y_n and p_n are the label and predicted probability that the item associated with the n th label is clicked, and N is the number of samples.

User-Dependent Gating

Analogous to an array of a logic gate in electronics, the gating mechanism converts each input vector to a 0-1 valued gating vector using the gating network, whose elements are used to scale the corresponding elements of a target vector individually by the Hadamard product. Let $\boldsymbol{e}_{u,t}^{\text{UDG}}$ be the user embedding specific for UDG obtained with another user embedding table $\boldsymbol{E}^{\text{UDG}}$ based on UID and Eqn. (1), the input

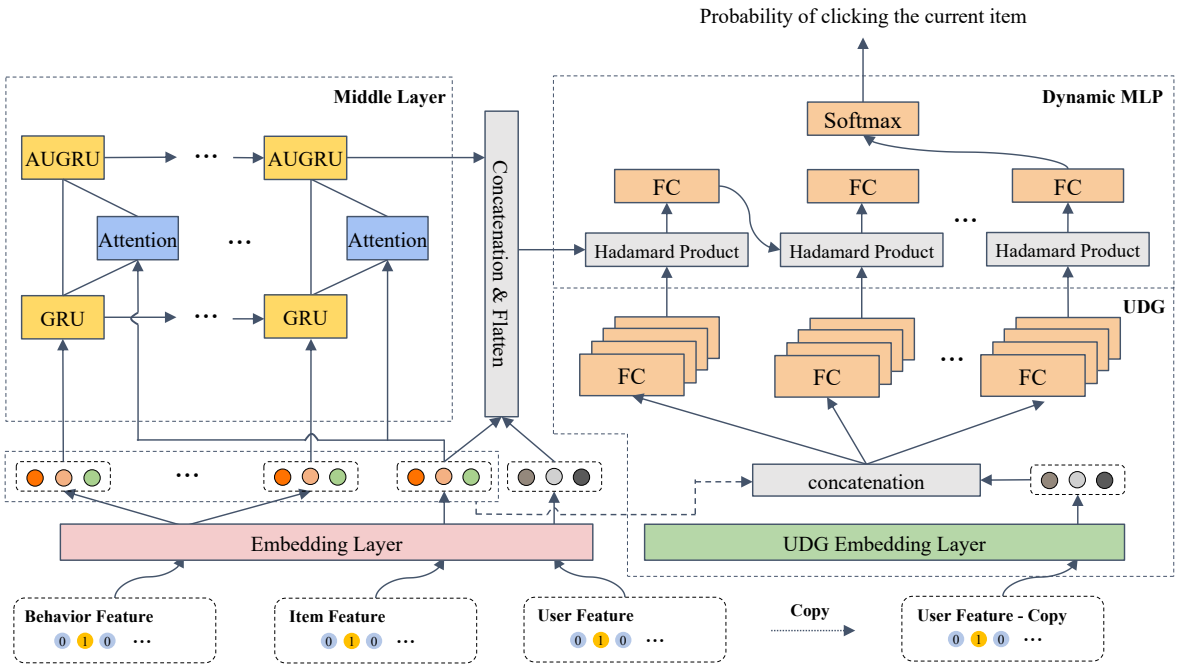


Figure 1: The sketch map of applying UDG to transform a static base model into distinct dynamic models. The base model is divided into the embedding layer, middle layer, and MLP. UDG takes each UID encoding as input to generate the gating vectors, which are used to scale the output vector derived from its relevant base model hidden layer using the Hadamard product.

vector used by each gating network is set to be the concatenation of $\boldsymbol{e}_i^{\text{item}}$, $\boldsymbol{e}_t^{\text{behv}}$ and $\boldsymbol{e}_{u,t}^{\text{UDG}}$. Then, the concatenation is fed into the l-th gating network to generate the gating vector $\boldsymbol{g}_{u,i,t}^{(l)}$, which is defined as

$$\boldsymbol{g}_{u,i,t}^{(l)} = \text{GN}^{(l)}(\text{Concat}(e_i^{\text{item}}, e_t^{\text{behv}}, e_{u,t}^{\text{UDG}})) \tag{6}$$

where $\text{GN}^{(l)}$ is the l-th gating network consisting of three fully connected layers (as shown in Fig. 1). The gating network *w.r.t.* each base model hidden layer l is a separate MLP with static parameters defined also with Eqn. (4), whose output layer size is $d^{(l)}$ that is also the size of the l-th layer of the base model. Sigmoid is used as the gating network output function. Similar to the user embedding table of the base model, $\boldsymbol{E}^{\text{UDG}}$ is also updated during training and is shared across all gating networks.

Since the gating vector $\boldsymbol{g}_{u,i,t}^{(l)}$ is associated with $\boldsymbol{h}_{u,i,t}^{(l)}$, Eqn. (4) can be modified as

$$\boldsymbol{h}_{u,i,t}^{(l)} = \text{ReLU}\left(\boldsymbol{W}^{(l)}\left(\boldsymbol{g}_{u,i,t}^{(l-1)} \odot \boldsymbol{h}_{u,i,t}^{(l-1)}\right) + \boldsymbol{b}^{(l)}\right), \tag{7}$$

where $\odot$ is the Hadamard product. Since $d^{(l)}$ is the size of the l th layer and also the row number of $\boldsymbol{W}^{(l)}$, by defining

$$\boldsymbol{W}_{u,i,t}^{(l)} = \boldsymbol{W}^{(l)} \odot \left[\left(\boldsymbol{g}_{u,i,t}^{(l-1)}\right)_{\times d^{(l)}}\right]^{\text{T}}, \tag{8}$$

Eqn. (7) can be re-written as

$$\boldsymbol{h}_{u,i,t}^{(l)} = \text{ReLU}\left(\boldsymbol{W}_{u,i,t}^{(l)}\,\boldsymbol{h}_{u,i,t}^{(l-1)} + \boldsymbol{b}^{(l)}\right). \tag{9}$$

Eqn. (9) has the same form as the MLP defined in Eqn. (4), apart from the use of dynamic weight matrices, $\boldsymbol{W}_{u,i,t}^{(l)}$, instead of the static ones, $\boldsymbol{W}^{(l)}$. This allows to use a distinct MLP regarding each user, item, and click, which increases the model representational ability.

Meta-Learning to Initialize User Embeddings

In CTR prediction, recommendation systems often suffers from the cold-start problem. It can be alleviated by the Meta-Embedding approach (Pan et al. 2019), which uses meta-learning (Finn, Abbeel, and Levine 2017) to learn to provide more sensible initial values of the item embeddings. In this paper, we adapt Meta-Embedding to handle the cold-start problem of new users, which applies meta-learning to provide sensible initial UDG user embedding values at test-time for the users unseen in the training set. The output values derived from the embedding generator are used as the initial UDG user embedding values for a new user when taking his/her user profile x_u^{prof} as the input. The objectives of using meta-learning to alleviate the cold-start problem of new users include:

- Cold-start: To improve the initial CTR prediction accuracy for a new user without any behavior feature;
- Warm-up: To improve the CTR prediction accuracy of a new user efficiently with only a few past clicks, which requires fast training convergence speed.

To achieve these objectives, the training procedure is divided into the cold-start phase (phase A) and warm-up phase

Algorithm 1: Training the UDG user embedding generator

Require: One-hot encoding $\boldsymbol{x}_u^{\text{prof}}$
Require: Disjoint sub-training sets $\mathcal{D}^{\text{A}}$ and $\mathcal{D}^{\text{B}}$
Require: Embedding generator $f_{\text{meta}}(\boldsymbol{x}_u^{\text{prof}}, \boldsymbol{\Theta})$
Require: Step size α and β for A and B respectively
Require: Minibatch size N for both A and B
Require: Interpolation coefficient of the loss functions γ

Initialize $\boldsymbol{\Theta}$ randomly
while not finished **do**
  Initialize UDG embedding $\hat{\boldsymbol{e}}_{u,t}^{\text{UDG}} = f_{\text{meta}}(\boldsymbol{x}_u^{\text{prof}}, \boldsymbol{\Theta}_t)$
  Sample a minibatch with N data points, $\mathcal{D}^{\text{a}}$, from $\mathcal{D}^{\text{A}}$
  Compute $\mathcal{L}_{\text{CE}}^{\text{A}}$ using $\hat{\boldsymbol{e}}_{u,t}^{\text{UDG}}$ and the data samples $\mathcal{D}^{\text{a}}$
  Update embedding $\boldsymbol{e}_{u,t}^{\text{UDG}} = \hat{\boldsymbol{e}}_{u,t}^{\text{UDG}} - \alpha \cdot \partial\, \mathcal{L}_{\text{CE}}^{\text{A}} / \partial\, \hat{\boldsymbol{e}}_{u,t}^{\text{UDG}}$
  Sample a minibatch with N data points, $\mathcal{D}^{\text{b}}$, from $\mathcal{D}^{\text{B}}$
  Compute $\mathcal{L}_{\text{CE}}^{\text{B}}$ using $\boldsymbol{e}_{u,t}^{\text{UDG}}$ and $\mathcal{D}^{\text{b}}$
  Compute loss $\mathcal{L}_{\text{meta}} = \gamma\, \mathcal{L}_{\text{CE}}^{\text{A}} + (1-\gamma)\, \mathcal{L}_{\text{CE}}^{\text{B}}$
  Update $\boldsymbol{\Theta}_{t+1} \leftarrow \boldsymbol{\Theta}_t - \beta \cdot \partial\, \mathcal{L}_{\text{meta}} / \partial\, \boldsymbol{\Theta}$
  Update $t \leftarrow t + 1$
end while

(phase B) in the Meta-Embedding approach, which use $\mathcal{D}^{\text{A}}$ and $\mathcal{D}^{\text{B}}$ as the training data of each phase separately. $\mathcal{D}^{\text{A}}$ and $\mathcal{D}^{\text{B}}$ are obtained by dividing the training set equally into two subsets to cover the same number of users. The cold-start phase performs training based on the predicted probabilities estimated using the UDG user embedding derived from the embedding generator network for new users with zero behavior features, whose loss function is denoted as $\mathcal{L}_{\text{CE}}^{\text{A}}$. The warm-up phase performs another training process based on the UDG user embedding obtained by updating the generator network to simulate the learning process with a few clicks, whose loss function is denoted as $\mathcal{L}_{\text{CE}}^{\text{B}}$. The two loss functions are interpolated as a joint loss function by

$$\mathcal{L}_{\text{meta}} = \gamma\, \mathcal{L}_{\text{CE}}^{\text{A}} + (1-\gamma)\, \mathcal{L}_{\text{CE}}^{\text{B}}, \tag{10}$$

where γ is the interpolation coefficient set to 0.1 in this paper. Both cold-start and warm-up objectives are optimised by the training.

The detailed procedure of training a UDG user embedding generator network $f_{\text{meta}}(\boldsymbol{x}_u^{\text{prof}}, \boldsymbol{\Theta})$ is presented in Algorithm 1, where $\boldsymbol{x}_u^{\text{prof}}$ is the one-hot vector of the profile of user u and $\boldsymbol{\Theta}$ is its trainable parameters. It is remarkable that when training the UDG user embedding generator on the training set based on the user profiles, the parameters of the base model and the gating networks are kept frozen since they have been pre-trained on the training set. A simple network structure for the embedding generator is used in this paper, which has one FC layer with the hyperbolic tangent activation function (tanh). That is

$$f_{\text{meta}}(\boldsymbol{x}_u^{\text{prof}}, \boldsymbol{\Theta}) = \tanh\left(\boldsymbol{W}^{\text{meta}}\, \boldsymbol{x}_u^{\text{prof}} + \boldsymbol{b}^{\text{meta}}\right), \tag{11}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{W}^{\text{meta}}, \boldsymbol{b}^{\text{meta}}\}$, and $\boldsymbol{W}^{\text{meta}}$ and $\boldsymbol{b}^{\text{meta}}$ are the weight matrix and bias vector of the FC layer.

4 Experiments

In this section, we first conduct offline experiments on three public datasets. Then apply UDG to private data collected from a large-scale commercial mobile video app.

Dataset Name		No. Users	No. Records
Public	MovieLens	26,083	573,534
	Amazon	33,658	236,892
	Alimama	114,465	513,877
Industry	Explore Feed	45M	50M
	Follow Feed	100M	200M

Table 1: The size of public datasets used in the offline experiments and industry datasets used in online evaluation.

Data and Models

Three public datasets are used: Amazon Electronics[1], MovieLens[2], and Alimama[3]. Table 1 summarizes the characteristics of the three datasets. The Amazon dataset contains product reviews and metadata from Amazon. We choose the Electronics subset in our experiments. The MovieLens dataset consists of 27 million movie ratings ranging from 1 to 5 with 0.5 increments. We label the samples with a rating above 3 to be positive and the rest to be negative and train it as a binary classification model. The Alimama dataset contains rich user profiles and user behaviors from Taobao ads display logs. The two industry datasets are collected from a popular mobile video app, including two recommendation services: Explore Feed and Follow Feed.

UDG is applied to the following CTR prediction models:

- **PNN** (Qu et al. 2016) introduces a product layer to capture high-order features and interactive patterns.
- **DeepFM** (Guo et al. 2017) performs end-to-end learning to obtain high-order features without the need to manually combining FM derived features.
- **W&D** (Cheng et al. 2016) learns both low-order and high-order features simultaneously by combining a wide linear model with a deep model.
- **DIN** (Zhou et al. 2018b) learns the representation of user interests adaptively from the users' historical behaviors w.r.t.given items.
- **DIEN** (Zhou et al. 2019) uses AUGRU to model the interest evolving processes, which leads to more expressive user interest representation.

Metrics

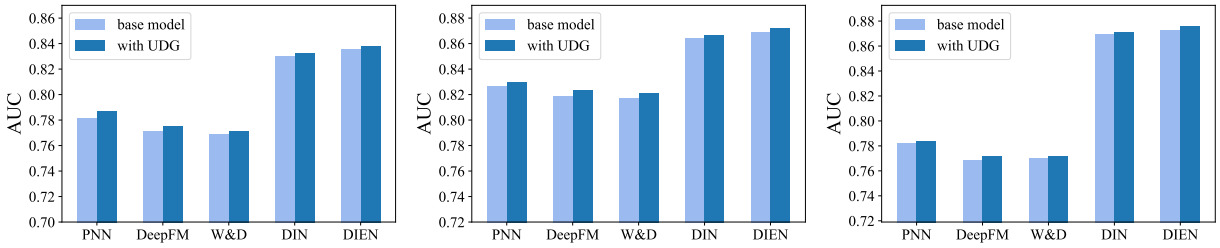
Four metrics are used to evaluate the model performance:

- **AUC:** Area Under Curve is defined as the area under the receiver operating characteristic curve. For CTR prediction, AUC is used to evaluate the rank order.
- **RIG:** Relative Information Gain measures the goodness of the output clicking probability, which makes up for the shortcomings of AUC (He et al. 2014; Ling et al. 2017).
- **Log Loss:** Log Loss is the output value from the CE loss function, as formulated in Eqn. (5).

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] https://grouplens.org/datasets/movielens/
[3] https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

Model	Amazon				MovieLens				Alimama			
	Log Loss	AUC	RMSE	RIG	Log Loss	AUC	RMSE	RIG	Log Loss	AUC	RMSE	RIG
PNN	0.5507	0.7811	0.4319	0.2055	0.5088	0.8264	0.4129	0.2660	0.5604	0.7822	0.4362	0.1915
PNN+UDG	**0.5475**	**0.7872**	**0.4310**	**0.2102**	**0.5080**	**0.8294**	**0.4121**	**0.2672**	**0.5582**	**0.7841**	**0.4351**	**0.1948**
DeepFM	0.5587	0.7709	0.4357	0.1940	0.5226	0.8190	0.4182	0.2460	0.5742	0.7690	0.4425	0.1716
DeepFM+UDG	**0.5572**	**0.7752**	**0.4347**	**0.1946**	**0.5219**	**0.8230**	**0.4177**	**0.2470**	**0.5724**	**0.7718**	**0.4416**	**0.1742**
W&D	0.5589	0.7685	0.4359	0.1937	0.5288	0.8172	0.4199	0.2372	0.5730	0.7700	0.4420	0.1734
W&D+UDG	**0.5574**	**0.7716**	**0.4352**	**0.1950**	**0.5273**	**0.8210**	**0.4182**	**0.2380**	**0.5702**	**0.7720**	**0.4415**	**0.1747**
DIN	0.5043	0.8302	0.4090	0.2725	0.4874	0.8643	0.3865	0.3045	0.4491	0.8694	0.3823	0.3523
DIN+UDG	**0.5023**	**0.8323**	**0.4082**	**0.2753**	**0.4861**	**0.8663**	**0.3856**	**0.3076**	**0.4483**	**0.8711**	**0.3818**	**0.3535**
DIEN	0.4993	0.8357	0.4060	0.2791	0.4850	0.8689	0.3839	0.3112	0.4449	0.8726	0.3808	0.3581
DIEN+UDG	**0.4982**	**0.8382**	**0.4052**	**0.2797**	**0.4838**	**0.8724**	**0.3817**	**0.3124**	**0.4439**	**0.8757**	**0.3802**	**0.3589**

Table 2: The different metric results of models with or without UDG on Amazon Electronics, MovieLens and Alimama dataset. UDG oviously improve the prediction accuracy on different models.



(a) Amazon Dataset (b) MovieLens Dataset (c) Alimama Dataset

Figure 2: The AUC metric results of models with or without UDG on Amazon Electronics, MovieLens and Alimama dataset.

- **RMSE:** Root Mean Square Error is the standard deviation of the prediction errors.

Results on Public Datasets

Results on the three public datasets are given in this section. All experiments were run for 5 times, and the mean values of all the runs are reported.

Experiments with different numbers of parameters: To demonstrate the effectiveness of UDG, experiments were conducted to compare the performance with or without UDG. As shown in Table 2, all models with UDG outperform the base model on all three datasets. For example, benefiting from the personalized MLP parameters, the results show that AUC increases by 0.25% to 0.78% and RIG by 0.21% to 2.28% in Amazon dataset. This improvement is evident and significant because a slight improvement in offline AUC will likely lead to a considerable increase in online CTR, bringing extra millions of dollars each year for a large-scale system (Gupta et al. 2020).

Experiments with the same number of parameters: Experiments with all systems having the same number of parameters were conducted to show that the benefits of using UDG does not come from a simple increase of the number of parameters. DIEN with UDG was compared against the modified DIEN whose embedding table size is twice as large as the original one, so that its number of parameters is even more than the system of DIEN with UDG. From the results given in Table 3, it is obvious that the models UDG performed much better than those with even twice as much parameters as in the base model.

Illustration of the effectiveness of using UDG: Experiments to visualize the effectiveness of using UDG is conducted with t-distributed stochastic neighbor embedding (t-SNE) (Maaten and Hinton 2008). The models used are PNNs with and without UDG trained on Alimama dataset. Two users, A and B, are selected from the test set to form the input to both models. Eighty-dimensional features are derived as the output vectors from the penultimate layer, which are projected to a two-dimensional space and shown shown in the Fig. 3. From the figure, both models can map their high-dimensional input features into two clusters (blue and orange clusters mean dislike and like respectively). The clustering results of the PNN+UDG model show better performance: the distance between the clusters is obviously larger and fewer points are assigned to wrong clusters. This gives an intuitive explanation that UDG extracts more discriminative features with the dynamic models.

Online Evaluation in Production System

Online evaluation is performed using a large-scale production system deployed to one of the largest mobile video apps in the world. UDG was mainly evaluated in two recommendation services: *Follow Feed* and *Explore Feed*. Follow Feed recommends the most attractive videos for users from the

Model	Operation	Amazon		MovieLens		Alimama	
		#parameters	AUC	#parameters	AUC	#parameters	AUC
PNN	Base model	10.74M	0.7811	14.79M	0.8264	33.86M	0.7822
	Doubling embedding size	21.48M	0.7800	29.57M	0.8259	67.70M	0.7815
	With UDG	15.54M	**0.7872**	18.61M	**0.8294**	49.93M	**0.7841**
DeepFM	Base model	10.80M	0.7709	14.80M	0.8190	34.07M	0.7690
	Doubling embedding size	21.51M	0.7695	29.58M	0.8188	67.87M	0.7678
	With UDG	15.61M	**0.7752**	18.65M	**0.8230**	50.16M	**0.7718**
W&D	Base model	10.83M	0.7685	14.83M	0.8172	34.12M	0.7700
	Doubling embedding size	21.56M	0.7680	29.64M	0.8170	67.97M	0.7665
	With UDG	15.64M	**0.7716**	18,65M	**0.8210**	50.20M	**0.7720**
DIN	Base model	10.94M	0.8302	14.95M	0.8643	34.18M	0.8694
	Doubling embedding size	21.85M	0.8295	29.83M	0.8635	68.08M	0.8720
	With UDG	17.25M	**0.8323**	18.67M	**0.8663**	52.19M	**0.8711**
DIEN	Base model	11.65M	0.8357	15.58M	0.8689	34.57M	0.8726
	Doubling embedding size	23.28M	0.8346	31.13M	0.8677	69.12M	0.8711
	With UDG	17.48M	**0.8382**	18.83M	**0.8724**	52.66M	**0.8757**

Table 3: The experimental result of increasing the number of model parameters, which proves that UDG gains from its design mechanism rather than additional parameters to achieve performance improvement.

	Follow Feed	Explore Feed
Offline AUC	+0.3%	+0.18%
Online AUC	+0.4%	+0.21%
Engagement Metric	+1.8%	+2.0%

Table 4: Production Evaluation Results

content they have followed. Explore Feed helps users discover more hot videos. Models used in both services are NN-based and trained by using billions of data samples everyday. The target audience is divided into experimental and control groups, each of them has ∼50 million users.

Table 4 summarizes the relative improvements brought by the use of UDG on the two services. It mainly reports three evaluation results: offline AUC, online AUC, and the engagement metric of online A/B preferred testing. The offline and online AUC measures the model performance tested over the offline datasets and online real-time datasets respectively. The engagement metric is a a linear combination of click-through rate (CTR), effective view rate (EVR), like rate (LR) and follow rate (FR), which reflects users' engagement activities defined as

$$\text{Engagement Metric} = \alpha\,\text{CTR} + \beta\,\text{EVR} + \gamma\,\text{LR} + \tau\,\text{FR},$$

where $\alpha, \beta, \gamma, \tau$ are pre-defined coefficients.

The increase of engagement metric indicates that users tend to click or watch videos more often and have more positive feedback to perform actions such as likes and follows. From Table 4, we can see the use of UDG improves all metrics compared to the original base models. Especially for the engagement metric, 1.8% and 2.0% improvements are considered to be very significant.

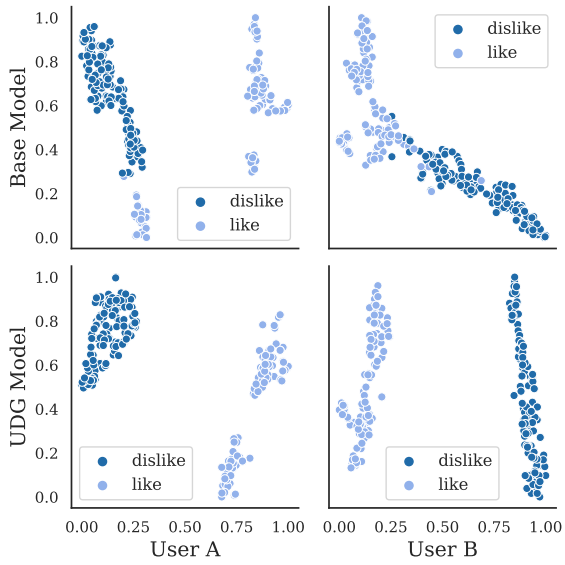


Figure 3: t-SNE visualization on 80-dimensional data. Different colors of every cluster represent users like or dislike the item.

Meta-Learning for Cold-start

Next, we study the use of meta-learning to alleviate the cold-start problem for users when UDG is used.

Experimental setup: The Alimama dataset is used here since it has rich user profile data, which is similar to real-world production system with rich user profile data. To compare meta-learning against the random initialization of user embedding fairly, we divide the labeled dataset into two sub-

Model	Target	Cold-Start phase		Warm-Up phase: a		Warm-Up phase: b		Warm-Up phase: c	
		w/o Meta	Meta	w/o Meta	Meta	w/o Meta	Meta	w/o Meta	Meta
PNN	AUC	0.00%	+0.46%	+5.29%	+6.93%	+9.22%	+11.47%	+11.68%	+14.07%
	Log Loss	0.00%	-0.02%	-0.59%	-0.72%	-1.07%	-1.27%	-1.41%	-1.65%
DeepFM	AUC	0.00%	+0.58%	+7.27%	+9.17%	+11.02%	+13.84%	+13.34%	+16.10%
	Log Loss	0.00%	-0.00%	-0.87%	-1.04%	-1.38%	-1.75%	-1.75%	-2.16%

Table 5: UDG performance with meta-learning (Meta) or without meta-learning (w/o Meta) in cold-start phase and warm-up phase. All the result columns calculate AUC and Log Loss by comparing with the first result column.

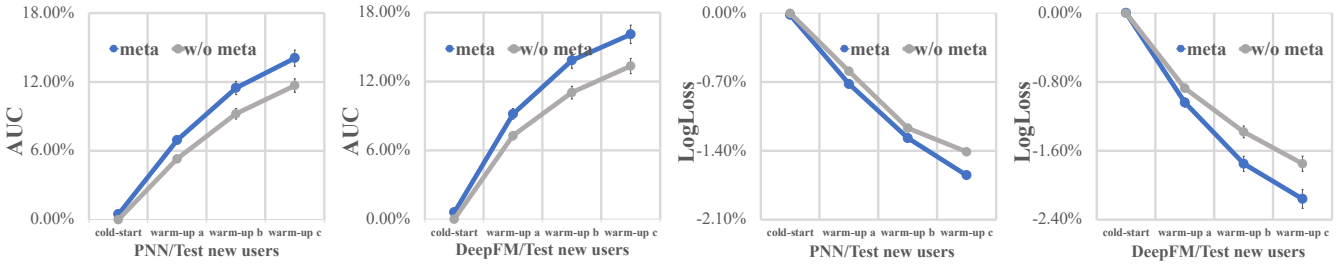


Figure 4: PNN+UDG results with meta-learning (meta) or without meta-learning (w/o meta) in cold-start and warm-up phases.

sets containing disjoint sets of users: *Active Users* and *New Users*. *Active Users* are users whose number of training instance is larger than 100 in the Alimama dataset. *New Users* are those whose number of training instances is larger than 80 but smaller than 100. For New Users, we divide their labeled instance into four batches sorted by time: batch-a, batch-b, batch-c, and the held-out batch. The number of instances in the four batches for each new user i are K, K, K, and L_i (where $K = 20$ and $L_i \geq 20$). The held-out batches of all new users form our held-out test set in this experiment.

The experiment process consists of four phases: *cold-start phase*, and *warm phases* a to c. The four phases are used to demonstrate that meta-embeddings not only give a better prediction for the first shot, but also help to learn embedding faster in the incoming data. In the cold-start phase, we mainly use the dataset of Active Users to pre-train the base models. The pre-trained models are tested with randomly initialized user embedding over the held-out test set. The embedding generators are also trained with the dataset of active users. Their performance is also tested over the hold-out test set by using meta-embedding for initial user embedding.

In the following three warm-up phases from a to c, we continue to update these models and their user embeddings over the New User dataset. For example, in the warm-up phase a, we update the models using all the batch-a data and replace every new user's embedding with those generated by the embedding generator. In the rest two warm-up phases, the models are updated as described above by using corresponding data from the New Users dataset. After each warm-up phase, the updated models are tested over the held-out test set.

Experimental results & analysis: The main results are summarized in Table 5 and Figure 4. For the cold phase, these user ID embedding still maintain a random initialization state and have not been updated. Therefore, the results are expected to be better when they are initialized by using meta-embedding. For example, AUC increased by 0.58% compared to the baseline models for DeepFM when we used embedding generator to initialize embedding. In the subsequent three warm-up phases, we observe that after using meta-embedding, the performance not only has been consistently better than random initialization, but also the relative improvement has gradually increased. This demonstrates that we can get better performance and faster training convergence speed by using meta-embedding.

5 Conclusion

In this paper, we propose the UDG mechanism to improve the performance of NN-based personalized recommendation systems. To improve the model representational ability regarding user interests, UDG creates dynamic sets of model parameters distinct for each user, each item and each click. This is achieved by scaling the hidden activation function output values of CTR prediction network with the gating vectors produced by a small UDG gating network, whose input include user, item and behavior features. Experimental results on three public datasets show that UDG can be applied to 5 mainstream CTR models to improve their performance with little changes. Furthermore, UDG is productionized in a large-scale commercial recommendation system, and the online evaluation results show that the engagement metrics are improved by 1.8% and 2.0%. A meta-learning-based technique is also developed to provide better initial values for new users that are unseen during training.

References

Bradbury, J.; Merity, S.; Xiong, C.; and Socher, R. 2017. Quasi-recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 1–12.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, 7–10.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, 191–198.

Davidson, J.; Liebald, B.; Liu, J.; Nandy, P.; Van Vleet, T.; Gargi, U.; Gupta, S.; He, Y.; Lambert, M.; Livingston, B.; et al. 2010. The YouTube video recommendation system. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*, 293–296.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic MetaLearning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 1126–1135.

Gai, K.; Zhu, X.; Li, H.; Liu, K.; and Wang, Z. 2017. Learning piece-wise linear models from large scale data for ad click prediction. In *arXiv preprint arXiv:1704.05194*, 1–12.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. 1725–1731.

Gupta, U.; Wu, C.-J.; Wang, X.; Naumov, M.; Reagen, B.; Brooks, D.; Cottel, B.; Hazelwood, K.; Hempstead, M.; Jia, B.; et al. 2020. The architectural implications of facebook's dnn-based personalized recommendation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 488–501. IEEE.

He, X.; and Chua, T.-S. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 355–364.

He, X.; Pan, J.; Jin, O.; Xu, T.; Liu, B.; Xu, T.; Shi, Y.; Atallah, A.; Herbrich, R.; Bowers, S.; et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the 8th International Workshop on Data Mining for Online Advertising (ADKDD)*, 1–9.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2015. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 1–10.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*, 9: 1735–1780.

Juan, Y.; Zhuang, Y.; Chin, W.-S.; and Lin, C.-J. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, 43–50.

Ling, X.; Deng, W.; Gu, C.; Zhou, H.; Li, C.; and Sun, F. 2017. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 689–698.

Lu, J.; Wu, D.; Mao, M.; Wang, W.; and Zhang, G. 2015. Recommender system application developments: a survey. *Decision Support Systems*, 74: 12–32.

Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1930–1939.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov): 2579–2605.

McMahan, H. B.; Holt, G.; Sculley, D.; Young, M.; Ebner, D.; Grady, J.; Nie, L.; Phillips, T.; Davydov, E.; Golovin, D.; et al. 2013. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1222–1230.

Pan, F.; Li, S.; Ao, X.; Tang, P.; and He, Q. 2019. Warm Up Cold-start Advertisements: Improving CTR Predictions via Learning to Learn ID Embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 695–704.

Parsana, M.; Poola, K.; Wang, Y.; and Wang, Z. 2018. Improving native Ads CTS prediction by large scale event embedding and recurrent networks. In *arXiv preprint arXiv:1804.09133*, 1–10.

Qu, Y.; Cai, H.; Ren, K.; Zhang, W.; Yu, Y.; Wen, Y.; and Wang, J. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 1149–1154.

Rendle, S. 2010. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, 995–1000.

Rendle, S. 2012. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology*, 3: 57:1–57:22.

Resnick, P.; and Varian, H. R. 1997. Recommender systems. *Communications of the ACM*, 40(3): 56–58.

Richardson, M.; Dominowska, E.; and Ragno, R. 2007. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, 521–530.

Shan, Y.; Hoens, T. R.; Jiao, J.; Wang, H.; Yu, D.; and Mao, J. 2016. 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*, 255–262.

Song, Y.; Elkahky, A. M.; and He, X. 2016. Multi-rate deep learning for temporal recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 909–912.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Training very deep networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, 2377–2385.

Xia, H.; Wang, Z.; Du, B.; Zhang, L.; Chen, S.; and Chun, G. 2019. Leveraging Ratings and Reviews with Gating Mechanism for Recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1573–1582.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T.-S. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 3119–3125.

Yang, Y.; Xu, B.; Shen, S.; Shen, F.; and Zhao, J. 2020. Operation-aware neural networks for user response prediction. *Neural Networks*, 121: 161–168.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 729–732.

Zhai, S.; Chang, K.-h.; Zhang, R.; and Zhang, Z. M. 2016. DeepIntent: Learning attentions for online advertising with recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1295–1304.

Zhang, S.; Yao, L.; Sun, A.; and Tay, Y. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1): 1–38.

Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T.; Bian, J.; Wang, B.; and Liu, T.-Y. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *In Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 1369–1375.

Zhou, C.; Bai, J.; Song, J.; Liu, X.; Zhao, Z.; Chen, X.; and Gao, J. 2018a. ATRank: An attention-based user behavior modeling framework for recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 4564–4571.

Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 5941–5948.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018b. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1059–1068.