Warm Up Cold-start Advertisements: Improving CTR Predictions via Learning to Learn ID Embeddings

Feiyang Pan*†
Institute of Computing Technology, Chinese Academy of Sciences

Shuokai Li†
Institute of Computing Technology, Chinese Academy of Sciences

Xiang Ao†
Institute of Computing Technology, Chinese Academy of Sciences

Pingzhong Tang
IIIS, Tsinghua University

Qing He†
Institute of Computing Technology, Chinese Academy of Sciences

ABSTRACT

Click-through rate (CTR) prediction has been one of the most central problems in computational advertising. Lately, embedding techniques that produce low-dimensional representations of ad IDs drastically improve CTR prediction accuracies. However, such learning techniques are data demanding and work poorly on new ads with little logging data, which is known as the cold-start problem.

In this paper, we aim to improve CTR predictions during both the *cold-start* phase and the *warm-up* phase when a new ad is added to the candidate pool. We propose *Meta-Embedding*, a meta-learning-based approach that learns to generate desirable initial embeddings for new ad IDs. The proposed method trains an embedding generator for new ad IDs by making use of previously learned ads through gradient-based meta-learning. In other words, our method learns how to learn better embeddings. When a new ad comes, the trained generator initializes the embedding of its ID by feeding its contents and attributes. Next, the generated embedding can speed up the model fitting during the warm-up phase when a few labeled examples are available, compared to the existing initialization methods.

Experimental results on three real-world datasets showed that Meta-Embedding can significantly improve both the cold-start and warm-up performances for six existing CTR prediction models, ranging from lightweight models such as Factorization Machines to complicated deep models such as PNN and DeepFM. All of the above apply to conversion rate (CVR) predictions as well.

KEYWORDS

CTR Prediction; Online Advertising; Cold-Start; Meta-Learning;

ACM Reference Format:
Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm Up Cold-start Advertisements: Improving CTR Predictions via Learning to Learn ID Embeddings. In *SIGIR'19: The 42nd International ACM SIGIR Conference on Research & Development in Information Retrieval, July 21-25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Corresponding author (panfeiyang@ict.ac.cn).
†At the Key Lab of Intelligent Information Processing of Chinese Academy of Sciences. Also at University of Chinese Academy of Sciences, China.

SIGIR'19, July 21-25, 2019, Paris, France
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The essence of online advertising is for the publisher to allocate ad slots in a way that maximizes social welfare. In other words, an ideal ad auction mechanism allocates, for each user, the best slot to the best ad, the second slot to the second best ad and so on. In order to determine which ad is the most valuable to this user, a key parameter is the so-called *click-through rate* also known as the CTR. A typical ad auction mechanism then allocates according to a descending order of the product of bid times its CTR. Since the bids are inputs from the advertisers, the main effort of the publisher is to estimate an accurate CTR for each ad-user pair in order to ensure optimal allocation. The same argument applies for conversion rate (CVR) if the publisher cares about how much these clicks successfully turn into business transactions. As a result, academia and major internet companies invest a considerable amount of research and engineering efforts on training a good CTR estimator.

A leading direction for predicting CTR has been the one that makes use of recent progress on deep learning [3, 9, 21, 23, 45]. These deep models can be typically decomposed into two parts: Embeddings and Multi-layer Perceptrons (MLP) [45]. First, an embedding layer transforms each field of raw input into a fixed-length real-valued vector. Typical usage of the embedding is to transform an ad identifier (ad ID) into dense vectors, which can be viewed as a latent representation of the specific ad. It has been widely known in the industry that a well-learned embedding for an ad ID can largely improve the prediction accuracy, compared to methods with no ID input [3, 9, 11, 23, 45]. Next, embeddings are fed into sophisticated models, which can be seen as different types of MLP. These models include Factorization Machine (FM) [24, 25], the extended family of FFM [11] and FwFM [21] that use inner products of embeddings to learn feature interactions; deeper models such as Wide&Deep [3], PNN [23] and DeepFM [9] that learn higher-order relations among features. These methods have achieved state-of-the-art performance across a wide range of CTR prediction tasks.

Despite the remarkable success of these methods, it is extremely data demanding to learn the embedding vectors. For each ad, a large number of labeled examples are needed to train a reasonable ID embedding. When a new ad is added to the candidate pool, it is unlikely to have a good embedding vector for its ID with the mentioned methods. Moreover, for "small" ads with a relatively small number of training samples, it is hard to train their embeddings

as good as for the "large" ads. These difficulties can all be regarded as the *cold-start* problem ubiquitous in the literature.

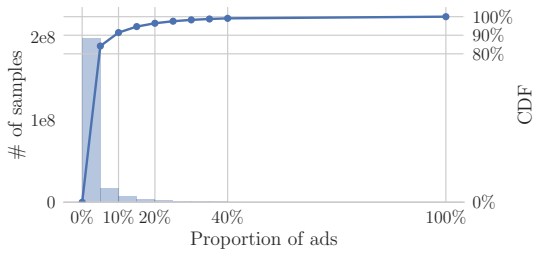


Figure 1: Histogram of the number of samples over different proportions of ads of the KDD Cup 2012 search ads dataset.

The cold-start problem has become a crucial obstacle for online advertising. For example, Figure 1 shows that in the KDD Cup 2012 dataset[1] of search ads, 5% of ads accounted for over 80% of samples, while the rest of 95% small-sized ads had a very small amount of data. Therefore, being good at cold-start can not only benefit the revenue but also improve the satisfaction of small advertisers.

In light of these observations, we aim to develop a new method that can warm up cold advertisements. Our high-level idea is to learn better initial embeddings for new ad IDs that can 1) yield acceptable predictions for cold-start ads and 2) to speed up the model fitting after a small number of examples are obtained. We propose a meta-learning approach that learns how to learn such initial embeddings, coined *Meta-Embedding* in this paper. The main idea of the proposed method includes leveraging a parameterized function of ad features as the ID embedding generator, and a two-phase simulation over old IDs to train the generator by gradient-based meta-learning to improve both cold-start and warm-up performances.

To start with, we list two important desiderata that we pursue:

(1) **Better at cold-start**: When making predictions for a new ad with no labeled data, one should make predictions with a smaller loss;
(2) **Faster at warming-up**: After observing a small number of labeled samples, one should speed up the model fitting so as to reduce the prediction losses for subsequent predictions.

In order to achieve these two desiderata, we design a two-phase simulation over the "large" ads in hand. The simulation consists of a cold-start phase and a warm-up phase. At the cold-start phase, we need to assign an initial embedding for the ID with no labeled data. At the warm-up phase when we have access to a minimal number of labeled examples, we update the embedding accordingly to simulate the model fitting procedure. In this way, we can learn how to learn.

With the two-phase simulation, we propose a meta-learning algorithm to train the Meta-Embedding generator. The essence of our method is to recast CTR prediction as a meta-learning problem, in which learning each ad is viewed as a task. We propose a gradient-based training algorithm with the advantage of Model-Agnostic Meta-Learning (MAML) [5]. MAML is successful for fast-adaptation in many areas, but it trains one model per task, so it cannot be used for CTR predictions if there are millions of tasks (ads). To this end, we generalize MAML into a content-based embedding generator. We construct our unified optimization objective that balances both cold-start and warm-up performance. Then the generated embedding cannot only yield acceptable predictions with no labeled data, but also adapt fast when a minimal amount of data is available. Lastly, our method is easy to implement and can be applied either offline or online with static or streaming data. It can also be used to cold-start other ID features, e.g., the user ID and the advertiser ID.

Note that all the models and experiments in this paper are under the general online supervised learning framework where the system passively collects data. So we do not address the trade-off of exploration and exploitation [15, 18–20, 31, 33], nor do we design interviews as in active learning [6, 10, 22, 46]. But our methodology can be easily extended to these cases.

To summarize, the contributions of this paper are as follows:

- We propose Meta-Embedding that learns how to learn embeddings for new ad IDs to address the cold-start problem. It generates initial embeddings for new ad IDs with the ad contents and attributes as inputs. To the best of our knowledge, it is the first work that aims to improve the performance of various state-of-the-art CTR prediction methods for both cold-start and warm-up phases for new ads under a general supervised learning framework.
- We propose a simple yet powerful algorithm to train the Meta-Embedding generator using gradient-based meta-learning by making use of second derivatives when back-propagating.
- The proposed method is easy to implement in the online cold-start setting. Once the embedding generator is trained, it can take the place of trivial random initializer for new ID embeddings so as to warm up cold-start for the new ad.
- We verify the proposed methods on three large-size real-world datasets. Experimental results show that six existing state-of-the-art CTR prediction methods can all be drastically improved when leveraging Meta-Embedding for both cold-start new ads and warm-up small-sized ads.

2 BACKGROUND AND FORMULATIONS

CTR prediction is a supervised binary classification task. Each instance $(\mathbf{x}, y)$ consists of the input features $\mathbf{x}$ and the binary label y. A machine learning system should approximate the probability $p = \Pr(y = 1 \mid \mathbf{x})$ for the instance with input $\mathbf{x}$.

In most cases, the inputs $\mathbf{x}$ can be splitted into three parts, i.e., $\mathbf{x} = (\mathbf{i}, \mathbf{u}_{[\mathbf{i}]}, \mathbf{v})$, including

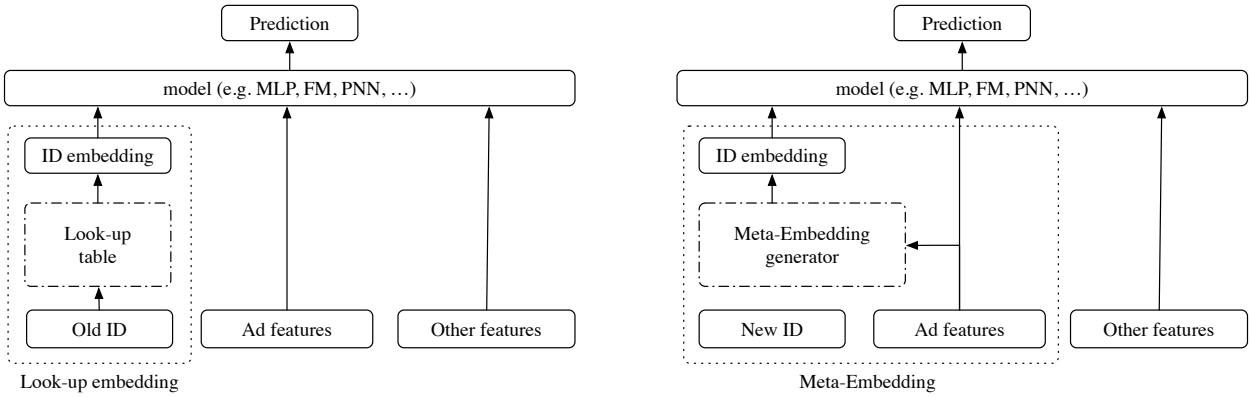
(1) $\mathbf{i}$, the ad identifier (ID), a positive integer to identify each ad;
(2) $\mathbf{u}_{[\mathbf{i}]}$, the features and attributes of the specific ad $\mathbf{i}$, may have multiple fields;
(3) $\mathbf{v}$, other features which do not necessarily relate to the ad, may have multiple fields, such as the user features and contextual information.

So we predict p by a discriminative function of these three parts,

$$\hat{p} = f(\mathbf{i}, \mathbf{u}_{[\mathbf{i}]}, \mathbf{v}). \tag{1}$$

Because the ID is an index, it has to be encoded into real-value to apply gradient-based optimization methods. One-hot encoding is a basic tool to encode it into a sparse binary vector, where all

[1] http://www.kddcup2012.org/c/kddcup2012-track2



(a) Warm-start: look-up embeddings for old ads

(b) Cold-start: Meta-Embedding for new ads

Figure 2: Comparison: Look-up Embedding for old IDs and Meta-Embedding for new IDs. For old ads, we look up the embedding from the look-up table trained with labeled samples previously. For new IDs in the cold-start phase, we use Meta-Embedding to generate an initial embedding by feeding the ad features.

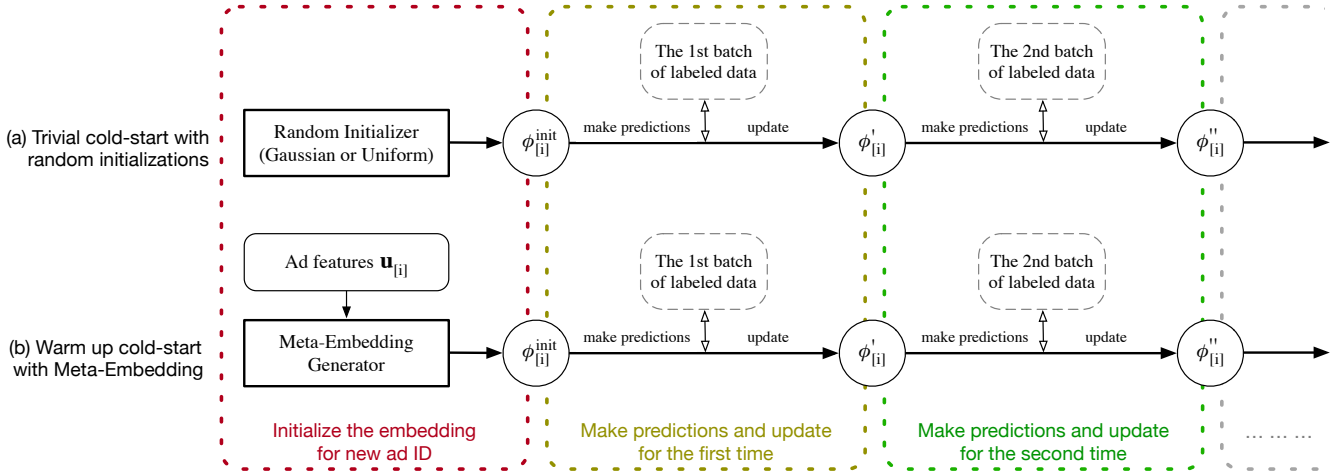


Figure 3: Meta-Embedding is easy to deploy. It is compatible with the standard updating scheme. From left to right, it first initializes the embedding before observing any labeled data. The second block is when we observe the data, receive the prediction loss, and update the embedding for the first time. The third block is when we apply the adapted embedding and receive losses again. We are motivated to find a good embedding generator that can reduce all these losses, to warm up cold-start.

components are 0 except for the $\mathbf{i}^{\text{th}}$ component is 1. For example,

$$\mathbf{i} = 3 \xrightarrow{\text{one-hot}} \mathbf{e}_\mathbf{i} = (0, 0, 1, 0, \dots)^T \in \mathbb{R}^n, \tag{2}$$

where n is the number of IDs in all. If followed up by a matrix multiplication, we will have a low-dimensional representation of the ID, i.e., given a dense matrix $\Phi \in \mathbb{R}^{n \times m}$, then

$$\phi_{[\mathbf{i}]} = \mathbf{e}_\mathbf{i}^T \Phi \in \mathbb{R}^m \tag{3}$$

is the resulted m-dimentional dense real-valued vector for ID $\mathbf{i}$. In this way, the ID is turned into a low-dimensional dense vector.

However, due to the fact that there are often millions of IDs, the one-hot vector can be extremely large. So nowadays it is more usual to use the look-up embedding as a mathematically equivalent alternative. For ID $\mathbf{i}$, it directly looks up the $\mathbf{i}^{\text{th}}$ row from Φ,

$$\mathbf{i} = 3 \xrightarrow{\text{look-up the i}^{\text{th}}\text{ row from }\Phi} \phi_{[\mathbf{i}]} \in \mathbb{R}^m \tag{4}$$

where the resulted $\phi_{[\mathbf{i}]}$ is the same as in (3). The dense matrix Φ is often referred to as the embedding matrix, or the look-up table.

Given Φ the embedding matrix and $\phi_{[\mathbf{i}]}$ the embedding for ID $\mathbf{i}$, we can get a parameterized function as the discriminative model,

$$\hat{p} = f_{\boldsymbol{\theta}}(\phi_{[\mathbf{i}]}, \mathbf{u}_{[\mathbf{i}]}, \mathbf{v}). \tag{5}$$

For readability, we refer to the function $f_{\boldsymbol{\theta}}$ as the *base model* throughout this paper, and let $\boldsymbol{\theta}$ denote its parameters. Figure 2(a) shows the basic structure of such a model.

The Log-loss is often used as the optimization target, i.e.

$$l(\boldsymbol{\theta}, \Phi) = -y \log \hat{p} - (1 - y) \log(1 - \hat{p}). \tag{6}$$

In offline training where a number of samples for each ad ID are available, $\boldsymbol{\theta}$ and Φ are updated simultaneously to minimize the Log-loss by Stochastic Gradient Descent (SGD).

What if there comes an ID $\mathbf{i}^*$ that the system has never seen before? It means that we have never got labeled data for the ad, so the corresponding row in the embedding matrix remains an initial state, for example, random numbers around zero. Therefore the testing accuracy would be low. It is known as the ad cold-start problem.

To address this issue, we aim to design an embedding generator. The structure is shown in Figure 2(b). Given an unseen ad $\mathbf{i}^*$, the generator inputs the features $\mathbf{u}_{[\mathbf{i}^*]}$ of the ad, and can somehow output a "good" initial embedding as a function of these features,

$$\phi^{\text{init}}_{[\mathbf{i}^*]} = h_{\boldsymbol{w}}(\mathbf{u}_{[\mathbf{i}^*]}).$$

Now that we explained why we want to design the embedding generator to address the cold-start problem. The problem is, how to train such an embedding generator? For what objective do we update its parameters? To accomplish our final goal of warming up cold-start, we propose a meta-learning based approach which learns how to learn new ad ID embeddings. Specifically, the parameters $\boldsymbol{w}$ in the embedding generator are trained by making use of previously learned ads through gradient-based meta-learning, which will be detailed in Section 3.

3 LEARNING TO LEARN THE ID EMBEDDINGS FOR NEW ADS

In this section, we propose Meta-Embedding, a meta-learning based method for learning to learn the ID embeddings for new ads, to solve the ad cold-start problem. First, we recast the CTR prediction problem as meta-learning. Then, we propose the framework of learning to learn ID embeddings and introduce our meta-learner induced by both the pre-trained model and a Meta-Embedding generator. Finally, we detail the model architectures and hyper-parameters.

3.1 Recasting CTR prediction as Meta-Learning

Recall that the final goal for CTR predictions as shown in (1) is to learn a predictive function $f(\cdot)$ with inputs of an ID and two sets of features. Only after the ID is transformed into a real-valued vector can we begin to learn the parameterized base model $f_{\boldsymbol{\theta}}(\phi_{[\mathbf{i}]}, \cdot, \cdot)$ as in (5). Therefore, for a given ad, the embedding of ID implicitly determines the hidden structure of the model.

To look at it from a meta-learning perspective, we introduce a new notation to write the predictive model for a fixed ID $\mathbf{i}$ as

$$\hat{p} = g_{[\mathbf{i}]}(\mathbf{u}_{[\mathbf{i}]}, \mathbf{v}) = f_{\boldsymbol{\theta}}(\phi_{\mathbf{i}}, \mathbf{u}_{[\mathbf{i}]}, \mathbf{v}). \tag{7}$$

Note that $g_{[\mathbf{i}]}(\cdot, \cdot)$ is exactly the same function as $f_{\boldsymbol{\theta}}(\phi_{[\mathbf{i}]}, \cdot, \cdot)$, whose parameters are $\boldsymbol{\theta}$ and $\phi_{[\mathbf{i}]}$.

In this way, we can see CTR prediction as an instance of meta-learning by regarding the learning problem w.r.t each ad ID as a distinguished task. For IDs $\mathbf{i} = 1, 2, \cdots$, the tasks $t_1, t_2, \cdots$ are to learn the task-specific models $g_{[1]}, g_{[2]}, \cdots$ respectively. They share the same set of parameters $\boldsymbol{\theta}$ from the base model, and meanwhile maintain their task-specific parameters $\phi_{[1]}, \phi_{[2]}, \cdots$.

Consider that we have access to prior tasks $t_{\mathbf{i}}$ with IDs $\mathbf{i} \in \mathcal{I}$, the set of all known IDs, as well as a number of training samples for each task. The original (pre-)training on this data gives us a set of well-learned shared parameters $\boldsymbol{\theta}$ and task-specific parameters $\phi_{[\mathbf{i}]}$ for all prior IDs $\mathbf{i} \in \mathcal{I}$. However, for a new ID $\mathbf{i}^* \notin \mathcal{I}$, we do not know $\phi_{[\mathbf{i}^*]}$. So we desire to learn how to learn $\phi_{[\mathbf{i}^*]}$ with the prior experience on those old IDs. This is how we see the cold-start problem of CTR prediction from a meta-learning point of view.

3.2 Meta-Embedding

In this section, we put forward Meta-Embedding that captures the skills of learning ID embeddings for new ads via meta-learning.

In the previous section, we introduced the shared parameters $\boldsymbol{\theta}$ and task-specific parameters $\phi_{\mathbf{i}}$ for old items $\mathbf{i} \in \mathcal{I}$. As far as $\boldsymbol{\theta}$ is usually trained previously with an extremely large amount of historical data, we are confident about its effectiveness. So, when training the Meta-Embedding, we freeze $\boldsymbol{\theta}$ and do no update it during the whole process. The only thing matters for the cold-start problem in this paper is how to learn the embeddings for new IDs.

Recall that the task-specific parameter $\phi_{[\mathbf{i}]}$ is unknown for any unseen ID. So we need to use a shared functional embedding generator to take its place. For a new ad with ID $\mathbf{i}^*$, we let

$$\phi^{\text{init}}_{[\mathbf{i}^*]} = h_{\boldsymbol{w}}(\mathbf{u}_{[\mathbf{i}^*]}), \tag{8}$$

as the generated initial embedding for simplicity of notations. Then the model induced by the generated embedding is

$$g_{\text{meta}}(\mathbf{u}_{[\mathbf{i}^*]}, \mathbf{v}) = f_{\boldsymbol{\theta}}\big(\phi^{\text{init}}_{[\mathbf{i}^*]}, \mathbf{u}_{[\mathbf{i}^*]}, \mathbf{v}\big). \tag{9}$$

So here $g_{\text{meta}}(\cdot, \cdot)$ is a model (a meta-learner) that inputs the features and outputs the predictions, without involving the embedding matrix. The trainable parameter for it is the meta-parameter $\boldsymbol{w}$ from $h_{\boldsymbol{w}}(\cdot)$.

Now we describe the detailed procedure to simulate cold-start with old IDs as if they were new. Consider that for each task $t_{\mathbf{i}}$ w.r.t an old ID $\mathbf{i}$, we have already got training samples $\mathcal{D}_{[\mathbf{i}]} = \big\{(\mathbf{u}_{[\mathbf{i}]}, \mathbf{v}_j)\big\}_{j=1}^{N_{\mathbf{i}}}$, where $N_{\mathbf{i}}$ is the number of samples for the given ID.

To begin with, we randomly select two disjoint minibatches of labeled data, $\mathcal{D}^a_{[\mathbf{i}]}$ and $\mathcal{D}^b_{[\mathbf{i}]}$, each with K samples. It is assumed that the minibatch size is relatively small, i.e., $K << N_{\mathbf{i}}/2$.

3.2.1 Cold-start phase. We first make predictions using $g_{\text{meta}}(\cdot, \cdot)$ on the first minibatch $\mathcal{D}^a_{[\mathbf{i}]}$, as

$$\hat{p}_{aj} = g_{\text{meta}}(\mathbf{u}_{[\mathbf{i}]}, \mathbf{v}_{aj}) = f_{\boldsymbol{\theta}}(\phi^{\text{init}}_{[\mathbf{i}]}, \mathbf{u}_{[\mathbf{i}]}, \mathbf{v}_{aj}) \tag{10}$$

where the subscript 'aj' is for the j^{th} sample from batch $\mathcal{D}^a_{[\mathbf{i}]}$. Then we calculate the average loss over these samples

$$l_a = \frac{1}{K}\sum_{j=1}^{K}\Big[-y_{aj}\log\hat{p}_{aj} - (1-y_{aj})\log(1-\hat{p}_{aj})\Big]. \tag{11}$$

By far, we have done with the cold-start phase: we generated the embedding $\phi^{\text{init}}_{[\mathbf{i}]}$ by the generator $h_{\boldsymbol{w}}(\cdot)$, and evaluated it on the first batch of data to get a loss l_a.

3.2.2 Warm-up phase. Next, we simulate the learning process for the warm-up phase with the second batch of data $\mathcal{D}^b_{[\mathbf{i}]}$.

By computing the gradient of l_a w.r.t the initial embedding $\phi^{\text{init}}_{[\mathbf{i}]}$ and take a step of gradient descent, we get a new adapted embedding

$$\phi'_{[\mathbf{i}]} = \phi^{\text{init}}_{[\mathbf{i}]} - a\,\frac{\partial l_a}{\partial \phi^{\text{init}}_{[\mathbf{i}]}} \tag{12}$$

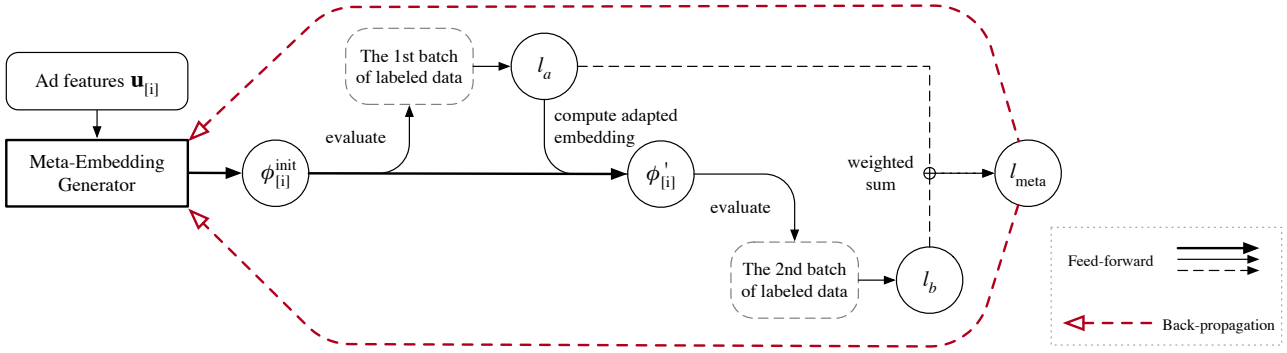


Figure 4: Demonstration of how to train the parameters of the Meta-Embedding generator, corresponding to the inner for-loop of Alg.1. The use of second derivatives make our method powerful for warm up cold-start. The training for Meta-Embedding can either use offline or online data, or both.

where $a > 0$ is the step size of gradient descent.

Now that we have a new embedding which is trained with a minimal amount of data, we can test it on the second batch of data. Similar to the previous part, we make predictions

$$\hat{p}_{bj} = g'_{[\mathbf{i}]}(\mathbf{u}_{[\mathbf{i}]}, \mathbf{v}_{bj}) = f_{\boldsymbol{\theta}}(\phi'_{[\mathbf{i}]}, \mathbf{u}_{[\mathbf{i}]}, \mathbf{v}_{bj}) \tag{13}$$

and compute the average loss

$$l_b = \frac{1}{K}\sum_{j=1}^{K}\Big[- y_{bj}\log\hat{p}_{bj} - (1-y_{bj})\log(1-\hat{p}_{bj})\Big]. \tag{14}$$

3.2.3 A unified optimization objective. We suggest to evaluate the goodness of the initial embeddings from two aspects:

(1) The error of CTR predictions for the new ad should be small;
(2) After a small number of labeled examples are collected, a few gradient updates should lead to fast learning.

Surprisingly, we find that the two losses l_a and l_b can perfectly suit these two aspects respectively. On the first batch, since we make predictions with generated initial embeddings, l_a is a natural metric to evaluate the generator in the cold-start phase. For the second batch, as the embeddings have been updated once, it is straight-forward that l_b can evaluate the sample efficiency in the warm-up phase.

To unify these two losses, we propose our final loss function for Meta-Embedding as a weighted sum of l_a and l_b,

$$l_{\text{meta}} = \alpha l_a + (1-\alpha) l_b, \tag{15}$$

where $\alpha \in [0, 1]$ is a coefficient to balance the two phases.

As l_{meta} is a function of the initial embedding, we can back-prop the gradient w.r.t the meta-parameter $\boldsymbol{w}$ by the chain rule:

$$\frac{\partial l_{\text{meta}}}{\partial \boldsymbol{w}} = \frac{\partial l_{\text{meta}}}{\partial \phi^{\text{init}}_{[\mathbf{i}]}}\frac{\partial \phi^{\text{init}}_{[\mathbf{i}]}}{\partial \boldsymbol{w}} = \frac{\partial l_{\text{meta}}}{\partial \phi^{\text{init}}_{[\mathbf{i}]}}\frac{\partial h_{\boldsymbol{w}}}{\partial \boldsymbol{w}}, \tag{16}$$

where

$$\frac{\partial l_{\text{meta}}}{\partial \phi^{\text{init}}_{[\mathbf{i}]}} = \alpha\frac{\partial l_a}{\partial \phi^{\text{init}}_{[\mathbf{i}]}} + (1-\alpha)\frac{\partial l_b}{\partial \phi'_{[\mathbf{i}]}} - a(1-\alpha)\frac{\partial l_b}{\partial \phi'_{[\mathbf{i}]}}\frac{\partial^2 l_a}{\partial {\phi^{\text{init}}_{[\mathbf{i}]}}^2}. \tag{17}$$

Although it involves second derivatives, i.e., a Hessian-vector, it can be efficiently implemented by existing deep learning libraries that allows automatic differentiation, such as TensorFlow [1].

Finally, we come to our training algorithm, which can update the meta-parameters by stochastic gradient descent in a mini-batch manner, see Alg.1. A demonstration of training procedure w.r.t each single ID (the inner for-loop of Alg.1) is also shown in Figure 4.

Note that Meta-Embedding not only can be trained with offline data set, but also can be trained online with minor modifications by using the emerging new IDs as the training examples.

Algorithm 1 Train Meta-Embedding by SGD

Input: $f_{\boldsymbol{\theta}}$: the pre-trained base model.
Input: $\mathcal{I}$: the set of all existing IDs.
Input: α: hyper-parameter, the coefficient for meta-loss.
Input: a, b: step sizes.

Randomly initialize $\boldsymbol{w}$
while not done **do**
  Randomly sample n IDs $\{\mathbf{i}_1, \ldots, \mathbf{i}_n\}$ from $\mathcal{I}$
  for $\mathbf{i} \in \{\mathbf{i}_1, \ldots, \mathbf{i}_n\}$ **do**
    Generate the initial embedding: $\phi^{\text{init}}_{[\mathbf{i}]} = h_{\boldsymbol{w}}(\mathbf{u}_{[\mathbf{i}]})$
    Sample mini-batches $\mathcal{D}^a_{[\mathbf{i}]}$ and $\mathcal{D}^b_{[\mathbf{i}]}$ each with K samples
    Evaluate loss $l_a(\phi^{\text{init}}_{[\mathbf{i}]})$ on $\mathcal{D}^a_{[\mathbf{i}]}$
    Compute adapted embedding: $\phi'_{[\mathbf{i}]} = \phi^{\text{init}}_{[\mathbf{i}]} - a\frac{\partial l_a(\phi^{\text{init}}_{[\mathbf{i}]})}{\partial \phi^{\text{init}}_{[\mathbf{i}]}}$
    Evaluate loss $l_b(\phi'_{[\mathbf{i}]})$ on $\mathcal{D}^b_{[\mathbf{i}]}$
    Compute loss: $l_{\text{meta},\mathbf{i}} = \alpha l_a(\phi^{\text{init}}_{[\mathbf{i}]}) + (1-\alpha) l_b(\phi'_{[\mathbf{i}]})$
  Update $\boldsymbol{w} \leftarrow \boldsymbol{w} - b\sum_{\mathbf{i}\in\{\mathbf{i}_1,\ldots,\mathbf{i}_n\}}\frac{\partial l_{\text{meta},\mathbf{i}}}{\partial \boldsymbol{w}}$

3.3 Architecture and hyper-parameter

Finally, we discuss how to choose the architecture of the embedding generator along with the hyper-parameters.

3.3.1 Architectures of $h_{\boldsymbol{w}}(\cdot)$. In principle, we want the embedding generator to be simple yet strong. So we suggest using the neural network as the parameterized function. Any common net architecture can be used for Meta-Embedding.

Here we would like to show an instance of the generator whose effectiveness is tested in our experiments, see Figure 5. It is simple

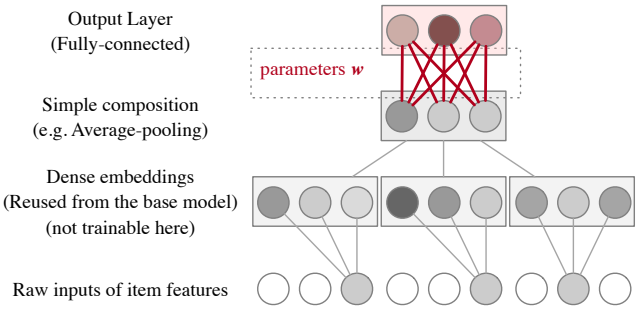


Figure 5: An example of the Meta-Embedding generator, with frozen reused embedding layers reused from the base model and lightweight compositional layers.

and lightweight, thus can be a basic choice if anyone would like to start using Meta-Embedding.

The input of the generator network is the ad feature $\mathbf{u}_{[i]}$. However, we do not need to train the generator from scratch. Recall that, in the base model, there is already embedding layers for ad features, so we can directly reuse these layers. We suggest freezing the weights of the reused layers so as to reduce the number of trainable parameters. The embeddings from different fields can be aggregated by average-pooling, max-pooling, or concatenating. Finally, we use a dense layer to get the outputs, which is the only trainable part of the generator. We can have numerically stable outputs by using three tricks on the final layer: 1) use the tanh activation, 2) do not add the bias term, and 3) use L2 regularization to penalize the weights.

3.3.2 The hyper-parameter α. The coefficient $\alpha \in [0, 1]$ balances the cold-start loss and the warm-up loss. Since the warm-up phase usually involves more time steps then the cold-start phase in practice, we suggest to set it to a small value so that the meta-learner will pay more attention to the warm-up phase to achieve fast adaptation. Empirically, we found that our algorithm is robust to the hyper-parameter. We simply set α to 0.1 for all our experiments.

4 EXPERIMENTS

4.1 Datasets

We evaluate Meta-Embedding on three datasets:

MovieLens-1M[2]: One of the most well-known benchmark data. The data consists of 1 million movie ranking instances over thousands of movies and users. Each movie can be seen as an ad in our paper, with features including its title, year of release, and genres. Titles and genres are lists of tokens. Other features include the user's ID, age, gender, and occupation. We first binarize ratings to simulate the CTR prediction task, which is common to test binary classification methods [13, 38]. The ratings at least 4 are turned into 1 and the others are turned into 0. Although this dataset is not a dataset for CTR prediction, its small size enables us to easily verify the effectiveness of our proposed method.

Tencent CVR prediction dataset for App recommendation: The public dataset for the Tencent Social Ads competition in 2018[3] with over 50 million clicks. Each instance is made up by an ad, a user and a binary label (conversion). Each ad has 6 categorical attributes: advertiser ID, ad category, campaign ID, app ID, app size, and app type. Other features are user attributes, including the user's gender, age, occupation, consumption ability, education level, and city.

KDD Cup 2012 CTR prediction dataset for search ads[4]: The dataset contains around 200 million instances derived from session logs of the Tencent proprietary search engine, *soso.com*. Each ad has three features: keywords, title, and description. These features are lists of anonymized tokens hashed from natural language. Other features consist of the query (also a list of tokens), two session features and the user's gender and age.

4.2 Base models

Because our Meta-Embedding is model-agnostic, it can be applied upon various existing models that in the Embedding & MLP paradigm. To show the effectiveness of our method, we conduct experiments upon the following representative models:

- **FM**: the 2-way Factorization Machine [24]. Originally it uses one-hot encodings and matrix multiplications to get embedding vectors, while in our implementation we directly use look-up embeddings. For efficiency, we use the same embedding vectors for the first- and the second-order components.
- **Wide & Deep**: proposed by Google in [3] which models both low- and high-order feature interactions. The wide component is a Logistic Regression that takes one-hot vectors as inputs. The deep component has embedding layers and three dense hidden layers with ReLU activations.
- **PNNs**: Product-based Neural Networks [23]. It first feeds the dense embeddings into a dense layer and a product layer, then concatenates them together and uses another two dense layers to get the prediction. As suggested by their paper, we use three variants: **IPNN**, **OPNN**, and **PNN***. IPNN is the PNN with an inner product layer, OPNN is the PNN with an outer product layer, and PNN* has both inner and outer products.
- **DeepFM** [9]: a recent state-of-the-art method that learns both low- and high-level interactions between fields. It feeds dense embeddings into an FM and a deep component, and then concatenates their outputs and gets the final prediction by a dense layer. For the deep component, we use three ReLU layers as suggested in [9].

These base models share the common Embedding & MLP structure. The dimensionality of embedding vectors of each input field is fixed to 256 for all our experiments. For natural language features in MovieLens and the KDD Cup dataset, we first embed each token (word) into a 256-dimensional word-embedding, then use AveragePooling to get the field-(sentence-)level representation. In other words, after the embedding layer, every field is embedded into a 256-dimensional vector respectively.

4.3 Experiment Set-Up

4.3.1 Dataset splits. To evaluate the performance of cold-start advertising in the two phases, we conduct the experiments by splitting the datasets. First, we group the advertisements by their sizes:

[2] http://www.grouplens.org/datasets/movielens/
[3] http://algo.qq.com/
[4] https://www.kaggle.com/c/kddcup2012-track2

Table 1: The statistics for data splitting

Dataset	Minibatch size K	Old ads			New ads		
		# of IDs	# of samples	# of samples used to train Meta-Embedding	# of IDs	# of samples	# of samples in the hold-out set
MovieLens-1M	20	1127	0.76 M	0.09 M	1058	0.19 M	0.12 M
Tencent CVR data	200	572	49.33 M	0.45 M	443	5.00 M	4.74 M
KDD Cup 2012	200	6534	148.55 M	5.22 M	9299	28.71 M	23.13 M

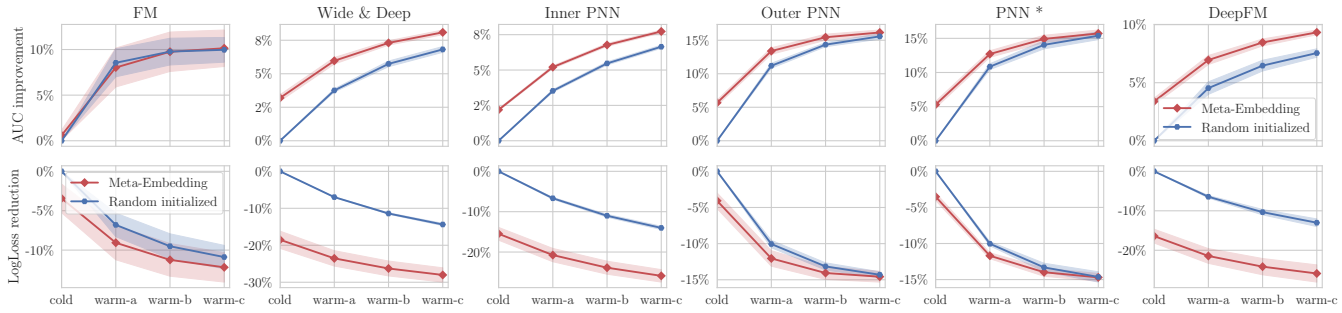


Figure 6: Cold-start performance in percentage on MovieLens-1M, over six popular base models. The solid lines are averaged scores and the bands are standard deviations over three runs.

- Old ads: the ads whose number of labeled instances is larger than a threshold N. Since the volumes of the three datasets are different, we used N of 300, 20000, and 5000 for MovieLens-1M, Tencent CVR data, and KDD Cup data, respectively.
- New ads: the ads whose number of labeled instances is less than N, but larger than $N_{\text{min}} > 3 \times K$, where K is the mini-batch size mentioned in Section 3. The reason for setting the minimum number of samples is to guarantee that after cutting $3 \times K$ samples for the warm-up phase, there still remains a number of samples for testing. N_{min} is set to 80, 2000, and 2000 for the three datasets, respectively.

The summary of the data split can be seen in Table 1. We set the thresholds to make the number of new ads be close to the number of old ads, although the data size of new ads is much smaller.

4.3.2 Experiment pipeline. To start with, we use the old ads to pre-train the base models. Then we use them to conduct offline training for Meta-Embedding. The number of samples used to train Meta-Embedding can be also found in Table 1. After training, it is tested over new ads. For each new ad, we train on 3 mini-batches one by one as if they were the warming-up data, named batch-a, -b, and -c, each with K instances. Other instances for new ads are hold-out for testing.

The experiments are done with the following steps:

0. Pre-train the base model with the data of old ads (1 epoch).
1. Train the Meta-Embedding with the training data (2 epochs).
2. Generate initial embeddings of new ad IDs with (random-initialization or Meta-Embedding).
3. Evaluate cold-start performance over the hold-out set;
4. Update the embeddings of new ad IDs with batch-a and compute evaluation metrics on the hold-out set;
5. Update the embeddings of new ad IDs with batch-b and compute evaluation metrics on the hold-out set;
6. Update the embeddings of new ad IDs with batch-c and compute evaluation metrics on the hold-out set;

The training effort for Meta-Embedding is very small compared to the pre-train step. We train it with $2 \times K$ samples per epoch/ad, so the number of samples involved is only (# of old IDs) $\times K \times 2 \times 2$. The actual number could be found in Table 1.

4.3.3 Evaluation metrics. We take two common metrics to evaluate the results, the Log-loss and the AUC score (Area Under Receiver Operator Characteristic Curve). All the metrics are evaluated only on the hold-out set.

To increase readability, we will show the relative improvements in percentage over the baseline cold-start score:

$$\text{LogLoss\_precentage} = \left(\frac{\textbf{LogLoss}}{\text{LogLoss}_{\text{base-cold}}} - 1 \right) \times 100\%$$

$$\text{AUC\_precentage} = \left(\frac{\textbf{AUC}}{\text{AUC}_{\text{base-cold}}} - 1 \right) \times 100\%$$

where "base-cold" refer to cold-start scores produced by the base model with random initialized embeddings for unseen IDs. Thus, for LogLoss, it is better if the percentage is more negative. For AUC, the percentage is positive, the larger the better. Note that this "base-cold" is a strong baseline because it has all the warm contents to use, and is the commonly used in industry.

4.4 Experiment results

For each experiment, we took three runs and reported the averaged performance. The main experimental results are shown in Table 2.

We first see the results on MovieLens-1M. Figure 6 exhibits the detailed performance of each tested model. We observed that

Table 2: Experimental results: Average performances on tested datasets, over six base models, three runs for each.

Dataset	Metrics	Cold-Start phase		Warm-Up phase: a		Warm-Up phase: b		Warm-Up phase: c	
		baseline	Meta	baseline	Meta	baseline	Meta	baseline	Meta
MovieLens-1M	AUC percentage	0.0%	+3.36%	+7.02%	+8.66%	+9.25%	+10.40%	+10.27%	+11.15%
	Logloss percentage	0.0%	-10.23%	-7.83%	-16.42%	-11.45%	-18.93%	-13.53%	-20.20%
Tencent CVR data	AUC percentage	0.0%	+0.50%	+0.60%	+0.99%	+1.16%	+1.47%	+1.57%	+1.83%
	LogLoss percentage	0.0%	-0.19%	-0.30%	-0.44%	-0.55%	-0.65%	-0.71%	-0.79%
KDD Cup 2012 Search Ads CTR	AUC percentage	0.0%	+0.73%	+2.34%	+2.76%	+3.56%	+3.87%	+4.36%	+4.61%
	LogLoss percentage	0.0%	-1.52%	-1.97%	-2.99%	-3.06%	-3.82%	-3.79%	-4.40%

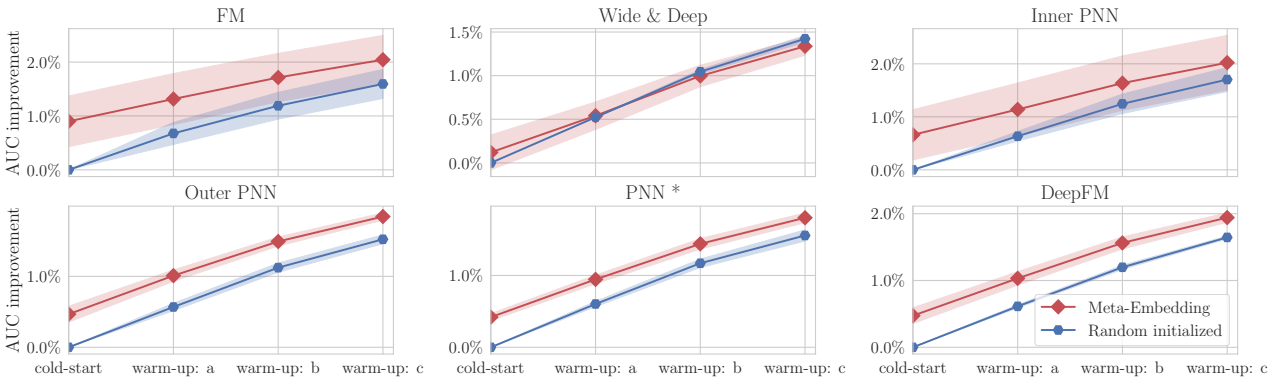


Figure 7: AUC improvements in percentage on the Tencent CVR prediction data, over six base models.

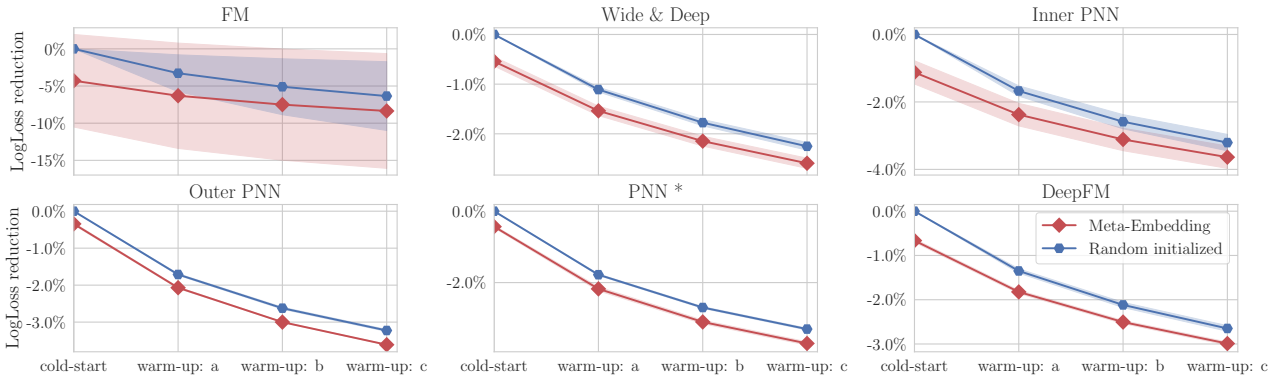


Figure 8: LogLoss reductions in percentage on the KDD Cup 2012 CTR prediction dataset for search ads, over six models.

when using Meta-Embedding, most models yielded better AUC and LogLoss results comparing to the random initialization baseline. Although the hyper-parameter of coefficient α was set to 0.1 so the major attention was on the warm-up loss, it still performed really well in the cold-start phase. For example, when using Wide&Deep, IPNN and DeepFM as the base models, Meta-Embedding provided with a LogLoss reduction of over 15% in the cold-start phase. After three times of warm-up updates, Meta-Embedding still outperformed the baseline, thus we are confident that Meta-Embedding increases the sample efficiency for new ads.

On the two larger datasets, Meta-Embedding also achieved considerable improvements as shown in Table 2. Due to the space limitation, we only show part of the detailed results: Figure 7 shows the AUC improvements on the Tencent CVR prediction dataset, and Figure 8 shows the LogLoss on the KDD Cup search ad dataset.

An interesting finding is that the relative improvement was more significant on smaller datasets (MovieLens) than the larger ones. To explain, recall that the number of training examples in MovieLens is much smaller, so the shared pre-trained model parameters $\boldsymbol{\theta}$ may not be good enough. Therefore, in this case, there could be a larger improvement space for cold-start ads, for example, Meta-Embedding can sometimes double the improvement compared to the random initialization baseline.

For most of the experiments, the scores were greatly improved by replacing random initialization with Meta-Embedding. But there

were also some cases where the performances were mixed. For example, for MovieLens with FM as the base model, Meta-Embedding had similar AUC with the baseline. But the LogLoss of Meta-Embedding was smaller, so we can still be sure that Meta-Embedding can help FM. For Wide & Deep in the Tencent CVR data (see Figure 7), the baseline tended to surpass Meta-Embedding after the third warm-up update. However, it only happens for the Tencent CVR data only. We think it was due to the "Wide" component which directly used raw inputs. We conjecture it mainly because this dataset had categorical inputs only. In the other two datasets, there are sequential inputs, so the "Wide" component can not dominate the performance.

Overall, we observe that Meta-Embedding is able to significantly improves CTR and CVR predictions for cold-start ads, and can work for almost all the tested base models and datasets in our comparisons.

5 RELATED WORK

A. Methods addressing the cold-start problem. There are two types of methods. The first type actively solve cold-start by designing a decision making strategy, such as using contextual-bandits [2, 15, 18, 19, 31, 33], or designing interviews to collect information for the cold item or user [6, 10, 22, 46].

This paper belongs to the second type which treats cold-start within an online supervised learning framework. It is common to use side information for the cold-start phase, e.g., using user attributes [26, 30, 37, 43], item attributes [8, 17, 28, 29, 35, 37, 43], or relational data [16, 41, 44]. These methods can be viewed as our "base-cold" baseline with different inputs and model structures without using ad IDs. They can certainly improve the cold-start performance comparing to classic methods that do not use those features. But they do not consider the ad ID neigher do they optimize to improve the warm-up phase. On the contrary, our method not only uses all available features, but also aims to improving both cold-start and warm-up performance.

For online recommendation, a number of studies aid to improve online learning and accelerate model fitting with incremental data. To name some, [27, 32] adjust matrix factorization factors with incremental data; [40] uses a rating comparison strategy to learn the latent profiles. These methods can learn faster with a small amount of data, but they could not be directly applied if there is no sample for new ads. Moreover, these methods are mostly designed for matrix factorization, so it is not straight-forward to apply them to deep feed-forward models. Different from these methods, Meta-Embedding improve CTR prediction in both the cold-start and warm-up phase, and is designed for deep model with an Embedding & MLP structure.

Dropout-Net [37] handles missing inputs by applying Dropout to deep collaborative filtering models, which can be viewed as a successful training method for pre-training the base models. But since this paper focuses on general CTR predictions than collaborative filtering, we did not include it in our experiments.

B. Meta-Learning. It learns how to learn new tasks by using prior experience with related tasks [5, 34, 36]. It led to an interest in many areas, such as recommendations [35], natural language processing [12, 39], and computer vision [4, 5].

We study how to warm up cold-start ads. In meta-learning literature, this problem relates to few-shot learning [14] and fast adaptation [5, 7]. Specifically, we follow similar spirit of MAML [5], a gradient-based meta-learning method which learns shared model parameters across tasks and achieves fast adaptation towards new tasks. However, MAML cannot be applied directly to CTR prediction because it is designed to learn one model per task, which is unacceptable if there is millions of tasks (ads).

A recent work of Vartak et al. [35] also utilizes meta-learning for item cold-start. It represents each user by an averaged representation of items labeled by the user previously. Our approach does not model the user activity and only focus on learning to learn ID embeddings for items (ads).

In natural language processing, there is another so-called "Meta-Embedding" that learns word-embeddings for a new corpus by aggregating pre-trained word vectors from prior domains [12, 39, 42]. It is specific for NLP and do not has common spirit with ours.

6 CONCLUSION AND DISCUSSION

In this paper, we proposed a meta-learning approach to address the cold-start problem for CTR predictions. We propose Meta-Embedding, which focuses on learning how to learn the ID embedding for new ads, so as to apply to state-of-the-art deep models for CTR predictions. Built on the pre-trained base model, we suggest a two-phased simulation over previously learned ads to train the Meta-Embedding generator. We seek to improve both cold-start and warm-up performance by leveraging a unified loss function for training. Afterward, when testing, the trained generator can initialize the ID embeddings for new ads, which yields significant improvement in both cold-start and warm-up phases. To the best of our knowledge, it is the first work aiming to improve CTR prediction for both cold-start and warm-up phases for new ads under a general online supervised learning framework. We verified Meta-Embedding on three real-world datasets over six state-of-the-art CTR prediction models. Experiments showed that, by replacing trivial random initialization with Meta-Embedding, the cold-start and warm-up performances can be significantly improved.

This is an interesting line of work to connect representation learning with learning to learn, especially for online learning tasks including advertising and recommendations. This paper focuses on learning to learn the embedding of ad IDs, which is a specific component of knowledge representation for online CTR/CVR predictions. For the future work, this insight can also be extended to other tasks, for example, learning to learn the model when the distribution of features and labels evolutes over time, or learning to tune the hyper-parameters for specific sub-tasks.

7 ACKNOWLEDGEMENTS

This work is partially supported by the National Key Research and Development Program of China under Grant No. 2017YFB1002104, the National Natural Science Foundation of China under Grant No. U1811461, 61602438, 91846113, 61573335, CCF-Tencent Rhino-Bird Young Faculty Open Research Fund No. RAGR20180111. This work is also funded in part by Ant Financial through the Ant Financial Science Funds for Security Research. Xiang Ao is also supported by Youth Innovation Promotion Association CAS.

Pingzhong Tang was supported in part by the National Natural Science Foundation of China Grant 61561146398, a China Youth 1000-talent program and an Alibaba Innovative Research program.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*. 265–283.

[2] Stéphane Caron and Smriti Bhagat. 2013. Mixing bandits: A recipe for improved cold-start recommendations in a social network. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*. ACM, 11.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[4] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. 2017. Deep Meta Learning for Real-Time Visual Tracking based on Target-Specific Feature Space. *arXiv preprint arXiv:1712.09153* (2017).

[5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*. 1126–1135.

[6] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2011. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 595–604.

[7] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930* (2018).

[8] Quanquan Gu, Jie Zhou, and Chris Ding. 2010. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *Proceedings of the 2010 SIAM international conference on data mining*. SIAM, 199–210.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 1725–1731.

[10] Abhay S Harpale and Yiming Yang. 2008. Personalized active learning for collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 91–98.

[11] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. 2017. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 680–688.

[12] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic Meta-Embeddings for Improved Sentence Representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1466–1477.

[13] László Kozma, Alexander Ilin, and Tapani Raiko. 2009. Binary principal component analysis in the Netflix collaborative filtering task. In *IEEE International Workshop on Machine Learning for Signal Processing, 2009.* IEEE, 1–6.

[14] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science* 350, 6266 (2015), 1332–1338.

[15] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 661–670.

[16] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2013. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 283–292.

[17] Kaixiang Mo, Bo Liu, Lei Xiao, Yong Li, and Jie Jiang. 2015. Image Feature Learning for Cold Start Problem in Display Advertising.. In *IJCAI*. 3728–3734.

[18] Hai Thanh Nguyen, Jérémie Mary, and Philippe Preux. 2014. Cold-start problems in recommendation systems via contextual-bandit algorithms. *arXiv preprint arXiv:1405.7544* (2014).

[19] Feiyang Pan, Qingpeng Cai, Pingzhong Tang, Fuzhen Zhuang, and Qing He. 2018. Policy Gradients for Contextual Recommendations. *arXiv preprint arXiv:1802.04162v3* (2018).

[20] Feiyang Pan, Qingpeng Cai, An-Xiang Zeng, Chun-Xiang Pan, Qing Da, Hualin He, Qing He, and Pingzhong Tang. 2018. Policy Optimization with Model-based Explorations. *arXiv preprint arXiv:1811.07350* (2018).

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

[22] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. 2006. Naïve filterbots for robust cold-start recommendations. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 699–705.

[23] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

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

[25] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 635–644.

[26] Sujoy Roy and Sharath Chandra Guntuku. 2016. Latent factor representations for cold-start video recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 99–106.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*. Citeseer, 27–28.

[28] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 89–96.

[29] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 253–260.

[30] Yanir Seroussi, Fabian Bohnert, and Ingrid Zukerman. 2011. Personalised rating prediction for new users using latent factor models. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia*. ACM, 47–56.

[31] Parikshit Shah, Ming Yang, Sachidanand Alle, Adwait Ratnaparkhi, Ben Shahshahani, and Rohit Chandra. 2017. A practical exploration system for search advertising. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1625–1631.

[32] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Investigation of various matrix factorization methods for large recommender systems. In *IEEE International Conference on Data Mining Workshops, 2008.* IEEE, 553–562.

[33] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. 2015. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 323–332.

[34] Joaquin Vanschoren. 2018. Meta-Learning: A Survey. *arXiv preprint arXiv:1810.03548* (2018).

[35] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A Meta-Learning Perspective on Cold-Start Recommendations for Items. In *Advances in Neural Information Processing Systems*. 6904–6914.

[36] Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial Intelligence Review* 18, 2 (2002), 77–95.

[37] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *Advances in Neural Information Processing Systems*. 4957–4966.

[38] Maksims Volkovs and Guang Wei Yu. 2015. Effective latent models for binary feedback in recommender systems. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 313–322.

[39] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Lifelong Domain Word Embedding via Meta-Learning. *arXiv preprint arXiv:1805.09991* (2018).

[40] Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. 2017. R a P are: A Generic Strategy for Cold-Start Rating Prediction Problem. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1296–1309.

[41] Yuan Yao, Hanghang Tong, Guo Yan, Feng Xu, Xiang Zhang, Boleslaw K Szymanski, and Jian Lu. 2014. Dual-regularized one-class collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, 759–768.

[42] Wenpeng Yin and Hinrich Schütze. 2016. Learning Word Meta-Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Vol. 1. 1351–1360.

[43] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 73–82.

[44] Wayne Xin Zhao, Sui Li, Yulan He, Edward Y Chang, Ji-Rong Wen, and Xiaoming Li. 2016. Connecting social media to e-commerce: Cold-start product recommendation using microblogging information. *IEEE Transactions on Knowledge and Data Engineering* 28, 5 (2016), 1147–1159.

[45] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.

[46] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 315–324.