Learning Graph Meta Embeddings for Cold-Start Ads in Click-Through Rate Prediction

Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Kun Zhang, Jinmei Luo, Zhaojie Liu, Yanlong Du
Alibaba Group
{maiwei.oywt,xiuwu.zxw,shukui.rsk,ll98745,jerry.zk,cathy.jm,zhaojie.lzj,yanlong.dyl}@alibaba-inc.com

ABSTRACT

Click-through rate (CTR) prediction is one of the most central tasks in online advertising systems. Recent deep learning-based models that exploit feature embedding and high-order data nonlinearity have shown dramatic successes in CTR prediction. However, these models work poorly on cold-start ads with new IDs, whose embeddings are not well learned yet. In this paper, we propose Graph Meta Embedding (GME) models that can rapidly learn how to generate desirable initial embeddings for new ad IDs based on graph neural networks and meta learning. Previous works address this problem from the new ad itself, but ignore possibly useful information contained in existing old ads. In contrast, GMEs simultaneously consider two information sources: the new ad and existing old ads. For the new ad, GMEs exploit its associated attributes. For existing old ads, GMEs first build a graph to connect them with new ads, and then adaptively distill useful information. We propose three specific GMEs from different perspectives to explore what kind of information to use and how to distill information. In particular, GME-P uses Pre-trained neighbor ID embeddings, GME-G uses Generated neighbor ID embeddings and GME-A uses neighbor Attributes. Experimental results on three real-world datasets show that GMEs can significantly improve the prediction performance in both cold-start (i.e., no training data is available) and warm-up (i.e., a small number of training samples are collected) scenarios over five major deep learning-based CTR prediction models. GMEs can be applied to conversion rate (CVR) prediction as well.

CCS CONCEPTS

• **Information systems → Online advertising**;

KEYWORDS

Online advertising; CTR prediction; Cold start; Deep learning

ACM Reference Format:
Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Kun Zhang, Jinmei Luo, Zhaojie Liu, Yanlong Du. 2021. Learning Graph Meta Embeddings for Cold-Start Ads in Click-Through Rate Prediction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462879

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462879

1 INTRODUCTION

Click-through rate (CTR) prediction plays an important role in online advertising systems. It aims to predict the probability that a user will click on a specific ad. The predicted CTR impacts both the ad ranking strategy and the ad charging model [28, 57]. For example, the ad ranking strategy generally depends on CTR × bid, where bid is the benefit the system receives if an ad is clicked. Moreover, according to the cost-per-click (CPC) or the optimized cost-per-click (oCPC) charging model, advertisers are only charged once their ads are clicked by users. Therefore, in order to maintain a desirable user experience and to maximize the revenue, it is crucial to estimate the CTR accurately.

CTR prediction has attracted lots of attention from both academia and industry [7, 14, 29, 36, 45, 50, 55, 57]. In recent years, deep learning-based models such as Deep Neural Network (DNN) [7], Product-based Neural Network (PNN) [37], Wide&Deep [7], DeepFM [11], xDeepFM [21] and AutoInt [45] are proposed to automatically learn latent feature representations and complicated feature interactions in different manners. These models generally follow an Embedding and Multi-layer perceptron (MLP) paradigm, where an embedding layer transforms each raw input feature into a dense real-valued vector representation in order to capture richer semantics and to overcome the limitations of one-hot encoding [26].

Despite the remarkable success of these models, it is extremely data demanding to well learn the embedding vectors. It has been widely known that a well-learned embedding for an ad ID can largely improve the CTR prediction accuracy [7, 11, 15, 28, 37, 57]. When a new ad is added to the candidate pool, its ID is never seen in the training data and therefore no embedding vector is available. A randomly generated ID embedding is unlikely to lead to good prediction performance. Moreover, for ads with a small number of training samples, it is hard to train their embeddings as good as those with abundant training data. These difficulties are known as the cold-start problem in CTR prediction.

In the domain of cold-start recommendation, some methods propose to use side information, e.g., user attributes [40, 43, 54] and/or item attributes [41, 42, 47]. However, in the CTR prediction task, side information is already used. The aforementioned CTR prediction models are all feature-rich models, which already take user attributes and ad attributes as input.

Another possible way to tackle this problem is to actively collect more training data in a short time. For example, [20, 27, 44, 46] use contextual-bandit approaches and [10, 12, 34, 58] design interviews to collect specific information with active learning. However, these approaches still cannot lead to satisfactory prediction performance before sufficient training data are collected.

We tackle the cold-start problem for new ads from a different perspective, which is to generate desirable initial embeddings for

new ad IDs in a meta learning framework, even when the new ads have no training data at all. Along this line, Pan et al. propose the Meta-Embedding model [32] by exploiting the associated attributes of the new ad. However, this model only considers the new ad itself, but ignores possibly useful information contained in existing old ads that may help boost the prediction performance. Another meta learning-based model MeLU [19] is proposed to estimate a new user's preferences with a few consumed items. It locally updates a user's decision-making process based on the user's item-consumption pattern. This model does not apply to our problem and it also considers the target user alone.

In this paper, we propose Graph Meta Embedding (GME) models to learn how to generate desirable initial embeddings for new ad IDs based on graph neural networks and meta learning. GMEs contain two major components: 1) embedding generator (EG) and 2) graph attention network (GAT) [48], where the aim of EG is to generate an ID embedding and the aim of GAT is to adaptively distill information. The main idea of GMEs is to simultaneously consider two information sources: 1) the new ad itself and 2) existing old ads. For the new ad, GMEs exploit its associated attributes. For existing old ads, GMEs first build a graph to connect them with new ads, and then utilize the GAT to adaptively distill useful information. This process is non-trivial, and we propose three specific GMEs from different perspectives. In particular, GME-P uses Pre-trained neighbor ID embeddings, GME-G uses Generated neighbor ID embeddings and GME-A uses neighbor Attributes. In other words, although the three GME models all exploit the GAT, they differ in what kind of information to use and how to distill information.

In order to train GMEs, we use a gradient-based meta learning approach [32], which generalizes Model-Agnostic Meta-Learning (MAML) [9]. We view the learning of ID embedding of each ad as a task. We use meta learning because the number of unique ads is much smaller than the number of samples and we need fast adaptation. The loss function considers two aspects: 1) cold-start phase: when a new ad comes in, one should make predictions with a small loss and 2) warm-up phase: after observing a small number of labeled samples, one should speed up the model fitting to reduce the loss for subsequent prediction. As a result, GMEs can improve the CTR prediction performance on new ads in both the cold-start phase (i.e., no training data is available) and the warm-up phase (i.e., a small number of training samples are collected).

The main contributions of this work are summarized as follows:

- We address the cold-start CTR prediction problem for new ads from a different perspective. The main idea is to build an ad graph and learn to generate desirable initial embeddings for new ad IDs by taking into account of both the new ad itself and other related old ads over the graph.
- We propose three specific Graph Meta Embedding (GME) models to generate initial embeddings for new ad IDs from different perspectives. In particular, GME-P uses Pre-trained neighbor ID embeddings, GME-G uses Generated neighbor ID embeddings and GME-A uses neighbor Attributes. We make the implementation code publicly available[1].

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

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

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

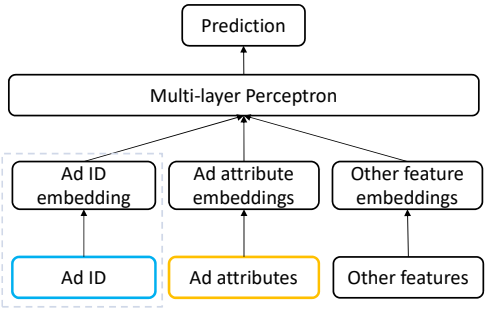


Figure 1: Illustration of typical deep CTR prediction models.

- We conduct experiments on three large-scale real-world datasets. Experimental results show that GMEs can significantly improve the prediction performance in both cold-start and warm-up scenarios over five major deep learning-based CTR prediction models.

2 BACKGROUND

2.1 Problem Formulation

The task of **CTR prediction** in online advertising is to build a prediction model to estimate the probability of a user clicking on a specific ad. Each instance can be described by multiple *fields* such as user information ("User ID", "City", "Age", etc.) and ad information ("Ad ID", "Category", "Title", etc.). The instantiation of a field is a *feature*. For example, the "User ID" field may contain features such as "2135147" and "3467291". Table 1 shows some examples.

During model training, we learn parameters corresponding to training features. After that, we make predictions on test data by using these learned parameters. However, new ads have features that are not seen in the training data, e.g., the ad ID. This causes the **cold-start CTR prediction** problem of new ads, where we need to make predictions in the absence of certain model parameters. We will make this problem more concrete in the context of deep CTR prediction models introduced below.

2.2 Typical Deep CTR Prediction Models

Typical deep learning-based CTR prediction models such as Deep Neural Network (DNN) [7], Product-based Neural Network (PNN) [37], Wide&Deep [7], DeepFM [11], xDeepFM [21] and AutoInt [45] all follow an Embedding and MLP paradigm (Figure 1). We present the modules of DNN below as an example.

Input: The input to the model is feature indices $\{i\}$.

Embedding layer: $i \rightarrow \mathbf{e}_i$. This module encodes the input into dense vector representations (i.e., embeddings) $\mathbf{e}_i$ through an embedding matrix $\mathbf{E}$ (to be learned). The ith column of the

Table 2: List of notations.

Notation	Meaning
ID_0	ID of the new ad
$\mathbf{x}_0$	associated attributes of the new ad
$\mathbf{z}_0$	concatenated embedding vector acc. to $\mathbf{x}_0$
$\tilde{\mathbf{z}}_0$	refined embedding vector w.r.t. $\mathbf{z}_0$
$\mathbf{g}_0$	generated (preliminary) ID emb. of the new ad
$\mathbf{r}_0$	initial ID emb. of the new ad in CTR prediction
ID_i	ID of the ith ngb. ($i = 1, \cdots, N$)
$\mathbf{x}_i$	associated attributes of the ith ngb.
$\mathbf{z}_i$	concatenated embedding vector acc. to $\mathbf{x}_i$
$\mathbf{p}_i$	pre-trained ID embedding of the ith ngb.
$\mathbf{g}_i$	generated ID embedding of the ith ngb.

embedding matrix $\mathbf{E}$ holds the embedding vector for the ith feature. The embedding vector $\mathbf{e}_i$ for feature index i is given by $\mathbf{e}_i = \mathbf{E}[:, i]$.

Concatenation layer: $\{\mathbf{e}_i\} \rightarrow \mathbf{s}$. This module concatenates the embeddings of all the input features as a long embedding vector $\mathbf{s} = [\mathbf{e}_1 \| \mathbf{e}_2 \| \mathbf{e}_3 \| \cdots]$, where $\|$ is the concatenation operator.

Hidden layers: $\mathbf{s} \rightarrow \mathbf{s}'$. This module transforms the long embedding vector $\mathbf{s}$ into a high-level representation vector $\mathbf{s}'$ through several fully-connected (FC) layers to exploit data nonlinearity and high-order feature interactions. In particular, $\mathbf{s}' = f_L(\cdots f_2((f_1(\mathbf{s}))))$, where L is the number of FC layers and f_j is the jth FC layer.

Prediction layer: $\mathbf{s}' \rightarrow \hat{y}$. This module predicts the click-through probability $\hat{y} \in [0, 1]$ of the instance based on the high-level representation vector $\mathbf{s}'$ through a sigmoid function.

Model training: The model parameters are learned through the cross-entropy loss on a training data set $\mathbb{Y}$. The loss function is

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

where $y \in \{0, 1\}$ is the true label corresponding to $\hat{y}$.

When a new ad comes in, its ID has not been trained yet and the model cannot find its embedding in the embedding matrix. In order to predict the CTR, a commonly used approach is to randomly generated an embedding for the new ad ID. However, this approach usually leads to poor prediction performance.

3 MODEL DESIGN

In the following, we propose Graph Meta Embedding (GME) models to learn how to generate desirable initial embeddings (i.e., not random) for new ad IDs based on graph neural networks and meta learning. These initial embeddings can lead to improved CTR prediction performance in both cold-start and warm-up scenarios.

3.1 Overview

For ease of presentation, we list the notations used in Table 2.

The proposed GME models are only activated for new ads. We illustrate the difference of ID embeddings for old ads and new ads in Figure 2. When an ad ID is given, we first lookup the trained embedding matrix. If the ID's embedding can be found, then it is an old ad and we use the found embedding [Figure 2(a)]. Otherwise, it is a new ad and we activate a GME model to generate an initial embedding for the ID by using the attributes of the new ad and information from its graph neighbors [Figure 2(b)].

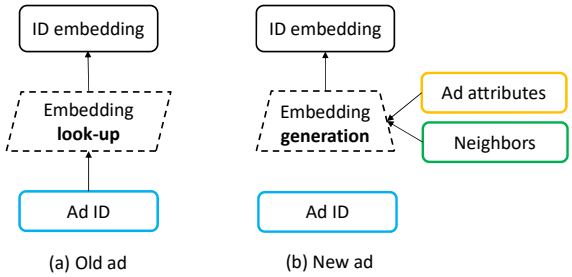


Figure 2: Illustration of ID embedding of old and new ads.

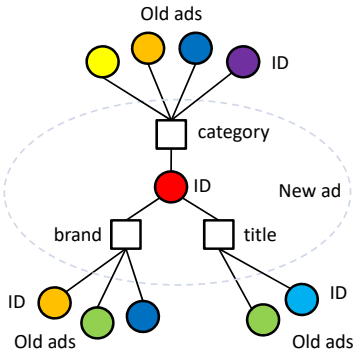


Figure 3: Illustration of the ad graph. Connections between ads are established based on their shared features. The dashed circle illustrates an ad with the ID and three attribute features (e.g., category, brand and title) shown.

GMEs contain two major components: 1) embedding generator (EG) and 2) graph attention network (GAT) [48], where the aim of EG is to generate an ID embedding and the aim of GAT is to adaptively distill information. GME models differ in what kind of information to use and how to distill information.

For convenience, we use the same notations for model parameters ($\mathbf{W}$, $\mathbf{V}$ and $\mathbf{a}$) in the following. Parameters with the same notation in different models have the same functionality, but possibly different dimensions (which are clear in each specific context).

3.2 Graph Creation

As the GME models utilize both the new ad and related old ads, the first step is to build a graph to connect them. However, unlike social networks where exist follower-followee or friendship relationships, there is no natural graph on ads. One possible way is to exploit the co-click relationships, but this approach is clearly not suitable for new ads. In this paper, we build connections between ads based on their features (illustrated in Figure 3).

Typically, one can use an adjacency matrix $\mathbf{A}$ [4], where the ith row represents a new ad i, the jth column represents an existing old ad j and $[\mathbf{A}]_{ij}$ is the adjacency score between i and j. This approach is highly time-consuming because it needs to repeatedly scan the whole set of existing old ads for each new ad.

Instead, we use the following approach for fast graph creation. Given a new ad, we obtain its ID and associated attributes (e.g., category, brand and title). For each attribute, we can retrieve old ads which have the same attribute. The union of these old ads then form the graph neighbors of the new ad.

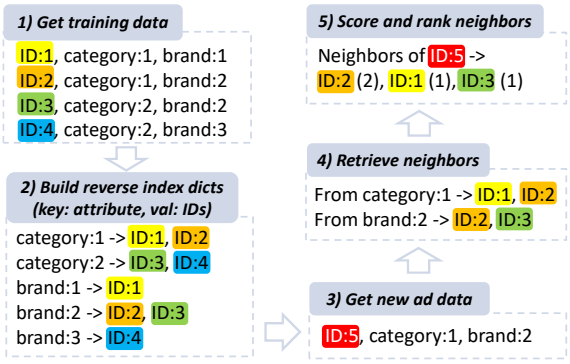


Figure 4: Illustration of fast graph creation.

In particular, we implement this idea as follows (summarized in Figure 4, where Steps 1-2 are performed only once for old ads).

- Build a reverse index dictionary, where the key is the attribute and the value is the set of ad IDs which have this attribute. For example, "category:1 → ID:1, ID:2"; "brand:2 → ID:2, ID:3".
- Given a new ad, retrieve its neighbors based on each attribute. For example, the new ad is "ID:5, category:1, brand:2", we then retrieve its neighbors based on the two attributes. The retrieved neighbors are ID:1, ID:2 and ID:3.
- Calculate the similarity score w.r.t. each neighbor and keep the top-N neighbors (break the tie randomly). We define the score as the number of attributes that the neighbor can be retrieved from. For example, ID:2 has a score as 2 because it can be retrieved from 2 attributes. This step can keep the most useful neighbors for subsequent processing.

The above approach is much faster because we only need to scan the set of old ads once (to build the reserve index dictionary) instead of multiple times. Without loss of generality, we denote the new ad as ID_0 and the set of its graph neighbors as $\mathcal{N} = \{ID_i\}_{i=1}^{N}$.

3.3 GME-P: Using Pre-trained Neighbor ID Embeddings

The first GME model we present is GME-P, which exploits the attributes of the new ad and the pre-trained ID embeddings of neighboring old ads. We illustrate its structure in Figure 5(a).

The idea is that: As we have obtained the pre-trained ID embeddings $\{\mathbf{p}_i\}$ of neighboring old ads by the main CTR prediction model (e.g. DNN in §2.2), we would like to exploit useful information in these embeddings. However, we only have attribute embeddings rather than ID embedding of the new ad. Therefore, we first generate a preliminary ID embedding $\mathbf{g}_0$ for the new ad by using its associated attributes. As both $\{\mathbf{p}_i\}$ and $\mathbf{g}_0$ are ID embeddings, we can then leverage useful information contained in $\{\mathbf{p}_i\}$ to improve $\mathbf{g}_0$ and obtained a refined ID embedding $\mathbf{r}_0$ for the new ad. Formally, GME-P contains the following two steps.

3.3.1 ***ID Embedding Generation.*** We generate a preliminary ID embedding for a cold-start ad by using its associated attribute features (such as category, brand and title) instead of randomly. Formally, let's denote the features of an instance as $[ID_0, \mathbf{x}_0, \mathbf{o}_0]$, where ID_0 is the identity of the new ad, $\mathbf{x}_0$ is the ad attribute features, and $\mathbf{o}_0$ is other features which do not necessarily relate to the ad such as user features and context features.

Although ID_0 of the new ad is not seen in the training data, the associated ad attributes $\mathbf{x}_0$ are usually observed. We then lookup the embeddings corresponding to $\mathbf{x}_0$ and obtain a long concatenated embedding vector $\mathbf{z}_0$. Based on $\mathbf{z}_0$, we generate a preliminary embedding $\mathbf{g}_0$ for ID_0 through an embedding generator (EG) which implements $\mathbf{g}_0 = f(\mathbf{z}_0)$. We use a simple instantiation of the EG as

$$\mathbf{g}_0 = \gamma \tanh(\mathbf{W}\mathbf{z}_0), \tag{2}$$

where $\mathbf{W}$ is the parameter (to be learned) of a fully connected layer, tanh is the activation function and $\gamma \in (0, 1]$ is a scaling hyperparameter. We use γ to restrict the range of $\mathbf{g}_0$ in $[-\gamma, \gamma]$.

3.3.2 ***ID Embedding Refinement.*** We then generate a refined ID embedding $\mathbf{r}_0$ for the new ad based on its preliminary ID embedding $\mathbf{g}_0$ and pre-trained ID embeddings $\{\mathbf{p}_i\}$ of its neighbors.

A simple way is to take the average of these ID embeddings and obtain $\mathbf{r}_0 = \text{average}(\mathbf{g}_0, \mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N)$. But this is clearly not a wise choice because some old ads may not be quite informative.

Alternatively, we resort to the Graph Attention Network (GAT) [48], which is proposed to operate on graph-structured data and to learn high-level data representations. It allows for assigning different importances to different graph nodes within a neighborhood through the attention mechanism [2] while dealing with different sized neighborhoods.

We first compute the attention coefficient between $\mathbf{g}_0$ and $\mathbf{p}_i$ as

$$c_{0i} = \mathcal{F}(\mathbf{V}\mathbf{g}_0, \mathbf{V}\mathbf{p}_i),$$

where $\mathcal{F}$ is a function to implement the attention mechanism and $\mathbf{V}$ is a shared weight parameter which transforms the input into higher-level features and obtains sufficient expressive power. We also compute the attention coefficient for the new ad itself as

$$c_{00} = \mathcal{F}(\mathbf{V}\mathbf{g}_0, \mathbf{V}\mathbf{g}_0).$$

To make coefficients easily comparable across different nodes, we normalize them using the softmax function. We implement the attention mechanism $\mathcal{F}$ using a single-layer feedforward neural network, parameterized by a weight vector $\mathbf{a}$, and applying the LeakyReLU nonlinearity (with negative input slope 0.2) [48]. The normalized coefficients α_{0i} can then be expressed as

$$\alpha_{0i} = \frac{\exp(c_{0i})}{\sum_{j=0}^{N}\exp(c_{0j})} = \frac{\exp\left(LeakyReLU(\mathbf{a}^T[\mathbf{V}\mathbf{g}_0 \| \mathbf{V}\mathbf{p}_i])\right)}{\sum_{j=0}^{N}\exp\left(LeakyReLU(\mathbf{a}^T[\mathbf{V}\mathbf{g}_0 \| \mathbf{V}\mathbf{p}_j])\right)},$$

where we define $\mathbf{p}_0 \triangleq \mathbf{g}_0$ for notational simplicity. LeakyReLU allows to encode both positive and small negative signals [24].

Note that the index j ranges from 0 to N. That is, the summation includes the new ad itself (index 0) and its neighbors (index 1 to N).

We then compute a weighted sum of the preliminary ID embedding $\mathbf{g}_0$ of the new ad (with importance α_{00}) and the pre-trained ID embeddings $\{\mathbf{p}_i\}$ of neighbors (with importance α_{0i}), to serve as the refined ID embedding $\mathbf{r}_0$ for the new ad as

$$\mathbf{r}_0 = ELU\left(\sum_{i=0}^{N}\alpha_{0i}\mathbf{V}\mathbf{p}_i\right),$$

where ELU is the exponential linear unit activation function [48], it also allows to encode both positive and small negative signals.

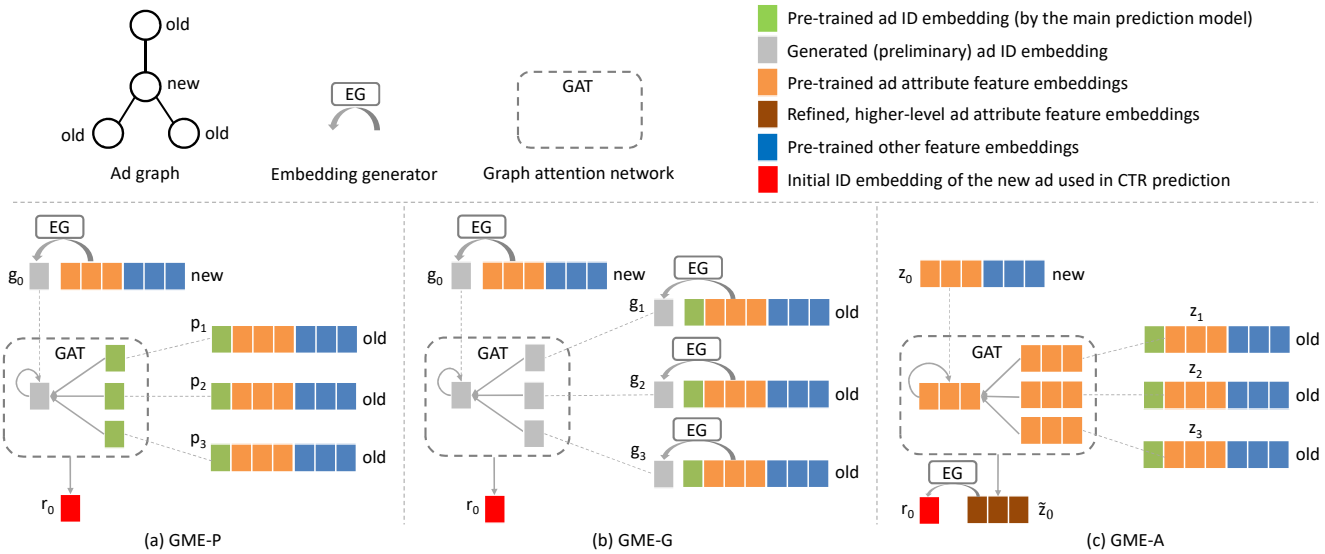


Figure 5: Graph Meta Embedding (GME) models (best viewed in color). Please refer to §3.3 to §3.5 for more details.

3.3.3 ***Analysis.*** GME-P seems reasonable. However, the pre-training process (e.g., based on the DNN model) does not impose any constraint between attributes and the ID embedding as in Eq. (2) and all the embeddings are randomly initialized. It is possible that given the same attributes, the corresponding $\mathbf{p}_0$ and $\mathbf{g}_i$ are quite different (because they correspond to two different IDs). It makes the attention computation between $\mathbf{p}_0$ and $\mathbf{g}_i$ meaningless.

3.4 GME-G: Using Generated Neighbor ID Embeddings

To overcome the limitation of GME-P, we propose GME-G in this section. Instead of using pre-trained ID embeddings of old ads, GME-G reuses the EG for the new ad, and generates ID embeddings $\{\mathbf{g}_i\}$ for old ads using their corresponding attributes as well. We illustrate its structure in Figure 5(b).

3.4.1 ***ID Embedding Generation.*** We use the same EG to generate the preliminary ID embedding $\mathbf{g}_0$ for the new ad and the ID embeddings $\{\mathbf{g}_i\}$ for existing old ads as

$$\mathbf{g}_0 = \gamma \tanh(\mathbf{W}\mathbf{z}_0), \ \mathbf{g}_i = \gamma \tanh(\mathbf{W}\mathbf{z}_i).$$

By doing so, we can guarantee that when ad attributes are the same (i.e., $\mathbf{z}_i = \mathbf{z}_0$), we have $\mathbf{g}_i = \mathbf{g}_0$. Subsequently, the attention computation between $\mathbf{g}_i$ and $\mathbf{g}_0$ makes more sense.

3.4.2 ***ID Embedding Refinement.*** The attention coefficients α_{0i} between the new ad and the ith neighboring old ad is then given by

$$\alpha_{0i} = \frac{\exp\left(LeakyReLU(\mathbf{a}^T[\mathbf{V}\mathbf{g}_0 \| \mathbf{V}\mathbf{g}_i])\right)}{\sum_{j=0}^{N} \exp\left(LeakyReLU(\mathbf{a}^T[\mathbf{V}\mathbf{g}_0 \| \mathbf{V}\mathbf{g}_j])\right)}.$$

Finally, the refined ID embedding $\mathbf{r}_0$ for the new ad is given by a linear combination of the generated ID embedding of the new ad and those of the neighboring old ads as

$$\mathbf{r}_0 = ELU\left(\sum_{i=0}^{N} \alpha_{0i}\mathbf{V}\mathbf{g}_i\right).$$

3.4.3 ***Analysis.*** GME-G does overcome the limitation of GME-P and it makes the attention coefficients between the new ad and old ads meaningful. However, GME-G repeatedly performs ID embedding generation for old ads. As the generated ID embedding could contain certain noise, the repetition can spread the noise.

3.5 GME-A: Using Neighbor Attributes

Given the limitation of GME-G, we further propose GME-A in this section, whose structure is shown in Figure 5(c). GME-A reverses the order of the "generation" step and the "refinement" step. Moreover, GME-A refines the attribute representation rather than the preliminary ID embedding.

3.5.1 ***Attribute Embedding Refinement.*** GME-A first obtains a refined attribute representation of the new ad, which aggregates useful information from the new ad itself and its neighboring old ads on the attribute level. Formally, the attention coefficients between the new ad and the ith neighboring old ad is computed based on attribute embedding vectors $\mathbf{z}_0$ and $\mathbf{z}_i$ as

$$\alpha_{0i} = \frac{\exp\left(LeakyReLU(\mathbf{a}^T[\mathbf{V}\mathbf{z}_0 \| \mathbf{V}\mathbf{z}_i])\right)}{\sum_{j=0}^{N} \exp\left(LeakyReLU(\mathbf{a}^T[\mathbf{V}\mathbf{z}_0 \| \mathbf{V}\mathbf{z}_j])\right)}.$$

We then obtain a refined, high-level attribute embedding vector $\tilde{\mathbf{z}}_0$ for the new ad by performing a linear combination of the original embedding vectors as

$$\tilde{\mathbf{z}}_0 = ELU\left(\sum_{i=0}^{N} \alpha_{0i}\mathbf{V}\mathbf{z}_i\right).$$

3.5.2 ***ID Embedding Generation.*** Given this refined attribute representation, we then generate the initial ID embedding of the new ad as

$$\mathbf{r}_0 = \gamma \tanh(\mathbf{W}\tilde{\mathbf{z}}_0).$$

3.5.3 ***Analysis.*** GME-A directly compares the attributes of the new ad and neighboring old ads, thus avoiding the "incomparable"

problem between the generated ID embedding and the pre-trained ID embeddings in GME-P. GME-A only uses the EG once, thus also avoiding the "repetition" issue in GME-G.

3.6 Model Learning

We first train a main model (e.g., DNN) for CTR prediction using old ads. We then obtain the model parameters Θ, including the embedding vectors of features and other weight parameters. As Θ is usually trained with a large amount of data, we are confident about its effectiveness. Therefore, when training the GME models, we freeze Θ and only learn the parameters $\Psi \triangleq \{\mathbf{W}, \mathbf{V}, \mathbf{a}\}$ that are specific to these models.

As can be seen, the number of unique ad IDs matters in the training of parameters Ψ. As the number of unique ad IDs is much smaller than the number of samples, we resort to meta learning for fast adaptation. We view the learning of ID embedding of each ad as a task and use a gradient-based meta learning approach [32], which generalizes Model-Agnostic Meta-Learning (MAML) [9].

The loss consider two aspects: 1) the error of CTR prediction for the new ad should be small and 2) after a small number of labeled examples are collected, a few gradient updates should lead to fast learning. This is achieved by combining the following two losses l_a and l_b.

For a given training old ad ID_0, we randomly select two disjoint minibatches of labeled data $\mathcal{D}^a$ and $\mathcal{D}^b$, each with M samples. We first make predictions using the initial ID embedding $\mathbf{r}_0$ produced by a GME model on the first minibatch $\mathcal{D}^a$. For the jth sample, we obtain its prediction as $\hat{y}_{aj}$. The average loss over these samples is given by

$$l_a = \frac{1}{M}\sum_{j=1}^{M}[-y_{aj}\log\hat{y}_{aj} - (1-y_{aj})\log(1-\hat{y}_{aj})],$$

where y_{aj} is the true label.

Next, by computing the gradient of l_a w.r.t. the initial embedding and taking a step of gradient descent, we get a new adapted embedding

$$\mathbf{r}_0' = \mathbf{r}_0 - \eta\frac{\partial l_a}{\partial \mathbf{r}_0},$$

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

We then test this new adapted embedding $\mathbf{r}_0'$ on the second minibatch $\mathcal{D}^b$, and obtain the average loss

$$l_b = \frac{1}{M}\sum_{j=1}^{M}[-y_{bj}\log\hat{y}_{bj} - (1-y_{bj})\log(1-\hat{y}_{bj})].$$

The final loss for learning the parameters Ψ is given by

$$l = \beta l_a + (1-\beta)l_b,$$

where $\beta \in [0,1]$ is a coefficient to balance the two losses that consider the aforementioned two aspects.

4 EXPERIMENTS

4.1 Datasets

We evaluate the performance of the proposed GME models on three real-world datasets, whose statistics are listed in Table 3.

(1) **MovieLens-1M (ML-1M) dataset**[2]**.** It is one of the most well-known benchmark dataset. This dataset contains 1 million movie rating instances over thousands of movies and users. Each movie has an ID and can be seen as an ad in our scenario. The associated attribute features include year of release, title and genres. Other features include user ID, gender, age and occupation. We convert the ratings that are at least 4 to label 1 and others to label 0. This is a common practice for evaluation in implicit feedback scenarios such as CTR prediction [13].

(2) **Taobao ad dataset**[3]**.** It is gathered from the traffic logs in Taobao [23] and is originally used for the conversion rate (CVR) prediction task. Each ad has an ID and the associated attribute features include category ID, shop ID, brand ID and intention node ID. Other features include user features and context features such as user ID, gender, age and categorical ID of user profile.

(3) **News feed ad dataset.** It is gathered from an industrial news feed advertising system and is used for CTR prediction. Each ad has an ID and the associated attribute features include industry ID, source ID, account ID and title. Other features include user features and context features such as user ID, gender, age and OS.

4.2 Experimental Settings

4.2.1 ***Main CTR Prediction Models.*** Because GMEs are model-agnostic (they only generate initial embeddings for new ad IDs), they can be applied upon various existing CTR prediction models that require feature embeddings. We conduct experiments on the following representative CTR prediction models:

(1) **DNN**. Deep Neural Network in [7]. It contains an embedding layer, several FC layers and an output layer.
(2) **PNN**. Product-based Neural Network in [37]. It introduces a production layer into DNN.
(3) **Wide&Deep**. Wide&Deep model in [7]. It combines logistic regression (LR) and DNN.
(4) **DeepFM**. DeepFM model in [11]. It combines factorization machine (FM) [38] and DNN.
(5) **AutoInt**. AutoInt model in [45]. It consists of a multi-head self-attentive network with residual connections and DNN.

There are other CTR prediction models that take additional information into consideration. For example, Deep Interest Network (DIN) [57] models user interest based on historical click behavior. Deep Spatio-Temporal Network (DSTN) [28] jointly exploits contextual ads, clicked ads and unclicked ads for CTR prediction. As most datasets do not contain behavior sequence information or position information, we do not include these models in our experiments.

4.2.2 ***Cold-Start ID Embedding Models.*** For each main CTR prediction model, we evaluate the following cold-start ID embedding models, which generate initial embeddings for new ad IDs.

(1) **RndEmb**. It uses a randomly generated embedding for the new ad ID.
(2) **MetaEmb**. MetaEmbedding model in [32]. It uses the attributes $\mathbf{x}_0$ of the new ad to generate an initial embedding of the new ad ID. MetaEmb serves as a baseline which only considers the new ad.

[2] http://www.grouplens.org/datasets/movielens/
[3] https://tianchi.aliyun.com/dataset/dataDetail?dataId=408

Table 3: Statistics of experimental datasets.

Dataset	# fields	# old ad IDs	# samples to train the main prediction model	# old ad IDs	# samples to train the cold-start ID embedding model	# new ad IDs	# samples for warm up training	# samples for testing
ML-1M	8	1,058	765,669	1,058	42,320	1,127	67,620	123,787
Taobao	23	62,209	835,450	3,177	254,160	531,593	808,806	896,615
News feed	30	5,563	3,088,542	1,761	352,000	8,379	603,335	1,346,504

(3) **NgbEmb**. It uses pre-trained ID embeddings of neighboring old ads to generate an initial ID embedding of the new ad as $\gamma \tanh(\mathbf{W}\frac{1}{N}\sum_{i=1}^{N}\mathbf{p}_i)$. NgbEmb serves as a baseline which only considers neighbor information.
(4) **GME-P**. Graph Meta Embedding model which uses Pre-trained ID embeddings $\{\mathbf{p}_i\}$ of neighboring old ads and the attributes $\mathbf{x}_0$ of the new ad to generate an initial embedding of the new ad ID. It is described in §3.3.
(5) **GME-G**. Graph Meta Embedding model which uses Generated ID embeddings $\{\mathbf{g}_i\}$ of the neighboring old ads and the attributes $\mathbf{x}_0$ of the new ad to generate an initial embedding of the new ad ID. It is described in §3.4.
(6) **GME-A**. Graph Meta Embedding model which uses the Attributes $\{\mathbf{x}_i\}$ of neighboring old ads and the attributes $\mathbf{x}_0$ of the new ad to generate an initial embedding of the new ad ID. It is described in §3.5.

4.2.3 ***Parameter Settings.*** We set the dimension of the embedding vector for each feature as 10, the balancing parameter as $\beta = 0.1$ and the number of graph neighbors for each ad as $N = 10$. For an ad ID, if the number of labeled instances is larger than a threshold, we regard it as an old ad. This threshold is set to 300, 40 and 100 for the three datasets respectively. Old ads are used to train the main CTR prediction model. We further sample old ads to train the cold-start ID embedding models, where each old ad has 20, 40 and 100 samples in each minibatch for the three datasets respectively. For the new ads, we hold out a proportion for warm up training (also serve as validation data) and use the remaining for testing. Details are listed in Table 3. All the models are implemented in Tensorflow [1] and optimized by the Adam algorithm [17]. We run each model 3 times and report the average result.

4.2.4 ***Evaluation Metrics.***

(1) **AUC**: Area Under the ROC Curve over the test set. It is a widely used metric for CTR prediction. It reflects the probability that a model ranks a randomly chosen positive instance higher than a randomly chosen negative instance. The larger the better. A small improvement in AUC is likely to lead to a significant increase in online CTR [7, 11, 28, 57].
(2) **Loss**: the value of Eq. (1) of the main prediction model over the test set. The smaller the better.

4.3 Performance Comparison

4.3.1 ***Effectiveness in the Cold-Start Phase.*** Table 4 lists the performance of various ID embedding models based on different CTR prediction models in the cold-start phase. It is observed that MetaEmb performs better than RndEmb, showing that using associated attributes of the new ad can contribute useful information and alleviate the cold-start problem. NgbEmb sometimes performs better and sometimes performs worse than MetaEmb, showing that simply considering the average of pre-trained neighbor ID embeddings is not quite effective.

GME-P leads to marginal performance improvement or even degraded performance compared with MetaEmb. It is because the pre-trained neighbor ID embeddings and the generated ID embedding from ad attributes are incomparable. As a consequence, GAT in GME-P cannot well extract useful information from neighbors.

In contrast, GME-G performs much better than MetaEmb. Different from GME-P, GME-G uses generated rather than pre-trained neighbor ID embeddings. As the preliminary ID embedding of the new ad is also generated from ad attributes, these embeddings are comparable. GAT can thus distill informative signals from neighbor ID embeddings and improve the new ad's ID embedding. GME-A further outperforms GME-G in most cases. It is because GME-A directly aggregates useful information from the neighbors on the attribute level and avoids the "repetition" issue in GME-G.

These results demonstrate that considering neighbor information and appropriately distilling useful information from them could help alleviate the cold-start problem of new ads.

4.3.2 ***Effectiveness in the Warm-up Phase.*** Figure 6 plots the performance of various models in the warm-up phase. We perform two rounds of warm-up training. In the first warm-up training, we provide a small number of training examples (related to new ads) to the main CTR models, but with different initial ID embeddings given by different embedding generation models. In the second warm-up training, we provide another small number of training examples (related to new ads) to the main CTR models, but based on different ID embeddings learned after the first warm-up training. It is observed that a model that results in good performance in the cold-start phase generally leads to good performance in the warm-up phase. GME-A not only performs best in the cold-start phase, but also in the two warm-up rounds.

4.4 Ablation Studies

4.4.1 ***Effect of the Scaling Parameter.*** Figure 7 plots the AUC of various models vs. the value of the scaling parameter γ. On the ML-1M dataset, it is observed that GME-P is relatively insensitive to γ. Differently, GME-G and GME-A perform much better when γ is large. On the Taobao dataset, GME-P performs better when γ is small while GME-G and GME-A perform better when γ is large. GME-A performs well on a relatively wide range of γ values.

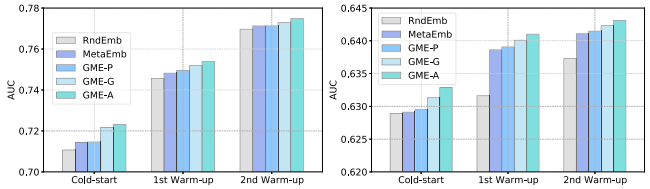
4.4.2 ***Effect of the Number of Neighbors.*** Figure 8 plots the AUC of various models vs. the number of graph neighbors. It is observed that generally when more neighbors are available, the

Table 4: Test AUC and Loss. Pred. model: Prediction model. Emb. model: ID embedding generation model. AUC (↑) is the larger the better. Loss (↓) is the smaller the better.

		ML-1M		Taobao		News Feed	
Pred. model	**Emb. model**	AUC	Loss	AUC	Loss	AUC	Loss
DNN	RndEmb	0.7107	0.6491	0.6289	.03177	0.7350	.03602
	MetaEmb	0.7144	0.6439	0.6291	.03177	0.7362	.03578
	NgbEmb	0.7131	0.6442	0.6294	.03177	0.7356	.03601
	GME-P	0.7146	0.6437	0.6295	.03177	0.7358	.03602
	GME-G	0.7217	0.6389	0.6323	.03172	0.7371	.03562
	GME-A	**0.7232**	**0.6368**	**0.6336**	**.03168**	**0.7389**	**.03553**
PNN	RndEmb	0.7162	0.6260	0.6325	.03172	0.7334	.03681
	MetaEmb	0.7164	0.6256	0.6327	.03172	0.7365	.03669
	NgbEmb	0.7163	0.6254	0.6330	.03171	0.7329	.03684
	GME-P	0.7164	0.6258	0.6330	.03172	0.7352	.03672
	GME-G	0.7172	0.6261	0.6343	.03166	0.7381	.03623
	GME-A	**0.7198**	**0.6233**	**0.6354**	**.03161**	**0.7392**	**.03617**
Wide& Deep	RndEmb	0.7122	0.6509	0.6305	.03164	0.7368	.03565
	MetaEmb	0.7149	0.6510	0.6306	.03164	0.7381	.03561
	NgbEmb	0.7125	0.6512	0.6306	.03165	0.7354	.03567
	GME-P	0.7149	0.6510	0.6306	.03166	0.7375	.03529
	GME-G	0.7166	0.6487	0.6332	**.03142**	0.7404	.03514
	GME-A	**0.7179**	**0.6425**	**0.6338**	.03143	**0.7413**	**.03503**
DeepFM	RndEmb	0.7143	0.6462	0.6294	.03174	0.7315	.03584
	MetaEmb	0.7146	0.6484	0.6297	.03171	0.7352	.03538
	NgbEmb	0.7142	0.6467	0.6299	.03171	0.7321	.03585
	GME-P	0.7146	0.6478	0.6298	.03175	0.7346	.03541
	GME-G	0.7195	0.6457	0.6337	**.03157**	0.7378	.03524
	GME-A	**0.7206**	**0.6449**	**0.6345**	.03160	**0.7389**	**.03517**
AutoInt	RndEmb	0.7152	0.6322	0.6331	.03193	0.7381	.03685
	MetaEmb	0.7167	0.6224	0.6336	.03166	0.7401	.03672
	NgbEmb	0.7154	0.6251	0.6335	.03164	0.7377	.03691
	GME-P	0.7168	0.6262	0.6335	.03167	0.7394	.03676
	GME-G	0.7204	0.6245	0.6402	.03154	0.7416	.03659
	GME-A	**0.7223**	**0.6218**	**0.6411**	**.03151**	**0.7432**	**.03647**

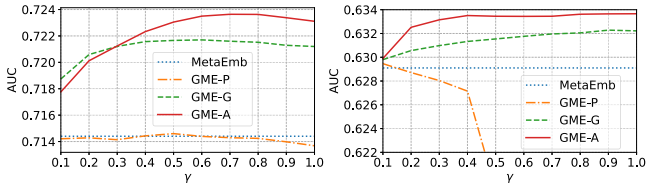
performance of various GME models also improves. But the performance may become flattened with enough number of neighbors, e.g., the performance of GME-G does not change much when the number of neighbors ranges from 6 to 10 on the Taobao dataset. Moreover, some GME models may not outperform MetaEmb when the number of neighbors is too small (e.g., 2 neighbors on the Taobao dataset). This is possibly because the neighbors also contain noisy information and it is hard to extract enough useful information from too few neighbors. Therefore, an enhanced approach to retrieving graph neighbors may lead to further improved performance.

4.4.3 ***Effect of the GAT.*** Table 5 lists the AUC of the GME models with and without the GAT component. When GAT is not used, we aggregates the corresponding representations using average pooling. It is observed that the inclusion of GAT can highly boost the AUC. For example, on the ML-1M dataset, GME-A performs much better than GME-A\GAT. Moreover, GME-A\GAT only slightly outperforms GME-G\GAT. But GME-A largely outperforms GME-G. These results show that GAT can better extract useful information from neighbors than simple average pooling by assigning different importance according to different neighbors' properties. Moreover, applying GAT on ad attributes leads to better performance than applying GAT on generated ID embeddings.



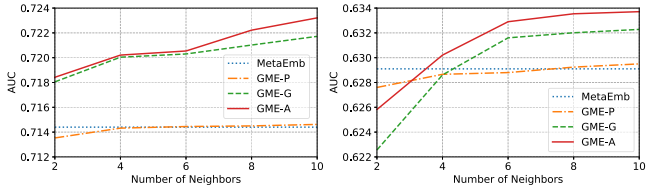
(a) ML-1M (b) Taobao

Figure 6: Performance in the warm-up phase. Main prediction model: DNN.



(a) ML-1M (b) Taobao

Figure 7: Effect of the scaling parameter. Main prediction model: DNN.



(a) ML-1M (b) Taobao

Figure 8: Effect of the number of neighbors. Main prediction model: DNN.

Table 5: Effect of the GAT. Main prediction model: DNN.

	ML-1M		Taobao		News Feed	
Emb. Model	AUC	Loss	AUC	Loss	AUC	Loss
GME-P\GAT	0.7131	0.6447	0.6282	.03208	0.7343	.03636
GME-P	**0.7146**	**0.6437**	**0.6295**	**.03177**	**0.7358**	**.03602**
GME-G\GAT	0.7154	0.6435	0.6301	.03176	0.7355	.03598
GME-G	**0.7217**	**0.6389**	**0.6323**	**.03172**	**0.7371**	**.03562**
GME-A\GAT	0.7156	0.6434	0.6304	.03176	0.7358	.03596
GME-A	**0.7232**	**0.6368**	**0.6336**	**.03168**	**0.7389**	**.03553**

4.5 Lessons Learned

We discuss some lessons learned during the experimentation with GME models.

(1) **Importance of ad IDs.** One would apply the GME models only when the missing of ad IDs impacts the prediction performance significantly. It depends on the property of each specific dataset. In other words, if the exclusion of ad IDs does not degrade the AUC significantly, there is no need to generate better initial embeddings for these IDs.

(2) **Intrinsic ad attributes.** Ad attributes used to create the ad graph and to generate initial ID embeddings should be intrinsic ad attributes. That is to say, given an ad ID, the associated ad attributes should not change in different samples. Otherwise, we use some changing attributes to generate a fixed ID embedding vector, the model would not be well trained. For example, a specific ad may be displayed at position 1 in one impression and then at position 2 in another impression. Ad position thus can not be used in the aforementioned processes.

(3) **Positive samples.** When training the ID embedding models with meta learning, there should be some positive samples in most minibatches. One can set the number M larger for datasets with a small proportion of positive samples, perform random sampling multiple times and train the model multiple rounds.

5 RELATED WORK

CTR prediction. The task of CTR prediction in online advertising is to estimate the probability of a user clicking on a specific ad.

As generalized linear models such as Logistic Regression (LR) [39] and Follow-The-Regularized-Leader (FTRL) [25] lack the ability to learn sophisticated feature interactions [5], Factorization Machine (FM) [3, 38], Field-aware FM [15] and Field-weighted FM [33] are proposed to address this limitation.

In recent years, deep learning-based models such as Deep Neural Network (DNN) [7], Product-based Neural Network (PNN) [37], Wide&Deep [7], DeepFM [11], xDeepFM [21] and AutoInt [45] are proposed to automatically learn latent feature representations and complicated feature interactions in different manners. Deep Matching and Prediction (DeepMP) model [30] combines two subnets to learn more representative feature embeddings for CTR prediction.

Some other models exploit auxiliary information. For example, Deep Interest Network (DIN) [57] and Deep Interest Evolution Network (DIEN) [56] model user interest based on historical click behavior. Xiong et al. [51] and Yin et al. [53] consider various contextual factors such as ad interaction, ad depth and query diversity. Deep Spatio-Temporal Network (DSTN) [28] jointly exploits contextual ads, clicked ads and unclicked ads for CTR prediction. Mixed Interest Network (MiNet) [31] models long- and short-term interests in the news and ads for cross-domain CTR prediction.

However, these models do not specifically address the cold-start problem and they usually have unsatisfactory performance on new ads whose IDs are not seen in the training data.

Cold-start recommendation / Cold-start CTR prediction. Recommender systems aim to model users' preference on items based on their past interactions. Popular recommendation techniques such as matrix factorization (MF) [18], neural matrix factorization (NeuMF) [13] and their families only utilize user IDs and item IDs. Some methods thus propose to use side information for the cold-start scenario, e.g., using user attributes [40, 43, 54] and/or item attributes [41, 42, 47, 54]. However, in the CTR prediction task, side information is already used. The aforementioned CTR prediction models are all feature-rich models, which already take user and ad attributes as input.

Another way to tackle this problem is to actively collect more training data in a short time. For example, [20, 27, 44, 46] use contextual-bandit approaches and [10, 12, 34, 58] design interviews to collect specific information with active learning. However, these approaches still cannot lead to satisfactory prediction performance before sufficient training data are collected.

We tackle the cold-start CTR prediction problem for new ads from a different perspective, which is to generate desirable initial embeddings for new ad IDs in a meta learning framework, even when the new ads have no training data at all. Along this line, Pan et al. propose the Meta-Embedding model [32] by exploiting the associated attributes of the new ad. However, this model only considers the new ad itself, but ignores possibly useful information contained in existing old ads. Another meta learning-based model MeLU [19] is proposed to estimate a new user's preferences with a few consumed items. This model does not apply to our problem and it also considers the target user alone.

Meta Learning. Meta learning intends to design models that can learn new skills or adapt to new environments rapidly with a few training examples. It has been successfully applied in various areas such as recommendation [19, 22, 49], natural language processing [6, 16, 52] and computer vision [8, 9, 35].

There are three common meta learning approaches: 1) metric-based: learn an efficient distance metric, 2) model-based: use (recurrent) networks with external or internal memory, and 3) optimization-based: optimize the model parameters explicitly for fast learning. The meta learning approach we used to train GMEs is optimization-based, which generalizes Model-Agnostic Meta-Learning (MAML) [9]. We view the learning of ID embedding of each ad as a task. We use meta learning because the number of unique ads is much smaller than the number of samples and we need fast adaptation.

6 CONCLUSION

In this paper, we address the cold-start CTR prediction problem for new ads whose ID embeddings are not well learned yet. We propose Graph Meta Embedding (GME) models that can rapidly learn how to generate desirable initial embeddings for new ad IDs based on graph neural networks and meta learning. Unlike previous works that consider the new ad itself, GMEs simultaneously consider two information sources: the new ad and existing old ads. GMEs build a graph to connect new ads and old ads, and adaptively distill useful information from neighboring old ads w.r.t. each given new ad. We propose three specific GMEs from different perspectives. Experimental results show that GMEs can significantly improve the prediction performance in both cold-start and warm-up scenarios over five major deep learning-based CTR prediction models. GME-A which uses neighbor attributes performs best in most cases. In the future, we would consider enhanced approaches to retrieving more informative graph neighbors and alternative ways to distilling more representative information from neighbors.

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 *OSDI*. USENIX, 265–283.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[3] Mathieu Blondel, Akinori Fujino, Naonori Ueda, and Masakazu Ishihata. 2016. Higher-order factorization machines. In *NIPS*. 3351–3359.

[4] Sudhanshu Chanpuriya and Cameron Musco. 2020. Infinitewalk: Deep network embeddings as Laplacian embeddings with a nonlinearity. In *KDD*. ACM, 1325–1333.

[5] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and scalable response prediction for display advertising. *ACM TIST* 5, 4 (2015), 61.

[6] Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *AAAI*, Vol. 32.

[7] 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 *DLRS*. ACM, 7–10.

[8] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. 2019. Deep meta learning for real-time target-aware visual tracking. In *CVPR*. IEEE, 911–920.

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

[10] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2011. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM*. ACM, 595–604.

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

[12] Abhay S Harpale and Yiming Yang. 2008. Personalized active learning for collaborative filtering. In *SIGIR*. ACM, 91–98.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

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

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

[16] Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018. Dynamic Meta-Embeddings for Improved Sentence Representations. In *EMNLP*. 1466–1477.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[19] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: meta-learned user preference estimator for cold-start recommendation. In *KDD*. 1073–1082.

[20] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*. ACM, 661–670.

[21] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *KDD*. ACM, 1754–1763.

[22] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In *KDD*. ACM, 1563–1573.

[23] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *SIGIR*. ACM, 1137–1140.

[24] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, Vol. 30. 3.

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

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

[27] 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).

[28] Wentao Ouyang, Xiuwu Zhang, Li Li, Heng Zou, Xin Xing, Zhaojie Liu, and Yanlong Du. 2019. Deep spatio-temporal neural networks for click-through rate prediction. In *KDD*. ACM, 2078–2086.

[29] Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Zhaojie Liu, and Yanlong Du. 2019. Click-through rate prediction with the user memory network. In *DLP-KDD*. 1–4.

[30] Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Chao Qi, Zhaojie Liu, and Yanlong Du. 2019. Representation Learning-Assisted Click-Through Rate Prediction. In *IJCAI*. 4561–4567.

[31] Wentao Ouyang, Xiuwu Zhang, Lei Zhao, Jinmei Luo, Yu Zhang, Heng Zou, Zhaojie Liu, and Yanlong Du. 2020. MiNet: Mixed Interest Network for Cross-Domain Click-Through Rate Prediction. In *CIKM*. ACM, 2669–2676.

[32] 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*. ACM, 695–704.

[33] 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 *WWW*. IW3C2, 1349–1357.

[34] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. 2006. Naive filterbots for robust cold-start recommendations. In *KDD*. ACM, 699–705.

[35] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. 2020. Incremental few-shot object detection. In *CVPR*. IEEE, 13846–13855.

[36] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *SIGIR*. ACM, 2347–2356.

[37] 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 *ICDM*. IEEE, 1149–1154.

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

[39] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. IW3C2, 521–530.

[40] Sujoy Roy and Sharath Chandra Guntuku. 2016. Latent factor representations for cold-start video recommendation. In *RecSys*. ACM, 99–106.

[41] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *RecSys*. ACM, 89–96.

[42] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *SIGIR*. ACM, 253–260.

[43] Yanir Seroussi, Fabian Bohnert, and Ingrid Zukerman. 2011. Personalised rating prediction for new users using latent factor models. In *HT*. ACM, 47–56.

[44] Parikshit Shah, Ming Yang, Sachidanand Alle, Adwait Ratnaparkhi, Ben Shahshahani, and Rohit Chandra. 2017. A practical exploration system for search advertising. In *KDD*. ACM, 1625–1631.

[45] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*. ACM, 1161–1170.

[46] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. 2015. Personalized recommendation via parameter-free contextual bandits. In *SIGIR*. ACM, 323–332.

[47] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A Meta-Learning Perspective on Cold-Start Recommendations for Items. In *NIPS*.

[48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[49] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems.. In *NIPS*. 4957–4966.

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

[51] Chenyan Xiong, Taifeng Wang, Wenkui Ding, Yidong Shen, and Tie-Yan Liu. 2012. Relational click prediction for sponsored search. In *WSDM*. ACM, 493–502.

[52] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2018. Lifelong domain word embedding via meta-learning. In *IJCAI*. 4510–4516.

[53] Dawei Yin, Shike Mei, Bin Cao, Jian-Tao Sun, and Brian D Davison. 2014. Exploiting contextual factors for click modeling in sponsored search. In *WSDM*. ACM, 113–122.

[54] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. 2014. Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *SIGIR*. ACM, 73–82.

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

[56] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.

[57] 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 *KDD*. ACM, 1059–1068.

[58] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *SIGIR*. ACM, 315–324.