Task-distribution-aware Meta-learning for Cold-start CTR Prediction

Tianwei Cao[1], Qianqian Xu,[2*]
Zhiyong Yang[3,4], Qingming Huang[1,2,5,6*]
[1]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
[2]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, China
[3]State Key Laboratory of Information Security, Institute of Information Engineering, CAS, Beijing, China
[4]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[5]Key Laboratory of Big Data Mining and Knowledge Management, Chinese Academy of Sciences, Beijing, China
[6]Peng Cheng Laboratory, Shenzhen, Guangdong, China
caotianwei19@mails.ucas.ac.cn,xuqianqian@ict.ac.cn,yangzhiyong@iie.ac.cn,qmhuang@ucas.ac.cn

ABSTRACT

Nowadays, click-through rate (CTR) prediction has achieved great success in online advertising. However, making desirable predictions for unseen ads is still challenging, which is known as the cold-start problem. To address such a problem in CTR prediction, meta-learning methods have recently emerged as a popular direction. In these approaches, the predictions for each user/item are regarded as individual tasks, then training a meta-learner on them to implement zero-shot/few-shot learning for unknown tasks. Though these approaches have effectively alleviated the cold-start problem, two facts are not paid enough attention, 1) the diversity of the task difficulty and 2) the perturbation of the task distribution. In this paper, we propose an adaptive loss that ensures the consistency between the task weight and difficulty. Interestingly, the loss function can also be viewed as a description of the worst-case performance under distribution perturbation. Moreover, we develop an algorithm, under the framework of gradient descent with max-oracle (GDmax), to minimize such an adaptive loss. Then we prove the algorithm can return to a stationary point of the adaptive loss. Finally, we implement our method on top of the meta-embedding framework and conduct experiments on three real-world datasets. The experiments show that our proposed method significantly improves the predictions in the cold-start scenario.

KEYWORDS

CTR Prediction, Cold-start, Meta-learning, Min-max Optimization

ACM Reference Format:

Tianwei Cao, Qianqian Xu, Zhiyong Yang and Qingming Huang. 2020. Task-distribution-aware Meta-learning for Cold-start CTR Prediction. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12-16, 2020, Seattle, WA, USA.*. ACM, New York, NY, USA, 9 pages. https://doi.org/10. 1145/3394171.3413739

*Corresponding author.

MM '20, October 12–16, 2020, Seattle, WA, USA.

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413739

1 INTRODUCTION

Click-through rate (CTR) prediction plays a key role in online multimedia platforms. One of its most typical applications is the cost-per-click (CPC) advertising system. In this system, the most crucial task is to attract clicks from users by using their personality, which is exactly the goal of the CTR prediction models. Consequently, CTR prediction has attracted a considerable amount of researchers from the machine learning and data mining community.

Currently, most popular CTR prediction models can learn embedding vectors for each discrete feature, e.g. ad identifier (ad ID) or user gender. The related researches include [3, 9–11, 13, 15–17, 21, 23, 25–30, 35, 41–43, 45, 47, 48], etc. Unfortunately, these models are well known for suffering from the cold-start problem, which refers to the inability to provide an ID embedding for unseen items. Specifically, the challenge of the cold-start problem lies in two aspects: 1) making desirable predictions for purely new items whose embeddings are never trained, and 2) improving the predictive results on these items significantly when a small number of corresponding instances are collected.

So far, various methods have been proposed for addressing such a problem. The main idea of these methods is to replace the role of ID embedding with different kinds of side-information. [7, 31, 34, 40, 44] use user features to handle the cold-start situation. [12, 46] use relational information to avoid cold-start problem. Moreover, another line of researches pay more attention to sharing the common knowledge between the known and unknown items. [8, 19, 24, 32, 33, 36, 38] learn to produce representations for unseen items by using item attributes. [20] proposes a cross-domain adaptation method that achieves state-of-the-art performance. Among this line of researches, the meta-learning approaches have recently emerged as a popular direction. As an early trail, [38] proposes a meta-learning approach to learn personalized models for tweet recommendation. It regards the users as individual tasks and aggregates their historically click behaviors to make predictions for new tweets. On the contrary, [24] proposes a method that regards each ad as a task, using an algorithm inspired by model-agnostic meta-learning (MAML) [6] to learn a generator of initial ID embedding (meta-embedding).

Despite the remarkable success of these methods, there exist two facts are not paid enough attention. Firstly, since the data set often covers a wide spectrum of tasks, it is natural to observe different tasks showing different difficulties during the training process. Specifically, the loss function for those easy tasks reduces rapidly until it reaches a sufficient small magnitude, while the hard ones often suffer from a relatively high training loss and slow training progress. Blindly assigning equal weight to all the tasks then provides limited spaces for the improvement of the hard ones. In this way, a proper set of task weights should be captured along with the model itself. Secondly, the perturbation of task distribution was not paid enough attention from the majority of the historical studies. Actually, with such a problem left uncontrolled, a slight change of test distribution might lead to significant performance degradation. This issue could be largely alleviated if the task weights are leveraged to gain the distributional robustness. Notably, the task distribution could have subtle relationships with the task difficulty if the difficulty is defined by the training loss, while these two problems are actually different in their essence. We will discuss this point in later sections.

In this paper, we aim to develop a task-distribution-aware method that could solve the aforementioned issue, thus improving CTR prediction in the cold-start scenario. Our main idea is to adaptively adjust the task distribution. Specifically, we perform the training process with consistency constraints between the task weights and difficulty for the sake of modeling the diversity of task difficulty. Meanwhile, if we regard the task weights as a description of empirical task distribution, then the consistency constraints on task weights can be viewed as an adversarial perturbation which aims to disturb the training process by slightly changing the task distribution. Such a process is actually equivalent to a min-max problem that optimizing the expected performance of tasks under the worst-case distribution taken within an uncertainty set, which is called distributionally robust optimization (DRO) [1]. For this min-max optimization problem, we develop an algorithm under the framework of gradient descent with max-oracle (GDmax) [18], which uses the projected gradient ascent to solve the max-oracle on the uncertainty set of the task distribution. In this way, more attention could be paid to the hard tasks via the adaptive weights and the better resistance to distribution perturbation.

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

- We highlight two important facts in meta-learning, 1) the diversity of the task difficulty and 2) the perturbation of the task distribution. Motivated by these facts, we propose an adaptive loss function that can pay more attention to the hard tasks during the training process, while gaining the distributional robustness.
- To solve this min-max optimization, we develop an algorithm under the framework of GDmax. Specifically, we realize a projected gradient ascent method to solve the max-oracle on the uncertainty set of the task distribution. Moreover, we prove the algorithm can finally convergence to a stationary point. Since our algorithm only involves standard gradient operations and simple projections, it can be easily implemented by existing deep learning libraries.
- We realize our method on top of the meta-embedding framework and conduct systematic experiments on three real-world datasets. Experimental results show the effectiveness of our proposed method under the cold-start scenario.

2 METHODOLOGY

In this section, first introduce the meta-learning setting for CTR prediction. After that, we systematically elaborate two elements of the proposed method: the adaptive loss function and its optimization algorithm.

2.1 Meta-learning for CTR Prediction

The CTR prediction can be regarded as a binary classification problem. Specifically, given an ad ID $i \in \mathbb{R}$ with its attributes $\boldsymbol{a}^{(i)}$ and other ad-independent features $\boldsymbol{v}$, our goal is to find a model $f_\theta(\cdot)$ to predict the probability $P(y = 1|i, \boldsymbol{a}^{(i)}, \boldsymbol{v})$, where the $\boldsymbol{\theta}$ is the model parameter and the label $y \in \{0, 1\}$ indicates whether the ad will be clicked:

$$P(y = 1|i, \boldsymbol{a}^{(i)}, \boldsymbol{v}) = f_\theta(i, \boldsymbol{a}^{(i)}, \boldsymbol{v}), \tag{1}$$

Note that most popular CTR prediction models have an embedding mechanism, by which the ad ID i can be mapped into a real-valued vector Φ_i. Therefore, the ID embedding, for each ad, can be regarded as a hidden architecture of an ad-specific model. In other words, the CTR prediction model $f_\theta(i, \cdot, \cdot)$, for each fixed ad ID i, can be regarded as an ad-specific model $g^{(i)}(\cdot)$:

$$P(y = 1|i, \boldsymbol{a}^{(i)}, \boldsymbol{v}) = f_\theta(i, \boldsymbol{a}^{(i)}, \boldsymbol{v}) = g^{(i)}(\boldsymbol{a}^{(i)}, \boldsymbol{v}), \tag{2}$$

From such a perspective, we can naturally cast the CTR prediction as a form of a multi-task problem, viewing each ad ID as individual tasks. As a result, the cold-start CTR prediction can be regarded as learning to produce task-specific models for unseen ad IDs.

In this scenario, we notice that the CTR prediction problem is equivalent to a meta-learning problem [39], where the objective is to learn a meta-learner, from a set of tasks, to produce models for new tasks with few shots or even zero shots.

Recall that the task-specific ID embedding is unknown for new ads. So a possible solution for this problem is to add a meta-model on top of the base model $f_\theta(\cdot)$ that is pre-trained on the examples from all known ads. The meta-model produces effective initial embeddings for unknown ads, while the base model takes the embeddings to make predictions in the cold-start scenario. This is called meta-embedding in [24].

More formally, our goal is to learn a meta-model $h_\gamma(\cdot)$, from the known tasks, to produce the N_e-dimensional ID embedding $\Phi^{(i)} \in \mathbb{R}^{N_e \times 1}$ for each unseen ad ID i by utilizing its attributes $\boldsymbol{a}^{(i)}$:

$$\Phi^{(i)}(\boldsymbol{\gamma}) = h_\gamma(\boldsymbol{a}^{(i)}), \tag{3}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^{N_h \times 1}$ represents N_h parameters of the meta-model h.

Notably, the produced ID embedding need to have two advantages, 1) better predictions for the unseen ads and 2) fast adaption when a minimal amount of data is collected. For these purposes, the meta-embedding loss consists of two components:

- To describe the model performance on the unseen ads, the cold-start loss $l_c^{(i)}$ for each task i is defined as the following:

$$l_c^{(i)}(\boldsymbol{\gamma}) = \frac{1}{K}\sum_{j=1}^{K} l(\Phi_i(\boldsymbol{\gamma}), y_j, \hat{y}_j), \tag{4}$$

  where l is the cross-entropy loss for binary classification; K is the number of examples in each task; y_j and $\hat{y}_j$ represent the label and prediction for the example j respectively.
- To describe the adaptability to a minimal amount of newly collected data, the warm-up loss $l_w^{(i)}$, for the task i, is in the form of:

$$l_w^{(i)}(\boldsymbol{\gamma}) = \frac{1}{K}\sum_{j=K+1}^{2K} l(\Phi_i(\boldsymbol{\gamma}) - \beta\nabla_{\Phi_i} l_c^{(i)}(\boldsymbol{\gamma}), y_j, \hat{y}_j), \tag{5}$$

  where $\Phi_i(\boldsymbol{\gamma}) - \beta\nabla_{\Phi_i} l_c^{(i)}(\boldsymbol{\gamma})$ represents the embedding value after a gradient update with the learning rate $\beta \in \mathbb{R}^+$. This loss function follows the spirit of MAML [6], which describes the model performance after learning from K samples.

Note that the embedding vector Φ_i is produced by the meta-model $h_\gamma(\cdot)$ in Eq.(3). Thus, $\boldsymbol{\gamma}$ is the only trainable parameter in both losses.

Unifying these two losses, meta-embedding loss l_{meta} is defined as the following:

$$l_{meta}(\boldsymbol{\gamma}) = \frac{1}{T}\sum_{i=1}^{T} \alpha l_c^{(i)}(\boldsymbol{\gamma}) + (1-\alpha) l_w^{(i)}(\boldsymbol{\gamma}), \tag{6}$$

where the $\alpha \in [0, 1]$ is a coefficient to trade-off the two losses; T is the number of the tasks. In this way, the optimization objective is provided in the form of:

$$\min_{\boldsymbol{\gamma}} \frac{1}{T}\sum_{i=1}^{T} \alpha l_c^{(i)}(\boldsymbol{\gamma}) + (1-\alpha) l_w^{(i)}(\boldsymbol{\gamma}), \tag{7}$$

where $\boldsymbol{\gamma}$ represents the parameters of the meta-model in Eq.(3) that is utilized to produce the initial embedding, while other parameters are reused from the base model.

On top of this framework, we propose a meta-learning method that involves two crucial elements: 1) The adaptive loss that dynamically modulates the task distribution in the training process; 2) The optimization algorithm to minimize the adaptive loss.

In the following, we will elaborate on these two elements respectively.

2.2 Adaptive Loss

Firstly, we reformulate the meta-embedding loss l_{meta} in Eq.(6) as the following:

$$l_{meta}(\boldsymbol{\gamma}) = \sum_{i=1}^{T} \left(p_i \cdot l_{meta}^{(i)}(\boldsymbol{\gamma}) \right), \tag{8}$$

where each p_i is equally assigned as $\frac{1}{T}$. The $l_{meta}^{(i)}$ is the task-wise loss for each task i:

$$l_{meta}^{(i)}(\boldsymbol{\gamma}) = \alpha l_c^{(i)}(\boldsymbol{\gamma}) + (1-\alpha) l_w^{(i)}(\boldsymbol{\gamma}), \tag{9}$$

which is also regarded as a description of the ***task difficulty***. In other words, the harder the task i, the larger the $l_{meta}^{(i)}$.

In such a formal setting, the meaning of p_i can be explained from two different perspectives simultaneously:

- The task-weight perspective, where we regard each p_i as the weight of the task i to capture the diversity of the task difficulty. In this condition, we expect a strong consistency between the p_i and task difficulty.
- The task-distribution perspective, where we regard each p_i as the probability that the task i is sampled. In this condition, we expect to model the distribution perturbation by p_i to gain the distributional robustness.

From the task-weight perspective, l_{meta} in Eq.(8) can be regarded as the inner product of the $\boldsymbol{p} = [p_1, p_2, \ldots, p_T]^\top$ and $\boldsymbol{l} = [l_{meta}^{(1)}(\boldsymbol{\gamma}), l_{meta}^{(2)}(\boldsymbol{\gamma}), \ldots, l_{meta}^{(T)}(\boldsymbol{\gamma})]^\top$, which can describe the consistency between the task weight and task difficulty. In other words, the greater the inner product, the stronger the consistency between the task weight and task difficulty. Based on this observation, we can slightly modify the l_{meta} in Eq.(8) to get a loss function $\hat{l}_{meta}$:

$$\begin{aligned} &\hat{l}_{meta}(\boldsymbol{\gamma}, \boldsymbol{p}) = \max_{\boldsymbol{p}} \boldsymbol{p}^\top \boldsymbol{l}, \\ &s.t.\ \boldsymbol{p} \geq 0, \boldsymbol{p}^\top \mathbf{1} = 1 \end{aligned} \tag{10}$$

where the $\max(\cdot)$ describes the pursuit of the strongest weight-difficulty consistency, while the constraints ensure the weights $\boldsymbol{p}$ can be simultaneously regarded as probabilities from the task-distribution perspective.

Unfortunately, despite that the loss function $\hat{l}_{meta}$ involves some good ideas, it is actually ineffective. The reason is that to gain the maximum inner product of the $\boldsymbol{p}^\top \boldsymbol{l}$, the weight of the hardest task will be assigned as 1, while the others equally assigned as 0. In such a situation, the hardest task will dominate the gradient in the training process, while other tasks neglected.

In view of these facts, we propose the adaptive loss $\tilde{l}_{meta}$ by adding an extra constraint to $\hat{l}_{meta}$. The adaptive loss can ensure the balance of the task weights while pursuing the strongest weight-difficulty consistency. Specifically, we restrict the value of the task weights to an uncertainty set around the uniform weights $\frac{1}{T}$, thus ensuring the task weights will not differ too much from each other. We now provide the formulation of the adaptive loss $\tilde{l}_{meta}$:

$$\begin{aligned} &\tilde{l}_{meta}(\boldsymbol{\gamma}, \boldsymbol{p}) = \max_{\boldsymbol{p}} \boldsymbol{p}^\top \boldsymbol{l}, \\ &s.t.\ \boldsymbol{p} \geq 0, \boldsymbol{p}^\top \mathbf{1} = 1, \frac{1}{2}\left\|\boldsymbol{p} - \frac{\mathbf{1}}{T}\right\|_2^2 \leq \frac{\rho}{T} \end{aligned} \tag{11}$$

where $\frac{1}{2}\left\|\boldsymbol{p} - \frac{\mathbf{1}}{T}\right\|_2^2$ is the above-mentioned uncertainty set, and $\rho \in \mathbb{R}^+$ is a coefficient to control the distance between $\boldsymbol{p}$ and $\frac{\mathbf{1}}{T}$.

Consider that if we assume that $\boldsymbol{p} = \frac{\mathbf{1}}{T} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \in \mathbb{R}^{T\times 1}$, then plugging this into $\frac{1}{2}\left\|\boldsymbol{p} - \frac{\mathbf{1}}{T}\right\|_2^2 \leq \frac{\rho}{T}$ yields that $\boldsymbol{\epsilon} \geq -\sqrt{2\rho/T}$. So if we set ρ such that $\frac{1}{T} - \sqrt{\frac{2\rho}{T}} > 0$, then we have:

$$\boldsymbol{p} = (\frac{\mathbf{1}}{T} + \boldsymbol{\epsilon}) \geq (\frac{1}{T} - \sqrt{\frac{2\rho}{T}}) > 0. \tag{12}$$

In other words, if ρ is sufficiently small, all of the task weights will be positive values around $\frac{1}{T}$, which can effectively avoid the hardest task completely overwhelming the others in the training process. Additionally, it is not hard to observe from Eq.(11) that,

when coefficient ρ is assigned to 0, p is fixed as $\frac{1}{T}$. The adaptive loss function $\tilde{l}_{meta}$ then is downgraded to original meta-embedding loss l_{meta} in Eq.(8). Thus the proposed $\tilde{l}_{meta}$ can be viewed as a generalization of meta-embedding loss by slightly relaxing the constraint on task weights p. Such a relaxation makes it possible to dynamically up-weight the harder tasks and down-weight the easier ones in the training process.

More interestingly, $\frac{1}{2}\left\|\boldsymbol{p}-\frac{\mathbf{1}}{T}\right\|_2^2 \leq \frac{\rho}{T}$ in Eq.(11), from the task-distribution perspective, can also be regarded as a distance constraint between the distribution $\boldsymbol{p}$ and $\frac{1}{T}$. Specifically, given a function $f(t)=\frac{1}{2}(t-1)^2$, then the χ^2-divergence [37] between distributions $\mathcal{S}$ and $\mathcal{Q}$ can be defined as the following:

$$D_{\chi^2}(\mathcal{S}||\mathcal{Q}) = \int f(\frac{d\mathcal{S}}{d\mathcal{Q}})d\mathcal{Q}. \tag{13}$$

Viewing the $\boldsymbol{p}$ and $\frac{1}{T}$ as two distributions, we notice that $D_{\chi^2}(\boldsymbol{p}||\frac{1}{T}) = \frac{T}{2}\left\|\boldsymbol{p}-\frac{\mathbf{1}}{T}\right\|_2^2 \leq \rho$. In this way, these constraints are able to model the distribution perturbation.

Specifically, the three constraints of $\tilde{l}_{meta}$ can be regarded as a family of distributions:

$$\mathbb{P} = \left\{\boldsymbol{p} : \boldsymbol{p} \geq 0, \boldsymbol{p}^\top \mathbf{1} = 1, \frac{1}{2}\left\|\boldsymbol{p}-\frac{\mathbf{1}}{T}\right\|_2^2 \leq \frac{\rho}{T}\right\}, \tag{14}$$

where each $\boldsymbol{p} \in \mathbb{P}$ has a restricted divergence with the uniform distribution. Meanwhile, the sum of all $p_i \cdot l_{meta}^{(i)}$ can be regarded as the expected loss of the tasks. In this way, the max$(\cdot)$ in $\tilde{l}_{meta}$ describes finding the worst case distribution within $\mathbb{P}$ to maximize the expected value of $l_{meta}^{(i)}$.

Given the Eq.(11) and Eq.(14), the optimization objective then can be formulated as:

$$\min_{\boldsymbol{\gamma}} \max_{\boldsymbol{p}\in\mathbb{P}} \left[obj(\boldsymbol{\gamma}, \boldsymbol{p}) = \boldsymbol{p}^\top \boldsymbol{l}\right], \tag{15}$$

where $\boldsymbol{\gamma}$ represents the parameters of the meta-model as in Eq.(3). By solving such a min-max problem, we expect to gain a sufficiently small l_{meta} on the worst-case distribution, where the worst-case distribution is taken from $\mathbb{P}$ [1]. In other words, if only a set of tasks are drawn from a distribution contained by $\mathbb{P}$, the supremum of its meta-embedding loss function would be provided. In this condition, the issue of distribution purturbation could be largely alleviated.

Unifying both perspectives above-mentioned, we can say that the adaptive loss $\tilde{l}_{meta}$ can not only capture the diversity of the task difficulty but also possess the robustness to distribution perturbation. From the analysis above, we can see that if we use the value of training loss to describe the task difficulty, the two issues will finally converge into a unified form. In other words, the two issues actually come from the two perspectives of the same fact.

2.3 Optimization

2.3.1 GDmax for the Min-max Problem. We now come to the optimization for the min-max objective function $obj(\cdot,\cdot)$ in Eq.(15). Consider that if the parameter $\boldsymbol{p}$ is given, the problem is then casted as minimization for a non-convex function $obj(\cdot, \boldsymbol{p})$. Conversely, if the $\boldsymbol{\gamma}$ is given, the problem is then casted as finding the maximum for a concave function $obj(\boldsymbol{\gamma}, \cdot)$. To solve such a nonconvex-concave min-max problem, a straightforward way is to calculate the inner max-oracle exactly and then solve the outer non-convex minimization, which is called GDmax in [18]. Next, we develop an algorithm based on this idea, which is shown in Alg.1. In this algorithm, the max-oracle is solved by the procedure `findp`, and then we utilize the gradient descent to find a stationary point of the function $obj(\cdot, \boldsymbol{p})$.

Algorithm 1 GDmax for the min-max problem

Input: the pre-trained CTR model f_θ;
Input: the user-click dataset $\mathbb{D}$;
All known ad IDs: $\mathbb{I} \leftarrow \{1, 2, \ldots, T\}$
Randomly initialize $\boldsymbol{\gamma}$
for *epoch* $= 1, 2, \ldots$ **do**
  Update $\boldsymbol{p} \leftarrow \texttt{findp}(\boldsymbol{p}, \boldsymbol{l})$
  Initialize $\mathbb{V} \leftarrow \{\}$
  while $\mathbb{V} \neq \mathbb{I}$ **do**
    $\mathbb{I}' \leftarrow$ Sample n IDs from $\mathbb{I}$
    for $i \in \mathbb{I}'$ **do**
      $\mathbb{D}_c^{(i)} \leftarrow K$ samples for ad i from $\mathbb{D}$
      $\mathbb{D}_w^{(i)} \leftarrow$ Another K samples for ad i from $\mathbb{D}$
      Produce the initial embedding: $\Phi_i \leftarrow h_\gamma(\boldsymbol{a}_i)$
      Cold-start loss: $l_c^{(i)} \leftarrow \frac{1}{K}\sum_{j\in\mathbb{D}_c^{(i)}} l(\Phi_i, y_j, \hat{y}_j)$
      Update embedding: $\Phi_i' \leftarrow \Phi_i - \beta\nabla_{\Phi_i} l_c^{(i)}$
      Warm-up loss: $l_w^{(i)} \leftarrow \frac{1}{K}\sum_{j\in\mathbb{D}_w^{(i)}} l(\Phi_i', y_j, \hat{y}_j)$
      Task-wise loss: $l_{meta}^{(i)} \leftarrow \alpha l_c^{(i)} + (1-\alpha) l_w^{(i)}$
    end for
    Update $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} - \mu\sum_{i\in\mathbb{I}'} \nabla_{\boldsymbol{\gamma}}(p_i \cdot l_{meta}^{(i)})$
    Update $\mathbb{V} \leftarrow \mathbb{V} \cup \mathbb{I}'$
  end while
end for

Moreover, we also prove that Alg.1 can return to a stationary point of Eq.(15). For the limitation of the space, we put these theoretical analysis into our supplementary materials.

2.3.2 Solving the Max-oracle. We now come to the procedure `findp` in Alg.1, which is used for solving the max-oracle of $obj(\gamma, \cdot)$. More formally, the objective for this maximization problem is as the following:

$$\max_{\boldsymbol{p}\in\mathbb{P}} \left[obj_\gamma(\boldsymbol{p}) = \boldsymbol{p}^\top \boldsymbol{l}\right], \tag{16}$$

where $obj_\gamma(\boldsymbol{p})$ is the maximization objective.

Since $obj_\gamma(\boldsymbol{p})$ is a concave function and its feasible region $\mathbb{P}$ is a convex set, we can realize the maximization by projected gradient ascent, which can be easily implemented by using deep learning libraries. Specifically, the overall process of `findp` is as shown in Alg.2, where $\texttt{project}_{\mathbb{P}}$ is a procedure that can project the updated $\boldsymbol{p}$ onto its feasible region $\mathbb{P}$.

2.3.3 Projection. We now come to the $\texttt{project}_{\mathbb{P}}$ in Alg.2, a procedure that can project the $\boldsymbol{p}$ onto $\mathbb{P}$. More formally, the objective of $\texttt{project}_{\mathbb{P}}$ is to find a $\tilde{\boldsymbol{p}} \in \mathbb{P}$ with the smallest Euclidean distance from $\boldsymbol{p}$:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{p}}} \frac{1}{2}\|\tilde{\boldsymbol{p}} - \boldsymbol{p}\|_2^2, \\ &s.t.\ \tilde{\boldsymbol{p}} > 0, \tilde{\boldsymbol{p}}^\top \mathbf{1} = 1, \frac{1}{2}\left\|\tilde{\boldsymbol{p}} - \frac{\mathbf{1}}{T}\right\|_2^2 \leq \frac{\rho}{T} \end{aligned} \tag{17}$$

Algorithm 2 Procedure `findp`

Input: Task distribution $\boldsymbol{p}$;
Input: $\boldsymbol{l} = [l_{meta}^{(i)} : i \in [T]]$;
All known ad IDs: $\mathbb{I} \leftarrow \{1, 2, \ldots, T\}$
for *epoch* $= 1, 2, \ldots$ **do**
  for $i \in \mathbb{I}$ **do**
    Update $p_i \leftarrow p_i + \delta \frac{\partial \boldsymbol{p}^\top \boldsymbol{l}}{\partial p_i}$
  end for
  Projection $\boldsymbol{p} \leftarrow \texttt{project}_{\mathbb{P}}(\boldsymbol{p})$
end for

To solve this minimization problem, we take a partial dual of it, then maximize the dual to find the optimal $\tilde{\boldsymbol{p}}$. Specifically, we introduce the dual variable $\lambda \geq 0$ for the constraint $\frac{1}{2}\left\|\tilde{\boldsymbol{p}} - \frac{\mathbf{1}}{T}\right\|_2^2 \leq \frac{\rho}{T}$ and perform the standard min-max swap [2] yields the maximization problem:

$$\begin{aligned} &\max_{\lambda \geq 0} \min_{\tilde{\boldsymbol{p}}} \mathcal{F}_\lambda(\tilde{\boldsymbol{p}}), \\ &s.t.\ \tilde{\boldsymbol{p}} > 0, \tilde{\boldsymbol{p}}^\top \mathbf{1} = 1 \end{aligned} \tag{18}$$

where the dual function $\mathcal{F}_\lambda(\tilde{\boldsymbol{p}})$ is as the following:

$$\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}) = \frac{\lambda}{2}(T\|\tilde{\boldsymbol{p}}\|_2^2 - 1) - \lambda\rho + \frac{1}{2}\|\tilde{\boldsymbol{p}} - \boldsymbol{p}\|_2^2, \tag{19}$$

where we transform the original inequality constraint $\frac{1}{2}\left\|\tilde{\boldsymbol{p}} - \frac{\mathbf{1}}{T}\right\|_2^2 - \frac{\rho}{T} \leq 0$ into $\frac{1}{2}(T\|\tilde{\boldsymbol{p}}\|_2^2 - 1) - \rho \leq 0$, by some algebraic manipulations.

According to Danskin's theorem [4], given the optimal solution $\tilde{\boldsymbol{p}}^\star$ for inner convex function, we can calculate the gradient $\frac{d\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)}{d\lambda}$ by the following equation:

$$\frac{d\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)}{d\lambda} = \frac{1}{2}(T\left\|\tilde{\boldsymbol{p}}^\star\right\|_2^2 - 1) - \rho,\ \tilde{\boldsymbol{p}}^\star = \arg\min_{\tilde{\boldsymbol{p}}} \mathcal{F}_\lambda(\tilde{\boldsymbol{p}}), \tag{20}$$

In this way, we can perform the binary search over λ, similar to [22], to find the maximum of $\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)$ where $\frac{d\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)}{d\lambda} = 0$. Specifically, if we have a $\lambda = a$ such that $(\frac{d\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)}{d\lambda}\Big|_{\lambda=a}) > 0$, which means there exists a $\lambda > a$ such that $\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star) > (\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)\Big|_{\lambda=a})$, we then increase the value of λ. Meanwhile, if there is a $\lambda = a$ such that $(\frac{d\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)}{d\lambda}\Big|_{\lambda=a}) < 0$, which means there exists a $\lambda < a$ such that $\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star) > (\mathcal{F}_\lambda(\tilde{\boldsymbol{p}}^\star)\Big|_{\lambda=a})$, we then decrease the value of λ. We would perform such a process recursively, until the maximum is achieved.

To solve the inner minimization, we cast the $\mathcal{F}_\lambda(\tilde{\boldsymbol{p}})$, in this paper, into a form of Euclidean projection, thus the minimization problem can be regarded as the projection of a vector to the probability simplex [5].

Specifically, we reformulate the $\mathcal{F}_p(\tilde{\boldsymbol{p}})$ as the following:

$$\begin{aligned} \mathcal{F}_p(\tilde{\boldsymbol{p}}) &= \int \mathcal{F}'_p(\tilde{\boldsymbol{p}})\, d\tilde{\boldsymbol{p}} \\ &= \int [T\lambda\tilde{\boldsymbol{p}} + (\tilde{\boldsymbol{p}} - \boldsymbol{p})]\, d\tilde{\boldsymbol{p}} \\ &= (T\lambda + 1) \int [\tilde{\boldsymbol{p}} - \frac{\boldsymbol{p}}{T\lambda + 1}]\, d\tilde{\boldsymbol{p}} \\ &= \frac{T\lambda + 1}{2}\left\|\tilde{\boldsymbol{p}} - \frac{\boldsymbol{p}}{T\lambda + 1}\right\|_2^2 + const. \end{aligned} \tag{21}$$

In this way, the $\min_{\tilde{\boldsymbol{p}}} \mathcal{F}_\lambda(\tilde{\boldsymbol{p}})$ in Eq.(18) is then equivalent to projecting the vector $\boldsymbol{v} \in \mathbb{R}^{T\times 1}$ onto the probability simplex:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{p}}} \frac{T\lambda + 1}{2}\|\tilde{\boldsymbol{p}} - \boldsymbol{v}\|_2^2, \\ &s.t.\ \tilde{\boldsymbol{p}} > 0, \tilde{\boldsymbol{p}}^\top \mathbf{1} = 1 \end{aligned} \tag{22}$$

where $\boldsymbol{v} = \frac{\boldsymbol{p}}{T\lambda+1}$. Similar to [5], the projection $\tilde{\boldsymbol{p}}$ can be formulated as $\tilde{\boldsymbol{p}} = (\boldsymbol{v} - \frac{\eta}{T\lambda+1})_+$, where $\eta \in \mathbb{R}$ is selected such that $\tilde{\boldsymbol{p}}^\top \mathbf{1} = 1$. To find such a value η, we assume that $\boldsymbol{v}$ is sorted in descending order $v_1 \geq v_2 \geq \cdots \geq v_T$. In this way, finding a value η is cast as finding the index i such that $i = \max(\{i \in [T] : \sum_{j=1}^{i}(v_j - v_i) < 1\})$. After gaining the index i, we can get the value of η through the algebraic manipulations: $\eta = \frac{1}{i}\sum_{j=1}^{i} p_j - \frac{T\lambda+1}{i}$,. In this way, we now get the procedure `solveInner` for the inner minimizaton problem Eq.(18), which is detailed in Alg.3.

Algorithm 3 `solveInner`

Input: $\boldsymbol{p}$;
Input: λ;
Output: i, η;
$\boldsymbol{p}' \leftarrow$ Sort $\boldsymbol{p}$ in descending order
$\boldsymbol{v} \leftarrow \frac{\boldsymbol{p}'}{T\lambda+1}$
$i \leftarrow \max(\{i \in [T] : \sum_{j=1}^{i}(v_j - v_i) < 1\}$
$\eta \leftarrow \frac{1}{i}\sum_{j=1}^{i} p'_j - \frac{T\lambda+1}{i}$

Given the solution of the inner minimization, we can define the procedure $\texttt{project}_{\mathbb{P}}$ as the above-mentioned binary search, which can solve the dual problem in Eq.(18), see Alg.4.

Algorithm 4 $\texttt{project}_{\mathbb{P}}$

Input: Original Parameter $\boldsymbol{p}$;
Output: Projected Parameter $\tilde{\boldsymbol{p}}$;
$\lambda_{min} \leftarrow 0$
$\lambda_{max} \leftarrow \frac{1}{T}(\frac{T \cdot max(\boldsymbol{p})}{\sqrt{2\rho+1}} - 1)$
$\lambda'_{max} \leftarrow \lambda_{max}$
while $|\lambda_{max} - \lambda_{min}| > \tau\lambda'_{max}$ **do**
  $\lambda \leftarrow \frac{\lambda_{max}+\lambda_{min}}{2}$
  $(i, \eta) \leftarrow \texttt{solveInner}(\boldsymbol{p}, \lambda)$
  $grad_\lambda \leftarrow \frac{d\mathcal{F}_\lambda(\tilde{\boldsymbol{p}})}{d\lambda}$
  if $grad_\lambda > 0$ **then**
    $\lambda_{min} \leftarrow \lambda$
  else
    $\lambda_{max} \leftarrow \lambda$
  end if
end while
$\lambda \leftarrow \frac{\lambda_{max}+\lambda_{min}}{2}$
$(i, \eta) \leftarrow \texttt{solveInner}(\boldsymbol{p}, \lambda)$
$\boldsymbol{v} \leftarrow \frac{\boldsymbol{p}}{T\lambda+1}$
$\tilde{\boldsymbol{p}} = (\boldsymbol{v} - \frac{\eta}{T\lambda+1})_+$

Now we have entirely detailed our algorithm for optimizing the min-max objective function in Eq.(15). Since the algorithm only

involves standard gradient operations and simple projections, it can be easily implemented by existing deep learning libraries.

3 EXPERIMENTS

Recall that our method is proposed to improve the CTR prediction in the cold-start scenario. In real-world applications, the models for cold-start CTR prediction usually goes through three phases: 1) The training phase, where we use the examples from known ads to train a predictive model, which we called the base model in the previous section; 2) The cold-start phase, where the base model can only make predictions for unknown ads with the randomly initialized ID embeddings. Thus, we expect that our proposed method can produce more effective initial embeddings; 3) The warm-up phase, where we expect the produced embeddings can be significantly improved, after training them on ***newly collected examples*** with only a few amounts. To simulate these three phases, each dataset in our experiments is split into three subsets: 1) Training set: the collection that includes all examples of known ads. In our experiments, we take the ads with relatively large amounts of examples as the simulation of known ads, because their embeddings can be well trained; 2) Warm-up set: Besides the known ads, we then select the unknown ads from the rest ones. Specifically, the example amounts of each unknown ad should be larger than $3 \times K$, where K is the task size mentioned in the previous section. We take $3 \times K$ examples from each unknown ad as the newly collected examples that compose the warm-up set; 3) Test set: Besides $3 \times K$ examples in the warm-up set, we use rest examples of the unknown ads to compose the test set. This subset is used to provide evaluations in both the cold-start and warm-up phase. In our experiments, we use cross-entropy, i.e., logLoss, and AUC score as the evaluation metrics in both the cold-start and warm-up phase. To highlight the relative improvements, we will show the percentage of improvements in two metrics: 1) $logLoss_{impr.} = \left(\frac{logLoss}{logLoss_{base\_cold}} - 1\right) \times 100\%$ and 2) $AUC_{impr.} = \left(\frac{AUC}{AUC_{base\_cold}} - 1\right) \times 100\%$. Recall the aforementioned base model that randomly initialize the embeddings for unknown ads. The $logLoss_{base\_cold}$ and AUC_{base_cold} here refer to the logLoss and AUC score of such a base model in the cold-start phase respectively. For the limitation of the space, we put $logLoss_{impr.}$ results on public datasets into supplementary materials. Notably, our proposed method, similar to the meta-embedding (**ME**) [24], needs to be applied upon the base model, where our method is essentially a meta-model that produces effective embeddings for unknown ads. Thus the training process consists of two parts, 1) pre-train the base model and 2) train our meta-model on top of the base model. In this way, our experiments are done with the following steps:

1. Pre-train the base model with the entire training set that consisting of all known ads (1 epoch).
2. Train the meta-model on the training set that is divided into tasks (3 epochs).
3. Produce initial embeddings of each unknown ad, randomly or through the meta-model.
4. Evaluate the prediction performance on the test set (performance in the cold-start phase).
5. Train the generated embeddings on the warm-up set (3 epochs).
6. Evaluate the prediction performance on the test set (performance in the warm-up phase).

3.1 Competitors

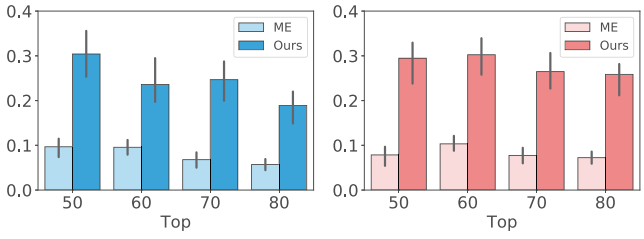
The following models are selected as the base models:

- **FM**: Factorization Machine [30] which uses factorization techniques to model second-order feature interactions.
- **DNN**: Abbreviation of Deep Neural Network. In this paper, it refers to a popular baseline network [47, 48] with an embedding layer and multiple dense layers;
- **W&D**: Wide&Deep model uses a wide component to handle the manually designed features and a **DNN** to extracts nonlinear relations. Here we follow the practice in [24] to take raw features as wide inputs.
- **DeepFM**: A recent state-of-the-art CTR prediction model [9]. It uses **FM** as the wide model in **W&D** to reduce the feature engineering jobs.

We conduct the comparisons between **ME** and our method, both of which are applied on top of four base models respectively. Note that when hyper-parameter ρ in Eq.(11) is assigned to 0, our method then degenerates into **ME**. Thus the comparison can also be regarded as an ablation study about our weighting scheme.

3.2 Performance on Public Datasets

3.2.1 MovieLens-1M Dataset. The MovieLens-1M dataset is a benchmark dataset for Recommendation System. The dataset contains 1 million ratings from 6000 users on 4000 movies, where the attributes of each movie include title, year of release and genre tags. In our experiments, the ratings larger than 4 are labeled as clicked, while others as unclicked. In this dataset, the ads with more than 300 instances are regarded as known ads, while the task size K is assigned as 20.



(a) Cold-start phase (b) Warm-up phase

Figure 1: Average AUC Scores on Top-N Hardest Tasks w.r.t Movielens-1M

In our experiments, we implement the meta-model as a simple neural network with the adaptive loss as Eq.(11). In making predictions, the network composes the pre-trained 128-dimensional embeddings of ad attributes and then get the outputs by a dense layer. The dense layer takes `Tanh` as the activation function and has no bias term. Note that the attributes embeddings are frozen in this process, so only the weights in the dense layer are trainable parameters. Moreover, Adam optimizer [14] is adopted as the

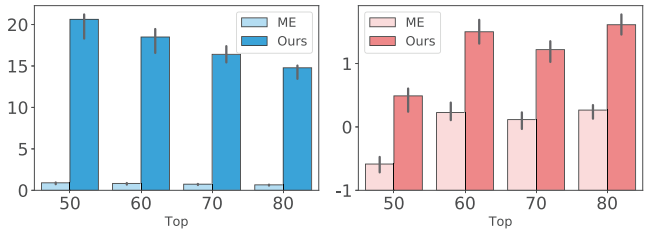
Table 1: AUC Improvements on Movielens-1M

Phase	Cold-start			Warm-up-1			Warm-up-2			Warm-up-3		
Model	Base	ME	Ours	Base	ME	Ours	Base	ME	Ours	Base	ME	Ours
FM	0.00	1.50	2.26	2.13	3.28	3.97	4.26	5.04	5.69	6.05	6.55	7.20
DNN	0.00	2.54	3.08	0.87	3.19	3.68	1.79	3.93	4.37	2.66	4.62	5.01
W&D	0.00	0.26	1.18	1.48	1.62	2.34	3.00	3.04	3.56	4.40	4.35	4.70
DeepFM	0.00	2.48	3.73	1.01	3.28	4.37	2.06	4.16	5.10	3.04	4.97	5.77

implementation of the gradient descent process in Alg.1. The hyper-parameters, such as ρ in Eq.(14) or α in Eq.(9), are obtained by grid search. The experimental results on the MovieLens-1M dataset are shown in Table 1, where **Base** refers to the afore-mentioned base models with randomly initialized embeddings for unknown ads. The blue parts of Table 1 show the improvements on Logloss and AUC score respectively, while the red parts show that of the warm-up phase where all of the 3 epochs are shown to illustrate the continuous process of warming up the cold-start model. From these tables, we can observe that our method outperforms other benchmarks in both phases on this dataset. Moreover, we calculate the average AUC scores on the hardest tasks in the test set, as Figure 1, where the worse the performance, the harder the task we considered. By comparing **ME** and our method in this figure, we observe that our method achieves better performance on the top-N hardest tasks, where N ranges from 50 to 80. In this way, we conclude that our proposed method can improve the prediction performance on hard tasks in this dataset.

3.2.2 Amazon Dataset. Amazon dataset is a collection of user browsing logs over e-commerce products with reviews and product metadata from Amazon Inc. [29, 47, 48]. We conduct experiments on a subset named Electronics, which contains about 200000 users and 60000 ads. Ad attributes include title, category, brand and price. Similar with MovieLens-1M, instances with ratings larger than 4 are regarded as clicked. The ads with more than 150 instances, in this dataset, are regarded as known ads, while the task size K is assigned as 20. The experimental results are shown in Table 2, where the implementation details are the same as the previous ones. Similarly, we observe that our proposed method achieves better performance than others. Moreover, Fig.2 shows the AUC scores on the hardest tasks are improved comparing with **ME**. This implies that our method is helpful to the hard tasks.

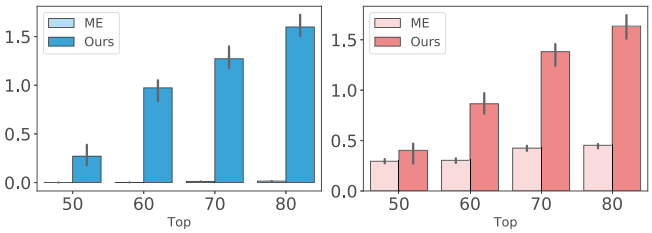
3.2.3 Tencent 2018 Dataset. The Tencent 2018 dataset is collected from Tencent Social Ads competition in 2018. This dataset contains over 50 million instances, where each instance consists of an ad, a user and the user-click behavior. The attributes of each ad include advertiser ID, ad category, campaign ID, app ID, app size, and app type. In this dataset, the ads with more than 20000 instances are regarded as known ads, while the task size K is assigned as 200. We conduct experiments on this dataset, with the same implementations as the MovieLens-1M. The experimental results are shown in Table 3. From these results, we also observe that our method outperforms other benchmarks, but, interestingly, the relative improvements on this dataset are less evident than the other two datasets. To explain, the size of Tencent 2018 dataset is much larger than Movielens-1M and Amazon, so the base models



(a) Cold-start phase (b) Warm-up phase

Figure 2: Average AUC Scores on Top-N Hardest Tasks w.r.t Amazon

were better pre-trained on this dataset. In other words, our method would provide a larger improvement when the base models are not sufficiently trained. Additionally, Fig.3 shows that the improvements of AUC score on the hardest tasks. We can see the advantage of our method over **ME** on the hard tasks. Taken the results on three datasets together, we can conclude that our proposed method can effectively improve the cold-start CTR prediction.



(a) Cold-start phase (b) Warm-up phase

Figure 3: Average AUC Scores on Top-N Hardest Tasks w.r.t Tencent 2018

3.3 Hyper-parameter Study

In this section, we study the impact of hyper-parameters on our predictive models, including 1) α which is used to trade-off the cold-start and warm-up loss, and 2) ρ which is used to control the distance between $\boldsymbol{p}$ and empirical distribution $\frac{1}{T}$. In the experiments, we take the FM as our base model and calculate the AUC score on MovieLens-1M dataset with respect to various α and ρ. Figure 4 shows the impact of hyper-parameter α. We can observe form

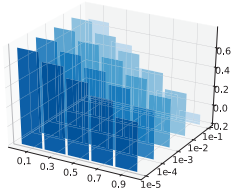
Table 2: AUC Improvements on Amazon

Phase	Cold-start			Warm-up-1			Warm-up-2			Warm-up-3		
Model	Base	ME	Ours	Base	ME	Ours	Base	ME	Ours	Base	ME	Ours
FM	0.00	0.20	1.85	3.18	7.38	9.69	7.57	14.79	16.65	11.53	20.84	22.52
DNN	0.00	0.07	0.43	2.40	4.38	4.63	5.25	8.56	8.71	7.53	11.49	11.52
W&D	0.00	0.30	5.91	3.45	5.58	11.84	9.73	16.28	21.90	17.46	29.33	32.57
DeepFM	0.00	0.09	0.18	0.36	3.13	3.23	0.58	6.60	6.71	1.30	9.58	9.64

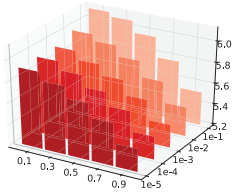
Table 3: AUC Improvements on Tencent 2018

Phase	Cold-start			Warm-up-1			Warm-up-2			Warm-up-3		
Model	Base	ME	Ours	Base	ME	Ours	Base	ME	Ours	Base	ME	Ours
FM	0.00	0.10	0.46	0.03	0.14	0.51	0.06	0.20	0.54	0.09	0.27	0.58
DNN	0.00	0.02	0.08	0.02	0.08	0.12	0.02	0.10	0.13	0.03	0.10	0.13
W&D	0.00	0.09	3.17	0.02	0.40	3.37	0.04	0.69	3.58	0.07	0.98	3.81
DeepFM	0.00	0.05	0.09	0.01	0.16	0.20	0.02	0.19	0.22	0.02	0.21	0.25

this figure that 1) in the cold-start phase, the performance starts degradation when it is larger than 0.1; 2) after warming up, the best performance is also achieved when α is 0.1. In other words, if α is assigned as 0.1, the best performance can be achieved in both cold-start and warm-up phases. Figure 5 shows the impact of the hyper-parameter ρ, i.e. the strength of adversarial perturbation on task distribution. From this figure, we can observe that 1) in the cold-start phase, model performance is stable when ρ is assigned with small values and starts degradation when ρ is sufficiently large; 2) after the warm-up phase, the final performance is relatively stable when ρ smaller and increases when ρ arrives 0.01. In other words, the impact of ρ shows the opposite trend in two phases. So there is a trade-off with respect to ρ between the cold-start and warm-up performance.



(a) Cold-start phase

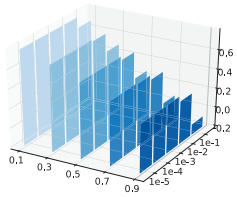


(b) Warm-up phase

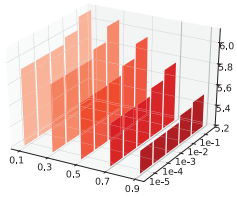
Figure 4: AUC Improvements w.r.t Hyper-parameter α. In both sub-figures, each 2D histograms means how α impacts $AUC_{impr.}$. The colors from dark to light are values of ρ. The bars in each 2D histogram are values of α. The height of each bar is $AUC_{impr.}$.

4 CONCLUSION

In this paper, our goal is to improve CTR prediction in the cold-start scenario. We first recast the cold-start CTR prediction as a meta-learning problem by viewing each ad as individual task, then propose an adaptive loss that can cope with the diversity of task difficulty and distribution perturbation. Moreover, we implement an algorithm to minimize the adaptive loss. Finally, the experimental results show that our proposed method can improve the predictions on those hardest unknown tasks, thus alleviating the cold-start problem in CTR prediction.



(a) Cold-start phase



(b) Warm-up phase

Figure 5: AUC Improvements w.r.t Hyper-parameter ρ. In both sub-figures, each 2D histograms means how ρ impacts $AUC_{impr.}$. The colors from dark to light are values of α. The bars in each 2D histogram are values of ρ. The height of each bar is $AUC_{impr.}$.

ACKNOWLEDGEMENTS

This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0102003, in part by National Natural Science Foundation of China: 61620106009, U1636214, 61931008, 61836002, 61672514 and 61976202, in part by Key Research Program of Frontier Sciences, CAS: QYZDJ-SSW-SYS013, in part by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDB28000000, in part by Beijing Natural Science Foundation (4182079), and in part by Youth Innovation Promotion Association CAS.

REFERENCES

[1] Aharon Ben-Tal and Arkadi Nemirovski. 2002. Robust optimization–methodology and applications. *Mathematical Programming* 92, 3 (2002), 453–480.

[2] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[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 *workshop on DLRS*. 7–10.

[4] John M. Danskin. 1966. The Theory of Max-Min, with Applications. *SIAM J. Appl. Math.* 14, 4 (1966), 641–664.

[5] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. 2008. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*. 272–279.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. 1126–1135.

[7] Wenjing Fu, Zhaohui Peng, Senzhang Wang, Yang Xu, and Jin Li. 2019. Deeply Fusing Reviews and Contents for Cold Start Users in Cross-Domain Recommendation Systems. In *AAAI*. 94–101.

[8] Quanquan Gu, Jie Zhou, and Chris Ding. 2010. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *SDM*. 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 *IJCAI*. 1725–1731.

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

[11] Fuxing Hong, Dongbo Huang, and Ge Chen. 2019. Interaction-Aware Factorization Machines for Recommender Systems. In *AAAI*. 3804–3811.

[12] Liang Hu, Songlei Jian, Longbing Cao, Zhiping Gu, Qingkui Chen, and Artak Amirbekyan. 2019. HERS: Modeling Influential Contexts with Heterogeneous Relations for Sparse and Cold-Start Recommendation. In *AAAI*. 3830–3837.

[13] Yuchin Juan, Damien Lefortier, and Olivier Chapelle. 2017. Field-aware Factorization Machines in a Real-world Online Advertising System. In *WWW*. 680–688.

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

[15] Liang Lan and Yu Geng. 2019. Accurate and Interpretable Factorization Machines. In *AAAI*. 4139–4146.

[16] 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 *SIGKDD*. 1754–1763.

[17] Ming Lin, Shuang Qiu, Jieping Ye, Xiaomin Song, Qi Qian, Liang Sun, Shenghuo Zhu, and Rong Jin. 2019. Which Factorization Machine Modeling Is Better: A Theoretical Answer with Optimal Guarantee. In *AAAI*. 4312–4319.

[18] Tianyi Lin, Chi Jin, and Michael I Jordan. 2019. On Gradient Descent Ascent for Nonconvex-Concave Minimax Problems. *arXiv preprint arXiv:1906.00331* (2019).

[19] Greg Linden, Brent Smith, and Jeremy York. 2003. Industry Report: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Distributed Systems Online* 4, 1 (2003).

[20] Jiawei Liu, Zheng-Jun Zha, Di Chen, Richang Hong, and Meng Wang. 2019. Adaptive Transfer Network for Cross-Domain Person Re-Identification. In *CVPR*. 7202–7211.

[21] Weiwen Liu, Ruiming Tang, Jiajin Li, Jinkai Yu, Huifeng Guo, Xiuqiang He, and Shengyu Zhang. 2018. Field-aware probabilistic embedding neural network for CTR prediction. In *RecSys*. 412–416.

[22] Hongseok Namkoong and John C Duchi. 2017. Variance-based regularization with convex objectives. In *NIPS*. 2971–2980.

[23] 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 *SIGKDD*. 2078–2086.

[24] 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*.

[25] 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*. 1349–1357.

[26] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *SIGKDD*. ACM, 2671–2679.

[27] Surabhi Punjabi and Priyanka Bhatt. 2018. Robust Factorization Machines for User Response Prediction. In *WWW*. 669–678.

[28] 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*. 1149–1154.

[29] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, and Kun Gai. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*. 565–574.

[30] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.

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

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

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

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

[35] 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.

[36] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.

[37] Alexandre B. Tsybakov. 2009. *Introduction to Nonparametric Estimation*.

[38] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. In *NIPS*. 6904–6914.

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

[40] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *NIPS*. 4957–4966.

[41] Qianqian Wang, Fang'ai Liu, Shuning Xing, Xiaohui Zhao, and Tianlai Li. 2019. Research on CTR Prediction Based on Deep Learning. *IEEE Access* (2019), 12779–12789.

[42] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *ADKDD, 2017*. 12:1–12:7.

[43] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *IJCAI*. 3119–3125.

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

[45] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data - - A Case Study on User Response Prediction. In *ECIR*. 45–57.

[46] 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 (2016), 1147–1159.

[47] 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*. 5941–5948.

[48] 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 *SIGKDD*. 1059–1068.