MAMDR: A Model Agnostic Learning Method for Multi-Domain Recommendation

Linhao Luo[1], Yumeng Li[2], Buyu Gao[2], Shuai Tang[3], Sinan Wang[2], Jiancheng Li[2], Tanchao Zhu[2], Jiancai Liu[2], Zhao Li[4], Binqiang Zhao[2], Ziyang Zheng[3], and Shirui Pan[1]*
[1]Monash University, [2]Alibaba Group, [3]Independent Researcher
[4]Alibaba-ZJU Joint Research Institute of Frontier Technologies, Zhejiang University
{linhao.luo,shirui.pan}@monash.edu,bellium@163.com,zhao_li@zju.edu.cn,ziyang.zheng0517@gmail.com
{lym174806,gaobuyu.gby,sinan.wsn,ljc250108,tanchao.zhutc,jiancai.ljc,binqiang.zhao}@alibaba-inc.com

ABSTRACT

Large-scale e-commercial platforms in the real-world usually contain various recommendation scenarios (domains) to meet demands of diverse customer groups. Multi-Domain Recommendation (MDR), which aims to jointly improve recommendations on all domains, has attracted increasing attention from practitioners and researchers. Existing MDR methods often employ a shared structure to leverage reusable features for all domains and several specific parts to capture domain-specific information. However, data from different domains may conflict with each other and cause shared parameters to stay at a compromised position on the optimization landscape. This could deteriorate the overall performance. Despite the specific parameters are separately learned for each domain, they can easily overfit on data sparsity domains. Furthermore, data distribution differs across domains, making it challenging to develop a general model that can be applied to all circumstances. To address these problems, we propose a novel model agnostic learning method, namely MAMDR, for the multi-domain recommendation. Specifically, we first propose a Domain Negotiation (DN) strategy to alleviate the conflict between domains and learn better shared parameters. Then, we develop a Domain Regularization (DR) scheme to improve the generalization ability of specific parameters by learning from other domains. Finally, we integrate these components into a unified framework and present MAMDR which can be applied to any model structure to perform multi-domain recommendation. MAMDR is scalable and has been deployed in Taobao. Extensive experiments on various real-world datasets and online applications demonstrate both the effectiveness and generalizability of MAMDR.

CCS CONCEPTS

• **Information systems → Information retrieval**.

*Corresponding author

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Multi-Domain Learning, Recommender System, Meta-Learning

ACM Reference Format:
Linhao Luo, Yumeng Li, Buyu Gao, Shuai Tang, Sinan Wang, Jiancheng Li, Tanchao Zhu, Jiancai Liu, Zhao Li, Binqiang Zhao, Shirui Pan, and Ziyang Zheng. 2022. MAMDR: A Model Agnostic Learning Method for Multi-Domain Recommendation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommender systems have been widely applied in many applications to provide personalized experiences for users. Conventional recommender systems [6, 14] are trained and predicted on samples collected from a single domain. However, large e-commercial platforms such as Taobao and Amazon need to build recommender systems for various domains to satisfy diverse user demands and stimulate users' purchases. For example, Taobao contains multiple business domains such as "what to take when traveling", "how to dress up yourself for a party", and "things to prepare when a baby is coming". Each domain contains a set of products that are related to the domain's topic and promotion strategies to stimulating purchasing. Thus, multiple domains are expected to be jointly considered to enable effective recommendation. Unlike cross-domain recommendation (CDR) [15] that only focuses on one or more target domains, multi-domain recommendation (MDR) [33, 38] aims to simultaneously improve recommendation results of all domains.

The main challenge of MDR is that the data distribution is not consistent across domains. Because distinct domains have only partially overlapping user and item groups, and varied domain marketing tactics result in diverse user behavior patterns. For example, during the domain of "Singles Day Promotion (Double 11)", China's largest online shopping event, customers are more inclined to click recommended goods due to the significant discounts, whereas some minor domains could have fewer user activities. These distinctions result in domain-specific data distributions and pose challenges for using the same model for all domains, as shown in Figure 1 (a). A simple solution is to build a separated model for each domain as shown in Figure 1 (b). However, this is not feasible for the large-scale MDR problem. Moreover, some domains do not have enough data to optimize a great separated model; meanwhile, the separated models disregard the shared information between domains and lead to a sub-optimal result.

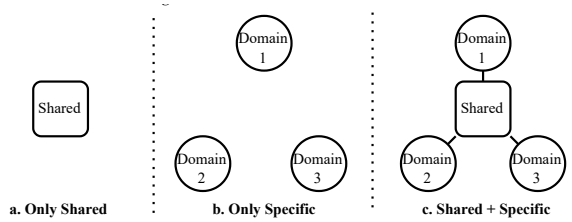


Figure 1: Existing Muti-Domain recommendation models. (a) one model for all domains, (b) one model for each domain, (c) one model with shared and domain-specific parameters.

Conventional MDR methods [25, 33, 44], inspired by Multi-task learning (MTL), treat each domain as a task. Most of them split model parameters into a set of shared parameters and domain-specific parameters, as shown in Figure 1 (c). The shared parameters are optimized by using all domains' data to leverage multi-domain features, and the specific parameters are optimized by using the domain-specific data to capture the domain distinction. However, these methods still have the following limitations:

(1) Shared parameters suffer from the domain conflict problem.
(2) Specific parameters are inclined to overfitting.
(3) Existing MDR models cannot generalize to all circumstances.

(Limit. 1) the domain conflict problem is also occurred in MTL [22, 43] and Domain Generalization (DG) [27]. The shared parameters are trained using gradient descent, in which gradients guide the parameters as they travel through the loss landscape. The gradients from each domain contain the domain-specific information, which could lead to disagreements among domains, i.e., gradients point to opposite directions. This ends up causing the shared parameters to stay at a compromised position on the optimization landscape and deteriorate the overall performance. **(Limit. 2)** the specific parameters are separately optimized on each domain's data. Thus, they can easily overfit on data sparsity domains and cannot generalize well to the unseen data. **(Limit. 3)** previous research [28] showed that the structure of shared and specific parameters are diverse in domains and essential to the final results. Due to the distinctive data distributions, it is difficult for existing MDR models to accommodate all circumstances. Despite some research efforts that can be adopted to address these limitations [4, 10, 18, 20, 43], they are either lack of scalability for the large-scale MDR problem or required to change the current model structure. This largely limits their applications in the real-world.

To address the aforementioned problems, we propose a novel model agnostic learning method for the multi-domain recommendation, denoted as MAMDR. In MAMDR, we simultaneously consider the optimization of shared parameters and specific parameters. Specifically, we first propose a scalable algorithm called Domain Negotiation (DN) (to address **Limit. 1**). DN optimizes the shared parameters by combining data from all domains. It can not only minimize the recommendation loss on each domain, but also reduce the domain conflict. DN encourages domain agreements by effectively maximizing the inner-products of gradients between domains, whose effectiveness and scalability are proved by both theoretical analyses as well as empirical studies. Second, we present a strategy denoted as Domain Regularization (DR) for specific parameters (to address **Limit. 2**). DR optimizes the domain-specific parameters using data from other domains. Extending the analysis of DN, DR enables the optimization process on other domains to minimize the loss on the specific domain. With the help of other domains, DR can alleviate the overfitting problem on data sparsity domains. Third, our method nicely integrates DN and DR into a unified framework for multi-domain recommendation. As MAMDR does not require any constraints on the model structure, it can be readily applied to any existing recommendation model and greatly improves its performance (to address **Limit. 3**).

The main contributions of this paper are summarized as follows:

- We propose two scalable algorithms: Domain Negotiation (DN) and Domain Regularization (DR) to alleviate the domain conflict and overfitting problem in MDR. Theoretical analyses are also provided to ensure the effectiveness.
- We propose a novel general model agnostic learning method: MAMDR, to address the MDR problem, which are compatible with arbitrary model structures. Extensive experiments on various public real-world datasets show that MAMDR can significantly improve the performance of existing MDR methods.
- We have provided a disturbed implementation of MAMDR to support large-scale applications, which has been deployed in Taobao to improve the MDR performance. Recommendation results on 69,102 different domains demonstrate both the effectiveness and scalability of MAMDR.

2 RELATED WORK

2.1 Multi-Domain Recommendation

Recommender system has been a long-standing research topic. Recently, many deep learning-based recommender systems have been proposed (e.g., WDL [6], NeurFM [14], and AutoInt [35]). However, most of them are designed for single domain recommendation. In the real-world, huge data are collected from multi-domains. Related to MDR, cross-domain recommendation (CDR) aims to improve the performance of the target domain with the help of auxiliary domains [17]. CDCF [24] extended the traditional matrix factorization to CDR with interacting information from auxiliary domains to target domains. Following the idea of transfer learning, CoNet [15] enabled a dual knowledge transfer across domains by introducing cross-connections from two networks. s^2Meta [8] unified the meta-learning and CDR and learned generic initial parameters for different domains. In contrast to CDR, our multi-domain recommendation (MDR) problem aims to improve the results of all domains.

The MDR problem has recently garnered considerable attention. MCF [44] applied several collaborative filtering tasks in various domains at the same time to investigate domain connections. Similarly, ICAN [40] also focused on the interactions across domains. Ma et al. [26] enabled the cross-media content features to be used in multi-domain collaborative filtering. Furthermore, AFT [12] proposed a novel adversarial feature translation to learn features translation between different domains. Additionally, by considering each domain as a task, multi-task approaches (e.g., Shared-Bottom [31]) could be simply deployed to tackle the MDR problem. MMoE [25] extended

the Shared-Bottom by designing several domain expert networks and a novel multi-gate strategy. On the top of MMoE, PLE [37] proposed a shared-specific expert network to progressively extract specific and multi-domain features. Inspired by MTL, STAR [33] separated the model parameters into shared and domain-specific parts. Meanwhile, STAR proposed a novel partitioned normalization to model the distinct distribution of different domains. However, these methods still suffer from the domain conflict and overfitting problems as discussed in the section 1.

2.2 Multi-Domain Learning

Multi-Domain Learning (MDL) [41] has been widely applied in the real-world. Some MDL research can be extended to solve the problems in MDR. The domain generalization (DG) methods seek to distill the common knowledge from multi-domains and learn more robust features that are potentially useful for unseen domains. Existing research [19] assumed that any domain can be factorized into a shared and domain-specific component. Mansilla et al. [27] analyzed the multi-domain gradient conflict and adapted the PCGrad [43] into the multi-domain setting. The PCGrad alleviates the gradient conflict of two domains by projecting their gradients into the non-conflict direction. However, this strategy is inefficient for scenarios with more than two domains. MLDG [20] improved the generalizability of learned parameters by extending the idea of meta-learning. By using meta-learning, Dou et al. [7] introduced two complementary losses to explicitly regularize the semantic structure of the feature space. Similarity, Sicilia et al. [34] adopted meta-learning to balance the losses between different domains. MetaReg [2] also adopted the meta-learning as a regularization term to achieve good multi-domain generalization.

2.3 Meta-Learning

Meta-learning (e.g., MAML [9] and Reptile [29]) aims to learn generalized initialized weights that can be readily tailored to new tasks, which is agnostic to model structure. Meta-learning acquires the shared knowledge across tasks and enables specific information through a few-step finetuning [36]. Meta-learning may be used to the MDR problem by treating each task as a domain. To address the gradient conflict, MT-net [18] enables the parameters to be learned on task-specific subspaces with distinct gradient descent. WarpGrad [10] further proposed a Warped Gradient Descent that facilitates the gradient descent across the task distribution. L2F [1] introduced a task-and-layer-wise attenuation mask on the initial parameters to alleviate the conflict among tasks. GradDrop [39] presented a meta-learning-based Gradient Dropout to avoid overfitting for certain tasks. TADAM [30] developed a metric scaling method to provide task-dependent metric space for optimizing specific parameters. HSML [42] introduced a hierarchical task clustering structure to preserve generalization knowledge among tasks, while also maintaining the specific information.

Even some multi-domain learning and meta-learning methods can be applied to the MDR problem, they are not as effective as the proposed MAMDR. Our method enables scalable implementation in the large-scale MDR problem and is compatible with the existing recommendation models.

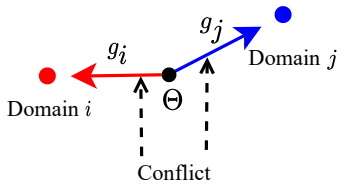


Figure 2: The illustration of domain conflict. The conflict happens when the inner-product of gradients from different domains is negative.

3 PRELIMINARY

3.1 Problem Definition

Multi-Domain Recommendation (MDR) [45] aims to recommend a group of products to a set of users from multiple domains. This can be defined as follows:

Definition 3.1 (Multi-Domain Recommendation). Given n different domains $\mathcal{D} = \{D^1, \cdots, D^n\}$, each domain $D^i = \{U^i, V^i, T^i\}$ includes a set of users $u \in U^i$ and items $v \in V^i$, where users and items could overlap across domains. The T^i denotes a set of user-item interaction histories $(u, v, y) \in T^i$, where $y \in \{1, 0\}$ indicates whether user u clicked item v or not. Thus, Multi-Domain Recommendation aims to use interaction histories of all domains $\mathcal{T} = \{T^1, \cdots T^n\}$ to train a model with parameter Θ, which could successfully predict interesting items for users in all domains. The object function of MDR can be formulated as follows:

$$O_M = \min_{\Theta} \sum_{i=1}^{n} L(\Theta, T^i), \tag{1}$$

where L denotes the recommendation loss function, and $L(\Theta, T^i)$ denotes the loss on domain D^i with training data T^i.

As we discussed in the section 1, some MDR methods split the model parameters Θ into a set of shared parameters θ^S and a set of domain-specific parameters $\{\theta^i | i \in [1, n]\}$. The θ^S are optimized by data from all domains to capture multi-domain features, and the θ^i is updated on specific domain to capture distinction. Thus, the objective function of these models can further formulated as:

$$O_M = \min_{\Theta=\{\theta^S, \theta^i\}} \sum_{i=1}^{n} L(\theta^S, T^i) + L(\theta^i, T^i), \tag{2}$$

where θ^S are optimized by data from all domains, and θ_i are only optimized by corresponding domain's data.

3.2 Domain Conflict

Directly optimizing equations 1 and 2 may deteriorate the recommendation performance. A primary cause for this phenomena is known as domain conflict, which is shown in the Figure 2.

For parameters optimized across domains, the gradient from each domain D^i is denoted as $g_i = \nabla L(\Theta, T^i)$. The Θ are optimized following the direction of g_i, i.e., $\Theta \leftarrow \Theta - \alpha \cdot g_i$, where α is the learning rate. However, the gradients from different domains may conflict with each other when they point to dissimilar directions. This dissimilarity could be represented by the inner-product between gradients. Thus, the conflict happens when the inner-product

of gradients from different domains is negative, i.e., $\langle g_i, g_j \rangle < 0$, where $\langle \cdot, \cdot \rangle$ denotes the inner-product between two gradients. As observed by previous research [22, 43], this conflict will impair the optimization and lead parameters to stay at a compromise point at the loss landscape, which also results in poor recommendation results.

3.3 Prior Attempts and Limitations

Some research efforts have been made to solve domain conflict. In the area of MTL, Alex et al. [16] utilizes a learned weighted loss to balance the gradients from different domains. PCGrad [43] relieves the conflict by projecting the gradient into the normal plane of others. GradDrop [5] samples gradients at an activation layer based on their level of consistency. In the area of meta-learning, the conflicts of gradients could be averse by projecting them into a common space [10] or minimizing the inner-product of them [20].

However, the MTL methods that manipulate the gradients could face the convergence problem and stay at a sub-optimal point of the loss landscape [22]. Meanwhile, the meta-learning methods are originally proposed for applying in unseen domains, which might not fit well to existing domains' data. Last, the above methods either are required to modify the model structure or lack scalability for large MDR. Despite some MDR methods propose using domain-specific parameters, as shown in equation 2, their share parameters θ^S still suffer the aforementioned problems. Besides, the uses of specific parameters also meet the overfitting problem when the domain data is insufficient.

Thus, in MAMDR, we introduce the Domain Negotiation (DN) and Domain Regularization (DR) strategies to solve the aforementioned challenges.

4 APPROACH

In this section, we will first introduce the Domain Negotiation and Domain Regularization in subsection 4.1 and 4.2, respectively. Then, the theoretical analyses for DN and DR will be discussed in subsection 4.3. Last, we will introduce the overall algorithm of MAMDR in subsection 4.4 and the large-scale implementation of MAMDR in subsection 4.5.

4.1 Domain Negotiation (DN)

Domain Negotiation (DN) is proposed to mitigate the domain conflict problem. Given n different domains, the Domain Negotiation (DN) is performed as follows:

As shown in the Algorithm 1, DN consists two training loops: the *outer loop* (line 1-8) and *inner loop* (line 4-6). At the beginning of each inner loop, the $\widetilde{\Theta}_1$ are initialized by Θ. Then, during the inner loop, the $\widetilde{\Theta}_i$ are sequentially updated on each domain i with random order, which can be formulated as:

$$\widetilde{\Theta}_{i+1} \leftarrow \widetilde{\Theta}_i - \alpha \cdot \triangledown L(\widetilde{\Theta}_i, T^i), \tag{3}$$

where T^i is the data from domain i, and α denotes the inner-loop learning rate. After inner loop, we treat $\widetilde{\Theta}_{n+1} - \Theta$ as the gradient for outer loop optimization, which directly updates the parameters Θ. This can be formulated as:

$$\Theta \leftarrow \Theta + \beta \cdot (\widetilde{\Theta}_{n+1} - \Theta), \tag{4}$$

Algorithm 1: Domain Negotiation (DN)

Input: n different domains $\mathcal{D}$, initial model parameters Θ, learning rate α and β, maximum training epoch N.
Output: Θ

1 **for** $epoch = 1, \cdots, N$ **do**
2 $\widetilde{\Theta}_1 \leftarrow \Theta$;
3 Randomly shuffle $\mathcal{D}$;
4 **for** $i \leftarrow 1, \cdots, n$ **do**
5 Update $\widetilde{\Theta}_{i+1} \leftarrow \widetilde{\Theta}_i - \alpha \cdot \triangledown L(\widetilde{\Theta}_i, T^i)$;
6 **end**
7 Update $\Theta \leftarrow \Theta + \beta \cdot (\widetilde{\Theta}_{n+1} - \Theta)$;
8 **end**
9 **return** Θ

where β denotes the outer-loop learning rate. When β is set to 1, DN will degrade to Alternate Training in MTL [23], which could corrupt the performance as shown in the parameters analysis (section 5.4).

Noticeably, both the inner loop and outer loop can use arbitrary optimizers, such as traditional SGD, Adam or Parallelized SGD [3] for distributed training in large-scale applications.

4.2 Domain Regularization (DR)

Despite DN being a powerful strategy for mitigating domain conflict of shared parameters, the specific parameters are still prone to overfit on data sparsity domains. In this section, we will introduce Domain Regularization (DR) for optimizing domain-specific parameters, which greatly improves the performance of MAMDR.

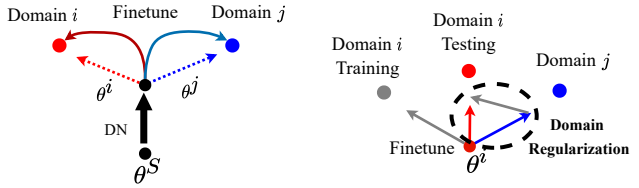
Traditionally, after optimizing across domains, the model will be finetuned on each specific domain to obtain several domain-specific models. Recently, some MDR methods [33] proposed to use domain-specific parameters to replace the finetune process. The domain-specific parameters θ^i have the same dimension as the shared parameters θ^S, but they are only optimized by domain-specific data. The θ^i are integrated with shared parameters θ^S by an element-wise add operation, which can be formulated as:

$$\Theta = \theta^S + \theta^i. \tag{5}$$

As shown in Figure 3 (a), the θ^i can be treated as the direction pointing to the endpoint of the finetune process, thus it can achieve similar results as finetune.

However, one major issue of finetune is that it is easy to overfit on some data sparsity domains. For example, as shown in Figure 3 (b), the θ^i is optimized on a data sparsity domains i. Though θ^i could perfectly fit on the training data of domain i, it cannot generalize well on the testing data as shown in the gray arrow. The Domain regularization (DR), as circled in Figure 3 (b), optimizes θ^i with the help of other domains' data to improve θ^i's generalizability. The details of Domain Regularization are shown in Algorithm 2.

Given a target domain D^i and its specific parameters θ^i, DR samples k different domains $\widetilde{\mathcal{D}}$ from $\mathcal{D}$. For each $D^j \in \widetilde{\mathcal{D}}$, DR first update $\widetilde{\theta}^i$ on D^j, then update it on D^i as regularization, which can



a. Specific Parameters and Finetune **b. Domain Regularization**

Figure 3: (a) The similarity between domain-specific parameters and finetune. (b) The proposed Domain Regularization.

Algorithm 2: Domain Regularization (DR)

Input: n different domains $\mathcal{D}$, target domain D^i, specific parameters θ^i, learning rate α, γ, sample number k
Output: θ^i

Sample k domains from $\mathcal{D}$ as $\widetilde{\mathcal{D}}$;
for D^j *in* $\widetilde{\mathcal{D}}$ **do**
  $\widetilde{\theta^i} \leftarrow \theta^i$;
  Update $\widetilde{\theta^i} \leftarrow \widetilde{\theta^i} - \alpha \cdot \nabla L(\widetilde{\theta^i}, T^j)$ # Update on domain j;
  Update $\widetilde{\theta^i} \leftarrow \widetilde{\theta^i} - \alpha \cdot \nabla L(\widetilde{\theta^i}, T^i)$ # Using domain i as regularization;
  Update $\theta^i \leftarrow \theta^i + \gamma \cdot (\widetilde{\theta^i} - \theta^i)$;
end
return θ^i

be formulated as:

$$\widetilde{\theta^i} \leftarrow \theta^i, \tag{6}$$

$$\widetilde{\theta^i} \leftarrow \widetilde{\theta^i} - \alpha \cdot \nabla L(\widetilde{\theta^i}, T^j), \tag{7}$$

$$\widetilde{\theta^i} \leftarrow \widetilde{\theta^i} - \alpha \cdot \nabla L(\widetilde{\theta^i}, T^i). \tag{8}$$

At last, the $\widetilde{\theta^i} - \theta^i$ is denoted as the gradient to update θ^i, which can be formulated as:

$$\theta^i \leftarrow \theta^i + \gamma \cdot (\widetilde{\theta^i} - \theta^i), \tag{9}$$

where γ is the learning rate for DR.

Unlike DN, in which the domain order is random at each iteration, the optimization sequence is fixed in DR. We first update specific parameters on domain j, then update them on the target domain i. In this way, we can make sure that only the helpful information from domain j is extracted for the target domain. The detailed analysis can be found at section 4.3.

4.3 Theoretical Analysis

In this section, we first provide theoretical analyses to explain why DN could mitigate the domain conflict problem. Extending the analysis of DN, we also provide the explanation for DR.

Unlike previous methods, DN does not directly manipulate gradients or change model structure. Instead, DN tries to maximize the inner-products of gradients between domains during the optimization. As discussed in section 3.2, the domain conflict happens when the inner-product of gradients is negative. Thus, previous methods like PCGrad [43] ensues the positiveness of the inner-product by projecting one gradient to the normal plane of others. This can be formulated as:

$$g_i = g_i - \frac{\langle g_i, g_j \rangle}{||g_j||^2} g_j. \tag{10}$$

Clearly, PCGrad has a $O(\frac{n(n-1)}{2})$ time complexity, which is not feasible for large-scale applications. Despite it proposed a sampling strategy to reduce computational complexity, the sampling also introduces more uncertainties and could lead the model to converge at any point in the loss landscape [22]. Another solution is to maximize the inner-products of gradients between domains during the optimization, which can be formulated as:

$$O_C = \max \sum_i^n \sum_j^n \langle g_i, g_j \rangle. \tag{11}$$

However, directly optimizing O_C also requires a $O(n^2)$ complexity.

In DN, we sequentially perform training on each domain, which provides a sequence of loss $L(\widetilde{\Theta}_i, T^i)$ that can be simplified as $L_i(\widetilde{\Theta}_i)$. We also define the following notations to facilitate analysis:

$$g_i = L_i'(\widetilde{\Theta}_i) \quad \text{(gradients from domain } i\text{)}, \tag{12}$$

$$\overline{g}_i = L_i'(\widetilde{\Theta}_1) \quad \text{(gradients from domain } i \text{ at initial point)}, \tag{13}$$

$$\overline{H}_i = L_i''(\widetilde{\Theta}_1) \quad \text{(Hessian at initial point)}, \tag{14}$$

$$\widetilde{\Theta}_i = \widetilde{\Theta}_1 - \alpha \sum_{j=1}^{i-1} g_j \quad \text{(sequence of gradient descent)}. \tag{15}$$

We can perform the Taylor expansion on the g_i when α is small enough, which can be formulated as:

$$g_i = L_i'(\widetilde{\Theta}_1) + L''(\widetilde{\Theta}_1)(\widetilde{\Theta}_i - \widetilde{\Theta}_1) + O(\alpha^2), \tag{16}$$

$$= \overline{g}_i + \overline{H}_i(\widetilde{\Theta}_i - \widetilde{\Theta}_1) + O(\alpha^2), \tag{17}$$

$$= \overline{g}_i - \alpha \overline{H}_i \sum_{j=1}^{i-1} g_j + O(\alpha^2), \tag{18}$$

$$= \overline{g}_i - \alpha \overline{H}_i \sum_{j=1}^{i-1} \overline{g}_j + O(\alpha) \quad (\text{using } g_j = \overline{g}_j + O(\alpha)). \tag{19}$$

Then, the gradients $\widetilde{\Theta}_{n+1} - \Theta$ for outer loop can be formulated as:

$$-(\widetilde{\Theta}_{n+1} - \Theta)/\alpha = \sum_{i=1}^n g_j = \sum_{i=1}^n \overline{g}_i - \alpha \sum_{i=1}^n \sum_{j=1}^{i-1} \overline{H}_i \overline{g}_j + O(\alpha). \tag{20}$$

In equation 20, the first term $\sum_{i=1}^n \overline{g}_i$ servers to jointly minimize loss on each domain. This makes sure the convergence of DN. The second term $\sum_{i=1}^n \sum_{j=1}^{i-1} \overline{H}_i \overline{g}_j$, which is more interesting, serves to maximize the inner-products of gradients between domains. Thus, we denote the expectation of $\overline{H}_i \overline{g}_j$ as the *InnerGrad*. Since the sequence of domains is shuffled at every epoch in the inner loop, the InnerGrad can be formulated as:

$$\text{InnerGrad} = \mathbb{E}(\overline{H}_i \overline{g}_j) = \mathbb{E}(\overline{H}_j \overline{g}_i), \tag{21}$$

$$= \frac{1}{2}\mathbb{E}(\overline{H}_i \overline{g}_j + \overline{H}_j \overline{g}_i), \tag{22}$$

$$= \frac{1}{2}\mathbb{E}\big(\frac{\partial}{\partial \Theta}\langle \overline{g}_i, \overline{g}_j \rangle\big). \tag{23}$$

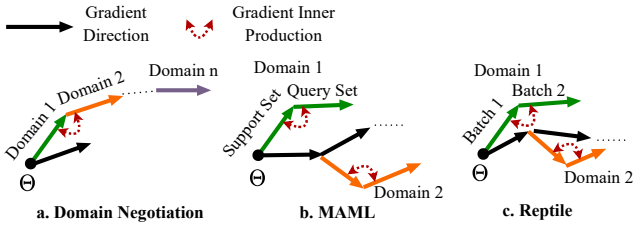


Figure 4: The illustration of (a) Domain Negotiation, (b) MAML, and (c) Reptile.

Clearly, the (-InnerGrad) is the direction that increases the inner-products of gradients between domains. Therefore, the gradient shown in equation 20 can not only minimize O_M for multi-domain recommendation, but also maximize O_C for mitigating domain conflict. What is more, the overall computational complexity of DN is $O(n)$, which makes it more suitable for large-scale applications.

Similar conclusion about maximizing the gradients' inner production has also been used by meta-learning methods (i.e., MAML [9] and Reptile [29]). However, as shown in Figure 4, MAML and Reptile maximize the gradients' inner production within the same domain, which only improves its single-domain generalizability, but is unable to mitigate conflict across domains. In DN, we successfully maximize gradients' inner production across domains, which is a key contribution to mitigating domain conflict.

The theoretical analysis of DR can be extended from equation 19 and 20. For each D^j, we first optimize $\widetilde{\theta^i}$ on D^j, then update on target domain D^i for regularization. The gradients $\widetilde{\theta^i} - \theta^i$ for updating specific parameters can be formulated as:

$$-(\widetilde{\theta^i} - \theta^i)/\alpha = g_j + g_i = \overline{g}_j + \overline{g}_i - \alpha \overline{H}_i \overline{g}_j. \tag{24}$$

Because the update sequence of D^j and D^i is fixed, the conclusion in equation 23 can not be applied to $\overline{H}_i \overline{g}_j$ here. The $\overline{H}_i$ denotes the Hessian matrix of target domain, thus the $\overline{H}_i \overline{g}_j$ in DR is to regularize the gradients $\overline{g}_j$ of D^j to best serve the optimization of target domain D^i. In this way, DR can easily adopt data from other domains to improve performance on the target domain D^i.

4.4 The MAMDR Algorithm

The DN and DR could be integrated into a unified framework, i.e., MAMDR. The overall process of MAMDR is illustrated in Algorithm 3. Given n different domains and arbitrary model structures with parameters Θ, we split Θ into the shared parameters θ^S and specific parameters $\{\theta^1, \cdots, \theta^n\}$. In each iteration, we first update shared parameters θ^S using Domain Negotiation (line 2) to mitigate domain conflict. Then, for each specific parameter, we update θ^i using Domain Regularization (line 4) to improve generalizability. The overall complexity of MAMDR is $O\big((k+1)n\big)$. From Algorithm 3, we can see that our MAMDR is agonist to model structure and can be applied to any MDR method to meet varied circumstances.

4.5 Large-scale Implementation

To support large-scale applications, we adopt the PS-Worker architecture [21] for distribute training. PS-Worker is a commonly used data-parallel method for scaling model training on multiple machines, which contains two parts of machine: parameter servers and workers. Parameter servers store model parameters, and workers calculate the gradients.

The overall architecture is illustrated in Figure 5. (1) we first distribute the training data into m worker machines. (2) each worker obtains the parameters from the parameter server and stores as the local parameter. (3) the MAMDR algorithm is implemented in each worker to update the local parameters and compute the local gradient. (4) workers send their local gradients to the parameter server. (5) the parameter server synthesizes the local gradients and updates the model parameters. In this way, we can implement MAMDR in our applications to deal with billions of data.

Algorithm 3: MAMDR

Input: n different domains $\mathcal{D}$, shared parameters θ^S, domain-specific parameters $\{\theta^1, \cdots, \theta^n\}$, learning rate α, β, γ, sample size k, maximum training epoch N.

Output: $\Theta = \{\theta^S, \{\theta^1, \cdots, \theta^n\}\}$

1 **for** *epoch* $= 1, \cdots, N$ **do**
2 Update θ^S using Domain Negotiation (Algorithm 1);
3 **for** $i = 1, \cdots, n$ **do**
4 Update θ^i using Domain Regularization (Algorithm 2);
5 **end**
6 **end**
7 **return** $\Theta = \{\theta^S, \{\theta^1, \cdots, \theta^n\}\}$

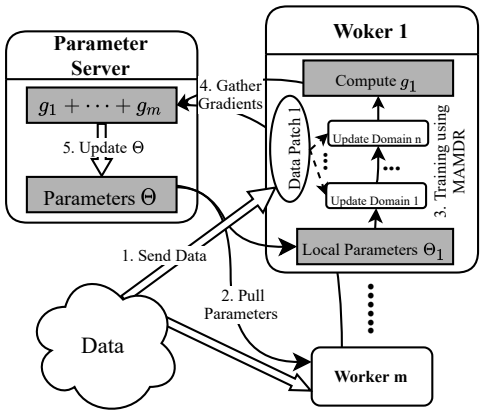


Figure 5: Implementation of MAMDR in PS-Worker architecture.

5 EXPERIMENT

In this section, we will detail the experiments to evaluate the performance of the proposed MAMDR. Code and data used in experiments are available at: https://github.com/RManLuo/MAMDR.

5.1 Dataset and Metric

In experiments, we conduct CTR prediction on two public datasets: Amazon [13] and Taobao [8]. The Amazon dataset is a large-scale dataset containing product reviews collected from Amazon.com. We split domains by the category of products. For each domain, we keep the existing user-product reviews as positive samples. We respectively select 6 and 13 domains forming two sub-datasets: Amazon-6 and Amazon-13. The Taobao dataset is from the click log of Cloud Theme in the Taobao app. It has already split domains by theme of purchase. Each domain contains a set of users and items, thus we treat the positive samples as the items clicked by users. We randomly select 10, 20, and 30 domains from the dataset to form three sub-datasets: Taobao-10, Taobao-20, and Taobao-30. For each dataset, we randomly select items that haven't been reviewed or clicked by users as the negative samples. The ratio of the negative sampling is also randomly set across domains to simulate domain distinction. The statistic of datasets is illustrated in Table 1. More details about the datasets can be found in Appendix A.1.

The area under the ROC curve (AUC) is the common metric used to evaluate the performance of CTR prediction. Thus, we select the AUC to evaluate the performance of the proposed MAMDR. The average AUC of all domains is adopted as the final evaluation metric.

Table 1: The statistic of datasets.

Dataset	# Domain	# User	# Item	# Train	# Val	# Test	Sample / Domain
Amazon-6	6	445,789	172,653	9,968,333	3,372,666	3,585,877	2,821,146
Amazon-13	13	502,222	215,403	11,999,607	4,100,756	4,339,523	1,572,299
Taobao-10	10	23,778	6,932	92,137	37,645	43,502	17,328
Taobao-20	20	58,190	16,319	243,592	96,591	106,500	22,334
Taobao-30	30	99,143	29,945	394,805	151,369	179,252	24,180

5.2 Performance Comparison

In this section, we will show the performance of the proposed MAMDR compared with other baseline models. We select two kinds of methods: (1) Single-Domain methods (e.g., MLP, WDL [6], NeurFM [14], AutoInt [35], and DeepFM [11]); (2) Multi-Task (Domain) methods (e.g., Shared-bottom [31], MMOE [25], PLE [37], and Star [33]). Since our MAMDR is agnostic to model structure, we just use the simplest multi-layer perceptron (MLP) with three fully connected layers as the base model structure. These baselines are jointly trained using data from all domains, and the parameters settings are detailed in Appendix A.2.

The comparison results are shown in Table 2, where the best results are highlighted with bold type. From Table 2, we can see that MAMDR (DN + DR) greatly improves the performance of MLP and achieves the state-of-the-art results throughout all datasets.

In Amazon-6, AutoInt outperforms other single domain methods, but is beat by multi-domain methods that consider the domain distinction. However, with the number of domains increasing, both the performance of single-domain and multi-domain methods deteriorates in Amazon-13. This is because domain conflict is more likely to occur when facing more domains. Despite multi-domain modes capturing the distinction using specific parameters, their shared parameters still suffer from the conflict problem. The domain negotiation (DN), on the other hand, alleviates the domain conflict and improves the performance of MLP by 2.9% in Amazon-6 and 4.5% in Amazon-13. Furthermore, domain regularization (DR) takes advantage of the information from other domains and boosts the improvement of MLP by 6.6% and 8.0% in Amazon-6 and Amazon-13, respectively.

Table 2: Comparison with multi-domain recommendation methods under average AUC metric.

Method	Amazon-6	Amazon-13	Taobao-10	Taobao-20	Taobao-30
MLP	0.7464	0.7016	0.7022	0.7255	0.7416
WDL	0.7449	0.7026	0.7154	0.7235	0.7559
NeurFM	0.6505	0.6152	0.7374	0.7461	0.7673
AutoInt	0.7531	0.7214	0.7302	0.7471	0.7623
DeepFM	0.7333	0.6976	0.7271	0.7347	0.7484
Shared-bottom	0.7794	0.7088	0.7197	0.7572	0.7714
MMOE	0.7816	0.7381	0.7250	0.7494	0.7717
PLE	0.7801	0.7114	0.7287	0.7603	0.7725
Star	0.7719	0.7209	0.7202	0.7324	0.7483
MLP+DN	0.7678	0.7331	0.7204	0.7501	0.7619
MLP+MAMDR (DN+DR)	**0.7957**	**0.7577**	**0.7445**	**0.7613**	**0.7750**

In Taobao dataset, the performance of each model improves with domains' numbers increasing. The possible reason is that training samples of each domain are more sparse in Taobao dataset, which are shown in the Table 1 and Appendix A.1. Therefore, more domains introduce more training samples and improve the overall performance. This also indicates the importance of shared information for multi-domain recommendations. Although more domains could facilitate the performance, DN still enables to improve the performance of MLP. Because, according to equation 20, DN can simultaneously optimize losses of all domains to capture shared information and mitigate domain conflict. On top of DN, DR alleviates the overfitting problem for sparse domains and promotes the performance of MLP to the best place among all Taobao datasets.

Last, even some MDR models have complex structures (e.g., NeurFM, AutoInt, MMOE, and PLE), their performance cannot outperform a simple MLP optimized under proposed MAMDR. What is more, their performances are diverse from different datasets. For example, NeurFM achieves the best result among single-domain models in Taobao-30, but the worst in Amazon-6 and Amazon-13. This indicates that the existing models' structure is not suitable for all circumstances. In contrast, MAMDR has no restriction on model structure and could easily fit any datasets without burdensome hyper-parameters turning.

5.3 Learning Method Comparison

In this section, we will compare our MAMDR with other model agnostic learning methods under different model structures. The selected learning methods can be grouped into three categories: (1) traditional learning methods (e.g., Joint and Joint+Finetune); (2) multi-task learning methods (e.g., Weighted Loss [16] and PCGrad [43]); (3) meta-learning methods (e.g., MAML [9], Reptile [29], and MLDG [20]). The Joint method directly trains model with all domains' data, and the Joint+Finetune denotes finetuning on each domain after the Joint training process. More details about other methods can be found in the Appendix A.3. We conduct experiments on Taobao-10, and the results are shown in Table 3.

Table 3: Comparison with other learning methods under average AUC metric.

Method	Joint	Joint+Finetune	Weighted Loss	PCGrad	MAML	Reptile	MLDG	DN	MAMDR (DN+DR)
MLP	0.7022	0.7126	0.7157	0.7254	0.6896	0.7117	0.7074	0.7204	**0.7445**
WDL	0.7154	0.7040	0.7098	0.7153	0.6945	0.7212	0.7182	0.7295	**0.7376**
NeurFM	0.7154	0.7465	0.7393	0.7526	0.7479	0.7579	0.7543	0.7572	**0.7609**
DeepFM	0.7271	0.7280	0.7259	0.7562	0.7237	0.7402	0.7480	0.7352	**0.7581**
Shared-bottom	0.7197	0.7225	0.7171	0.7269	0.6816	0.7255	0.7195	0.7233	**0.7339**
Star	0.7202	0.7303	0.7297	0.7221	0.7228	0.7353	0.7181	0.7328	**0.7520**

From the results in Table 3, we can clearly find that MAMDR outperforms all learning methods with respect to all model structures. For traditional learning methods, simply finetuning on each domain could improve performance for most models. But the performance of WDL slightly drops after finetuning, which may be due to the overfitting on certain domains.

Among multi-task learning methods, PCGrad performs better than Weighted Loss. PCGrad tries to solve domain conflict problems by projecting gradients from two domains into the non-conflict direction. But Weighted Loss only adjusts the weight of loss for different domains, which cannot fundamentally solve the domain conflict problem. In addition, Weighted Loss could give bigger weight to the domain that is easy to train. In this way, the model could end up staying at the local optimal point of that domain rather than the global optimal point for all domains. Though the effectiveness of PCGrad, its gradient manipulation could lead model to stay at random points during training. Meanwhile, the complexity of PCGrad is $O(n^2)$, which is unacceptable for large-scale MDR.

Meta-learning methods try to maximize the inner-product between gradients and improve generalizability. Among them, MAML achieves the worst results. The possible reason is that MAML was originally proposed for few-shot learning on unseen domains. It splits the training samples into two subsets (query and support set), which cannot fully utilize the training sets. Reptile and MLDG, on the other hand, do not split the training samples and thus reach better results. However, they only maximize the inner-product within the same domain rather than across domains.

For our methods, DN has already reached the best results in most datasets by mitigating domain conflict. On top of DN, DR takes advantage of the information from other domains further improving the performance and reaching the best results.

5.4 Parameters Analysis

In this section, we will analyze the parameters setting of our MAMDR. First, we analyze the number of sample domains used in Domain Regularization. Experiments are conducted on Taobao-30 with various sample number k. From the results shown in the Figure 6, we can see that with the sample number increasing, the performance of the model first improves and drops at $k = 5$. Because updating using too many domains will lead specific parameters deviating much from the shared parameters and impairing results. In addition, this also shows that DR would not need many domains to enhance the performance, which guarantee the computational efficiency.

Second, we analyze the effect of inner-loop learning rate α and outer-loop learning rate β in DN. From the results shown in the Figure 7, we can see that the best performance is achieved with $\alpha = 1e^{-3}$ and $\beta = [0.5, 0.1]$. The reason is that according to the analysis in equation 19, the Taylor expansion only holds when α is small enough. Thus, the model is barely trained when $\alpha = 1e^{-1}$ or $1e^{-2}$. Besides, the results also show that using slightly bigger β would not impair the performance, and it can also improve the training speed. Noticeably, when the outer-learning rate is set to 1, the performance drops. Because when $\beta = 1$, the DN will degrade to Alternate Training in MTL, which could be affected by some data richness domains and cannot maximize the inner-gradient as DN does.

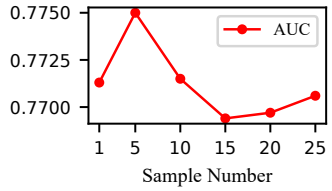


Figure 6: Results on Taobao-30 under different sample number k.

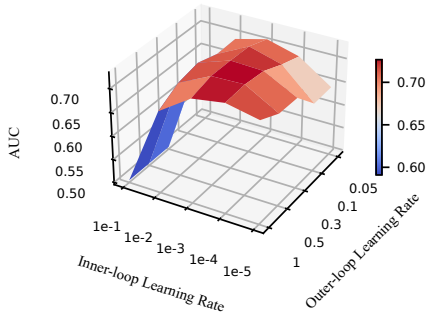


Figure 7: Results under different inner-loop learning rate α and outer-loop learning rate β.

5.5 Online Results in Taobao

In this section, we will report the results of our MAMDR in online applications. We have implemented MAMDR in Taobao, which is one of the largest online shopping applications in China. The dataset used in online experiments contains 69,102 different domains and 489,852,853 samples. More details of online dataset can be found in Appendix A.1.

Table 4: Results on online applications.

Methods	RAW+Separate	RAW+Joint	MMOE	CGC	PLE	RAW+DN	RAW+MAMDR
AUC	0.7460	0.7503	0.7497	0.7489	0.7513	0.7559	**0.7700**

Table 5: Results on 10 largest domains of online applications.

Methods	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
RAW+Separate	0.8127	0.7635	0.8285	0.7569	0.6896	0.7367	0.6701	0.7370	0.7283	0.6947
RAW+Joint	0.8202	0.7635	0.8439	0.7295	0.6962	0.7417	0.6661	0.7524	0.7540	0.6912
MMOE	0.8166	0.7597	0.8288	0.7694	0.6945	0.7453	0.6677	0.7315	0.7478	0.6941
CGC	0.8172	0.7640	0.8307	0.7747	0.7215	0.7392	0.6726	0.7444	0.7357	0.7019
PLE	0.8158	0.7643	0.8261	0.7768	0.7327	0.7284	0.6793	0.7410	0.7472	0.7038
RAW+DN	0.8173	0.7655	0.8397	0.7643	0.7188	0.7344	0.6664	0.7523	0.7505	0.7021
RAW+MAMDR	**0.8226**	**0.7704**	**0.8469**	**0.8090**	**0.7391**	**0.7648**	**0.6965**	**0.7666**	**0.7689**	**0.7150**

We apply MAMDR to our existing recommender model (denoted as *RAW*) and compare it with other methods (i.e., MMOE [25], CGC [37], and PLE [37]). We first show the average AUC of 69,102 domains in Table 4. From the results, we can see that MAMDR successfully improves the performance of existing models and reaches the best results. Besides, performance of MMOE and CGC is slightly worse than the Joint training. The reason is that some domains have limited samples, which could lead specific parameters to overfit on them. Our MAMDR can not only mitigate the domain conflict to leverage shared features but also alleviate the overfitting problem. In Table 5, we present the results of top 10 largest domains in online applications. From results, we can see that MAMDR achieves the best performance among all selected domains, which shows the effectiveness of MAMDR on data richness domains. Last, experiments on large-scale online applications also demonstrate the scalability of MAMDR in the real-world.

6 CONCLUSION

In this paper, we propose a novel model agnostic learning method for multi-domain recommendation, denoted as MAMDR. MAMDR unifies the Domain Negotiation (DN) and Domain Regularization (DR) in the same framework to address the domain conflict and overfitting problem. We also provide a distributed implementation of MAMDR to support large-scale applications. Extensive results on various real-world datasets and online applications demonstrate both the effectiveness and generalizability of MAMDR. Furthermore, instead of the multi-domain recommendation, the proposed DN and DR have the potential to be used for other problems such as multi-task learning, domain adaptation, and domain generalization.

REFERENCES

[1] Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. 2020. Learning to forget for meta-learning. In *CVPR*. 2379–2387.

[2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. 2018. Metareg: Towards domain generalization using meta-regularization. *NIPS* 31 (2018), 998–1008.

[3] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*. PMLR, 794–803.

[5] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just Pick a Sign: Optimizing Deep Multitask Models with Gradient Sign Dropout. In *NIPS*, Vol. 33. 2039–2050.

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

[7] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. 2019. Domain generalization via model-agnostic learning of semantic features. *NIPS* 32 (2019), 6450–6461.

[8] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential scenario-specific meta learner for online recommendation. In *KDD*. 2895–2904.

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

[10] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. 2019. Meta-Learning with Warped Gradient Descent. In *ICLR*.

[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] Xiaobo Hao, Yudan Liu, Ruobing Xie, Kaikai Ge, Linyao Tang, Xu Zhang, and Leyu Lin. 2021. Adversarial Feature Translation for Multi-domain Recommendation. In *KDD*. 2964–2973.

[13] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.

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

[15] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. Conet: Collaborative cross networks for cross-domain recommendation. In *CIKM*. 667–676.

[16] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*. 7482–7491.

[17] Muhammad Murad Khan, Roliana Ibrahim, and Imran Ghani. 2017. Cross domain recommender systems: a systematic literature review. *ACM Computing Surveys (CSUR)* 50, 3 (2017), 1–34.

[18] Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*. PMLR, 2927–2936.

[19] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2017. Deeper, broader and artier domain generalization. In *ICCV*. 5542–5550.

[20] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2018. Learning to generalize: Meta-learning for domain generalization. In *AAAI*.

[21] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *USENIX*. 583–598.

[22] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021. Conflict-Averse Gradient Descent for Multi-task Learning. *NIPS* 34 (2021).

[23] Lianjie Long, Faliang Huang, Yunfei Yin, and Youquan Xu. 2021. Multi-task learning for collaborative filtering. *International Journal of Machine Learning and Cybernetics* (2021), 1–14.

[24] Babak Loni, Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Cross-domain collaborative filtering with factorization machines. In *European conference on information retrieval*. Springer, 656–661.

[25] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *KDD*. 1930–1939.

[26] Weizhi Ma, Min Zhang, Chenyang Wang, Cheng Luo, Yiqun Liu, and Shaoping Ma. 2018. Your Tweets Reveal What You Like: Introducing Cross-media Content Information into Multi-domain Recommendation.. In *IJCAI*. 3484–3490.

[27] Lucas Mansilla, Rodrigo Echeveste, Diego H Milone, and Enzo Ferrante. 2021. Domain Generalization via Gradient Surgery. In *ICCV*. 6630–6638.

[28] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *CVPR*. 3994–4003.

[29] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* (2018).

[30] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. 2018. TADAM: task dependent adaptive metric for improved few-shot learning. In *NIPS*. 719–729.

[31] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[32] Weichen Shen. 2017. DeepCTR: Easy-to-use,Modular and Extendible package of deep-learning based CTR models. https://github.com/shenweichen/deepctr.

[33] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One Model to Serve All: Star Topology Adaptive Recommender for Multi-Domain CTR Prediction. In *CIKM*. 4104–4113.

[34] Anthony Sicilia, Xingchen Zhao, Davneet S Minhas, Erin E O'Connor, Howard J Aizenstein, William E Klunk, Dana L Tudorascu, and Seong Jae Hwang. 2021. Multi-Domain Learning By Meta-Learning: Taking Optimal Steps In Multi-Domain Loss Landscapes By Inner-Loop Learning. In *ISBI*. IEEE, 650–654.

[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*. 1161–1170.

[36] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer learning for few-shot learning. In *CVPR*. 403–412.

[37] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *RecSys*. 269–278.

[38] Tiffany Y Tang, Pinata Winoto, and Robert Ziqin Ye. 2011. Analysis of a multi-domain recommender system. In *The 3rd International Conference on Data Mining and Intelligent Information Technology Applications*. IEEE, 280–285.

[39] Hung-Yu Tseng, Yi-Wen Chen, Yi-Hsuan Tsai, Sifei Liu, Yen-Yu Lin, and Ming-Hsuan Yang. 2020. Regularizing meta-learning via gradient dropout. In *ACCV*.

[40] Ruobing Xie, Zhijie Qiu, Jun Rao, Yi Liu, Bo Zhang, and Leyu Lin. 2020. Internal and Contextual Attention Network for Cold-start Multi-channel Matching in Recommendation.. In *IJCAI*. 2732–2738.

[41] Yongxin Yang and Timothy Hospedales. 2015. A Unified Perspective on Multi-Domain and Multi-Task Learning. In *ICLR*.

[42] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. 2019. Hierarchically structured meta-learning. In *ICML*. PMLR, 7045–7054.

[43] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. *NIPS* 33 (2020).

[44] Yu Zhang, Bin Cao, and Dit-Yan Yeung. 2010. Multi-domain collaborative filtering. In *UAI*. 725–732.

[45] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. 2021. Cross-domain recommendation: challenges, progress, and prospects. *arXiv preprint arXiv:2103.01696* (2021).

A APPENDIX

In the appendix, we will first give more details of datasets. Following, we will introduce the compared methods, implementation details, and experiment settings used in sections 5.2 and 5.3. Last, we will present the detailed experimental results of Parameters Analysis to help better understand the performance of our methods.

A.1 Datasets

The public datasets used in our experiments can be found at: Amazon[1] and Taobao[2]. Amazon dataset originally contains 24 categories of products, such as "Musical Instruments" and "Video Games". Thus, we respectively select 6 and 13 categories of products as domains to construct the Amazon-6 and Amazon-13. As the Amazon dataset does not provide negative samples, we randomly select products not rated by a specific user as negative samples. The number of negative samples is selected by the given CTR Ratio for each domain. The CTR ratio is defined as:

$$\text{CTR Ratio} = \frac{\#\text{ positive samples}}{\#\text{ negative samples}}. \tag{25}$$

We randomly assign CTR Ratio for each domain ranging from $[0.2, 0.5]$ to simulate the domain distinction. For Amazon-6 and Amazon-13, the category, sample number, sample percentage, and CTR Ratio of each domain are listed in Table 6 and Table 7.

Taobao dataset contains users' click logs from different themes of purchase, e.g.,"what to take when traveling", "how to dress up yourself for a party", and "things to prepare when a baby is coming". Thus, we treat each theme as a domain and respectively select 10/20/30 themes to construct Taobao-10/20/30. Similarly, the CTR Ratio is randomly selected from $[0.2, 0.5]$. The statistics of the selected 30 domains are listed in Table 8. The first 10 and 20 domains are used for Taobao-10 and Taobao-20, respectively.

The details of Taobao online datasets are shown in Table 9.

A.2 Experiment settings of Performance Comparison

In this section, we will detail the baselines and experiment settings used in performance comparison.

A.2.1 Baselines. We select several state-of-the-art methods in CTR prediction as baselines, which can be roughly grouped into two categories: Single-Domain method and Multi-Task (Domain) method.

Single-Domain Method. This group of methods is originally proposed for single domain recommendation. Thus, they do not consider the domain distinction information.

- MLP: Multi-layer perceptron (MLP) is the simplest neural network model composed of multiple fully connected layers.
- WDL [6]: WDL is a widely used recommendation model in the industry. It contains a wide liner network and a deep neural network to simultaneously capture the cross-product features as well as the nonlinear features.
- NeurFM [14]: NeurFM proposes a bi-interaction pooling layer to learn feature interaction between embedding vectors. Then, it integrates the results with the logit output of a MLP layer.
- AutoInt [35]: AutoInt proposes the attention-based interaction layer to automatically identify meaningful high-order features.
- DeepFM [11] integrate the factorization machine with the deep neural network to improve the recommendation results.

Multi-Task (Domain) Method. This group of methods contains the multi-task and multi-domain methods. As discussed in the Introduction, the multi-task methods can be directly applied to MDR by treating each domain as a separate task.

- Shared-Bottom [31]: Shared-Bottom is a multi-task method that consists of shared-bottom networks and several domain-specific tower networks. Each domain has its specific tower network while sharing the same bottom network.
- MMOE [25]: MMOE adopts the Mixture-of-Experts (MoE) structure by sharing the expert modules across all domains, while having a gating network trained for each domain.
- PLE [37]: PLE separates shared components and task-specific components explicitly and adopts a progressive mechanism to extract features gradually.
- Star [33] is the state-of-the-art MDR method. It splits the parameters into shared and specific parts. Meanwhile, it proposes a Partitioned Normalization for distinct domain statistics.

A.2.2 Implementation Details. All the models in single-domain and multi-domain methods, except the Star, are implemented by DeepCTR [32], which is an open-source deep-learning based CTR package. Star is implemented by us according to the original paper.

To make a fair comparison, we try to set similar parameters for all the models. For all single-domain methods, the hidden layers are set to $[256, 128, 64]$; for AutoInt, its attention head number is set to 4; for Shared-bottom, its shared network is set to $[512, 256, 128]$ and its tower network is set to $[64]$; for MMOE, its expert networks are set to $[512, 256, 128]$, its tower network and gating network are set to $[64]$, and its expert number is set to 2; for PLE, its shared network is set to $[64]$, its tower network is set to $[256]$, its gating network is set to $[64]$, its shared and specific experts are set to 2 and 10, respectively, and the progressive level is set to 1; for Star, both the shared and specific networks are set as $[256, 128, 64]$. For all models, the dropout rate are set to 0.5, the embedding size is set to 128, and the learning rate is set to 0.001. For our MAMDR, the inner learning rate is set to 0.001, and the outer learning rate is set to 0.1; the sample number of DR is set to $[3, 5, 5, 5, 5]$ for each dataset respectively. We use Adam as the optimizer and Binary Cross Entropy as the recommendation loss function. Experiments are conducted on a RTX2080 GPU with 64G RAM.

A.3 Experiment Settings of Learning Method Comparison

In this section, we detail the learning methods used in learning method comparison, which can be roughly grouped into three categories: traditional learning methods, multi-task leaning method, and meta-learning methods.

Traditional Learning Methods.

- Joint: Joint learning is a conventional learning method that combines data from multiple domains and trains a model to

[1] https://nijianmo.github.io/amazon/index.html
[2] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649

Table 6: Statistics of Amazon-6 dataset.

Domain	Musical Instruments	Office Products	Patio Lawn and Garden	Prime Pantry	Toys and Games	Video Games
# Samples	1204340	3921259	3025218	694758	5382501	2698800
Percentage	7.11%	23.17%	17.87%	4.10%	31.80%	15.94%
CTR Ratio	0.22	0.23	0.32	0.23	0.47	0.21

Table 7: Statistics of Amazon-13 dataset.

Domain	Arts Crafts and Sewing	Digital Music	Gift Cards	Industrial and Scientific	Luxury Beauty	Magazine Subscriptions	Musical Instruments	Office Products	Patio Lawn and Garden	Prime Pantry	Software	Toys and Games	Video Games
# Samples	2419005	770132	11951	380386	87360	13103	814928	3178096	2317603	655970	11022	7541261	459646
Percentage	12.96%	4.13%	0.06%	2.04%	0.47%	0.07%	4.37%	17.03%	12.42%	3.52%	0.06%	40.41%	2.46%
CTR Ratio	0.22	0.23	0.32	0.23	0.47	0.21	0.36	0.30	0.46	0.25	0.30	0.30	0.27

Table 8: Statistics of Taobao-10/20/30 dataset.

Domain	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30
# Sample	13255	7007	20130	62460	11563	7190	4185	24052	5582	17860	29302	6472	8873	125586	15560	5458	14095	53910	12102	2936	4710	29256	41609	7354	68119	5308	24918	38919	24297	34253
Percentage	1.82%	0.96%	2.77%	8.60%	1.59%	0.99%	0.58%	3.31%	0.77%	2.46%	4.03%	0.89%	1.22%	17.29%	2.14%	0.75%	1.94%	7.42%	1.67%	0.40%	0.65%	4.03%	5.73%	1.01%	9.38%	0.73%	3.43%	5.36%	3.35%	4.72%
CTR Ratio	0.22	0.23	0.32	0.23	0.47	0.21	0.36	0.30	0.46	0.25	0.30	0.30	0.27	0.20	0.33	0.23	0.38	0.22	0.29	0.33	0.47	0.23	0.24	0.44	0.21	0.47	0.37	0.28	0.45	0.43

Table 9: Statistics of Taobao online dataset.

Dataset	# Sample	# User	# Item	# Domain	# Train	# Val	# Test	Sample / Domain	CTR Ratio
Taobao-online	489,852,853	84,307,785	16,385,662	69,102	420,097,203	23,340,352	46,415,298	7,088	0.0474

server all domains. In this way, it enables the model to learn shared features and improve performance.

- Joint + Finetune: On the top of the model learned by Joint learning, we finetune the model on each domain to obtain several domain-specific models. Each specific model can capture the domain distinction information.

Multi-Task Learning Methods.

- Weighted Loss [16]: Weighted Loss is a multi-task learning method that assigns weight to the loss of each task. Meanwhile, the weight is automatically optimized during the training to balance between different losses.
- PCGrad [43]: PCGrad is a powerful multi-task learning method. By projecting the gradients of each task into the normal plane of others, it successfully avoids the conflicting problem.

Meta-learning Methods.

- MAML [9]: MAML aims to train parameters through various tasks and acquires parameters that can quickly adapt to new tasks. We treat each domain as the task and split the training data into the support and query sets used for MAML.
- Reptile [29]: Reptile is a first-order meta-learning method, which trains parameters by rapidly sampling the tasks. It maximizes the inner-gradients within the task and leads parameters quickly to generalize to new tasks.
- MLDG [20]: MLDG proposes a novel meta-learning method for domain generalization. Its meta-optimization function improves the performance on both train and test domains.

We implement all the learning methods in Tensorflow and conduct experiments on the Taobao-10 dataset. Implementations are available at this link [3].

[3]https://github.com/RManLuo/MAMDR

Table 10: Detailed results on Taobao-30 under different sample numbers k.

Sample Num.	1	5	10	15	20	25
AUC	0.7713	**0.7750**	0.7715	0.7694	0.7697	0.7706

Table 11: Detailed results under different inner-loop learning rate α and outer-loop learning rate β.

β \ α	1e-1	1e-2	1e-3	1e-4	1e-5
1	0.5000	0.6691	0.7148	0.7257	0.7114
0.5	0.5000	0.6946	**0.7271**	**0.7291**	0.6897
0.3	0.5000	0.7038	**0.7267**	0.7237	0.6875
0.1	0.5887	0.6906	**0.7246**	0.7050	0.6417
0.05	0.5881	0.6918	0.7090	0.6902	0.6430

A.4 Detailed Experiment Results

In this section, we will illustrate the details of experiment results in Parameters Analysis. The detailed results of Figure 6 are shown in Table 10. The detailed results of Figure 7 are shown in Table 11.