MultiFS: Automated Multi-Scenario Feature Selection in Deep Recommender Systems

Dugang Liu*
Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University
Shenzhen, China
dugang.ldg@gmail.com

Chaohua Yang*
Shenzhen University
Shenzhen, China
ych981203@gmail.com

Xing Tang*
Tencent
Shenzhen, China
xing.tang@hotmail.com

Yejing Wang
City University of Hong Kong
Hongkong, China
yejing.wang@my.cityu.edu.hk

Fuyuan Lyu
McGill University
Montreal, Canada
fuyuan.lyu@mail.mcgill.ca

Weihong Luo
Tencent
Shenzhen, China
lobbyluo@tencent.com

Xiuqiang He
Tencent
Shenzhen, China
xiuqianghe@tencent.com

Zhong Ming†
Shenzhen Technology University
Shenzhen, China
mingz@szu.edu.cn

Xiangyu Zhao†
City University of Hong Kong
Hongkong, China
xianzhao@cityu.edu.hk

ABSTRACT

Multi-scenario recommender systems (MSRSs) have been increasingly used in real-world industrial platforms for their excellent advantages in mitigating data sparsity and reducing maintenance costs. However, conventional MSRSs usually use all relevant features indiscriminately and ignore that different kinds of features have varying importance under different scenarios, which may cause confusion and performance degradation. In addition, existing feature selection methods for deep recommender systems may lack the exploration of scenario relations. In this paper, we propose a novel automated multi-scenario feature selection (MultiFS) framework to bridge this gap, which is able to consider scenario relations and utilize a hierarchical gating mechanism to select features for each scenario. Specifically, MultiFS first efficiently obtains feature importance across all the scenarios through a scenario-shared gate. Then, some scenario-specific gate aims to identify feature importance to individual scenarios from a subset of the former with lower importance. Subsequently, MultiFS imposes constraints on the two gates to make the learning mechanism more feasible and combines the two to select exclusive features for different scenarios. We evaluate MultiFS and demonstrate its ability to enhance the multi-scenario model performance through experiments over two public multi-scenario benchmarks.

*Equal contribution
†Co-Corresponding authors

WSDM '24, March 4–8, 2024, Merida, Mexico

ACM ISBN 979-8-4007-0371-3/24/03…$15.00
https://doi.org/10.1145/3616855.3635859

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Feature selection, Multi-scenario learning, Deep recommender system, Hierarchical gate

ACM Reference Format:
Dugang Liu, Chaohua Yang, Xing Tang, Yejing Wang, Fuyuan Lyu, Weihong Luo, Xiuqiang He, Zhong Ming, and Xiangyu Zhao. 2024. MultiFS: Automated Multi-Scenario Feature Selection in Deep Recommender Systems. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24), March 4–8, 2024, Merida, Mexico.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3616855.3635859

1 INTRODUCTION

Deep recommender systems (DRS) play a key role in solving today's online information overloading problem [5, 9, 14, 22, 37]. With the development of industrial recommendation platforms, deep recommender systems increasingly need to serve a variety of scenarios [3, 11, 24, 26, 34, 36, 38]. For example, in an online financial recommendation platform, a user may be exposed to a certain fund in various scenarios, such as the homepage, balanced investment portfolio page, and aggressive investment portfolio page. Therefore, instead of training a model for each scene, training a unified model to serve multiple scenes simultaneously is receiving increasing research attention. Deploying a multi-scenario recommender system can capture shared information between scenarios, alleviate data sparsity, and reduce maintenance costs.

The key to multi-scenario learning is to capture the commonalities and discriminate the differences between the scenarios [21]. To this end, several *shared-specific* models have been proposed [2, 19, 25]. Typically, these models consist of both scenario-shared

and scenario-specific architecture. The scenario-shared architecture takes all features as input and outputs a shared representation, which aims to obtain shared information across the scenarios. Then, the scenario-specific architecture makes predictions based on shared representation in the corresponding scenario. As shown in Fig. 1, we adopt Shared-Bottom [1] to illustrate the overall framework of multi-scenario learning. The scenario-shared architecture typically contains an embedding table and scenario-shared layers shared between the scenarios. The scenario-specific architecture aims to learn specific representations based on the shared representation produced by the scenario-shared layer. Finally, the scenario-specific hidden representation is fed to the output layer to generate the predicted labels. Despite complex architecture, all these works ignore the different importance of input features to scenarios. For example, features related to financial marketing, e.g., China's *Shanghai Composite Index*, may be important for recommendation predictions for aggressive investment portfolio scenarios but useless for balanced investment portfolio scenarios. Therefore, selecting suitable feature subsets for different scenarios in MSRSs is necessary.

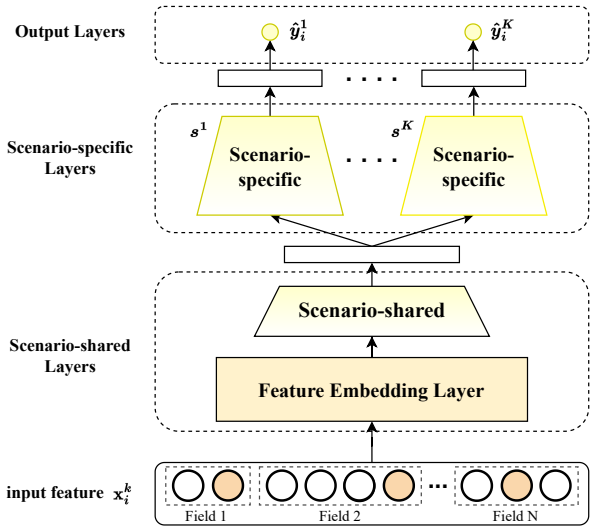

Figure 1: Overview of the Shared-Bottom framework in multi-scenario modeling.

To improve the performance and efficiency of the model, several feature selection methods have been proposed for single-scenario recommendation [12, 17, 18, 32]. However, directly extending these works to MSRSs will incur a high computational cost. For example, for AutoField [32] where the selection granularity is restricted to m feature field, K scenarios will require 2^{Km} search space, and for OptFS [17] with a granularity of n feature values, the search space will be 2^{Kn}, where $n \gg m$ is intractable. In addition, some recent works have also made many efforts in feature selection in multi-task recommendations [4, 7]. However, on the one hand, multi-task recommendation focuses on solving different problems [24, 35] compared to MSRSs. Specifically, MSRSs focus on the same task with the same label space, but the multi-task recommendation is oriented to different tasks with different label spaces [30]. On the other hand, all these methods focus on the feature field level and select features independently for each task, which is too coarse and expensive. Therefore, this motivates us to propose an effective and efficient feature selection framework for MSRSs to bridge the gap.

In this paper, we propose an automated Multi-scenario Feature Selection (MultiFS) framework to address the feature selection problem in multi-scenario settings. Aiming at the intractable oversized search space problem faced in fine-grained feature value selection in MSRSs, we address this problem by using scenario-shared and scenario-specific gates to determine features hierarchically. By decomposing the search space into two subsets, we greatly reduce the search space size, which can be easily handled in our framework. Specifically, MultiFS first efficiently obtains the feature importance across all the scenarios by a scenario-shared gate and its corresponding constraints. Then, by filtering out subsets with lower importance values from the above operations, some scenario-specific gates aim to identify their importance to individual scenarios from them. Finally, MultiFS combines the results of the two gates to select exclusive features for different scenarios.

2 RELATED WORK

In this section, we briefly review some relevant works on two research topics, including multi-scenario recommendations and feature Selection in deep recommender systems.

Multi-scenario Recommendations. The mainstream methods for tackling the multi-scenario modeling employ shared-specific architectures [11, 23, 24, 31, 34, 36, 38]. For example, HMoE [11] utilizes a multi-gate mixture-of-experts [19] to model shared representation and distinctions among multiple scenarios implicitly, and STAR [24] proposes the star topology, which consists of a centered network shared by all scenarios and the scenario-specific network tailored for each scenario. Based on these basic architectures, several works improve the performance by designing novel modules. SASS [38] design multi-layer scenario adaptive transfer module with scenario-adaptive gate units to fuse transfer information from the whole scenario in a fine-grained way. M2M [36] pays attention to advertiser modeling with a meta-attention module to capture diverse inter-scenario correlations. SAR-Net [23] learns users' cross-scenario interests via two specific attention modules, leveraging the scenario features and item features, respectively. However, these works only focus on designing architectures while neglecting input features. Our MultiFS first selects informative features in the multi-scenario modeling, which is an effective and efficient complementary for these works.

Feature Selection in Deep Recommender Systems. Feature selection is a critical component in prediction task [8, 13, 27]. Several works in the deep recommendation system are dedicated to this problem [10, 12, 17, 18, 32]. Inspired by the advancement of AutoML [15, 39], AutoField pioneers feature field selection by dynamically moderating the probability pairs deciding whether to select or drop feature fields in a deep recommender system. LPFS [10] adopt smoothed-$l0$ optimization [20] to discover informative fields. AdaFS [12] proposes a novel controller network to decide feature fields for each sample, which fits the dynamic recommendation. OptFS [17] is the first work to select feature value and feature interaction with a continuation learning regime [16]. Furthermore,

some recent works aim at feature selection for multi-task recommendation [4, 7, 29]. Unlike them, our MultiFS targets the feature value level selection, which extends to multi-scenario modeling and selects from a bigger search space than existing methods.

3 PROBLEM FORMULATION

We first formulate the multi-scenario learning in this section. For user and item pairs in a single scenario, we denote $\mathcal{X}$ and $\mathcal{Y}$ as feature space and label space, respectively. The $\mathcal{X}$ consists of user features, item features, and context features, and the $\mathcal{Y}$ denote user behaviors, which are usually represented as binary labels, such as {*click*,*non-click*}. When we have K scenarios $\mathcal{S} = \{s^1, \cdots, s^K\}$, the multi-scenario dataset can be denoted as $\mathcal{D} = \{(\mathbf{x}_i^k, y_i^k, s^k)\}_{k=1}^K$. Here s^k is the k-th scenario, $\mathbf{x}_i^k \in \mathcal{X}$ is the feature representation of the i-th instance in k-th scenario, and $y_i^k \in \{0, 1\}$ is corresponding label. Multi-scenario learning aims to train a model based on the above dataset, where this model consists of the shared and specific components [1],

$$\hat{y}_i^k = f(\mathbf{x}_i^k \mid \theta, \{\theta_{s^k}\}), \tag{1}$$

where $f(\cdot)$ is the mapping function corresponding to the model from features to labels, and θ and $\{\theta_{s^k}\}$ are the scenario-shared and scenario-specific parameters, respectively. The cross-entropy function is usually used to optimize the model,

$$\mathcal{L} = \sum_{k=1}^{K} \sum_{i=1}^{|s^k|} l(y_i^k, \hat{y}_i^k), \tag{2}$$

where $|s^k|$ denotes the number of instance in scenario s^k and $l(\cdot)$ is the cross-entropy loss.

We further formulate the feature selection problem for MSRSs based on the above formulation. Usually, there are *m feature fields* in a feature vector $\mathbf{x}_i^k$, which is denoted as,

$$\mathbf{x}_i^k = [x_{i1}^k, \cdots, x_{im}^k], \tag{3}$$

where one j-th field x_{ij}^k includes a set of *feature values*. We use x_{ij}^k as the feature value in the j-th field hereafter to simplify our notation. To better represent and utilize features, we usually employ an embedding table to convert feature values into low-dimensional and dense real-value vectors,

$$\mathbf{e}_{ij}^k = \mathbf{E} \times x_{ij}^k, \quad \mathbf{E} \in \mathbb{R}^{n \times d}, \tag{4}$$

where n is the number of feature values, d is a hyperparemeter for embedding dimension, and $\mathbf{E}$ denotes the embedding table shared across scenarios, which means $\mathbf{E} \in \theta$. Hence, the feature selection problem for MSRSs can be defined as applying gate mask operation on the embedding table for each scenario and producing the corresponding masked embedding table $\hat{\mathbf{E}}$,

$$\hat{\mathbf{E}}^k = \mathbf{E} \odot \mathbf{G}^k. \tag{5}$$

Here $\mathbf{G}^k \in \{0, 1\}^n$, and $\odot$ denotes element-wise multiplication. Note that we convert the feature selection from the feature values granularity to the problem of determining the gate vectors, and the search space is 2^{Kn}. As discussed above, MSRSs should consider scenario-shared and scenario-specific information [4]. To do this, we can define a specific $\mathbf{G}^k$ as the combination of a scenario-shared gate vector and the corresponding scenario-specific gate vector,

$$\mathbf{G}^k = g(\mathbf{g}^k, \mathbf{g}^{sh}), \tag{6}$$

where $\mathbf{g}^k$ and $\mathbf{g}^{sh}$ are masks for scenario k-specific and scenario-shared features, respectively. The $g(\cdot, \cdot)$ denotes the specific form of how to combine both two gates. Finally, with all these formulations, we formulate our problem as follows:

$$\min_{\Theta, \{\mathbf{G}^k\}} \mathcal{L}(\mathcal{D}). \tag{7}$$

where $\Theta = \{\theta, \{\theta_{s^k}\}_{k=1}^K\}$ denotes the network parameters.

4 THE PROPOSED FRAMEWORK

In this section, we will first illustrate the overall framework of our MultiFS. Then, we detail each part of MultiFS, including feature mask decomposition and hierarchical gating mechanism. Finally, we introduce how to combine our MultiFS with a multi-scenario recommendation model for effective feature selection.

4.1 Framework Overview

The overall framework of our MultiFS is illustrated in Fig. 2, which follows the main paradigm of multi-scenario learning in Fig. 1. Firstly, the feature mask is decomposed into two types as indicated in Eq.(6). The shared mask (marked as blue) focuses on selecting the shared features that are useful across all scenarios, and the specific masks (marked as other colors) serve for a particular scenario, which filters the useless features for a specific scenario. Therefore, the set of specific masks can be determined based on the shared mask. Note that one sample will only belong to one scenario and will be used to train the scenario-shared network and one of the corresponding scenario-specific networks. This means that the combination of a shared mask and one of the specific masks will produce the embedding mask for the embedding table, which further transforms the features of this sample. In summary, we hierarchically search masks and generate the corresponding embedding tables, which consist of a shared part and a set of specific parts.

4.2 Feature Mask Decomposition

To address the intractable large search space problem faced by the selection of feature value levels in MSRSs, the first step in our MultiFS is to decompose the feature mask, which determines the $g(\cdot, \cdot)$ in Eq.(6). Formally, we aim to determine a set of scenario-specific feature gates as shown in Eq.(6), i.e., $\boldsymbol{G} = \{\mathbf{G}^1, ..., \mathbf{G}^K\}$. Note that we focus on the feature value level selection in this paper, i.e., instead of making the field-level selection, we formulate scenario feature selection as assigning a binary gate $\mathbf{g}_{ij}^k \in \{0, 1\}$ for each feature embedding $\boldsymbol{e}_{ij}^k$. Therefore, each $\mathbf{G}^k \in \{0, 1\}^n$ refers to gating vectors indicating whether one feature is kept or dropped in scenario s^k. After selection, the feature embeddings can be formulated as follows,

$$\tilde{\boldsymbol{e}}_{ij}^k = \mathbf{g}_{ij}^k \odot \boldsymbol{e}_{ij}^k = \mathbf{g}_{ij}^k \odot (\boldsymbol{E} \times \boldsymbol{x}_{ij}^k), \tag{8}$$

when $\mathbf{g}_{ij}^k = 1$, feature $\boldsymbol{x}_{ij}^k$ is selected in the scenario s^k and vice versa.

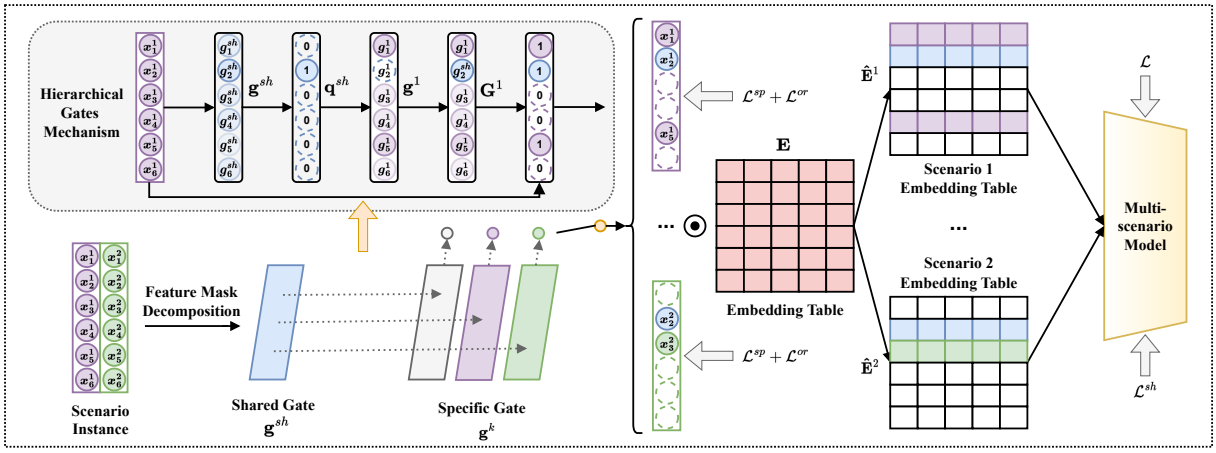

Figure 2: The architecture of our automated multi-scenario feature selection (MultiFS) framework.

However, there are usually many overlapping features between the scenarios for MSRSs, so optimizing each $\mathbf{G}^k$ independently cannot effectively utilize the shared information across scenarios. To effectively leverage the shared information, we introduce a scenario-shared feature gate $\mathbf{g}^{sh} \in \{0,1\}^n$, which is aimed to identify the useful shared features. As indicated in Eq.(6), we can decompose each $\mathbf{G}^k$ into scenario-shared feature selection and scenario-specific feature selection. We formulate the g as follows,

$$\mathbf{G}^k = g(\mathbf{g}^k, \mathbf{g}^{sh}) = \mathbf{g}^{sh} + (1 - \mathbf{g}^{sh}) \odot \mathbf{g}^k, \tag{9}$$

where $\mathbf{g}^k \in \{0,1\}^n$ represents the gate mask selecting informative features for scenario k. Note that $\mathbf{g}^k$ only needs to be searched in the remaining features after the scenario-shared gate selects n_{sh} features. Thus, the original search space for the set of G^k is $O(2^{Kn})$, and the search space is $O(2^n + 2^{K(n-n_{sh})})$ after decomposition. Note that $n - n_{sh} < n$ and n_{sh} will usually not be a small value in practice, thus reducing the space greatly.

4.3 Hierarchical Gating Mechanism

To obtain a universal and efficient $\mathbf{G}^k$, we have to overcome two challenges: (i) the binary gate vector $\mathbf{g}^{sh}$ and $\mathbf{g}^k$ are hard to compute gradient; (ii) optimizing gate mask set $\mathbf{G}^k$ and network parameters Θ together will harm the model's performance. To address these challenges, we introduce a hierarchical gating mechanism with the learning-by-continuation training scheme [17]. A schematic of this process is shown in the upper left of Fig. 2.

To efficiently optimize the gate mask set $\mathbf{G}^k$ with feature value level granularity, we introduce a continual scenario-specific gate vector set $\{\mathbf{g}_c^1, ..., \mathbf{g}_c^K\}$ and a continual scenario-shared gate vector $\mathbf{g}_c^{sh}$, where each $\mathbf{g}_c^k \in \mathbb{R}^n$ and $\mathbf{g}_c^{sh} \in \mathbb{R}^n$. Specifically, the continual gate $\mathbf{g}^{sh}$ and $\mathbf{g}^k$ are defined as,

$$\mathbf{g}^{sh} = \frac{\sigma\left(\mathbf{g}_c^{sh} \times \tau\right)}{\sigma\left(\mathbf{g}_c^{sh(0)}\right)}, \tau = \gamma^{t/T}, \tag{10}$$

$$\mathbf{g}^k = \frac{\sigma\left(\mathbf{g}_c^k \times \tau\right)}{\sigma\left(\mathbf{g}_c^{k(0)}\right)}, \quad \tau = \gamma^{t/T}, \tag{11}$$

where $\mathbf{g}_c^{sh(0)}$ and $\mathbf{g}_c^{k(0)}$ are initial value of the continual gate, respectively, $\sigma(\cdot)$ is the sigmoid function, t is the current training epoch number, T is the total training epoch and γ is the final value of τ after training for T epochs.

By replacing the mask with the continual gate vectors during training, we could optimize both gate vectors and network parameters in a differentiable manner. However, the continual gates will make the Eq.(9) hard to get the remaining features hierarchically. For this case, we use a straight-through estimator operation $S(\cdot)$ [6] on the scenario-shared gate during training, i.e., transforming it into a binary proxy mask during forward prediction while maintaining the backward differentiability property.

$$\mathbf{q}^{sh} = S(\mathbf{relu}(\mathbf{g}^{sh} - \epsilon)), \tag{12}$$

where ϵ is a learnable threshold. Thus, we further reformulate Eq.(9) as follows,

$$\mathbf{G}^k = \mathbf{q}^{sh} \odot \mathbf{g}^{sh} + (1 - \mathbf{q}^{sh}) \odot \mathbf{g}^k. \tag{13}$$

After training T epochs, the final gating vector $\mathbf{g}^{sh}$ and $\mathbf{g}^k$ are calculated through a unit-step function as follows:

$$\mathbf{g}^{sh} = \begin{cases} 0, & \mathbf{g}_c^{sh} \le 0 \\ 1, & \text{otherwise} \end{cases}, \tag{14}$$

$$\mathbf{g}^k = \begin{cases} 0, & \mathbf{g}_c^k \le 0 \\ 1, & \text{otherwise} \end{cases}. \tag{15}$$

4.4 Optimization

Next, we will introduce the required optimization constraints to ensure that a multi-specific recommendation model obtains the desired feature gates effectively. First, to encourage the sparsity of feature gate vectors, we also introduce the l_1 regularization.

$$\mathcal{L}^{sp} = (\left\|\mathbf{g}^{sh}\right\|_1 + \sum_{k=1}^{K} \left\|\mathbf{g}^k\right\|_1), \tag{16}$$

where $\|\cdot\|_1$ indicates the l_1 norm. Note that the l_0 norm can be approximated by l_1 norm given the fact that $\|\mathbf{g}\|_0 = \|\mathbf{g}\|_1$ for binary $\mathbf{g}$.

Second, an ideal MSRS should make the scenario-specific representations decoupled from each other. Therefore, we design an

Table 1: Statistics of the processed datasets.

Dataset		AliExpress-1			AliExpress-2			AliCCP			
		NL	FR	ALL	ES	US	ALL	SA	SB	SC	ALL
Train	#impress	12,402,036	18,924,921	31,326,957	22,168,599	19,174,829	41,343,428	14,296,532	286,913	23,487,225	38,070,670
	#click	266,815	380,148	646,963	589,547	314,701	904,248	571,542	12,600	895,607	1,479,749
Validation	#impress	2,657,580	4,055,340	6,712,920	4,750,414	4,108,892	8,859,306	1,588,839	31,960	2,608,436	4,229,235
	#click	57,254	81,662	138,916	126,668	67,704	194,372	63,241	1,323	99,943	164,507
Test	#impress	2,657,579	4,055,340	6,712,919	4,750,414	4,108,829	8,859,306	16,351,580	321,024	26,344,010	43,016,614
	#click	57,009	80,943	137,952	125,840	67,203	193,043	656,280	14,099	1,003,068	1,673,447

orthogonal penalty for $\{\mathbf{g}^k\}$ as follows, and with this term, our MultiFS can select informative features that are specific to the scenario.

$$\mathcal{L}^{or} = \sum_{p=1}^{K} \sum_{r=p+1}^{K} \left\| \mathbf{g}^p \odot \mathbf{g}^r \right\|_1 . \tag{17}$$

Then, the imbalanced data distribution in the multi-scenario dataset would affect the performance [24]. Similarly, this also will be harmful to multi-scenario feature selection. For example, suppose a scenario has too few samples, it will lead to difficulties in effectively optimizing the feature mask of this scenario, which will affect the performance of the entire model. To address this issue and ensure the generality of our approach, we introduce a single prediction on $\mathbf{g}^{sh}$ as an embedding mask, and the corresponding loss can be formulated as follows:

$$\hat{\mathbf{y}} = f(\mathbf{g}^{sh} \odot \mathbf{E} \times \mathbf{x}; \Theta), \tag{18}$$

$$\mathcal{L}^{sh} = \mathcal{L}(f(\mathbf{g}^{sh} \odot \mathbf{E} \times \mathbf{x}; \Theta), \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_k^{sh}(\hat{y}, y), \tag{19}$$

where $\mathcal{L}_k^{sh}$ indicates the prediction error loss on scenario s_k after selecting feature embeddings only through $\mathbf{g}^{sh}$. Notably, we get all the scenario predictions as a vector $\hat{\mathbf{y}}$ in Eq.(18), which achieves a transfer learning mechanism to address the problem of difficult optimization caused by an extreme imbalance in scenario samples. Finally, combining Eq.(7), Eq.(16), Eq.(17) and Eq.(19), we get the final training objective becomes,

$$\min_{\{\mathbf{G}^k\},\Theta} \mathcal{L}_{MultiFS} = \mathcal{L} + \lambda_1 \mathcal{L}^{sh} + \lambda_2 \mathcal{L}^{sp} + \lambda_3 \mathcal{L}^{or}, \tag{20}$$

where λ_1, λ_2, and λ_3 are the control weights of different optimization terms.

Finally, in the searching stage, all possible features are fed into the model to explore the optimal scenario-aware feature gate vector set $\mathbf{G}^k$. Thus, the useless features might hurt the model's performance. To address this problem, we must retrain the model after obtaining the optimal $\mathbf{G}^k$. After determining each gating vector $\mathbf{g}^k$ and $\mathbf{g}^{sh}$, we retrain the model parameters Θ as the corresponding values at T_c epoch, which is carefully tuned in our setting. The final parameters Θ are trained as follows:

$$\min_{\Theta} \mathcal{L} + \lambda_1 \mathcal{L}^{sh}. \tag{21}$$

5 EXPERIMENTS

In this section, we conduct experiments intending to answer the following four key questions. Note that the source codes are available at https://github.com/dgliu/WSDM24_MultiFS. Due to space limitations, more results can also be found in the repository.

- RQ1: How does our MultiFS perform compared to the baselines?
- RQ2: What is the role of some key components in our MultiFS?
- RQ3: What is the transferability of the features selected by our MultiFS?
- RQ4: What are the characteristics of feature subsets get by our MultiFS?

5.1 Experiment Setup

5.1.1 Datasets. To evaluate the effectiveness of our MultiFS, we conduct experiments on two public real-world datasets with multi-scenario information, including AliExpress[1] and AliCCP[2]. AliExpress consists of the user click and purchase records collected from real-world traffic logs of the search system in AliExpress, including Russia, Spain, French, the Netherlands, and America, and this can be regarded as 5 scenarios in the experiments. AliCCP is collected from real-world traffic logs of the Mobile Taobao recommender system, including user interaction records in three associated business scenarios. We denote them as SA, SB, and SC.

5.1.2 Dataset Preprocessing. For AliExpress, we combine two of these countries to form multiple subsets containing two scenarios, including AliExpress-1 consisting of the Netherlands (NL) and French (FR) and AliExpress-2 consisting of America (US) and Spain (ES). Since the original dataset includes both labels of click and purchase, we treat both purchases as clicks to keep only one label, i.e., $y = 1$ means a clicked interaction, and $y = 0$ means a non-clicked interaction. Following the previous work [18], we set a threshold of 2 to filter the low-frequency features and divide the training, validation, and test sets from all the original data of each scenario according to the ratio of 70%, 15%, and 15%. For AliCCP, following the previous work [33], we use 10 as a threshold to filter the low-frequency features. We then randomly sample 10% from the original training set for each scenario as a validation set and use the original test set with each scenario for testing. The statistics of the processed datasets are shown in Table 1.

5.1.3 Metrics. Following the setup of previous works [12, 32], we use the common evaluation metrics for deep recommender systems, i.e., the area under the ROC curve (AUC) and cross-entropy (log loss). Note that an improvement in AUC of more than 0.1% is considered significant [9]. Furthermore, to evaluate the feature selection capability, we will use the feature retention rate as a reference, which is the ratio between the number of remaining features and original features.

[1] https://tianchi.aliyun.com/dataset/74690
[2] https://tianchi.aliyun.com/dataset/408

5.1.4 Baselines Methods and Backbone Methods. We choose the representative methods from the research on feature selection in deep recommender systems summarized in Section 2. Specifically, we compare our MultiFS with the following single-scenario feature selection baselines, including AutoField [32], LPFS [10], OptEmbed [18] and OptFS [17]. We also compare our MultiFS with the multi-task feature selection baseline CFS [4]. To evaluate the generalization ability of all the methods, we integrate them with the mainstream skeleton models DNN, DeepFM [9], and DCN [28], respectively. In addition, to show the importance of feature selection for multi-scenarios, we employ two base models that preserve all the features, one modeled individually based on the data for each scenario (SD-Backbone) and the other modeled based on aggregated data for all the scenarios (Backbone). We also use representative methods that improve architectures for multi-scenario recommendation as the strong baselines, i.e., STAR [24] and HMoE [11].

5.1.5 Implementation Details. Next, we provide the implementation details of our MultiFS and the baselines. For general hyperparameters, we set the embedding dimension, batch size, and l_2 regularization weights as 16, 4096, and 3e-6, respectively. For the MLP layer in the backbone models, we use a three-layer fully connected network of size $[1024, 512, 256]$. We select the optimal learning ratio from {1e-3, 3e-4, 1e-4, 3e-5, 1e-5}. We use Adam optimizer, Batch Normalization, and Xavier initialization in the experiments. For the hyperparameters of MultiFS, we select the optimal regularization penalty λ_1, λ_3 and final value γ from {2e-8, 1e-8, 5e-9, 2e-9, 1e-9}, {0.1, 0.5, 1, 3, 5, 7, 9} and {50, 100, 200}, respectively, and set λ_2 to 1e-9. During re-training, we fix the optimal learning ratio and l_2 regularization weights and select the rewinding epoch T_c from $\{1,2,\cdots,T-1\}$. For other baseline methods, we use the open source implementations for AutoField [32], LPFS [10][3] and OptEmbed [18][4] and OptFS [17][5]. Due to the lack of available implementations for the CFS [4], STAR [24], and HMoE [11], we re-implement them based on the details provided by the original paper. Notice that all the baselines will also search for the best hyperparameter combination within the same hyperparameter range.

5.2 RQ1: Overall Performance

In this subsection, we conduct two studies comparing the baselines for feature selection and the baselines for the multi-scenario recommendation, respectively.

Feature Selection. In the upper part of Table 2, we report the overall performance of our MultiFS and other feature selection baseline methods on three different backbone models using three benchmark datasets. We can have the following observations: 1) A backbone model trained on the aggregated data of all the scenarios (i.e., Backbone) will usually be weaker than a backbone model trained on the individual data of each scenario (i.e., SD-Backbone), which indicates the importance of considering the specific information of each scenario in multi-scenario recommendation; 2) The baseline methods for single-scene scenario selection (e.g., LPFS, OptEmbed, and OptFS) usually outperform the above two basic baselines (i.e., SD-Backbone and Backbone), and this means that feature selection is still an effective step to improve model performance in multi-scenario recommendation. 3) The baseline method for multi-task feature selection (i.e., CFS) has a certain performance improvement on AliCCP but has worse performance on AliExpress-1 and AliExpress-2. This may be because the consensus between the scenarios in AliCCP is stronger than AliExpress-1 and AliExpress-2, and CFS can benefit more from it. 4) Unlike other feature selection baseline methods, our MultiFS can maintain a significant performance gain in most cases. This fully demonstrates that our MultiFS can effectively capture the beneficial shared information among multiple scenarios and identify the beneficial specific information for each scenario.

Multi-scenario Recommendation. In the lower part of Table 2, we report the overall performance of our MultiFS and multi-scenario recommendation baseline methods. We can find that compared to the baseline methods designed on the model architecture, by using our MultiFS on the basic backbone model, we can achieve a better result in a multi-scenario recommendation. This shows the importance of feature selection in multi-scenario recommendation and is expected to provide a new perspective for the research of multi-scenario recommendation.

5.3 RQ2: Ablation Study

We conduct an ablation study in this subsection to analyze the effect of some key optimization steps in our MultiFS. Specifically, we sequentially consider removing the retraining step (denoted 'n.re.'), the shared loss $\mathcal{L}^{sh}$ (denoted 'n.sh.'), the orthogonal loss $\mathcal{L}^{or}$ (denoted 'n.or.'), and the sparse loss $\mathcal{L}^{sp}$ (denoted 'n.sp.') from our MultiFS. We report the corresponding results in Table 3. We can observe that removing either the retraining step or the orthogonal loss $\mathcal{L}^{or}$ usually leads to a significant performance drop, which means that these are two critical steps for our MultiFS. Removing the shared loss $\mathcal{L}^{sh}$ also brings a certain degree of performance penalty, which means that it is advantageous to consider the distribution imbalance problem in multi-scenario feature selection. When considering the removal of the sparse loss $\mathcal{L}^{sp}$, we can see that this may add some small additional gains. However, we must point out that this gain comes from retaining too many shared and specific features. Specifically, in practice, we can observe that under the constraint of lacking a sparse loss, our MultiFS variant tends to preserve more subsets of features, especially on AliExpress-1 and AliExpress-2. More detailed results can be found in the appendix files in our MultiFS open-source repository. Considering that the increase in feature retention ratio is far greater than the gain it brings, a sparse loss is also necessary for our MultiFS to get a better trade-off in feature selection and model performance.

5.4 RQ3: Transferability Analysis

Next, we analyze the transferability of the subset of features selected by our MultiFS. A good transferability means that we can perform feature selection for multi-scenario recommendations with a more compact and lightweight backbone model without adding more overhead. This is particularly attractive for real-world applications. Therefore, we implement our MultiFS through three different backbone models. After obtaining the shared features and specific

[3]https://github.com/fuyuanlyu/AutoFS-in-CTR
[4]https://github.com/fuyuanlyu/OptEmbed
[5]https://github.com/fuyuanlyu/OptFS

Table 2: Results on all datasets, where the best and second best results are marked in bold and underlined, respectively. Note that * indicates a significance level of $p \leq 0.05$ based on a two-sample t-test between our method and the best baseline.

	Method	AliExpress-1				AliExpress-2				AliCCP					
		AUC↑		Logloss↓		AUC↑		Logloss↓		AUC↑			Logloss↓		
DNN	SD-Backbone	.7326	.7307	.0953	.0901	.7289	.7410	.1125	.0765	.6209	.5728	.6180	**.1651**	.1845	.1602
	Backbone	.7317	.7304	.0953	.0901	.7291	.7397	.1125	.0766	.6023	.5834	.5997	.1661	**.1788**	.1600
	CFS	.7251	.7209	.0962	.0910	.7172	.7297	.1134	.0769	.6253	.5948	.6234	.1666	.1789	.1600
	AutoField	.7289	.7280	.0955	.0903	.7296	.7418	.1124	.0763	.6241	.6023	.6216	.1663	.1796	.1603
	LPFS	.7332	.7320	.0952	.0900	.7304	.7430	.1123	.0763	.6257	.6030	.6234	.1663	.1799	.1602
	OptEmbed	.7353	.7343	.0951	.0898	.7319	.7417	.1122	.0763	.6268	.6033	.6242	.1659	.1795	.1600
	OptFS	.7351	.7322	.0952	.0901	.7307	.7414	.1124	.0764	.6270	.6024	.6245	.1655	.1789	**.1596**
	MultiFS	**.7425***	**.7385***	**.0944***	**.0895**	**.7351***	**.7504***	**.1119**	**.0758***	**.6277***	**.6038***	**.6248**	.1654	**.1788**	**.1596**
DeepFM	SD-Backbone	.7335	.7311	.0953	.0900	.7292	.7417	.1125	.0764	.6212	.5731	.6190	**.1651**	.1845	.1596
	Backbone	.7333	.7320	.0952	.0900	.7296	.7407	.1124	.0765	.6048	.5853	.6024	.1661	.1787	.1601
	CFS	.7257	.7204	.0958	.0911	.7163	.7280	.1134	.0771	.6245	.5950	.6225	.1665	**.1786**	.1602
	AutoField	.7305	.7294	.0954	.0903	.7303	.7425	.1123	.0763	.6240	.6026	.6216	.1663	.1796	.1603
	LPFS	.7354	.7333	.0950	.0899	.7308	.7438	.1123	.0763	.6253	.6030	.6232	.1661	.1794	.1602
	OptEmbed	.7353	.7343	.0950	.0898	.7334	.7433	.1121	.0762	.6265	.6034	.6245	.1656	.1791	.1596
	OptFS	.7360	.7342	.0949	.0897	.7322	.7441	.1123	.0762	.6274	.6035	.6253	.1654	.1787	**.1594**
	MultiFS	**.7437***	**.7402***	**.0941***	**.0893**	**.7370***	**.7504***	**.1116***	**.0756***	**.6278**	**.6049***	**.6260***	.1652	.1787	**.1594**
DCN	SD-Backbone	.7333	.7312	.0953	.0900	.7286	.7419	.1125	.0764	.6209	.5729	.6182	.1651	.1844	.1603
	Backbone	.7320	.7305	.0954	.0901	.7284	.7393	.1126	.0766	.6023	.5832	.5997	.1663	.1789	.1603
	CFS	.7206	.7203	.0961	.0909	.7163	.7301	.1135	.0770	.6242	.6002	.6206	.1660	**.1779**	.1593
	AutoField	.7315	.7297	.0953	.0902	.7280	.7387	.1126	.0765	.6243	.6017	.6220	.1660	.1795	.1600
	LPFS	.7332	.7308	.0952	.0900	.7304	.7431	.1123	.0763	.6259	.6029	.6236	.1658	.1793	.1598
	OptEmbed	.7344	.7341	.0952	.0899	.7317	.7417	.1122	.0763	.6260	.6038	.6236	.1659	.1794	.1598
	OptFS	.7365	.7330	.0949	.0899	.7312	.7427	.1124	.0763	.6263	.6033	.6242	.1657	.1795	.1596
	MultiFS	**.7411***	**.7381***	**.0946**	**.0896**	**.7355***	**.7500***	**.1119**	**.0758***	**.6280***	**.6041**	**.6254***	**.1649**	.1785	**.1588***
MS	Star	.7328	.7348	.0961	.0905	.7306	.7440	.1128	.0765	.6245	.5884	.6189	.1698	.1869	.1611
	HMoE	.7377	.7355	.0948	.0898	.7328	.7460	.1121	.0761	.6262	.6029	.6232	.1657	.1793	.1598
	MultiFS (DNN)	**.7425***	**.7385***	**.0944**	**.0895**	**.7351***	**.7504***	**.1119**	**.0758**	**.6277***	**.6038***	**.6248***	**.1654**	**.1788***	**.1596**

Table 3: Ablation Analysis on our MultiFS, where the best results are marked in bold.

Model	Methods	AliExpress-1				AliExpress-2				AliCCP					
		AUC↑		Logloss↓		AUC↑		Logloss↓		AUC↑			Logloss↓		
DNN	n.re.	.6284	.6312	.1910	.1905	.6440	.6486	.2809	.1712	.5677	.5524	.5664	.2243	.2431	.2146
	n.sh.	.7395	.7378	.0946	.0896	.7347	.7473	.1119	.0759	.6260	.5882	.6229	.1650	.1805	.1603
	n.or.	.6628	.6489	.1129	.1063	.7026	.7002	.1153	.0803	.6278	.6038	.6249	**.1654**	.1789	.1597
	n.sp.	**.7427**	**.7392**	**.0944**	**.0894**	**.7363**	**.7513**	**.1117**	**.0758**	**.6280**	**.6040**	**.6251**	.1655	.1789	.1597
	MultiFS	.7425	.7385	**.0944**	.0895	.7351	.7504	.1119	**.0758**	.6277	.6038	.6248	**.1654**	**.1788**	**.1596**
DeepFM	n.re.	.6564	.6407	.2165	.2158	.6216	.6240	.2855	.1810	.5663	.5524	.5644	.2086	.2283	.2029
	n.sh.	.7419	.7405	.0943	.0892	.7371	.7488	**.1116**	.0757	.6268	.5894	.6244	.1649	.1822	.1595
	n.or.	**.7447**	.7393	.0942	**.0893**	.7376	.7509	**.1116**	**.0756**	.6279	.6048	**.6262**	**.1651**	**.1787**	**.1594**
	n.sp.	.7440	.7399	**.0941**	**.0893**	**.7374**	.7502	**.1116**	**.0756**	**.6280**	**.6049**	.6261	**.1651**	**.1787**	.1595
	MultiFS	.7437	**.7402**	**.0941**	**.0893**	.7370	**.7504**	**.1116**	**.0756**	.6278	**.6049**	.6260	.1652	**.1787**	**.1594**
DCN	n.re.	.6196	.6309	.2073	.1926	.6472	.6390	.2744	.1672	.5699	.5559	.5676	.2212	.2426	.2149
	n.sh.	.7400	.7382	.0948	**.0896**	.7323	.7470	.1121	.0759	.6245	.5820	.6218	.1659	.1811	.1599
	n.or.	.6870	.6741	.1039	.0988	.7177	.7275	.1135	.0775	**.6282**	**.6044**	.6254	**.1649**	**.1785**	**.1588**
	n.sp.	**.7423**	**.7384**	**.0945**	**.0896**	.7347	**.7501**	**.1119**	**.0757**	.6281	**.6044**	**.6256**	.1650	**.1785**	.1589
	MultiFS	.7411	.7381	.0946	**.0896**	**.7355**	.7500	**.1119**	.0758	.6280	.6041	.6254	**.1649**	**.1785**	**.1588**

features of multiple scenarios, we integrate them into the downstream model, which uses three backbone models, respectively. We report the corresponding results in Table 4. From the results in Table 4, we can observe that the feature subsets obtained by our MultiFS with different backbone models can perform similarly on the downstream recommendation models with different backbone models. In addition, comparing with the results in Table 2, it can be found that the current downstream model can also maintain a good

Table 4: Transferability Analysis on all datasets, where the best results are marked in bold.

Target	Source	AliExpress-1				AliExpress-2				AliCCP					
		AUC↑		Logloss↓		AUC↑		Logloss↓		AUC↑			Logloss↓		
DNN	DNN	**.7425**	**.7385**	**.0944**	**.0895**	.7351	.7504	.1119	**.0758**	**.6277**	**.6038**	**.6248**	**.1654**	**.1788**	**.1596**
	DeepFM	.7417	.7382	.0945	**.0895**	.7346	.7489	.1119	.0759	.6263	.6037	.6241	.1656	.1789	**.1596**
	DCN	.7414	.7358	.0945	.0896	**.7361**	**.7514**	**.1118**	**.0758**	.6276	.6032	.6247	.1655	**.1788**	**.1596**
DeepFM	DeepFM	.7437	**.7402**	**.0941**	**.0893**	.7370	.7504	.1116	.0756	**.6278**	**.6049**	**.6260**	**.1652**	**.1787**	**.1594**
	DNN	.7440	.7400	**.0941**	**.0893**	**.7381**	.7513	**.1115**	.0756	.6276	.6042	.6259	.1653	**.1787**	**.1594**
	DCN	**.7444**	.7398	**.0941**	**.0893**	.7375	**.7519**	.1116	**.0755**	.6277	.6048	.6258	.1654	**.1787**	.1595
DCN	DCN	.7411	.7381	.0946	**.0896**	**.7355**	.7500	**.1119**	.0758	**.6280**	**.6041**	**.6254**	**.1649**	**.1785**	**.1588**
	DNN	**.7420**	**.7389**	**.0945**	**.0896**	.7343	**.7502**	**.1119**	**.0757**	.6274	.6035	.6247	.1654	.1788	.1596
	DeepFM	.7415	**.7389**	.0946	**.0896**	.7344	.7499	**.1119**	.0758	.6269	.6036	.6246	.1655	.1789	.1596

Table 5: Analysis of our MultiFS training results, including the feature retention ratio on each scenario and the feature overlap ratio between multiple scenarios.

Method	AliExpress-1				AliExpress-2				AliCCP						
	NL	FR	Shared	NL&FR	ES	US	Shared	ES&US	SA	SB	SC	Shared	SA&SB	SA&SC	SB&SC
MultiFS-DNN	.0123	.0131	.3938	3e-6	.0228	.0120	.3176	.0000	.0262	.0050	.0369	.7606	.0004	.0089	.0001
MultiFS-DeepFM	.0198	.0204	.3002	5e-6	.0386	.0217	.2390	.0000	.0509	.0112	.0701	.6195	.0006	.0233	.0002
MultiFS-DCN	.0234	.0297	.3280	3e-6	.0382	.0209	.2749	.0000	.0266	.0113	.0359	.7442	.0004	.0068	3e-5

enough multi-scenario recommendation performance. These results demonstrate that our MultiFS can generate a subset of features with good transferability for the multi-scenario recommendation.

5.5 RQ4: Analysis of Feature Subset

Finally, we analyze the feature subset obtained by MultiFS for various backbone models. As described in Section 4, our MultiFS aims to select a suitable subset of sparse shared features and specific feature subsets for different scenarios and for each scenario, respectively. Furthermore, the sparse specific features between different scenarios should remain sufficiently discriminative. Therefore, as shown in Table 5, we report the feature retention ratio in each scenario, the shared feature ratio, and the feature overlap ratio between different scenarios on the three benchmark datasets. The formula for calculating the feature overlap ratio between different scenarios is as follows:

$$\mathbf{g}^p \& \mathbf{g}^r = \frac{\sum_{i=1}^{n} (\mathbf{g}_i^p \times \mathbf{g}_i^r)}{\sum_{i=1}^{n} \mathbf{g}_i^p + \sum_{i=1}^{n} \mathbf{g}_i^r}, \tag{22}$$

where $\mathbf{g}^p$ and $\mathbf{g}^r \in \{\mathbf{g}^k\}$, and $p \neq r$. As expected, our MultiFS can effectively reduce redundant features to obtain the refined shared features and specific features, where the shared features will have a higher ratio than the specific features to better capture the commonality between different scenarios. In addition, we can also find that the feature overlap ratio between any two scenarios is maintained at a very small value. These findings above all indicate that our MultiFS really get a shared feature subset across scenario and a discriminative-specific feature subset for each scenario.

6 CONCLUSIONS AND FUTURE WORK

In this paper, different from previous multi-scenario recommendation methods aimed at architecture improvement, we focus on the feature selection problem of multi-scenario recommendation and propose a novel automated multi-scenario feature selection (MultiFS) framework to solve it. Specifically, we introduce a new framework consisting of scenario-shared gates and scenario-specific gates for multi-scenario architectures. These two gates determine the features input into the model in a hierarchical manner, where the scenario-shared gate aims to select useful features for all the scenarios, and the scenario-specific gate aims to select the useful features for individual scenarios. To alleviate these two feature gates' large search space, we adopt the sparse and orthogonal constraints on the gate vectors to make the learning mechanism more feasible. Furthermore, we introduce a shared loss on the scenario-shared features to improve the generalization ability of our MultiFS for distribution imbalance in multi-scenario recommendations. Finally, we conduct extensive experiments on two public multi-scenario datasets to demonstrate the effectiveness of our MultiFS. For future work, we are interested in exploring more optimization constraints that benefit our framework to achieve a more parsimonious subset of features and better model performance. In addition, we are also interested in conducting more explorations on efficient training for the multi-scenario recommendation, such as feature embedding dimensions, feature interactions, and architecture search.

ACKNOWLEDGMENTS

We thank the support of the National Natural Science Foundation of China (No.62302310, No.62272315), Research Impact Fund (No.R1015-23), APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for New Faculty of City University of Hong Kong), CityU - HKIDS Early Career Research Grant (No.9360163), Hong Kong ITC Innovation and Technology Fund Midstream Research Programme for Universities Project (No.ITS/034/22MS), Hong Kong Environmental and Conservation Fund (No. 88/2022), and SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074).

ETHICAL IMPLICATIONS

The proposed MultiFS has the same limitations as most existing recommendation methods, which may cause filter bubbles or echo chamber phenomena to service users. Therefore, in deploying our MultiFS, combining some existing model-independent debiasing recommendation techniques is necessary to reduce these adverse effects. In addition, our MultiFS strictly adheres to user privacy, i.e., using public datasets with anonymized information processing.

REFERENCES

[1] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[2] Jianxin Chang, Chenbin Zhang, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. 2023. Pepnet: Parameter and embedding personalized network for infusing with personalized prior information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3795–3804.

[3] Wei Chen, Wynne Hsu, and Mong Li Lee. 2013. Making recommendations from multiple domains. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 892–900.

[4] Zhongde Chen, Ruize Wu, Cong Jiang, Honghui Li, Xin Dong, Can Long, Yong He, Lei Cheng, and Linjian Mo. 2022. CFS-MTL: A causal feature selection mechanism for multi-task learning via pseudo-intervention. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3883–3887.

[5] 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.

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830* (2016).

[7] Ke Ding, Xin Dong, Yong He, Lei Cheng, Chilin Fu, Zhaoxin Huan, Hai Li, Tan Yan, Liang Zhang, Xiaolu Zhang, et al. 2021. MSSM: A multiple-level sparse sharing model for efficient multi-task learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2237–2241.

[8] Jérémie Donà and Patrick Gallinari. 2021. Differentiable feature selection, a reparameterization approach. In *Proceedings of the 2021 Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 414–429.

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

[10] Yi Guo, Zhaocheng Liu, Jianchao Tan, Chao Liao, Daqing Chang, Qiang Liu, Sen Yang, Ji Liu, Dongying Kong, Zhi Chen, et al. 2022. LPFS: Learnable polarizing feature selection for click-through rate prediction. *arXiv preprint arXiv:2206.00267* (2022).

[11] Pengcheng Li, Runze Li, Qing Da, An-Xiang Zeng, and Lijun Zhang. 2020. Improving multi-scenario learning to rank in e-commerce by exploiting task relationships in the label space. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 2605–2612.

[12] Weilin Lin, Xiangyu Zhao, Yejing Wang, Tong Xu, and Xian Wu. 2022. AdaFS: Adaptive feature selection in deep recommender system. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3309–3317.

[13] Dugang Liu, Pengxiang Cheng, Hong Zhu, Xing Tang, Yanyu Chen, Xiaoting Wang, Weike Pan, Zhong Ming, and Xiuqiang He. 2023. DIWIFT: Discovering instance-wise influential features for tabular data. In *Proceedings of the ACM Web Conference 2023*. 1673–1682.

[14] Dugang Liu, Yuhao Wu, Weixin Li, Xiaolian Zhang, Hao Wang, Qinjuan Yang, and Zhong Ming. 2023. Pairwise intent graph embedding learning for context-aware recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 588–598.

[15] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable architecture search. In *Proceedings of the 6th International Conference on Learning Representations*.

[16] Junjie Liu, Zhe Xu, Runbin Shi, Ray CC Cheung, and Hayden KH So. 2020. Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. In *Proceedings of the 8th International Conference on Learning Representations*.

[17] Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing feature set for click-through rate prediction. In *Proceedings of the ACM Web Conference 2023*. 3386–3395.

[18] Fuyuan Lyu, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. 2022. OptEmbed: Learning optimal embedding table for click-through rate prediction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*. 1399–1409.

[19] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1930–1939.

[20] Hosein Mohimani, Massoud Babaie-Zadeh, and Christian Jutten. 2008. A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm. *IEEE Transactions on Signal Processing* 57, 1 (2008), 289–301.

[21] Shanlei Mu, Penghui Wei, Wayne Xin Zhao, Shaoguo Liu, Liang Wang, and Bo Zheng. 2023. Hybrid contrastive constraints for multi-scenario Ad ranking. *arXiv preprint arXiv:2302.02636* (2023).

[22] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web: Methods and strategies of web personalization*. 291–324.

[23] Qijie Shen, Wanjie Tao, Jing Zhang, Hong Wen, Zulong Chen, and Quan Lu. 2021. Sar-Net: A scenario-aware ranking network for personalized fair recommendation in hundreds of travel scenarios. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 4094–4103.

[24] 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 *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 4104–4113.

[25] 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 *Proceedings of the 14th ACM Conference on Recommender Systems*. 269–278.

[26] Xing Tang, Yang Qiao, Yuwen Fu, Fuyuan Lyu, Dugang Liu, and Xiuqiang He. 2023. OptMSM: Optimizing multi-scenario modeling for click-through rate prediction. In *Proceedings of the 2021 Joint European Conference on Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track*. 567–584.

[27] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58, 1 (1996), 267–288.

[28] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD 2017*. 1–7.

[29] Yejing Wang, Zhaocheng Du, Xiangyu Zhao, Bo Chen, Huifeng Guo, Ruiming Tang, and Zhenhua Dong. 2023. Single-shot feature selection for multi-task recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 341–351.

[30] Yuhao Wang, Ha Tsz Lam, Yi Wong, Ziru Liu, Xiangyu Zhao, Yichao Wang, Bo Chen, Huifeng Guo, and Ruiming Tang. 2023. Multi-Task Deep Recommender Systems: A Survey. *arXiv preprint arXiv:2302.03525* (2023).

[31] Yuhao Wang, Xiangyu Zhao, Bo Chen, Qidong Liu, Huifeng Guo, Huanshuo Liu, Yichao Wang, Rui Zhang, and Ruiming Tang. 2023. PLATE: A prompt-enhanced paradigm for multi-scenario recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1498–1507.

[32] Yejing Wang, Xiangyu Zhao, Tong Xu, and Xian Wu. 2022. Autofield: Automating feature selection in deep recommender systems. In *Proceedings of the ACM Web Conference 2022*. 1977–1986.

[33] Dongbo Xi, Zhen Chen, Peng Yan, Yinger Zhang, Yongchun Zhu, Fuzhen Zhuang, and Yu Chen. 2021. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3745–3755.

[34] Senrong Xu, Liangyue Li, Yuan Yao, Zulong Chen, Han Wu, Quan Lu, and Hanghang Tong. 2023. MUSENET: Multi-scenario learning for repeat-aware personalized recommendation. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*. 517–525.

[35] Yongxin Yang and Timothy Hospedales. 2015. A unified perspective on multi-domain and multi-task learning. In *Proceedings of the 3rd International Conference on Learning Representations*.

[36] Qianqian Zhang, Xinru Liao, Quan Liu, Jian Xu, and Bo Zheng. 2022. Leaving no one behind: A multi-scenario multi-task meta learning approach for advertiser modeling. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 1368–1376.

[37] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *Comput. Surveys* 52, 1 (2019), 5(1)–5(38).

[38] Yuanliang Zhang, Xiaofeng Wang, Jinxin Hu, Ke Gao, Chenyi Lei, and Fei Fang. 2022. Scenario-adaptive and self-supervised model for multi-scenario personalized recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*. 3674–3683.

[39] Ruiqi Zheng, Liang Qu, Bin Cui, Yuhui Shi, and Hongzhi Yin. 2023. AutoML for deep recommender systems: A survey. *ACM Transactions on Information Systems* 41, 4 (2023), 1–38.