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

Dugang Liu[1], Chaohua Yang[1], Xing Tang[2], Yejing Wang[3], Fuyuan Lyu[4], Weihong Luo[2], Xiuqiang He[2], Zhong Ming[1], Xiangyu Zhao[3]

{dugang.ldg,ych981203}@gmail.com, xing.tang@hotmail.com, mingz@szu.edu.cn, xianzhao@cityu.edu.hk

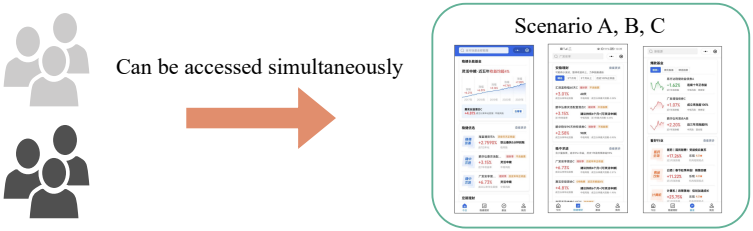
[1]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen University, China

[2]Financial Technology (FiT), Tencent, China

[3]City University of Hong Kong, China

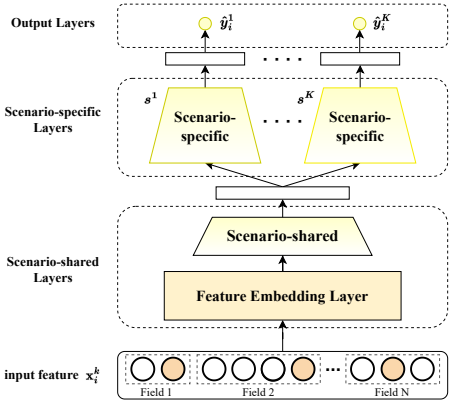
[4]McGill University, Canada

Multi-scenario Recommender Systems (MSRS)



- Multi-scenario recommender systems (MSRS) have been increasingly used in real-world industrial platforms for their excellent advantages in mitigating data sparsity and reducing maintenance costs [Sheng et al., 2021].
- Instead of training a model for each scenario, they aim to serve multiple scenarios simultaneously by training a unified model.

Multi-scenario Learning Architecture



- The key to multi-scenario learning is to capture the commonalities and differences between the scenarios [Mu et al., 2023].
- The scenario-shared architecture takes all features as input and outputs a shared representation, which aims to obtain shared information across the scenarios.
- The scenario-specific architecture makes predictions based on shared representation in the corresponding scenario.

Existing Problems

- Despite complex architecture, conventional MSRS usually use all relevant features indiscriminately and ignore the fact 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 single-scenario recommendation may lack the exploration of scenario relations, and directly extending these works to MSRS will incur a high computational cost.
- For example, for AutoField [Wang et al., 2022] where the selection granularity is restricted to m feature field, K scenarios will require 2^{Km} search space.
- This motivates us to propose an effective and efficient feature selection framework for MSRS to bridge the gap.

Multi-scenario Learning

- Let multi-scenario dataset with K scenarios denote 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,

$$\hat{y}_i^k = f(\mathbf{x}_i^k \mid \theta, \{\theta_{s^k}\}), \mathbf{x}_i^k = [x_{i1}^k, \cdots, x_{im}^k], \tag{1}$$

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

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

The Feature Selection Problem for MSRS

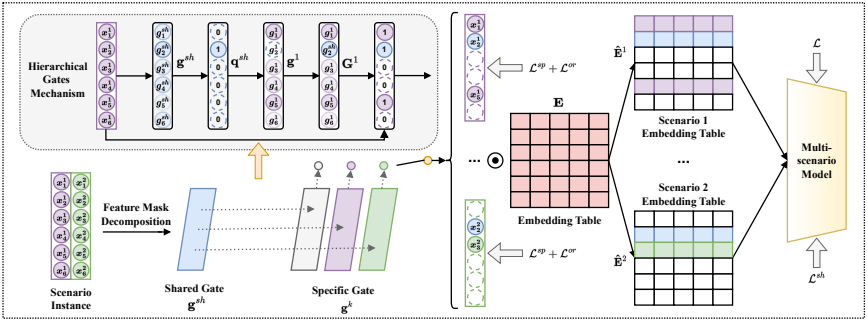
- To better represent and utilize features, we usually employ an embedding table $\mathbf{E} \in \mathbb{R}^{n \times d}$ to convert feature values into low-dimensional and dense real-value vectors, $\mathbf{e}_{ij}^k = \mathbf{E} \times x_{ij}^k$, where n is the number of feature values, d is a hyperparemeter 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 MSRS can be defined as applying gate mask operation on the embedding table for each scenario and producing the corresponding masked embedding table $\hat{\mathbf{E}}$,

$$\hat{\mathbf{E}}^k = \mathbf{E} \odot \mathbf{G}^k. \tag{3}$$

  Here $\mathbf{G}^k \in \{0, 1\}^n$, and $\odot$ denotes element-wise multiplication.
- Finally, with all these formulations, we formulate our problem as follows, where $\Theta = \{\theta, \{\theta_{s^k}\}_{k=1}^K\}$ denotes the network parameters.

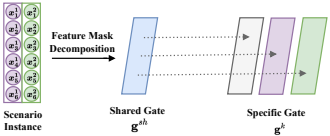
$$\min_{\Theta, \{\mathbf{G}^k\}} \mathcal{L}(\mathcal{D}). \tag{4}$$

Architecture



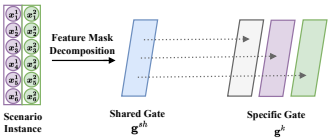
Our automated multi-scenario feature selection (MultiFS) framework consists of three key steps: feature mask decomposition, hierarchical gating mechanism, and optimization.

Feature Mask Decomposition (1/2)



- Since there are usually many overlapping features between the scenarios for MSRSs, 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.

Feature Mask Decomposition (2/2)

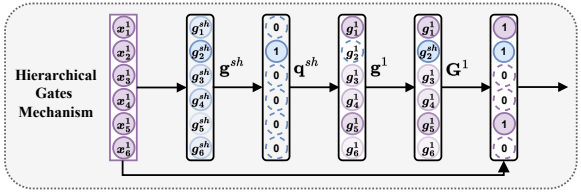


- We can decompose each $\mathbf{G}^k$ into scenario-shared feature selection and scenario-specific feature selection.

$$\mathbf{G}^k = g(\mathbf{g}^k, \mathbf{g}^{sh}) = \mathbf{g}^{sh} + (1 - \mathbf{g}^{sh}) \odot \mathbf{g}^k, \tag{5}$$

  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 search space for the set of G^k will change from $\mathcal{O}(2^{Kn})$ to $\mathcal{O}(2^n + 2^{K(n-n_{sh})})$.
- Note that $n - n_{sh} < n$ and n_{sh} will usually not be a small value in practice, thus reducing the space greatly.

Hierarchical Gating Mechanism (1/3)



- 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 [Lyu et al., 2023].

Hierarchical Gating Mechanism (2/3)

- Specifically, 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 $\mathbf{g}_c^{sh} \in \mathbb{R}^n$.
- The continual gate $\mathbf{g}^{sh}$ and $\mathbf{g}^k$ are defined as,

$$\mathbf{g}^{sh} = \frac{\sigma\left(\mathbf{g}_c^{sh} \times \tau\right)}{\sigma\left(\mathbf{g}_c^{sh(0)}\right)}, \tau = \gamma^{t/T}, \mathbf{g}^k = \frac{\sigma\left(\mathbf{g}_c^k \times \tau\right)}{\sigma\left(\mathbf{g}_c^{k(0)}\right)}, \quad \tau = \gamma^{t/T}, \tag{6}$$

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.

Hierarchical Gating Mechanism (3/3)

- Since the continual gates will make the Eq.(5) hard to get the remaining features hierarchically, we use a straight-through estimator operation $S(\cdot)$ on the scenario-shared gate during training [Courbariaux et al., 2016], where ϵ is a learnable threshold.

$$\mathbf{q}^{sh} = S(\mathbf{relu}(\mathbf{g}^{sh} - \epsilon)), \tag{7}$$

- Thus, we further reformulate Eq.(5) as $\mathbf{G}^k = \mathbf{q}^{sh} \odot \mathbf{g}^{sh} + (1 - \mathbf{q}^{sh}) \odot \mathbf{g}^k$.
- After training T epochs, the final gating vector $\mathbf{g}^{sh}$ and $\mathbf{g}^k$ are calculated through a unit-step function as follows:

$$\mathbf{g}^{sh} = \begin{cases} 0, & \mathbf{g}_c^{sh} \leq 0 \\ 1, & \text{otherwise} \end{cases}, \mathbf{g}^k = \begin{cases} 0, & \mathbf{g}_c^k \leq 0 \\ 1, & \text{otherwise} \end{cases}. \tag{8}$$

Optimization (1/2)

- First, to encourage the sparsity of feature gate vectors, we also introduce the l_1 regularization, 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 $\mathbf{g}$.

$$\mathcal{L}^{sp} = (\left\|\mathbf{g}^{sh}\right\|_1 + \sum_{k=1}^{K} \left\|\mathbf{g}^{k}\right\|_1), \tag{9}$$

- Second, an ideal MSRS should make the scenario-specific representations decoupled from each other. Therefore, we design an orthogonal penalty for $\{\mathbf{g}^k\}$ as follows, and with this term, our MultiFS can select informative features that are specific to the scenario.

$$\mathcal{L}^{or} = \sum_{p=1}^{K} \sum_{r=p+1}^{K} \left\|\mathbf{g}^{p} \odot \mathbf{g}^{r}\right\|_1 . \tag{10}$$

Optimization (2/2)

- Then, to address the imbalanced data distribution issue and ensure the generality of MultiFS, we introduce a single prediction on $\mathbf{g}^{sh}$ as an embedding mask, and the corresponding loss can be formulated as follows, where $\mathcal{L}_k^{sh}$ indicates the prediction error loss on scenario s_k after selecting feature embeddings only through $\mathbf{g}^{sh}$.

$$\hat{\mathbf{y}} = f(\mathbf{g}^{sh} \odot \mathbf{E} \times \mathbf{x}; \Theta), \tag{11}$$

$$\mathcal{L}^{sh} = \mathcal{L}(f(\mathbf{g}^{sh} \odot \mathbf{E} \times \mathbf{x}; \Theta), \mathbf{y}) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_k^{sh}(\hat{y}, y), \tag{12}$$

- Finally, combining Eq.(4), Eq.(9), Eq.(10) and Eq.(12), we can get the final training objective, where λ_1, λ_2, and λ_3 are the control weights of different optimization terms.

$$\min_{\{\mathbf{G}^k\}, \Theta} \mathcal{L}_{MultiFS} = \mathcal{L} + \lambda_1 \mathcal{L}^{sh} + \lambda_2 \mathcal{L}^{sp} + \lambda_3 \mathcal{L}^{or}, \tag{13}$$

Retrain Stage

- 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 $\mathbf{G}^k$.
- Specifically, 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:

$$\min_{\Theta} \mathcal{L} + \lambda_1 \mathcal{L}^{sh}. \tag{14}$$

Datasets

Table: 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

Baselines Methods and Backbone Methods

- To evaluate the generalization ability of all the methods, we integrate them with the mainstream skeleton models DNN, DeepFM [Guo et al., 2017], and DCN [Wang et al., 2017], respectively.
- We compare our MultiFS with the following single-scenario and multi-task feature selection baselines, including AutoField [Wang et al., 2022], LPFS [Guo et al., 2022], OptEmbed [Lyu et al., 2022], OptFS [Lyu et al., 2023] and CFS [Chen et al., 2022].
- 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 [Sheng et al., 2021] and HMoE [Li et al., 2020].

Implementation Details (1/2)

- 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 used 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 $\{1,2,\cdots,T-1\}$.

Implementation Details (2/2)

- For other baseline methods, we use the open source implementations for AutoField [Wang et al., 2022], LPFS [Guo et al., 2022] and OptEmbed [Lyu et al., 2022] and OptFS [Lyu et al., 2023].
- Due to the lack of available implementations for the CFS [Chen et al., 2022], STAR [Sheng et al., 2021], and HMoE [Li et al., 2020], we re-implement them based on the details provided by the original paper.
- Note that all the baselines will also search for the best hyperparameter combination within the same range.
- Following the setup of previous works [Wang et al., 2022, Lin et al., 2022], we use the common evaluation metrics for deep recommender systems, i.e., the area under the ROC curve (AUC) and cross-entropy (log loss).

RQ1: Performance Comparison (1/2)

	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**

RQ1: Performance Comparison (2/2)

- Unlike other feature selection baseline methods, our MultiFS can maintain a significant performance gain in most cases. This demonstrates that our MultiFS can effectively capture the beneficial shared information among multiple scenarios and identify the beneficial specific information for each scenario.
- We can find that compared to the baseline methods designed on the model architecture, 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.

RQ2: Ablation Study (1/2)

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**

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.

RQ2: Ablation Study (2/2)

- 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. 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.

RQ3: Transferability Analysis (1/2)

Table: 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

We implement our MultiFS through three backbone models. After obtaining the shared features and specific features of multiple scenarios, we integrate them into the downstream model, which uses three backbone models.

RQ3: Transferability Analysis (2/2)

- 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.
- 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, it can be found that the current downstream model can also maintain a good enough multi-scenario recommendation performance.

RQ4: Analysis of Feature Subset (1/2)

Table: 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

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.

RQ3: Analysis of Feature Subset (2/2)

- 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 capture the commonality between different scenarios better.
- In addition, we can also find that the feature overlap ratio between any two scenarios is maintained at a minimal value.
- These findings above all indicate that our MultiFS gets a shared feature subset across scenarios and a discriminative-specific feature subset for each scenario.

Conclusions

- In this paper, 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, 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.

Thank You!

- We thank the anonymous reviewers for their expert and constructive comments and suggestions.
- We thank the support of the National Natural Science Foundation of China (NSFC) Nos. 62302310 and 62272315.

- Codes and slides are available at: `https://github.com/dgliu/WSDM24_MultiFS`
- If you have any questions, please feel free to contact us.


Chen, Z., Wu, R., Jiang, C., Li, H., Dong, X., Long, C., He, Y., Cheng, L., and Mo, L. (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*, pages 3883–3887.


Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. (2016).
Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1.
arXiv preprint arXiv:1602.02830.


Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017).
Deepfm: A factorization-machine based neural network for ctr prediction.
In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1725–1731.


Guo, Y., Liu, Z., Tan, J., Liao, C., Chang, D., Liu, Q., Yang, S., Liu, J., Kong, D., Chen, Z., et al. (2022).
Lpfs: Learnable polarizing feature selection for click-through rate prediction.
arXiv preprint arXiv:2206.00267.


Li, P., Li, R., Da, Q., Zeng, A.-X., and Zhang, L. (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*, pages 2605–2612.


Lin, W., Zhao, X., Wang, Y., Xu, T., and Wu, X. (2022).
Adafs: Adaptive feature selection in deep recommender system.
In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3309–3317.


Lyu, F., Tang, X., Liu, D., Chen, L., He, X., and Liu, X. (2023).
Optimizing feature set for click-through rate prediction.
In *Proceedings of the ACM Web Conference 2023*, pages 3386–3395.


Lyu, F., Tang, X., Zhu, H., Guo, H., Zhang, Y., Tang, R., and Liu, X. (2022).
Optembed: Learning optimal embedding table for click-through rate prediction.
In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pages 1399–1409.

Mu, S., Wei, P., Zhao, W. X., Liu, S., Wang, L., and Zheng, B. (2023).
Hybrid contrastive constraints for multi-scenario ad ranking.
arXiv preprint arXiv:2302.02636.

Sheng, X.-R., Zhao, L., Zhou, G., Ding, X., Dai, B., Luo, Q., Yang, S., Lv, J., Zhang, C., Deng, H., 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*, pages 4104–4113.

Wang, R., Fu, B., Fu, G., and Wang, M. (2017).
Deep & cross network for ad click predictions.
In *Proceedings of the ADKDD 2017*, pages 1–7.

Wang, Y., Zhao, X., Xu, T., and Wu, X. (2022).
Autofield: Automating feature selection in deep recommender systems.
In *Proceedings of the ACM Web Conference 2022*, pages 1977–1986.