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]

[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

Background

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 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 deep recommender systems may lack the exploration of scenario relations.

The Feature Selection Problem for MSRS

- 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. Usually, there are m *feature fields* in a feature vector $\mathbf{x}_i^k$. 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 corresponding to the model from features to labels, and θ and $\{\theta_{s^k}\}$ are the scenario-shared and scenario-specific parameters, respectively.
- 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 MSRSs can be defined as applying gate mask operation on the embedding table for each scenario and producing the corresponding masked embedding table $\hat{\mathbf{E}}$,

$$\hat{\mathbf{E}}^k = \mathbf{E} \odot \mathbf{G}^k. \tag{2}$$

  Here $\mathbf{G}^k \in \{0, 1\}^n$, and $\odot$ denotes element-wise multiplication.
- As discussed above, MSRSs should consider scenario-shared and scenario-specific information. To do this, we can further define a specific $\mathbf{G}^k$ as the combination of a scenario-shared gate vector and the corresponding scenario-specific gate vector,

$$\mathbf{G}^k = g(\mathbf{g}^k, \mathbf{g}^{sh}), \tag{3}$$

  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:

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

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

Mask Decomposition

As indicated in Eq.(3), 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. We formulate the g as follows,

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

where $\mathbf{g}^k \in \{0, 1\}^n$ represents the gate mask 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 will be reduced from $\mathcal{O}(2^{Kn})$ to $\mathcal{O}(2^n + 2^{K(n-n_{sh})})$.

Hierarchical Gating Mechanism

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,

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

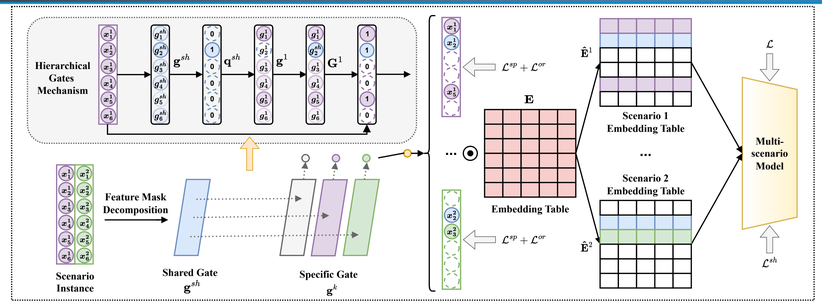
where $\mathbf{g}_c^{sh^{(0)}}$ and $\mathbf{g}_c^{k^{(0)}}$ are initial value of the continual gate, respectively, $\sigma(\cdot)$ is the sigmoid function, t is the current training epoch number, T is the total training epoch and γ is the final value of τ after training for T epochs. Since the continual gates will make the Eq.(5) hard to get the remaining features hierarchically, we use a straight-through estimator operation on the scenario-shared gate during training, $\mathbf{q}^{sh} = S(\mathbf{relu}(\mathbf{g}^{sh} - \epsilon))$, where ϵ is a learnable threshold. 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:

$$\mathbf{g}^{sh} = \begin{cases} 0, & \mathbf{g}_c^{sh} \leq 0 \\ 1, & \text{otherwise} \end{cases}, \mathbf{g}^k = \begin{cases} 0, & \mathbf{g}_c^k \leq 0 \\ 1, & \text{otherwise} \end{cases}. \tag{7}$$

Optimization

- To encourage the sparsity of feature gate vectors, $\mathcal{L}^{sp} = (\left\|\mathbf{g}^{sh}\right\|_1 + \sum_{k=1}^K \left\|\mathbf{g}^k\right\|_1)$.
- An orthogonal penalty for $\{\mathbf{g}^k\}$, $\mathcal{L}^{or} = \sum_{p=1}^K \sum_{r=p+1}^K \left\|\mathbf{g}^p \odot \mathbf{g}^r\right\|_1$.
- To address the imbalanced data distribution in MSRS, $\mathcal{L}^{sh} = \mathcal{L}(f(\mathbf{g}^{sh} \odot \mathbf{E} \times \mathbf{x}; \Theta), \mathbf{y}) = \frac{1}{K} \sum_{k=1}^K \mathcal{L}_k^{sh}(\hat{y}, y)$, where $\hat{\mathbf{y}} = f(\mathbf{g}^{sh} \odot \mathbf{E} \times \mathbf{x}; \Theta)$.

Architecture



The optimization objective function of our MultiFS is, $\min_{\{\mathbf{G}^k\}, \Theta} \mathcal{L}_{MultiFS} = \mathcal{L} + \lambda_1 \mathcal{L}^{sh} + \lambda_2 \mathcal{L}^{sp} + \lambda_3 \mathcal{L}^{or}$.

Result

	Method	AliExpress-1 AUC↑		AliExpress-1 Logloss↓		AliExpress-2 AUC↑		AliExpress-2 Logloss↓	
DNN	SD-Backbone	.7326	.7307	.0953	.0901	.7289	.7410	.1125	.0765
	Backbone	.7317	.7304	.0953	.0901	.7291	.7397	.1125	.0766
	CFS	.7251	.7209	.0962	.0910	.7172	.7297	.1134	.0769
	AutoField	.7289	.7280	.0955	.0903	.7296	.7418	.1124	.0763
	LPFS	.7332	.7320	.0952	.0900	.7304	.7430	.1123	.0763
	OptEmbed	.7353	.7343	.0951	.0898	.7319	.7417	.1122	.0763
	OptFS	.7351	.7322	.0952	.0901	.7307	.7414	.1124	.0764
	MultiFS	**.7425***	**.7385***	**.0944***	**.0895**	**.7351***	**.7504***	**.1119**	**.0758***
DeepFM	SD-Backbone	.7335	.7311	.0953	.0900	.7292	.7417	.1125	.0764
	Backbone	.7333	.7320	.0952	.0900	.7296	.7407	.1124	.0765
	CFS	.7257	.7204	.0958	.0911	.7163	.7280	.1134	.0771
	AutoField	.7305	.7294	.0954	.0903	.7303	.7425	.1123	.0763
	LPFS	.7354	.7333	.0950	.0899	.7308	.7438	.1123	.0763
	OptEmbed	.7353	.7343	.0950	.0898	.7334	.7433	.1121	.0762
	OptFS	.7360	.7342	.0949	.0897	.7322	.7441	.1123	.0762
	MultiFS	**.7437***	**.7402***	**.0941***	**.0893**	**.7370***	**.7504***	**.1116***	**.0756***
DCN	SD-Backbone	.7333	.7312	.0953	.0900	.7286	.7419	.1125	.0764
	Backbone	.7320	.7305	.0954	.0901	.7284	.7393	.1126	.0766
	CFS	.7206	.7203	.0961	.0909	.7163	.7301	.1135	.0770
	AutoField	.7315	.7297	.0953	.0902	.7280	.7387	.1126	.0765
	LPFS	.7332	.7308	.0952	.0900	.7304	.7431	.1123	.0763
	OptEmbed	.7344	.7341	.0952	.0899	.7317	.7417	.1122	.0763
	OptFS	.7365	.7330	.0949	.0899	.7312	.7427	.1124	.0763
	MultiFS	**.7411***	**.7381***	**.0946**	**.0896**	**.7355***	**.7500***	**.1119**	**.0758***
MS	Star	.7328	.7348	.0961	.0905	.7306	.7440	.1128	.0765
	HMoE	.7377	.7355	.0948	.0898	.7328	.7460	.1121	.0761
	MultiFS (DNN)	**.7425***	**.7385***	**.0944**	**.0895**	**.7351***	**.7504***	**.1119**	**.0758**