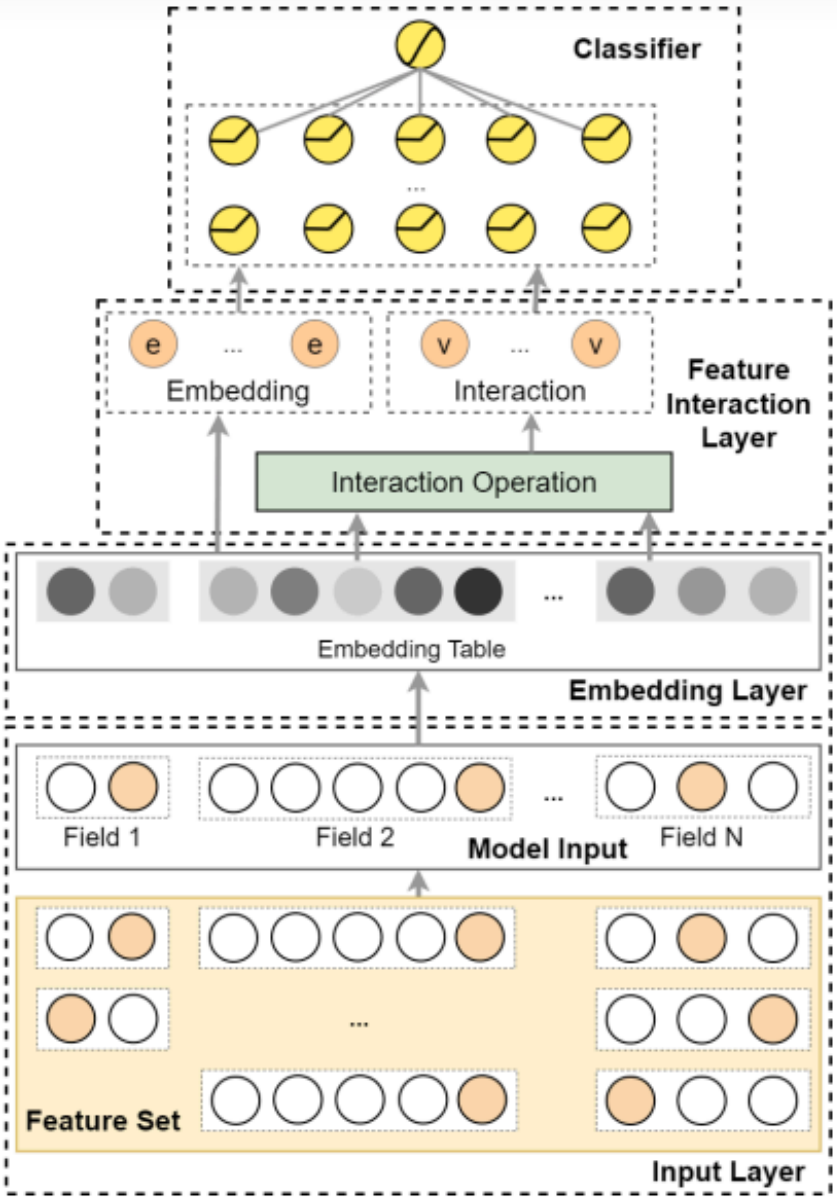

Figure 1: Overview of the general CTR framework.

腾讯出品 2023- OptFS：CTR特征选择新框架，引领特征选择新方向

SmartMindAI
专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我

已关注

Introduction

在CTR预测中，盲目将所有可用特征输入特征集并不理想。因为有些特征可能会对模型性能产生负面影响。首先，这些特征自身可能只增加了一些不必要的学习参数，导致模型容易过拟合。其次，它们可能导致一些无效的特征交互，带来不必要的噪声和复杂性，从而降低模型预测性能。因此，在选择特征集时，应当关注这些因素之间的关联性。

本文提出了一种名为OptFS的方法，用于解决最佳特征集查找的问题。在此方法中，我们面临着两个主要挑战。 第一挑战在于如何同时选择特征及其交互。如前所述，一种理想的特征集应避免引入无用的交互。为了解决这个问题，我们把每个特征交互的选取分解为两个相关的特征选取。因此，OptFS缩小了特征交互的搜索空间，并在整个模型上进行了训练，考虑到各种特征交互操作。

- 本文将特征集优化问题分为两个部分，一部分是特征级别，另一部分是特征交互级别。

知乎

- 本文进行了在三个大型公共数据集上的大量实验，结果表明提出的该方法的有效性和效率。

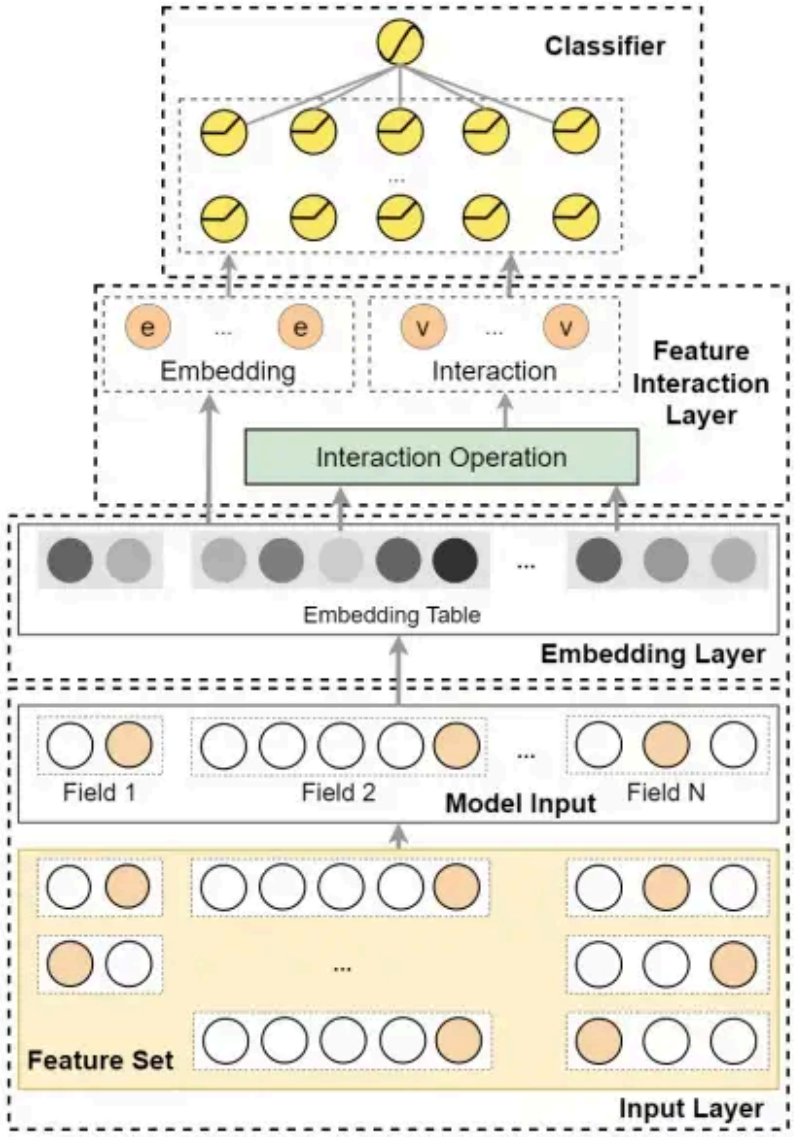

Figure 1: Overview of the general CTR framework.

OptFS

本文首先区分了Section中的特征集优化问题，并详细说明了OptFS如何进行特征选择+；然后，我们将详细介绍OptFS如何影响特征交互选择；最后，我们将详细介绍学习-by-continuation方法。

Problem Formulation

我们将所有可能的特征表示为$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$。$\mathbf{x}_i$是一个— hot 表示，非常稀疏且高维。如前所述，特征集优化问题旨在确定所有可能特征中有效的特征，即定义最优特征集$\mathbf{X}^{\mathbf{g}} \subset \mathbf{X}$。可以将其表述如下： 目标函数+的形式为：

$$f(\mathbf{X}) = L(\hat{\mathbf{y}}, \mathbf{y}) + \mu R(\mathbf{X}),$$

其中，$L(\hat{\mathbf{y}}, \mathbf{y})$是损失函数，用于评估预测效果；$\mu R(\mathbf{X})$是正则化项，用于防止过拟合；$\mathbf{X}^{\mathbf{g}}$是寻找的目标最优特征子集。

$$\begin{aligned} &\min_{\mathbf{W}} \mathcal{L}(\mathcal{D}|\mathbf{W}),\ \mathcal{D} = \{\mathbf{X}^{\mathbf{g}}, \mathbf{Y}\}, \\ s.t. &\forall \mathbf{x} \in \mathbf{X}^{\mathbf{g}}, \mathcal{L}(\mathbf{X}^{\mathbf{g}}) > \mathcal{L}(\mathbf{X}^{\mathbf{g}} - \{\mathbf{x}\}), \\ &\forall \mathbf{x} \notin \mathbf{X}^{\mathbf{g}}, \mathcal{L}(\mathbf{X}^{\mathbf{g}}) \geq \mathcal{L}(\mathbf{X}^{\mathbf{g}} + \{\mathbf{x}\}), \end{aligned}$$

知乎

Feature Selection

每个特征$\mathbf{z}_i$包含了所有可能特征的一部分比例，记作： $\mathbf{z}_i = \{\mathbf{x}_{k_i}\}, 1 \leq k_i \leq m,$
其中，表示的是特征与字段之间的1到多映射关系。在实践中，字段的数量远小于特征的数量。因此，可以从特征和字段的角度重写CTR模型的输入：

$$\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n] = [\mathbf{x}_{k_1}, \mathbf{x}_{k_2}, \cdots, \mathbf{x}_{k_n}],$$

其中，第二个等号表示对于输入$\mathbf{z}$，对应字段$\mathbf{z}_i$的特征为$\mathbf{x}_{k_i}$，
如图所示。我们通常使用嵌入表将$\mathbf{z}_i$s转换为低维和密集实值向量。这可以形式化为

$$\mathbf{e}_i = \mathbf{E} \times \mathbf{z}_i = \mathbf{E} \times \mathbf{x}_{k_i}, 1 \leq i \leq n, 1 \leq k_i \leq m,$$

其中$\mathbf{E} \in \mathbb{R}^{m \times D}$是嵌入表，$m$是特征值*的数量，$D$是嵌入的大小。然后将嵌入向量堆叠在一起形成一个嵌入向量*
$\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_n]$。 我们提出了特征选择方法的基本流程和步骤。首先，根据二进制门$\mathbf{g}_{k_i}$的值，判断特征$\mathbf{x}_{k_i}$是否应该被保留或删除。然后，将对应的特征嵌入$\mathbf{e}^{\mathbf{g}}_{k_i}$与其它特征嵌入结合起来，形成一个特征选择的嵌入向量$\mathbf{e}^{\mathbf{g}}$. 最后，模型的预测结果可以通过嵌入向量$\mathbf{e}^{\mathbf{g}}$来进行计算。

$$\hat{y} = \mathcal{F}(\mathbf{g} \odot \mathbf{E} \times \mathbf{x} | \mathbf{W}) = \mathcal{F}(\mathbf{E}^{\mathbf{g}} \times \mathbf{x} | \mathbf{W}),$$

其中，$\mathbf{g} \in \{0, 1\}^m$是指表示特定特征是否被选中的门向量，$\mathbf{E}^{\mathbf{g}} = \mathbf{g} \odot \mathbf{E}$是指被选中特征的特征表嵌入，$\mathbf{E}^{\mathbf{g}}$也可以被视为从嵌入表经过变换得到的特征集$\mathbf{X}^{\mathbf{g}}$后的特征表，表示为
$\mathbf{E}^{\mathbf{g}} = \mathbf{E} \times \mathbf{X}^{\mathbf{g}}$。

Feature Interaction Selection

特征交互可以增强模型的表现力，提高模型的泛化能力。在以前的研究中，有多种类型的特征交互，例如内积* 。两个特征$\mathbf{e}_i$和$\mathbf{e}_j$之间的交互一般表示为：
$\mathbf{v}_{(i,j)} = \mathcal{O}(\mathbf{e}_i, \mathbf{e}_j),$

其中，$\mathcal{O}$作为交互操作，可以从单层感知器到跨层不等。特征交互选择可以表示为为每种特征交互分配$\mathbf{g}'_{(i,j)}$。所有特征交互可以聚合在一起用于最终预测：

$\hat{y} = \mathcal{H}((\mathbf{g}' \odot \mathbf{v}) \oplus \mathcal{G}(\mathbf{e}^{\mathbf{g}})) = \mathcal{H}(\mathbf{v}^{\mathbf{g}'} \oplus \mathcal{G}(\mathbf{e}^{\mathbf{g}}))$, 其中符号$\oplus$表示连接操作，$\mathcal{G}(\cdot)$表示从嵌入空间到特征交互空间的转换函数*。$\mathcal{H}(\cdot)$表示预测函数*。
主流模型中的$\mathcal{G}(\cdot)$，$\mathcal{O}(\cdot)$和$\mathcal{H}(\cdot)$的组合在表格中总结如下：

在现实世界中探索所有可能的特征交互的方式。目前，直接通过引入特征交互矩阵来表示所有第二阶特征交互是一种可能的方法，但由于变量数量的巨大，这种方法几乎是不可行的。因此，一些研究人员采用了有限的搜索空间，例如特征域交互，以缩小搜索空间并避免过拟合的问题。然而，这种放松可能会导致无法有效地区分有用的特征交互和无用的特征交互。因此，在实际应用中，需要在搜索空间和模型性能之间找到合适的平衡点*。
$\mathbf{g}'_{(k_i,k_j)} = \mathbf{g}_{k_i} \times \mathbf{g}_{k_j},$

当两个特征都被认为是有用的时候，这两个特征之间的交互才会被认为是有效的。

知乎

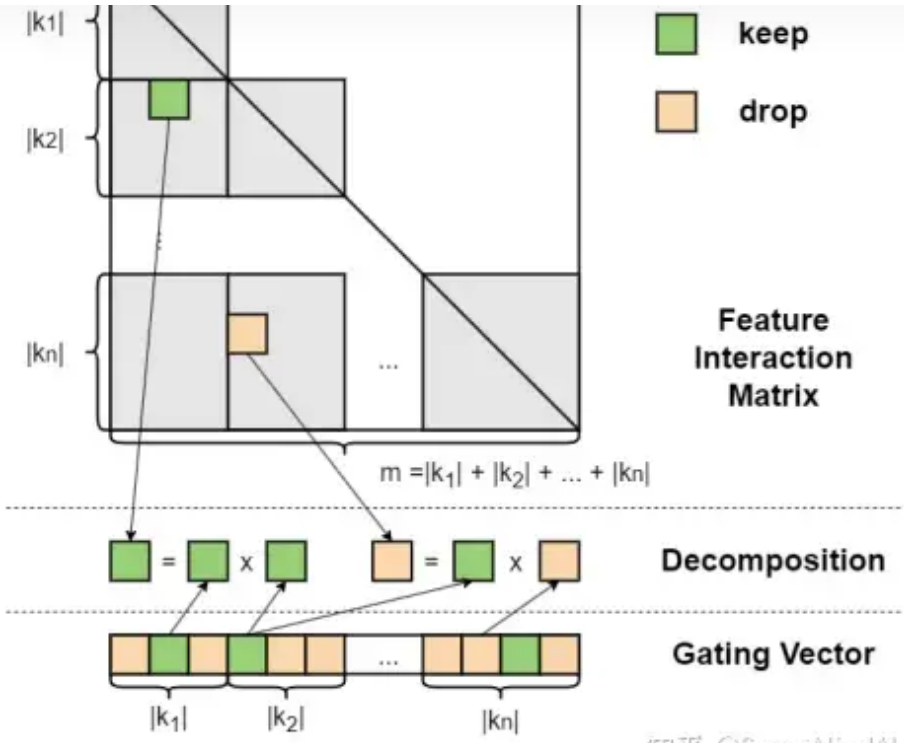

Figure 2: The Overview of OptFS.

这种特征交互的分解示例图已经在Figure中进行了展示。最终的预测结果可以通过上述步骤得到，并可以根据需要进一步优化。

$$\hat{y} = \mathcal{H}((\mathbf{g} \times \mathbf{g} \odot \mathbf{v}) \oplus \mathcal{G}(\mathbf{g} \odot \mathbf{e})),$$

通过选择特征交互的门向量，可以进一步缩小搜索空间，并以端到端的方式获得最优特征集。这样不仅可以提高模型的泛化能力，还可以简化模型的设计过程。

Table 1: Summary of $\mathcal{G}(\cdot)$, $\mathcal{O}(\cdot)$ and $\mathcal{H}(\cdot)$ in mainstream models

Model	$\mathcal{G}(\cdot)$	$\mathcal{O}(\cdot)$	$\mathcal{H}(\cdot)$
FM [26]	null	inner product	null
DeepFM [7]	MLP	inner product	average
DCN [31]	MLP	cross network	average
IPNN [24]	null	inner product	MLP
OPNN [24]	null	outer product	MLP
PIN [25]	null	MLP	MLP

Learning by Continuation

即使在Section中，搜索空间从$C_m^2 + m$缩小到了m，我们仍然需要确定是否保留或丢弃特征集合中的每个特征。为了有效地训练整个模型，我们引入了一个学习延续训练方案。在搜索阶段，通过计算梯度，我们可以有效地确定特征交互的门向量。在重训阶段，我们需要通过迭代的方式更新嵌入表和其他参数，以适应新的特征交互门向量。这种方法已经被证明是一种有效的方法来近似l_0归一化，从而提高模型的泛化能力。

Searching

为了高效地以特征级别对特征集进行优化，我们引入了连续的门向量$\mathbf{g}_c \in \mathbb{R}^m$。

在搜索阶段，我们引入了一个指数增加的温度值τ来近似L_0归一化*。通过这种方式，我们可以有效地控制特征的活跃程度，从而提高模型的泛化能力。

知乎

其中初始值$\mathbf{g}_c^{(0)}$是通过sigmoid函数计算得到的，而随着时间的推移，温度值τ会逐渐接近二进制门向量。此外，您还提到了预测的过程，即将方程（9）用于计算每个样本的预测结果。

$$\mathrm{CE}(y,\hat{y}) = y\log(\hat{y}) + (1-y)\log(1-\hat{y}),$$

其中y是用户点击的实际标签。最后准确率损失是通过比较用户的实际标签和模型的预测结果来计算的。 $\mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathbf{E},\mathbf{W}\}) = -\frac{1}{|\mathcal{D}|}\sum_{(\mathbf{x},y)\in\mathcal{D}}\mathrm{CE}(y,\mathcal{F}(\mathbf{E}\times\mathbf{x}|\mathbf{W})),$

其中$\mathcal{D}$是训练数据集，$\mathbf{W}$是除了嵌入表$\mathbf{E}$以外的网络参数。因此，最终的训练目标变为：

$$\min_{\mathbf{g}_c,\mathbf{E},\mathbf{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathbf{g}_c\odot\mathbf{E},\mathbf{W}\}) + \lambda\|\mathbf{g}\|_1,$$

其中λ是正则化惩罚项，$\|\cdot\|_1$表示ℓ_1范数以鼓励稀疏性。由于二进制$\mathbf{g}$的l_0范数等于l_1范数，因此我们重述为ℓ_0范数到ℓ_1范数。在训练了T个周期后，最终的门向量是由一个单位步函数计算出来的，这个函数在连续门向量的基础上添加了一个额外的约束条件，使其更接近于二进制门向量。

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

Retraining

在搜索阶段，将所有可能的特征输入模型来探索最优特征集$\mathbf{X}^{\mathbf{g}}$。因此，无用的特征可能会损害模型的性能。为了解决这个问题，我们需要在获得最优特征集$\mathbf{X}^{\mathbf{g}}$后重新训练模型。一旦确定了门向量$\mathbf{g}$，那么就可以重新训练模型参数$\mathbf{E}$和$\mathbf{W}$，并且这些参数会在T_c时期值被精心调整。这样做是为了避免过度拟合，并且在不同的初始化条件下也能获得良好的结果。

$$\min_{\mathbf{E},\mathbf{W}} \mathcal{L}_{\mathrm{CE}}(\mathcal{D}|\{\mathbf{g}\odot\mathbf{E},\mathbf{W}\}).$$

Algorithm 1 The OptFS Algorithm

Require: training dataset $\mathcal{D}$, initialization epoch T_c, total epoch T
Ensure: gating vector g, model parameters {E, W}

```
## Searching ##
t=0
while t < T do
    t = t + 1
    while D is not fully iterated do
        Sample a mini-batch from the training dataset
        {E_t, W_t}, g = Searching(D)            ▷ Equation 13
    end while
    if t == T_c then
        {Ê, Ŵ} ⇐ {E_t, W_t}
    end if
end while
g = Discretization({g_c})                       ▷ Equation 14
## Retraining##
Retrain {E, W} given g with {Ê, Ŵ} as initialization ▷ Equation 15
```


experiment

Metrics

为了衡量特征集的大小，使用了一个公式进行归一化，该公式基于每个特征的重要性得分。假设特征分数的分布符合正态分布N(μ, σ)，那么归一化后的特征分数是N(μ/σ, 1)。这个过程可以保证

差。这样，即使两个特征集的规模相差悬殊，也可以公平地进行比较。

Baseline Methods and Backbone Models

我们将提出的OptFS方法与以下特征选择方法进行比较：

(i)AutoField ： 这是基线，利用神经架构搜索技术[+] 在字段级别选择信息性特征；

(ii)LPFS ： 这是基线，设计了一个自定义的光滑-l_0-似函数，在字段级别选择信息性字段；

(iii)AdaFS ： 这是一个基线，通过一个新颖的控制器网络为每个样本选择最相关的特征。

我们在主流的FM、DeepFM、DCN和IPNN模型上应用上述基线。 我们还比较了提出的OptFS方法与一种特征交互选择方法：AutoFIS 。这个基线使用GRDA优化器以一种场级别的方式放弃不重要的特征交互。我们应用AutoFIS在FM和DeepFM后援模型上。我们只比较AutoFIS与FM和DeepFM后援模型，因为原始论文只提供了最优超参数设置并在此设置下发布源代码。

Overall Performance(RQ1)

分别比较了特征选择方法和特征交互选择方法。这两种方法都可以用于解决特征集优化问题。

Feature Selection

首先，与其他基准方法相比，我们的OptFS更有效率和高效。OptFS可以在不显著统计意义的情况下降低特征比例并实现更高的AUC。然而，OptFS带来的益处在不同数据集上有所不同。在Criteo上，OptFS倾向于减少特征集的大小。OptFS可以减少86%至96%的特征，同时改善不足被认为具有统计意义。在Avazu和KDD12数据集上，这种益处倾向于提高性能和减少特征。OptFS可以通过大致使用10%的特征显著增加AUC，而与基线模型相比，其AUC提高了0.01%至0.45%。 同时，其他基准特征选择方法往往会带来性能下降。这可能是因为他们采用了特征字段选择的设计。这种设计不可避免地会丢弃有用的特征或保留无用的特征。 其次，不同的数据集在特征冗余方面表现出不同的行为。与其他基线方法相比，所有数据集上OptFS都能获得更好的结果。

Table 2: Performance Comparison Between OptFS and Feature Selection Methods.

	Method	FM			DeepFM			DCN			IPNN		
		AUC↑	Logloss↓	Ratio↓	AUC↑	Logloss↓	Ratio↓	AUC↑	Logloss↓	Ratio↓	AUC↑	Logloss↓	Ratio↓
Criteo	Backbone	0.8055	0.4457	1.0000	0.8089	0.4426	1.0000	0.8107	0.4410	1.0000	0.8110	0.4407	1.0000
	LPFS	0.7888	0.4604	0.0157	0.7915	0.4579	0.2415	0.7802	0.4743	0.1177	0.7789	0.4705	0.3457
	AutoField	0.7932	0.4567	**0.0008**	0.8072	0.4439	0.3811	**0.8113**	**0.4402**	0.5900	0.8115	**0.4401**	0.9997
	AdaFS	0.7897	0.4597	1.0000	0.8005	0.4501	1.0000	0.8053	0.4472	1.0000	0.8065	0.4448	1.0000
	OptFS	**0.8060**	**0.4454**	0.1387	**0.8100***	**0.4415***	**0.0422**	0.8111	0.4405	**0.0802**	**0.8116**	**0.4401**	**0.0719**
Avazu	Backbone	0.7838	0.3788	1.0000	0.7901	0.3757	1.0000	0.7899	0.3755	1.0000	0.7913	0.3744	1.0000
	LPFS	0.7408	0.4029	0.7735	0.7635	0.3942	0.9975	0.7675	0.3889	0.9967	0.7685	0.3883	0.9967
	AutoField	0.7680	0.3862	**0.0061**	0.7870	0.3773	1.0000	0.7836	0.3782	0.9992	0.7865	0.3770	0.9992
	AdaFS	0.7596	0.3913	1.0000	0.7797	0.3837	1.0000	0.7693	0.3954	1.0000	0.7818	0.3833	1.0000
	OptFS	**0.7839**	**0.3784**	0.8096	**0.7946***	**0.3712***	**0.8686**	**0.7932***	**0.3718***	**0.8665**	**0.7950***	**0.3709***	**0.9118**
KDD12	Backbone	0.7783	0.1566	1.0000	0.7967	0.1531	1.0000	0.7974	0.1531	1.0000	0.7966	0.1532	1.0000
	LPFS	0.7725	0.1578	1.0000	0.7964	0.1532	1.0000	0.7970	**0.1530**	1.0000	0.7967	0.1532	1.0000
	AutoField	0.7411	0.1634	**0.0040**	0.7919	0.1542	0.9962	0.7943	0.1536	**0.8249**	0.7926	0.1541	0.8761
	AdaFS	0.7418	0.1644	1.0000	0.7917	0.1543	1.0000	0.7939	0.1538	1.0000	0.7936	0.1539	1.0000
	OptFS	**0.7811***	**0.1560***	0.5773	**0.7988***	**0.1527***	**0.9046**	**0.7981***	**0.1530**	0.8912	**0.7985***	**0.1530**	**0.7729**

Here * denotes statistically significant improvement (measured by a two-sided t-test with p-value < 0.05) over the best baseline. **Bold** font indicates the best-performed method.

Feature Interaction Selection

首先，与没有进行任何特征交互选择的基线模型相比，AutoFIS和OptFS取得了更高的性能。这种观察表明两种数据集上的无用特征交互的存在。 其次，OptFS和AutoFIS在不同的模型上表现出不同的性能。当特征集中的特征数量较少时，OptFS在FM上的表现几乎与AutoFIS相同，但在DeepFM上的表现却明显更好。这是因为OptFS专注于特征级别的交互，这比AutoFIS使用的字段级别交互更为精细。 最后，OptFS可以在不减少特征的情况下将特征集减少13%至96%，而AutoFIS则在所有可能的特征上进行操作，而不进行任何减少。

	Model	Method	Metrics		
			AUC↑	Logloss↓	Ratio↓
Criteo	FM	Backbone	0.8055	0.4457	1.0000
		AutoFIS	**0.8063**	**0.4449**	1.0000
		OptFS	0.8060	0.4454	**0.1387**
	DeepFM	Backbone	0.8089	0.4426	1.0000
		AutoFIS	0.8097	0.4418	1.0000
		OptFS	**0.8100**	**0.4415**	**0.0422**
Avazu	FM	Backbone	0.7838	0.3788	1.0000
		AutoFIS	**0.7843**	0.3785	1.0000
		OptFS	0.7839	**0.3784**	**0.8096**
	DeepFM	Backbone	0.7901	0.3757	1.0000
		AutoFIS	0.7928	0.3721	1.0000
		OptFS	**0.7946***	**0.3712***	**0.8686**

Transferability Study(RQ2)

我们将研究OptFS结果的可移植性。实验设置如下：首先，我们搜索一个模型的门向量$\mathbf{g}$，将其命名为源模型。然后，我们重新训练另一个基于该门向量的基线模型，将其称为目标模型。我们在Criteo和Avazu两个数据集上研究了DeepFM、DCN和IPNN等基线模型之间的转移性。根据表格中的结果显示，所有的变换都导致性能下降。这种下降甚至在Avazu数据集上被认为是显著的。因此，特征交互操作需要不同特征集来实现高性能。我们可以得出结论，选择特征集需要包含交互操作，进一步强调了选择同时考虑特征及其交互的操作的重要性。

Table 4: Transferability Analysis on Criteo and Avazu.

	Target	Source	Metrics		
			AUC↑	Logloss↓	Ratio↓
Criteo	DeepFM	DeepFM	**0.8100**	**0.4415**	**0.0422**
		DCN	0.8097	0.4419	0.0802
		IPNN	0.8097	0.4418	0.0719
	DCN	DCN	**0.8111**	**0.4405**	0.0802
		DeepFM	0.8106	0.4410	**0.0422**
		IPNN	0.8107	0.4410	0.0719
	IPNN	IPNN	**0.8116**	**0.4401**	0.0719
		DCN	0.8113	0.4404	0.0802
		DeepFM	0.8114	0.4403	**0.0422**
Avazu	DeepFM	DeepFM	**0.7946***	**0.3712***	0.8686
		DCN	0.7873	0.3754	**0.8665**
		IPNN	0.7872	0.3755	0.9118
	DCN	DCN	**0.7932***	**0.3718***	**0.8665**
		DeepFM	0.7879	0.3784	0.8686
		IPNN	0.7860	0.3762	0.9118
	IPNN	IPNN	**0.7950***	**0.3709***	0.9118
		DCN	0.7907	0.3747	**0.8665**
		DeepFM	0.7908	0.3748	0.8686

知乎

我们将进行重训练阶段的ablation研究，其详细说明见第Section 。在第Section中，我们提出了一种定制的初始化方法，即ci。这里我们将它与其他三种获取模型参数的方法进行比较：

（i）wo，即without re-training，直接继承搜索阶段的模型参数；

（ii）ri，即randomly initialize模型参数；

（iii）lt.h，即lottery ticket hypothesis，这是一种常见的用于重训练稀疏网络的方法。

具体来说，它以搜索阶段相同的种子初始化模型参数。实验在Criteo和Avazu基准上针对三个基线模型（DeepFM、DCN和IPNN）进行。根据表中显示的结果，我们可以基于以下观察做出以下结论。

首先，我们可以很容易地看出，无论重训练的设置如何，都能提高性能。不重训练神经网络会从搜索阶段继承劣化的模型参数，这是由门向量中的非二进制元素影响的。重训练在限制门向量的约束下改善了模型性能。 其次，ci始终优于其他两种重训练方法。这种性能差距在Criteo和Avazu的三个基线模型上被认为是显著的。这可能是因为在Avazu数据集上，基线模型通常只在过拟合之前训练一个周期。因此，它进一步增加了在重训练阶段引入自定义初始化的重要性。这个观察验证了CTR预测中引入自定义初始化的必要性。

Table 5: Ablation Study Regarding the Re-training Stage.

	Model	Metrics	Methods			
			w.o.	r.i.	l.t.h.	c.i.
Criteo	DeepFM	AUC↑	0.8012	**0.8100**	**0.8100**	**0.8100**
		Logloss↓	0.4686	0.4416	**0.4415**	**0.4415**
	DCN	AUC↑	0.8077	0.8109	0.8108	**0.8111**
		Logloss↓	0.4522	0.4407	0.4408	**0.4405**
	IPNN	AUC↑	0.7757	0.8113	0.8114	**0.8116**
		Logloss↓	0.4998	0.4404	0.4403	**0.4401**
Avazu	DeepFM	AUC↑	0.6972	0.7873	0.7883	**0.7946***
		Logloss↓	0.5017	0.3754	0.3790	**0.3712***
	DCN	AUC↑	0.7122	0.7870	0.7858	**0.7932***
		Logloss↓	0.4736	0.3801	0.3764	**0.3718***
	IPNN	AUC↑	0.7560	0.7912	0.7910	**0.7950***
		Logloss↓	0.4411	0.3745	0.3745	**0.3709***

Efficiency Analysis(RQ4)

Time Complexity

推理时间对于将模型部署到在线Web系统至关重要。我们将推理时间定义为推理一个批次所需的时间。结果是通过在验证集中对所有批次的推理时间进行平均获得的。 如图所示，OptFS实现的是最短的推理时间。这是因为由OptFS获取的特征集合通常具有最少的特征。同时，AdaFS需要最长的推理时间，甚至比基线模型还要长。这是因为它需要在运行过程中动态确定是否保留或丢弃每个特征。

知乎

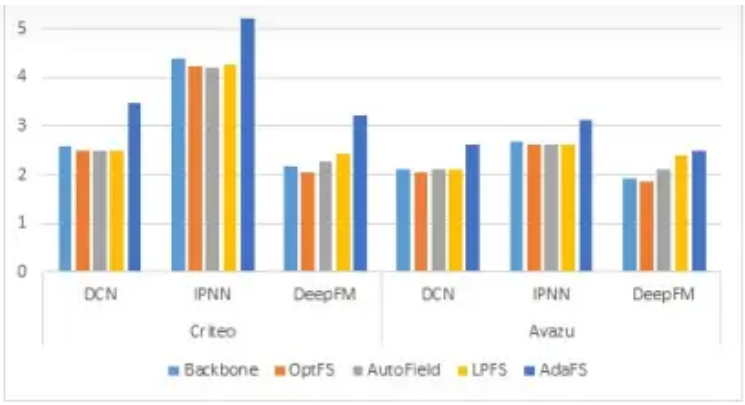

Figure 4: Inference Time of Different Models on Criteo and Avazu Dataset. The Y-axis represents the influence time, measured by ms

Case Study(RQ5)

在图中，我们展示了每字段上的特征比与其互信息的关系。设$\mathbf{z}_i = \{\mathbf{x}_{k_i}\}$和真实标签$\mathbf{y} = (y_1, y_2, \ldots, y_n)$，它们之间的互信息定义如下：

$$I(\mathbf{z}_i; \mathbf{y}) = -\sum_{y=1}^{n} p(y) \log_2 \frac{p(y|\mathbf{z}_i)}{p(y)}$$

其中，前一项是边缘熵，第二项是给定字段$\mathbf{z}_i = \{\mathbf{x}_{k_i}\}$的真实标签$\mathbf{y} = (y_1, y_2, \ldots, y_n)$的条件熵。注意，具有高互信息分数的字段更具有预测性（即更重要）。作为案例研究[+]，我们研究了每个字段上的特征比，如图所示。我们选择了在Criteo数据集上使用DeepFM、DCN和IPNN的结果。图显示了每个字段的互信息得分，这代表了每个字段对预测标签的可解释性。图、图和图显示了给定每个字段的特征比。如可以看见，具有较高互信息得分的字段可能会保留下更多的特征，这表明OptFS从字段角度获得了最佳特征集。

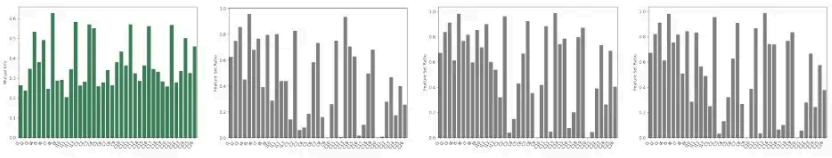

Figure 6: A Case Study of OptFS output on Criteo. In all subfigures, the X-axis indicates the field identifiers. Subfigure (a) plots the mutual information scores, while subfigures (b), (c) and (d) plot the feature set ratio of OptFS on DeepFM, DCN and IPNN.

编辑于 2024-02-21 18:31 · IP 属地北京

ctr预估 特征工程 腾讯

理性发言，友善互动

2 条评论 默认 最新

LuLuYao-YL
论文有吗？有的话，请分享😁
2023-11-23 · 广东 回复 喜欢

SmartMindAI 作者
论文原文《Optimizing Feature Set for Click-Through Rate Prediction》
2023-11-23 · 北京 回复 喜欢