ContextNet: A Click-Through Rate Prediction Framework Using Contextual information to Refine Feature Embedding

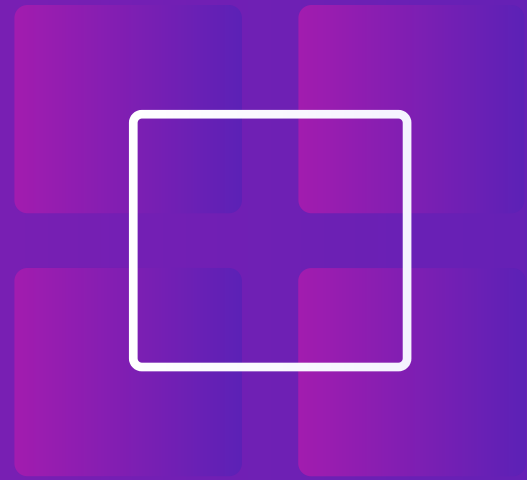
乔梁 2022.7.1

一、相关背景

二、模型结构

三、实验结果

四、学习收获



一、相关背景

CTR

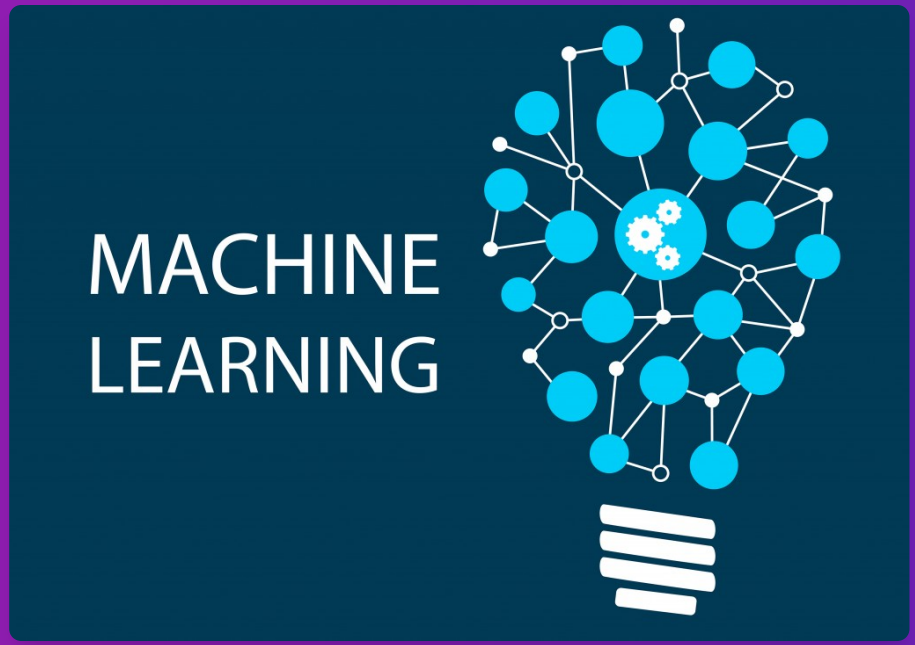
点击率（CTR）估计是`个性化广告`和`推荐系统`中的一项基本任务，对于排名模型有效捕获复杂的高阶特征非常重要。受ELMO和Bert在NLP领域的成功启发，他们根据单词出现的上下文句子信息动态优化单词嵌入，我们认为在CTR估计任务中，根据输入实例中包含的上下文信息逐层动态优化每个特征的嵌入也很重要。通过这种方式，我们可以有效地捕获每个特征的有用特征交互。在本文中，我们提出了一个新的CTR框架 ContextNet，该框架通过根据输入上下文动态细化每个特征的嵌入来隐式地建模高阶特征交互。具体来说，ContextNet 由两个关键组件组成：**上下文嵌入模块**和**ContextNet模块**。

- 上下文嵌入模块从输入实例中聚合每个特征的上下文信息，
- ContextNet 模块逐层维护每个特征的嵌入，并通过将上下文高阶交互信息合并到特征嵌入中来动态细化其表示。

CTR模型

传统模型

- 逻辑回归（LR）、多项式、基于树的模型
- 基于张量的模型、贝叶斯模型、FFM



CTR模型

深度学习模型

深度学习技术在计算机视觉、语音识别和自然语言理解等许多研究领域都取得了很好的成果。因此，使用DNN进行CTR估计也是该领域的一个研究趋势。一些基于深度学习的模型已经被引入并取得了成功，如：

- FNN、AFM、wide&deep
- DeepFM、xDeepFM、DIN等

模型的不足

- FNN和DeepFM，都使用浅层MLP层以隐式方式模拟高阶相互作用，这已被证明是无效的。
- xDeepFM，通过在网络结构中添加子网络，明确引入了高阶特征交互。然而，这将显著增加计算时间，并且很难在实际应用程序中部署它。
- AutoInt和Fi-GNN可以动态地改变特征嵌入，但这些模型的特征表示是一种成对交互的加权求和。这些模型在成对交互后遵循求和方式的特征聚合规则。

二、模型结构

ContextNet框架

如图1所示，我们提出了一个名为ContextNet的新CTR框架，该框架通过根据输入实例中包含的上下文信息动态细化特征嵌入来隐式地建模高阶特征交互。ContextNet由两部分组成：

- 上下文嵌入模块：上下文嵌入模块为每个特征聚合同一实例中的上下文信息，并将收集的上下文信息投影到同一低维特征嵌入所处的空间。请注意，上下文嵌入模块的输入始终来自特征嵌入层。
- ContextNet 模块：ContextNet模块通过先将上下文信息合并到每个特征的特征嵌入中来隐式地建模高阶交互，然后对合并的嵌入进行非线性变换，以更好地捕获高阶交互。我们可以逐块堆叠ContextNet以形成更深的网络，前一块的细化特征嵌入输出是下一块的输入。不同的ContextNet模块具有相应的上下文嵌入模块来优化每个特征的嵌入。最后一个ContextNet模块的输出被馈送到预测层，以给出实例的预测值。

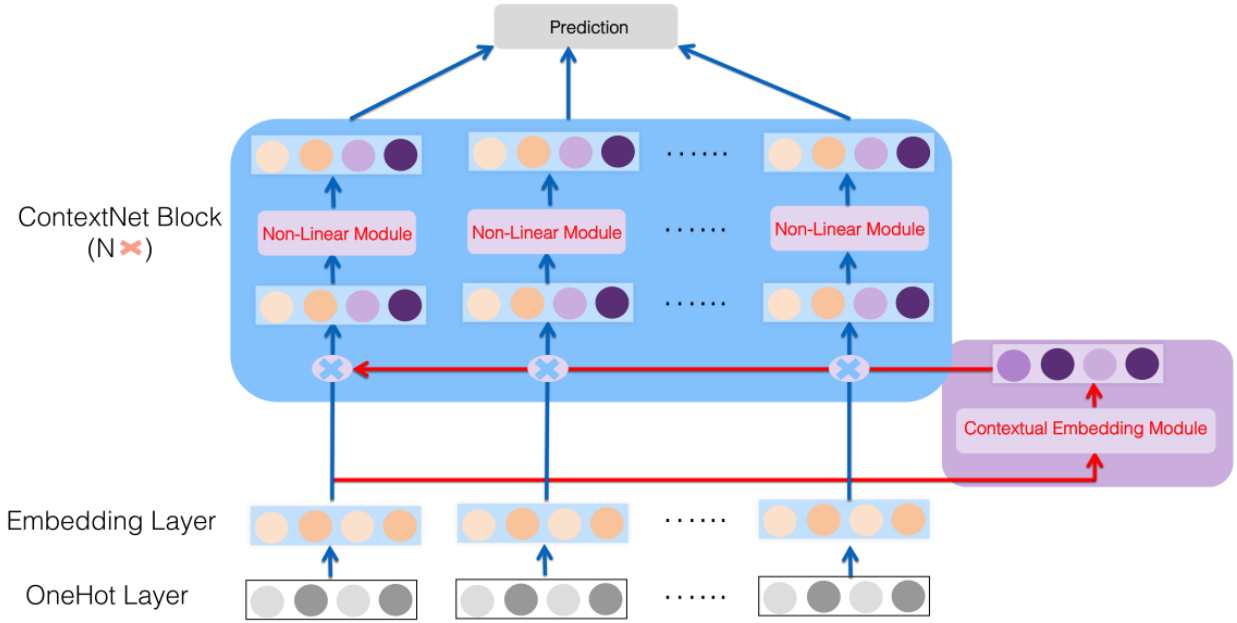


Figure 1: The Neural Structure of ContextNet Framework

特征嵌入

CTR任务的输入数据通常由稀疏和密集的特征组成。这些特征被编码为一个热向量，这通常会导致大型词汇的高维特征空间。这个问题的常见解决方案是引入嵌入层。通常，稀疏输入可以表示为：

$$x = [x_1, x_2, \ldots, x_f]$$

其中 f 表示字段的数量， $x_i \in \mathbb{R}^n$ 表示具有 n 个特征的分类字段 。我们可以获得独热向量 x_i 的特征嵌入 E_i ：

$$E_i = W_e x_i$$

其中 $W_e \in \mathbb{R}^{k \times n}$ 是 n 个特征的嵌入矩阵，k 是嵌入维数。数值特征 x_j 也可以通过以下方式转换为相同的低维空间：

$$E_j = V_j x_j$$

其中$V_j \in \mathbb{R}^k$ 是维度为 k 的对应字段嵌入。 通过上述方法，在原始特征输入上应用嵌入层，将其压缩为低维稠密实值向量。嵌入层的结果是一个广泛的串联向量：

$$E = \text{concat}(E_1, E_2, \ldots, E_i, \ldots, E_f)$$

上下文嵌入

ContextNet中的上下文嵌入模块有两个目标：

- 首先，ContextNet使用该模块来聚合输入实例中每个特征的上下文信息，即特征嵌入层。
- 其次，将收集到的一个特征的上下文信息投影到与特征嵌入所在的相同的低维空间。

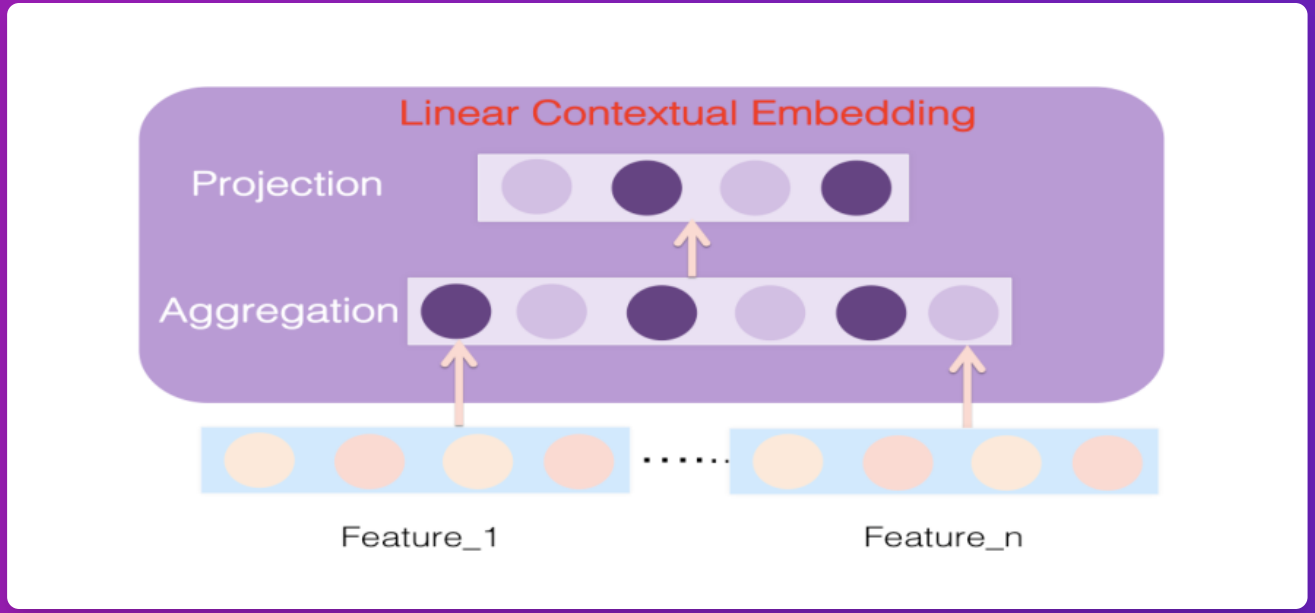
我们可以将此过程表述如下：

$$CE_i = \mathcal{F}_{\text{project}}\left(\mathcal{F}_{\text{agg}}\left(E_i, E; \Theta_a\right); \Theta_p\right)$$

其中 $CE_i \in \mathbb{R}^k$ 表示第 i 个特征的嵌入 E_i 的上下文嵌入，k 是字段嵌入的维度，$\mathcal{F}_{\text{agg}}\left(E_i, E; \Theta_a\right)$ 是使用嵌入层 E 和特征嵌入 E_i 作为输入的第 i 个特征字段的上下文信息聚合函数，Θ_a 表示聚合模型的参数。$\mathcal{F}_{\text{project}}\left(\mathcal{F}_{\text{agg}}; \Theta_p\right)$ 是将上下文信息投影到同一低维空间的映射函数，特征嵌入就在其中。Θ_p 表示投影模型的参数。

为了使该模块更加具体，本文提出了一种用于该模块的两层上下文嵌入网络（TCE）。我们将前馈网络调整为聚合函数 $\mathcal{F}_{\text{agg}}\left(E_i, E; \Theta_a\right)$ 和投影函数 $\mathcal{F}_{\text{project}}\left(\mathcal{F}_{\text{agg}}; \Theta_p\right)$。需要注意的是，TCE只是该模块的一个特定解决方案，还有其他选项值得进一步探索。上下文嵌入模块的输入应该来自包含原始和全局上下文信息的嵌入层。

Figure2 Two Layer Contextual Embedding



上下文嵌入

接下来，我们将描述上下文嵌入网络是如何工作的。假设我们有一个属于特征字段 d 的特征 E_i。如图2所示，TCE模块中使用了两个完全连接的（FC）层。第一个FC层被称为“聚合层”，这是一个相对较宽的层，用于从带有参数 W_d^a 的嵌入层收集上下文信息。第二个FC层是“投影层”，它通过特征嵌入将上下文信息投影到相同的低维空间中，并将维数降低到与特征嵌入相同的大小。投影层的参数为 W_d^p 。正式表示如下：

$$CE_i = \mathcal{F}_{\text{project}}\left(\mathcal{F}_{\text{agg}}\left(E_i, E; \Theta_a\right); \Theta_p\right) = W_d^p\left(RELU\left(W_d^a E\right)\right) \tag{6}$$

其中 $E \in \mathbb{R}^{m=f\times k}$ 表示输入实例的嵌入层， $W_d^a \in \mathbb{R}^{t\times m}$ 和 $W_d^p \in \mathbb{R}^{k\times t}$ 分别是字段 d 的TCE中聚合层和投影层的参数。t 和 k 分别表示聚合层和投影层的神经元数量。聚合层通常比投影层宽，因为投影层的大小要求等于特征嵌入大小 k。聚合层越宽，模型的表达能力就越强。

上下文嵌入

我们可以从公式（6）中看到，每个特征字段 d 分别为聚合层和投影层维护自己的参数 W_d^a 和 W_d^p。假设在嵌入层中有 f 个不同的特征字段，TCE的参数量将为 $f \times (W_d^a + W_d^p)$。我们可以通过在所有字段之间共享聚合层中的 W_d^a 来减少参数量。TCE的参数量将减少到 $W_d^a + f \times W_d^p$。

为了减少模型参数，我们采用了以下策略：在所有特征字段之间共享聚合层参数，同时保持每个特征字段的投影层参数私有。该策略有效地平衡了模型的复杂性和模型的表达能力，因为私有投影层将使每个特征独立地为其目的提取有用的上下文信息。这使得TCE模块看起来像多任务学习中的"共享底部"结构，其中底部隐藏层在任务之间共享。

ContextNet 模块

ContextNet 模块用于通过合并为该功能生成的上下文嵌入来动态优化每个功能的嵌入，以隐式捕获高阶功能交互。为了实现这一目标，ContenxtNet 模块中有两个相应的过程，如图1所示：嵌入合并和以下非线性转换。我们可以逐块堆叠ContextNet以形成深度网络，前一块的输出就是下一块的输入。

接下来，我们将描述ContextNet 模块是如何工作的。我们使用 E_i^l 来表示第 l 个block的输出特征嵌入，也就是说，E_i^l 是第 l 个 block 的第 i 个特征的输入嵌入。CE_i^{l+1} 表示TCE的第 l 个 block 中为第 i 个特征计算的相应上下文嵌入。我们可以将第 i 个特征的此过程描述如下：

$$E_i^{l+1} = \mathcal{F}_{\text{non-linear}}\left(\mathcal{F}_{\text{merge}}\left(E_i^l, CE_i^{l+1}; \Theta_m\right); \Theta_n\right)$$

其中，$E_i^{l+1} \in \mathbb{R}^k$ 表示第 $l+1$个 ContextNet 模块为 第 i 个特征 E_i^l 输出的微调特征嵌入， k 是字段嵌入的维数，$\mathcal{F}_{\text{merge}}\left(E_i^l, CE_i^{l+1}; \Theta_m\right)$是第 i 个特征的合并函数，第 i 个特征使用前一个模块的输出特征嵌入 E_i^l 和当前块中的上下文嵌入 CE_i^{l+1} 作为输入。 Θ_m 表示合并函数的参数。$\mathcal{F}_{\text{non-linear}}\left(F_{\text{merge}}; \Theta_n\right)$ 是映射函数，用于对合并嵌入进行非线性变换，以进一步捕获第 i 个特征的高阶交互。Θ_n 表示非线性变换函数的参数。

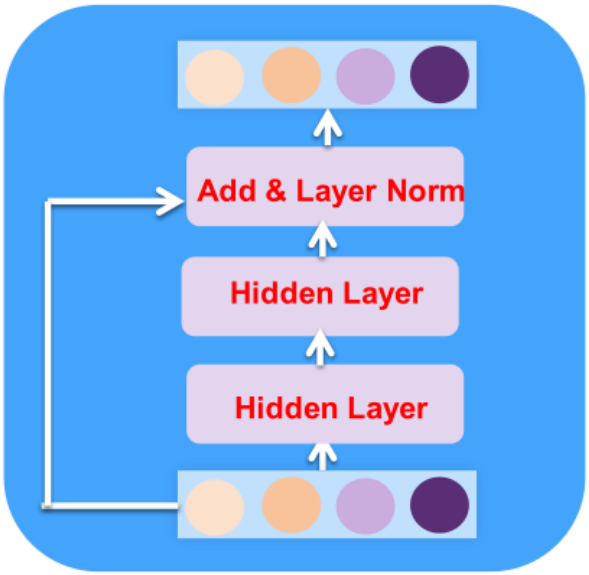
ContextNet Block

对于合并函数 $\mathcal{F}_{\text{merge}}\left(E_i^l, CE_i^{l+1}; \Theta_m\right)$，本文使用Hadamard乘积将特征嵌入 E_l^l 和相应的上下文嵌入 CE_i^{l+1} 合并，如下所示：

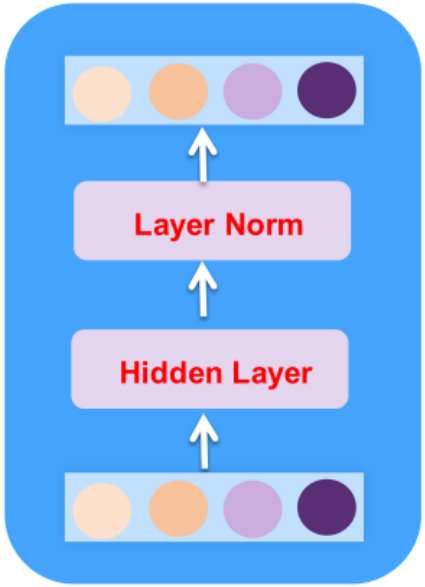
$$E_i^l \otimes CE_i^{l+1} = \left[E_{i1}^l \cdot CE_{i1}^{l+1}, \ldots, E_{ij}^l \cdot CE_{ij}^{l+1}, \ldots, E_{ik}^l \cdot CE_{ik}^{l+1}\right]$$

其中，k 是第 i 个特征的嵌入向量 E_i^l 和上下文嵌入 CE_i^{l+1} 的大小。对于非线性函数$\mathcal{F}_{\text{non-linear}}\left(F_{\text{merge}}; \Theta_n\right)$，我们提出了两种神经网络，如图3所示：逐点前馈网络和单层前馈网络。

Figure3 Structure of Non-Linear Transformation



Point-Wise FFN



Single-Layer FFN

逐点前馈网络

虽然ContextNet使用上下文嵌入模块聚合所有其他功能的嵌入，然后将其投影到固定的嵌入大小，但最终它仍然是一个线性模型。为了赋予模型非线性，以便更好地捕捉高阶相互作用，我们可以对所有 E_i 应用逐点两层前馈网络（在所有特征之间共享参数）：

$$\mathcal{F}_i = \mathrm{PFFN}\left(E_i; \Theta_n\right) = \mathrm{LN}\left(\mathrm{RELU}\left(E_i W^1\right) W^2 + E_i\right)$$

其中 W_1，W_2是 $k \times k$ 矩阵，k 是嵌入的维数。我们还采用了残差连接和层规范化。FFN 引入的额外参数数为 $W_1 + W_2$ ，由于FFN中的参数共享，该数目不太大。具有此版本FFN的ContextNet在本文的以下部分称为“ContextNet PFFN”。

单层前馈网络

通过减少剩余连接和ReLU非线性，我们提出了另一种更简单的单层前馈网络。我们可以将此转换应用于具有共享参数的所有 E_i ，如下所示：

$$\mathcal{F}_i = FFN\left(E_i; \Theta_n\right) = LN\left(E_i W^1\right)$$

其中，W^1是$k \times k$ 矩阵，并且偏置也被丢弃。虽然从映射函数中删除了ReLU，但层归一化将为高阶特征交互带来非线性。FFN引入的额外参数数为 W_1，由于FFN中的参数共享，该值很小。在本文的以下部分，我们将此版本的FFN称为ContextNet“ContextNet SFFN”。虽然与ContextNet PFFN相比，映射形式要简单得多，但ContextNet SFFN在许多数据集中具有相当甚至更好的性能。

不同的 ContextNet 模块具有相应的上下文嵌入模块，当我们堆叠多个块以形成更深的网络时，可以细化每个特征的嵌入。我们可以通过在每个 ContextNet 模块的TCE模块之间共享聚合层或投影层的参数来进一步减少模型参数。

预测层

综上所述，我们给出了模型输出的总体公式：

$$\hat{y} = \delta\left(w_0 + \sum_{i=1}^{f*k} w_i x_i\right)$$

其中，$\hat{y} \in (0,1)$ 是CTR的预测值，σ 是sigmoid函数，f 是特征字段数目，k是特征嵌入大小，x_i 是最后一个ContextNet块输出的所有特征嵌入向量的位值，w_i是每个位值的学习权重。

对于二分类问题，损失函数是对数损失：

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log\left(1 - \hat{y}_i\right)$$

其中，n 是训练实例的总数，y_i 是第 i 个实例的真实值，$\hat{y}$ 是预测的CTR。优化过程旨在最小化以下目标函数：

$$\mathfrak{L} = \mathcal{L} + \lambda\|\Theta\|$$

其中 λ表示正则化项，Θ 表示参数集。

ContextNet的可解释性

与LR等简单模型相比，DNN模型因缺乏可解释性而臭名昭著，因为广泛使用的MLP层引入了非线性。与大多数DNN模型相比，ContextNet的一个优点是它具有良好的模型解释性。最后一个 Context 模块的输出保持了每个特征的嵌入，这些特征融合了有用的高阶信息，ContextNet 的预测层实际上是一个LR模型。因此，在给定输入实例的情况下，我们可以轻松计算每个特征的权重和对最终预测分数的贡献，如下所示：

$$FW_j = \sum_{i=1}^{k} w_i E_{ji} \tag{14}$$

其中 $FW_j \in \mathbb{R}$ 是实例中第 j 个特征的权重分数。$E_j \in \mathbb{R}^k$ 是最后一个ContextBlock输出的W特征的嵌入，w_i 是位 E_{ji} 在预测层的学习权重。k 是特征嵌入的大小。正分数得到正标签，负分数得到负标签。

ContextNet的可解释性

对于一个特定的实例，我们可以检测到一些重要的特征，这些特征可以解释为什么该实例具有公式（14）所示的最终预测标签。我们还可以通过累积或平均分数计算整个训练集级别上的特征重要性，如下所示：

$$FW_j = \sum_{i=1}^{m} \left| FW_j^i \right| \tag{15}$$

$$FW_j = \left(\sum_{i=1}^{m} \left| FW_j^i \right| \right) / (n + \alpha) \tag{16}$$

其中$FW_j \in \mathbb{R}$是整组第 j 个特征的权重得分。FW_j^i是实例 i 中第 j 个特征的得分，训练集的大小为 m。此处采用绝对值，因为得分为正或负。N 是出现特征的实例集的大小，α是减少低频特征影响的范数。我们可以根据公式（16）找出重要特征，或者丢弃分数较小的不重要特征，根据公式（15）对模型进行压缩。

三、实验结果

实验细节

- 框架：Tensorflow
- 优化方法：最小批量为1024的Adam
- 学习率：0.0001。
- 所有模型的嵌入维数：10
- Hidden Layer：3
- 每层神经元数量：400
- 激活函数：ReLU。
- 除特别提及外，对ContextNet的默认设置如下：特征嵌入为10，默认模型有3个ContextBlocks，TCE模块的隐藏层大小为20。我们使用2个 Tesla $K40$ GPU进行实验。

研究问题

我们在四个真实数据集上评估了所提出的方法，并回答了以下研究问题：

- Q1：提出的方法是否比现有的基于深度学习的CTR模型表现更好？
- Q2：ContextNet的训练效率是多少？
- Q3：ContextNet体系结构中各种组件的影响是什么？
- Q4：网络的超参数如何影响ContextNet的性能？
- Q5：ContextNet真的可以逐步完善功能嵌入以捕获功能交互吗？那么特征重要性计算呢？

模型效率

为了减少线性上下文嵌入模块的参数，我们可以为不同的ContextNet Block共享TCE模块中的参数。我们有以下三种策略：

- 在每个块的相应TCE模块之间共享聚合层的参数（share-A）；
- 共享聚合层和投影层的参数（share-A&P）；
- 我们不共享参数（不共享任何内容），如果没有特别提及，这是所有其他实验的默认设置。

我们进行了一些实验，以探索不同的参数共享策略对模型性能的影响。在两个数据集上，无共享策略和共享A策略的效果是可比较的。但是，如果在聚合层和投影层共享参数，性能会大大降低。这表明，对于TCE模块来说，为不同ContextNet块的每个细化特征提取不同的高阶交互信息对于模型的良好性能至关重要。与无共享策略相比，Share-A可能是一个更好的选择，因为它具有较少的参数，并且可以保持良好的模型性能。

模型效率

为了比较不同模型的模型效率，我们使用每个 epoch 的运行时间作为评估指标。

DNN和DeepFM被视为效率基线，因为它们的网络结构相对简单，在许多实际应用中得到了广泛应用。

与所有其他模型相比，xDeepFM更耗时，这意味着xDeepFM很难在许多实际场景中应用。至于我们提出的ContextNet模型的训练效率，我们可以看到ContextNet PFFN和ContextNet SFFN都比AutoInt和xDeepFM具有更快的训练速度。如果我们共享所提出的两个ContextNet模型的聚合层参数，则可以进一步提高训练速度。带有Share-A策略的ContextNet SFFN模型的运行速度可能略慢于基线模型，这意味着ContextNet对于现实世界的应用程序来说足够有效。

四、学习收获

学习收获

- 公式书写、表达的规范性；
- 实验的完整性；
- 模型的准确性+可解释性。

Thank you