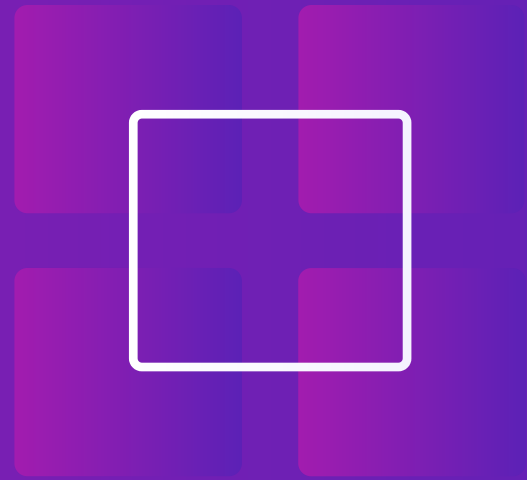
CAN: Feature Co-Action for Click-Through Rate Prediction

乔梁 2022.7.29

一、引言
二、架构
三、实验
四、总结
五、学习收获



一、引言

相关工作

一些研究致力于CTR预测中的模型特征交互。这些方法可以分为几类：聚合方法、基于图的方法和基于因子分解的方法。

- **Aggregation based methods**，基于汇聚的方法，DIN，DIEN，MIMN等都属于这类方法。基于汇聚的方法主要聚焦于如何从**用户历史行为序列」中汇聚得到比较有区分性的表征。为了进行有效的汇聚，这类方法通过建模目标item和历史序列item之间的co-action，并作为历史序列行为的权重**从而实现加权汇聚。也就是说，Co-Action在这类方法中仅仅是作为历史行为的**权重**，并用于**信息的汇聚**(information aggregation)。

- **Graph based methods**， 基于图的方法，GCN，HINE等。这类方法将特征作为结点，并相互连接形成图。此时，特征的co-action是作为**边的权重**来指导信息的传递，即：作为权重系数用于加权汇聚邻域结点的信息。仍然是用于**信息的汇聚**(information aggregation)。

- **Combinatorial embedding methods**， 基于组合嵌入的方法，如FM，DeepFM，PNN等。这类方法通过显式地组合和交叉不同的特征来建模co-action效应。这实际上是一种**信息的增强** (information augmentation)，而不是信息的汇聚。例如：PNN在两两特征之间使用内积或者外积来增强输入。但是这种方法的缺点在于，模型要同时负责**单一特征的表征学习**，又要负责**两两特征之间的co-action建模**，这二者之间实际上是会**有矛盾**的，会互相干扰和影响。即：特征co-action是通过单一特征的嵌入来间接表达的，单一特征的学习会影响特征co-action的学习。归根结底，主要是因为特征**co-action的学习过程不是独立的。**

问题

要对特征交互进行建模，最简单的方法是使用笛卡尔积。给定两个特征A和B，一旦选择了特征A和B，协同效应将被视为一个新特征并反馈到模型中。由于以附加输入的形式直接提供协同效应，因此训练过程变得更容易。笛卡尔乘积虽然简单有效，但它存在一些严重的缺陷，如参数量和特征空间大，泛化能力差。

与引入额外输入信息的笛卡尔乘积不同，一些研究工作致力于通过输入特征的仔细组合进行模型特征交互。

典型的例子是基于因子分解的方法，它们通过将潜在向量空间中的特征嵌入与各种算子直接结合来强调低阶和/或高阶特征交互。由于这些方法从模型角度考虑特征交互，并且算子设计良好，因此与笛卡尔乘积相比，它们通常更易于部署。然而，它们仍有一些不容忽视的缺点。遗憾的是，一些基于因子分解的方法的浅结构限制了其代表性。

更关键的是，由其操作生成的嵌入同时承担表示学习和特征交互建模的责任，这可能会阻碍训练过程。

这种组合降低了特征交互的记忆能力，从而降低了模型容量。

问题

为了解决这些问题，我们提出了协同网络（CAN），该网络可以捕捉特征交互，并有效利用不同特征对的相互和公共信息。具体来说，对于每个特征对，一侧（induction feature）的嵌入用于构造应用于共同作用单元中另一侧（feed feature）的MLP。

与笛卡尔积相比，这种特征交互建模范式将附加参数从$O(N^2 \times D)$减少到 $O(N \times (D' + D))$，N是特征的数量，D和 D'是协同作用单元中使用的参数的维数。此外，与传统的基于因子分解的方法相比，在协同作用单元中使用MLP可以提供更强大的拟合能力和更好的非线性。此外，可以区分表示学习和特征交互建模的参数空间，以避免训练中的相互干扰。进一步利用多阶增强和多级独立性来丰富CAN的表达能力。

贡献

这项工作的主要贡献总结如下：

- 我们研究了笛卡尔积的有效性，并指出了隐式特征交互建模的潜力。受笛卡尔积独立编码的启发，我们设计了一种新的特征交互范式，该范式与笛卡尔积具有相当的性能，但资源消耗更少。
- 我们提出了协同网络（CAN）来模拟输入阶段原始特征之间的特征交互。CAN中的每个特征ID将分发到单个微型MLP，以模拟与其他特征的交互。这样可以提高有限参数下建模特征交互的表达能力。与基于因子分解的方法中使用的普通算子相比，单个微MLP具有更强大的表达能力。
- 我们在公共和工业数据集上进行了广泛的实验。与其他最先进的竞争对手相比，这种一致的优势验证了CAN在建模特征交互方面的有效性。CAN的部署为阿里巴巴的显示广告系统带来了12%的点击率和8%的每千次展示收入提升。

二、架构

特征交互

在广告系统中，通过以下方式计算用户 u 点击广告 m 的预测CTR $\hat{y}$：

$$\hat{y} = \mathrm{DNN}\left(E\left(u_1\right), \cdots, E\left(u_I\right), E\left(m_1\right), \cdots, E\left(m_J\right)\right)$$

其中 $u = \{u_1, \ldots, u_I\}$ 是一组用户特征，包括浏览和点击历史记录、用户配置文件特征等，$m = \{m_1, \ldots, m_J\}$ 是一组项目特征。用户和项目特征通常是唯一的ID。$E(\cdot) \in \mathbb{R}^d$ 表示大小为 d 的嵌入，该嵌入将稀疏ID映射为可学习的稠密向量，作为DNN的输入。除这些一元项外，以前的工作将交互作用建模为二元项：

$$\hat{y} = \mathrm{DNN}\left(E\left(u_1\right), \cdots, E\left(u_I\right), E\left(m_1\right), \cdots, E\left(m_J\right), \left\{F\left(u_i, m_j\right)\right\}_{i \in [1,\cdots,I], j \in [1,\cdots,J]}\right)$$

其中，$F(u_i, m_j) \in \mathbb{R}^d$ 表示用户特征 u_i 和项目特征 m_j 之间的交互。由于特征交互的存在，该模型可以受益于特征交互，如“啤酒和尿布”示例所示。因此，如何有效地建模特征交互对于提高性能至关重要。

特征交互

仔细回顾以前的方法后，可以发现它们要么以特征交互作为权重，要么同时学习与其他目标的隐含相关性，这可能会产生不满意的结果。学习特征交互的最直接方法是将特征组合视为新特征，并直接学习每个特征组合的嵌入向量，例如笛卡尔积。笛卡尔积提供了独立的参数空间，因此具有足够的灵活性来学习协同作用信息，以提高预测能力。

然而，也存在一些严重的缺陷。

- 第一个是参数爆炸问题。尺寸为 N 的两个特征的笛卡尔乘积的参数空间将从 $O(N \times D)$ 扩展到 $O(N^2 \times D)$，其中 D 是嵌入的维数，这将给在线系统带来很大的负担。
- 此外，由于笛卡尔乘积将<A，B>和<A，C>视为完全不同的特征，因此组合之间没有信息共享，这也限制了表示能力。

考虑到笛卡尔乘积的优点和计算的服务效率，我们引入了一种新的特征交互建模方法。对于每个特征对，其笛卡尔乘积产生一个新特征和相应的嵌入。由于不同的特征对可能共享相同的特征，因此任何两个特征对之间都存在隐含的相似性，笛卡尔乘积忽略了这一点。如果可以有效地处理隐式相似性，则可以用比笛卡尔乘积更小的参数规模更有效地建模这些对之间的特征交互。在本文中，受笛卡尔乘积独立编码的启发，我们首先区分嵌入和特征交互的参数，以避免相互干扰。考虑到DNN具有强大的拟合能力，我们设计了一个协同单元，以微网络的形式参数化特征嵌入。由于不同的特征对可以共享同一个微网络，相似信息自然地学习并存储在该微网络中。

图2：从笛卡尔积到 CAN 的演变

其中A、B、C、D表示四种特征。N_A，N_B，N_C和N_D分别表示特征A、B、C、D的数量。h是特征嵌入的尺寸，d是来自 Co-Action Unit 的输出的尺寸。在此图中，我们使用特征A与其他三个特征进行交互。

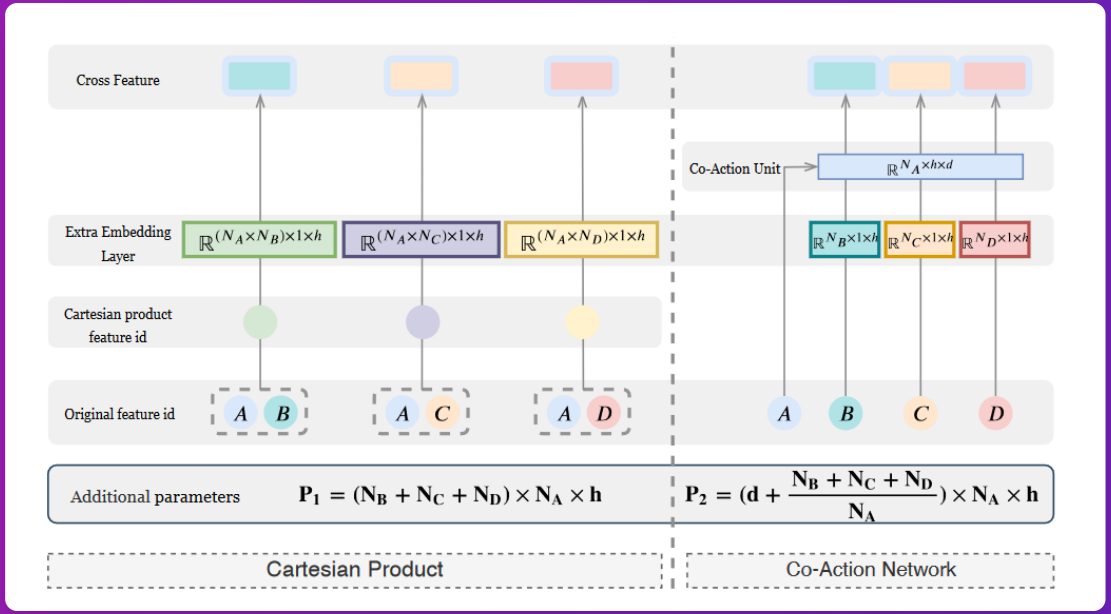
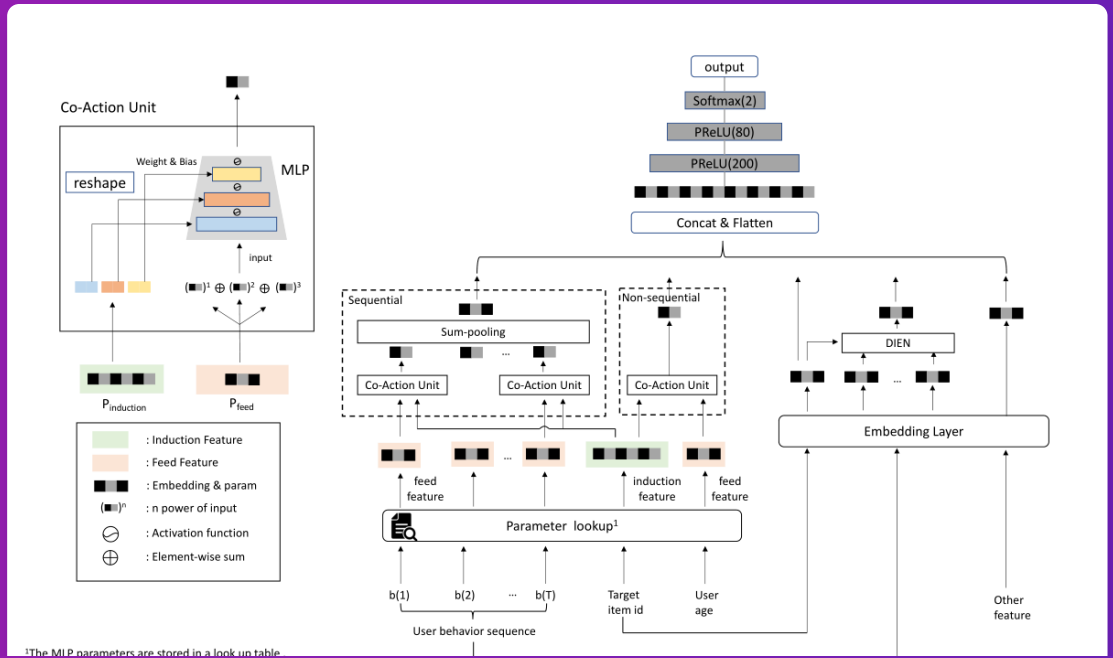


图3：Co-Action Network的总体框架

给定目标项和用户特征，嵌入层将稀疏特征编码为稠密嵌入。一些选定的特征分为两侧 $P_{induction}$ 和 P_{feed}，它们是 Co-Action Unit 的组成部分。$P_{induction}$ 参数化微MLP，P_{feed}作为输入。共同作用单元的输出，以及公共特征嵌入，用于进行最终的CTR预测。



CAN

CAN的整体架构如图3所示。用户和目标项目的特征 U 和 M 以两种方式输入CAN。

- 在第一种方式中，使用嵌入层对稠密向量 $E\left(u_1\right), \cdots, E\left(u_I\right)$ 和 $E\left(m_1\right), \cdots, E\left(m_J\right)$ 进行编码，并进一步分别连接为 e_{item} 和 e_{user}。
- 在第二种方式中，我们从 U 和 M 中选择子集 U_{feed} 和 $M_{induction}$，用我们提出的 Co-Action Unit 来建模交互特征 $\left\{F\left(u_i, m_j\right)\right\}_{u_i \in U_{\text{feed}}, m_j \in M_{\text{induction}}}$。 CAN 的公式如下：

$$\hat{y}=\operatorname{DNN}\left(e_{item}, e_{user},\left\{F\left(u_i, m_j\right)\right\}_{u_i \in U_{\text{feed}}, m_j \in M_{\text{induction}}} \mid \Theta\right)$$

其中 Θ 表示模型中的参数，$\hat{y} \in [0,1]$是点击行为的预测概率。真实点击信息表示为 $y \in \{0,1\}$. 我们最终最小化预测 $\hat{y}$ 和标签 y 之间的交叉熵损失函数：

$$\min_{\Theta} -y \log(\hat{y})-(1-y) \log(1-\hat{y})$$

Co-Action Unit

一般来说，协同单元是每个特征对的独立MLP，即具有特征对提供的MLP的权重、偏差和输入的微MLP。对于特定的用户特征ID $u_0' \in U_{feed}$，我们使用参数查找来获得可学习的参数 $P_{induction} \in \mathbb{R}^{D'}$，而项目特征ID $m_0 \in M_{induction}$用于 $P_{feed} \in \mathbb{R}^{D} (D < D')$。接下来，对 $P_{induction}$ 进行 reshape，并将其分解为微MLP的权重矩阵和偏差向量。这个过程可以公式化为：

$$\|_{i=0}^{L-1} (w_i \| b_i) = P_{\text{induction}}$$

$$\sum_{i=0}^{L-1} (|w_i| + |b_i|) = |P_{\text{induction}}| = D'$$

其中 w_i 和 b_i 表示microMLP的第 i 层的权重和偏差，|| 表示串联操作，L 确定微MLP的深度，$|\cdot|$获取变量的大小。这一过程的直观说明如图3的左半部分所示。

Co-Action Unit

然后将P_{feed} 馈入微MLP，并通过串联各层的输出来实现特征交互：

$$
\begin{aligned}
h_0 &= P_{\text{feed}} \\
h_i &= \sigma\left(w_{i-1} \otimes h_{i-1} + b_{i-1}\right), \quad i = 1, 2, \cdots, L \\
F\left(u_{o'}, m_o\right) &= H\left(P_{\text{induction}}, P_{feed}\right) = \|_{i=1}^{L} h_i
\end{aligned}
$$

其中 $\otimes$ 表示矩阵乘法，σ 表示激活函数，H表示具有向量输入 $P_{induction}$ 和 P_{feed} 的协同作用单元，而不是输入为特征 u_0'和 m_0 的原始符号 F.

对于像用户行为历史 $P_{seq} = \left\{P_{b(t)}\right\}_{t=1}^{T}$ 这样的序列特征，协同作用单元应用于每个点击行为，然后在序列上进行 sum pooling：

$$
H\left(P_{\text{induction}}, P_{\text{seq}}\right) = H\left(P_{\text{induction}}, \sum_{t=1}^{T} P_{b(t)}\right)
$$

Co-Action Unit

在我们的实现中，$P_{induction}$ 从项目特征中获取信息，而 P_{feed} 从用户特征中获取信息。然而，P_{feed} 也可以作为微MLP的参数，反之亦然

根据经验，在广告系统中，候选项目是所有项目的一小部分，因此其数量小于用户点击历史中的项目数量。因此，我们选择 $P_{induction}$ 作为微MLP参数，以减少总参数，从而使学习过程更容易和更稳定。

注意，微MLP层的数量取决于学习的难度。根据经验，较大的特征尺寸通常需要更深的MLP。

与其他方法相比，所提出的协同作用单元至少有三个优点。

1. 首先，与以往在不同类型的场间相互作用中使用相同潜在向量的工作不同，协同作用单元利用微观MLP的计算能力，动态耦合两个分量特征 $P_{induction}$ 和 P_{feed}，而不是固定模型，这提供了更大的容量来保证两个场特征的解纠缠更新。
2. 其次，需要学习较小规模的参数。例如，考虑具有两个 N ID的两个特征，其笛卡尔乘积的参数比例应为 $O(N^2 \times D)$，其中 D 是嵌入的维数。然而，通过使用协同作用单元，该比例将减少到 $O(N \times (D' + D))$，其中 D' 是协同作用单元中 $P_{induction}$ 的维数。更少的参数不仅有利于学习，而且可以有效地减轻在线系统的负担。
3. 第三，与笛卡尔乘积相比，协同作用单元对新的特征组合具有更好的泛化能力。给定一个新的特征组合，只要之前训练了它们的两侧嵌入，协同单元仍然可以工作。

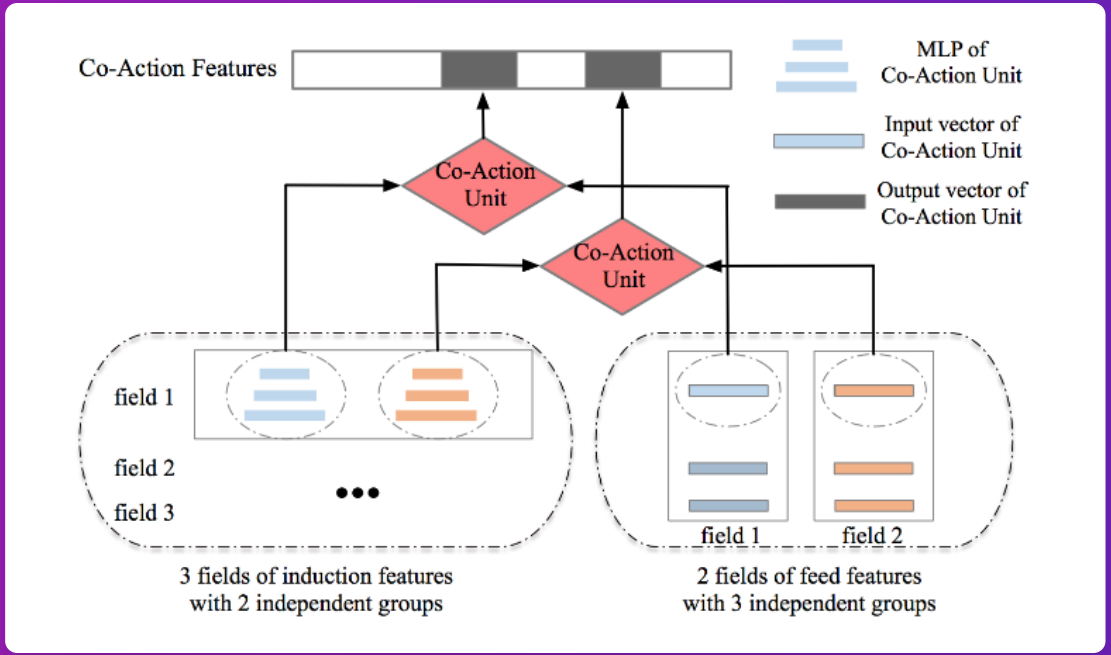
多阶增强

上述特征是在一阶特征的基础上形成的。然而，可以在高阶上估计特征交互。尽管考虑到微MLP的非线性，协同作用单元可以隐式学习高阶特征交互，但由于特征交互的稀疏性，学习过程被认为是困难的。为此，我们明确引入多阶信息以获得多项式输入。这是通过将微MLP应用于不同的 P_{feed} 阶来实现的：

$$H_{\text{Multi-order}}\left(P_{\text{induction}}, P_{\text{feed}}\right)=\sum_{c=1}^{C} H\left(P_{\text{induction}},\left(P_{feed}\right)^{c}\right)$$

其中 C 是阶数。我们利用 $Tanh$ 来避免由高阶项引起的数值问题。多阶增强有效地提高了模型的非线性拟合能力，而不会带来额外的计算和存储成本。

图4：组合独立性的说明


Co-Action Features
MLP of
Co-Action Unit
Input vector of
Co-Action Unit
Output vector of
Co-Action Unit
Co-Action
Unit
Co-Action
Unit
field 1
field 2
field 3
field 1
field 2
3 fields of induction features
with 2 independent groups
2 fields of feed features
with 3 independent groups

多层次独立性

学习独立性是特征交互建模的主要关注点之一。为了保证学习的独立性，我们从不同的方面提出了三级策略。

1. 参数独立，必须使用。如第4.2节所述，我们的方法解决了表示学习和特征交互建模的更新问题。参数独立性是CAN的基础。即每个特征都有两种初始的向量，1种是用于表征学习的，即前文提到的 e ；1种是用于co-action建模的，即前文提到的 P 。这两个参数是独立的。

2. 组合独立，推荐使用。随着特征组合数量的增加，特征交互呈线性增长。根据经验，目标项目特征（如“item_id”和“category_id”）被选为导入侧，而用户特征用于提要侧。由于一个 feed 侧微MLP可以与多个 induction 侧相结合，反之亦然，因此我们的方法可以轻松地将模型的表达能力成倍地扩大。我们在图4中说明了这一想法。形式上，如果 feed 侧和 induction 侧分别有 Q 组和 S 组，则特征交互的组合应满足：

$$|P_{\text{induction}}| = \sum_{s=1}^{S} \sum_{i=0}^{L_s-1} (|w_i(s)| + |b_i(s)|)$$

$$|P_{\text{feed}}| = \sum_{q=1}^{Q} |x(q)|$$

其中 $|x(q)|$ 是第 q 个微MLP的输入尺寸。在正向传递中，进给特征分为几个部分来实现每个微MLP。

多层次独立性

3. 阶数独立，可选。为了进一步提高多阶输入中特征交互建模的灵活性，我们的方法对不同阶进行了不同的 feed 侧嵌入。然而，这些嵌入的尺寸相应地增加了 C 倍。

多级独立性有助于特征交互建模，但同时也带来了额外的内存访问和计算。独立性水平和部署成本之间存在权衡。根据经验，模型使用的独立性级别越高，它需要的训练数据就越多。在我们的实际系统中，使用了三个独立级别，但由于缺乏训练样本，在公共数据集中只使用了参数独立性。

三、实验

工业部署

如前几节所述，笛卡尔积是特征交互建模中最直接的方法。然而，笛卡尔乘积通常会导致大量的资源消耗。一方面，模型尺寸将以极快的速度扩展。超大模型给存储和网络传输带来了巨大挑战，进一步影响了模型的实时更新。另一方面，随着特征在输入阶段的增加，它增加了应用程序请求中的嵌入查找操作，这导致了系统响应的延迟。

现有方法在工业部署中更友好。然而，我们也注意到，在数十亿数据的规模下，与笛卡尔积相比，提升非常有限。同时，简单地增加参数空间（如扩大嵌入大小）并没有带来额外的改进。

在这种情况下，CAN被设计为一种新的特征交互建模方案。在我们的广告系统中，选择了21种特征来构建特征交互，包括6种广告特征（例如，ad_id、item_id、shop_id等）和15种用户特征（例如，item_history、shop_history等）。我们注意到，与只有十分之一模型大小的笛卡尔积相比，可以实现相当的性能。

工业部署

考虑到21种特征，由于特征交互独立性，可以分配额外的21个嵌入。由于用户特征大多是长度超过100的行为序列，因此需要额外的内存访问，这会导致很大的响应延迟。此外，特征交互的计算成本根据特征组合的数量线性增长，这也给我们的系统带来了相当大的响应延迟。为了充分发挥CAN的所有能量，人们付出了许多努力来减少响应延迟。在工业上，我们可以从三个方面进行优化：

- 序列截止。16种用户特征的长度从50到200不等。我们巧妙地应用序列截止以减少内存成本，例如，所有长度为200的用户行为序列都被截断为长度50。保留最新的行为。序列截止将QPS（每秒查询数）提高了20%，但AUC降低了0.1%，这是可以接受的。
- 组合减少。6个广告功能和15个用户功能可以获得多达90个组合，这是一个沉重的负担。从经验上看，同类广告和用户特征的组合可以更好地模拟共现。根据这一原则，我们保留了“item_id”、“item_click_history”、“category_id”、“category_click_history”等组合，并删除了一些不相关的特征。场景中心RPM主页广告+11.4%+8.8%购后广告+12.5%+7.5%组合。这样，特征组合的数量从90减少到48，从而带来30%的QPS改进。

工业部署

- 计算内核优化。特征交互计算是指形状为$[batch_size \times M \times dim_in \times dim_out] \times [batch_size \times M \times T \times dim_in]$的$P_{induction}$和$P_{feed}$之间耗时的大矩阵乘法，其中$M$、$T$、dim_in和dim_out分别表示特征交互的数量、用户行为序列的长度、MLP输入和输出维度。在我们的例子中，dim_in和dim_out不是常用的形状，因此BLAS（基本线性代数子程序）无法很好地优化此类矩阵乘法。为了解决这个问题，重写了内部计算逻辑，从而提高了60%的QPS。此外，我们进行了内核融合，将多个操作（如Matmul和Tanh）组合为一个操作，以减少GPU内存I/O消耗。通过这样做，避免了矩阵乘法输出的中间GPU内存写入，这又带来了47%的QPS提升。

一系列优化使CAN能够在我们的广告系统的主要流量中稳定地在线服务。在我们的实践中，CAN的CTR预测步骤大约需要10毫秒。

四、总结

总结

在本文中，我们强调了特征交互建模的重要性，这是以前的工作没有充分探讨的。受笛卡尔积的启发，我们提出了一种新的特征交互范式，使用一种特殊设计的网络，协同网络（CAN）。

通过一个灵活的模块，协同作用单元，可以将表示学习和特征交互建模分离开来。此外，在协同单元中引入了多阶增强和多级独立性，进一步提高了特征交互建模的能力。实验表明，该算法的性能优于以往的工作。CAN已经部署在阿里巴巴的展示广告系统中，并服务于主要流量。我们相信，这项工作推动了特征交互学习向前迈进了一步，未来将进一步探索多特征和轻量级交互建模。

五、学习收获

学习收获

1. 学习到了一种新的特征交互方式，思考作者提出的“为特征交互走出一条新路”；
2. 模型部署上线的优化细节。

Thank you