xDeepFM理论与实践

# 背景

特征工程是很多预测任务成功与否的关键，然而人工特征工程具有以下缺陷：

1）重要的特征都是与应用场景息息相关的，挖掘出高质量的交互特征需要非常专业的领域知识，针对每一种应用场景，工程师们都需要首先花费大量时间和精力深入了解数据的规律之后才能设计、提取出高效的高阶交叉特征，因此人力成本高昂，在大型的推荐系统中，原生特征是海量的，手动挖掘交叉特征几乎不可能，此外，人工无法挖掘出肉眼不可见的交叉特征；

2）原始数据中往往包含大量稀疏的特征，例如用户和物品的ID，交叉特征的维度空间 是原始特征维度的乘积，因此很容易带来维度灾难的问题；

3）人工提取的交叉特征无法泛化到未曾在训练样本中出现过的模式中（即无法泛化到未曾在训练样本中出现的特征）。

因此自动学习特征间的交互关系是十分有意义的。于是，业界探索和实践了诸多基于因子分解机（FM）与DNN相结合的CTR预估模型。如Wide & Deep、Deep & Cross Network、DeepFM等，这些模型可以自动学习学习低阶和高阶组合特征。但是这种交互会学习所有的交叉特征，其中肯定会包含无用的交叉组合，无用的交叉特征会引入噪音并降低模型的表现。此外学习出的是隐式的交互特征，即并不知道模型进行了哪些特征交叉，也没有一种公式可以明确推论出最终学习出来的交叉特征到底是多少维的，其形式是未知的、不可控的；同时它们的特征交互是发生在元素级（bit-wise）而不是特征向量之间（vector-wise），这一点违背了因子分解机的初衷。来自Google的团队在KDD 2017 AdKDD&TargetAD研讨会上提出了DCN模型，旨在显式（explicitly）地学习高阶特征交互，其优点是模型非常轻巧高效，但缺点是最终模型的表现形式是一种很特殊的向量扩张，同时特征交互依旧是发生在元素级上。

特征交叉是一种重要的变换方式，计算交叉积可以称为 cross features或者 multi-way features。下面总结了之前有关特征交叉相关模型的优缺点，并引出xDeepFM。

1. FM： FM模型中的每个特征的embedding向量中的每一个元素，可以看作bit。经典的FM模型可以引入任意高阶的交互特征，但是这种交互会学习所有的交叉特征，其中肯定会包含无用的交叉组合，无用的交叉特征会引入噪音并降低模型的表现；
2. AFM：建模所有特征的交互关系有可能引入无用的特征，甚至给模型带来噪音；
3. FNN：在DNN之前使用FM预训练好的field embedding，可以学习高阶特征；
4. PNN：在embedding层和DNN层之间有product层，而且不需要依赖预训练。但是PNN和FNN都有共同的缺点：都聚焦在高阶交互特征上，而对低阶交互特征关注较少；
5. Wide&Deep：通过引入浅层模块和深层模块的组合结构，使得学习过程具有记忆性和泛化性，可以同时学习到低阶和高阶的交互特征，Wide部分的特征依然需要人工特征工程，Deep部分的特征交叉是隐式的；
6. DeepFM：通过引入浅层模块和深层模块的组合结构构，可以同时学习到低阶和高阶的交互特征，Deep部分的特征交叉是隐式的；
7. DCN：捕捉有界阶数的交互特征，但其本质是特征的线性变换，这限制了特征表达；
8. xDeepFM：以vector-wise形式来显式生成特征交互，使用CIN模块来代替DCN中的cross网络模块，CIN可以显式的学习交互关系，并且随着网络的加深，特征交互关系的阶数也在变大。并且仿照Wide&Deep和DeepFM的组合式网络结构，xDeepFM组合了显式高阶交互模块、隐式高阶交互模块。

可以看到xDeepFM相比于DeepFM模型多了一个显式高阶交互模块。该显式高阶交互模块是在vector-wise层面建模的。

|  |
| --- |
| 注：  在xDeepFM出现之前可以完成特征自动交叉的模型存在的主要问题是：虽然这些模型可以完成低阶和高阶特征交叉，学习任意函数的表示，但是这些特征交叉都是隐式的特征交叉，无法得知其最大特征交叉阶数为多少，并且是所有特征都会交叉，这会带来噪音，影响模型的效果。 |

# 简介

xDeepFM（eXtreme Deep Factorization Machine）模型是2018年由中科大、北邮、微软研究院在KDD上联合提出的模型。xDeepFM主要是针对DCN的Cross Network的特征交叉是bit-wise level，对其进行了改进，提出了一种新型的网络Compressed Interaction Network(压缩交互网络，简称CIN)，CIN可以像Deep & Cross Network(DCN)那样，随着Cross Network层数的递增，特征的交叉维度也会递增，并弥补了DCN存在的一些不足。在xDeepFM中，使用CIN替换DCN中的Cross Network，CIN是显示的高阶特征交互，并且是vector-wise level。CIN层的特征交叉学习部分是xDeepFM的核心，被称为艺术般的交叉，CIN提出的目的就是提高特征交互的能力。

CIN与CNN和RNN结构具有一些共性，xDeepFM将CIN与DNN结合，不仅可以显式的生成一定阶数内的特征交互关系，而且可以隐式的生成任意低阶和高阶的特征交互。

xDeepFM模型不仅能同时以显式和隐式的方式自动学习高阶的特征交互，使特征交互发生在向量级(vector-wise)，还兼具记忆与泛化的学习能力。其主要贡献在于，基于vector-wise的模式提出了新的显式交叉高阶特征的方法，并且与DCN一样，能够构造有限阶交叉特征。虽然xDeepFM在名称上与DeepFM相似，但其主要对比的是DCN模型。

paper中作者提到的三个主要贡献(其中前两个也是xDeepFM模型相比于其他模型的最主要特点和优势)：

1. 有效结合了显式高阶交互模块、隐式高阶交互模块，不需要人工特征工程；
2. 设计了CIN模块可以学习显式高阶交互，并且特征交互阶数往后每一层都会增加，以vector-wise方式代替普通DNN的bit-wise方式；
3. 在3个数据集上对比了其他的算法，证明了xDeepFM模型结构的有效性。

# 特征交互

论文提出两个概念的特征交叉：bit-wise level和vector-wise level。

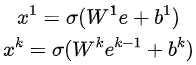
bit-wise level指的是神经网络中节点之间的交互。如DCN模型的Cross Network交互类型就是bit-wise level。bit-wise最早运用于FM模型中，通过特征隐向量之间的点积来表征特征之间的交叉组合。由FM的衍生模型，尤其是引入DNN模块后，常见的做法是，将embedding之后的特征向量拼接到一起，然后送入后续的DNN结构模拟特征交叉的过程，这属于bit-wise level。这种方式与vector-wise的区别在于，各特征向量concat在一起成为一个向量，抹去了不同特征向量的概念，后续模块计算时，对于同一特征向量内的元素会有交互计算的现象出现。

vector-wise level指的是embedding向量之间的特征交叉（例如FM、DeepFM、PNN的特征交叉），vector-wise level中特征交叉参与运算的最小单位为向量，且同一隐向量内的元素并不会有交叉乘积。vector-wise的方式其实在之前介绍的PNN、NFM、AFM与DeepFM中都有使用，但是并没有单独拎出来说明清楚。

xDeepFM主要是针对DCN的Cross Network的特征交叉是bit-wise level，对其进行了改进，使用CIN对Cross Network进行替代，CIN是显示的高阶特征交互，并且是 vector-wise level。

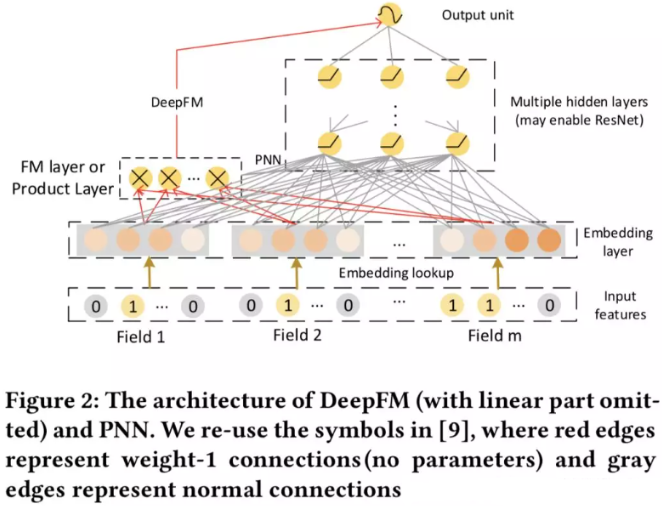
## 隐式特征交互

FNN、DCN、Wide&Deep等模型都是在field embedding向量e上使用前馈神经网络充分学习高阶特征交互的信息，前向的计算公式如下：

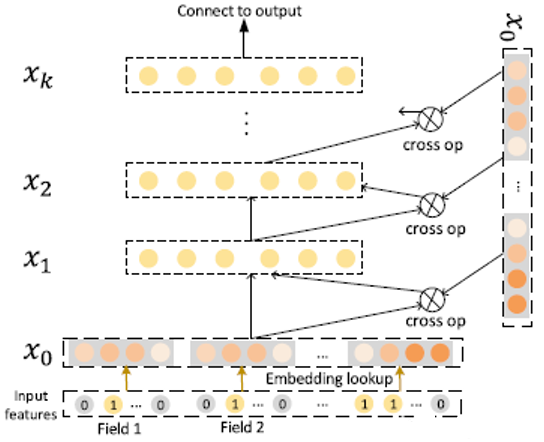


其中，k表示神经网络层的深度，σ表示激活函数，表示第k层神经网络的输出，这与Figure 2中的深度网络部分比较相似（当然不包括FM层或者Product层），这种前向神经网络的模型架构是以bit-wise的方式来建模交互关系的，也就是说，即使是同一个field embedding向量内的bit元素也会互相影响。

而FNN、DeepFM模型对上面的前向神经网络的模型架构进行了略微的调整，除了保持原有的前向DNN结构之外，还会加入一个two-way的交互层，因此在FNN、DeepFM模型中同时存在bit-wise和vector-wise的特征交互关系，而FNN、DeepFM结构的主要区别在于FNN是将product的输出连接到DNN上，而DeepFM是将FM层直接连接到输出单元上。



## 显式特征交互



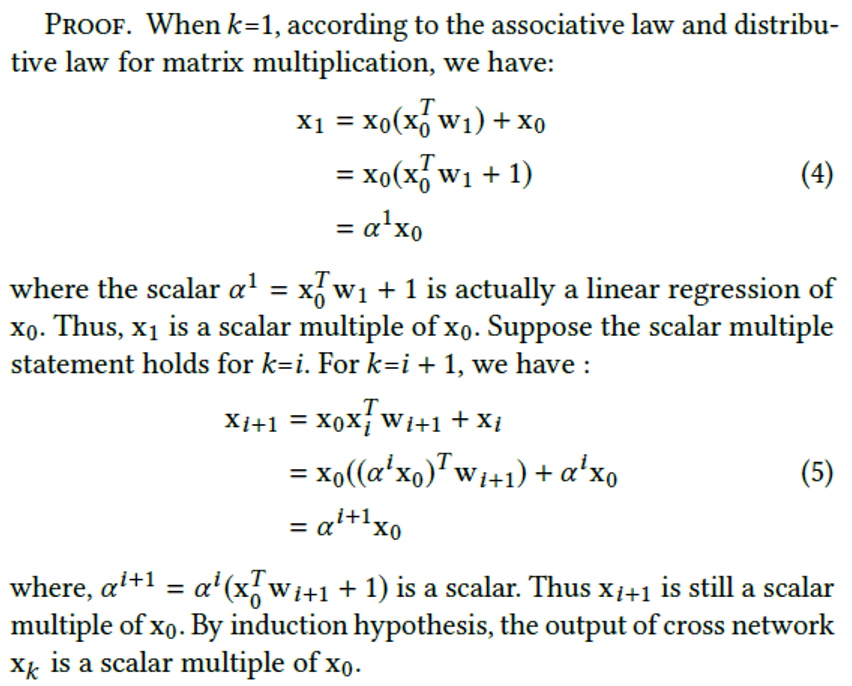
Deep Cross Network结构图

在如上图所示的DCN模型中，显式的建模了高阶特征，不像前向神经网络只能隐式的建模高阶特征，DCN建模高阶特征公式如下：



其中w，b和x分别表示网络的weights，bias和output，k表示第k层。

Cross Network可以显示地学习到高维的特征组合，但它存在一个问题，就是最终的xk 结果是x0乘以一个标量(scalar)。



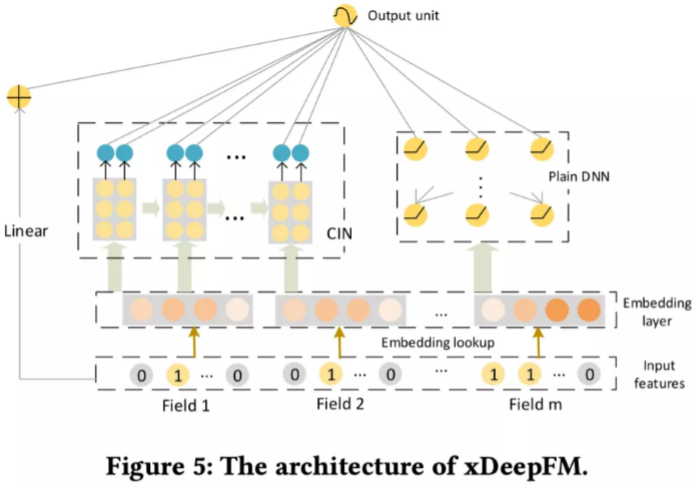
虽然xk结果是x0乘以一个标量，但是并不意味着x0和xk是线性关系的，因为是关于x0敏感的，对于不同的x0，这个标量是不一样的。 因此，Cross Network具有如下局限性：

* CrossNet的输出只能是指定的形式，即是关于x0的标量倍数；
* 特征交叉也是也是以bit-wise形式得到的。

# xDeepFM

将CIN和DNN结合起来，构造一个更强大的模型，这就是xDeepFM。

整体架构图如下所示：



从上图中可以发现xDeepFM可以细分为三大块，一块是基于底层特征的线性部分，一块是基于特征Embedding的DNN部分(implicit feature interactions，隐式特征交互)以及CIN部分(Explicit feature interactions，显示特征交互)。xDeepFM将线性模块、CIN模块、DNN模型三者组合起来互为补充，分别提供低阶特征、显式高阶特征、隐式高阶特征。下面分别阐述这三块的作用的机理。

## Linear 层

直接使用原生特征（one-hot编码）进行二分类任务。

这个部分可以是一个简单的LR，将原始特征（未经过Embedding）作为输入，表示为：



其中的a就是原始特征输入，是激活函数。

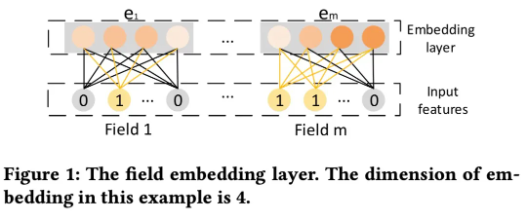
## Embedding Layer

在推荐系统中，不像cv和nlp领域的数据具有时空关联的特性，通常在大规模级别的推荐系统中，输入特征通常是高维离散特征，因此，对于一个任务中具有多个离散特征的情况也是比较常见的。

与所有经典的基于神经网络的推荐相同，Embedding层都是将多个领域（field）组成的高维稀疏分类特征通过神经网络嵌入到低维密集特征。这也可以看作是一种向量（vector-wise）之间的交互。Embedding通常有两种情况：

* 单个特征，例如特征名为‘性别’，该特征对应的embedding可以直接用于DNN的输入；
* 复合特征，或者说交叉特征，例如[user\_id=s02,gender=male, organization=msra,interests=comedy&rock]这种交叉特征，对于每个特征user\_id、gender、organization、interests都有他们自身对应的embedding，一般是计算交叉特征中单个特征embedding的sum pooling结果用于DNN的输入。

Embedding的具体过程如下图所示：



图中的 Field 1 是单个特征，而 Field m 应该是复合特征。

|  |
| --- |
| 注意：  在xDeepFM论文中作者将每个原始特征分别当做一个field，例如原始特征一共有10个，那么该数据集的fields数量为10。这里的field跟FFM模型中的field是一个意思;  paper中每一条样本为为一个instance。 |

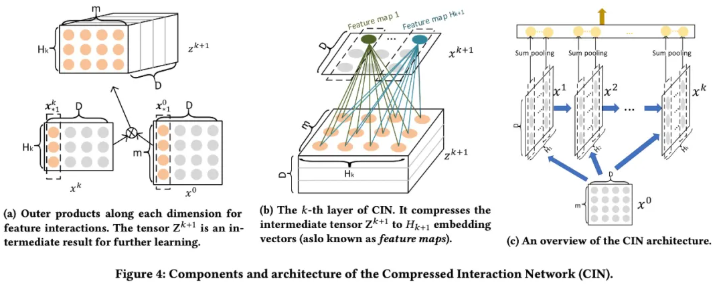
Embedding层的输出结果一般是表示成如下的wide concatenated vector形式，假设有m个field,最终我们将得到：



其中m表示特征（filed）的数量，D表示原始特征embedding后的维度。因此尽管每个instance的特征数量是不一样的（具体样本中有些特征取值为空），但是最终每个instance的wide concatenated vector的长度都是一样的，即m \* D。

## CIN(Compression Interaction Network)

整体如下图所示：

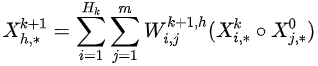


图(a)和(b)表示了如何从第k层的隐藏层（）和层（）来产生下一层隐藏层的（），图示所示计算方法是为了更好的展现为什么模型有CNN的思想，先通过X0和Xk的第i列做一个outer product(matrix multiplication)得到一个的矩阵(0<=i<D), 然后W就像是CNN中的filter,来过滤产生每个feature map的第i列，这样CNN中的"compressed"在CIN中就指代矩阵压缩为矩阵。

与DCN的bit-wise fashion不同的是，xDeepFM的CIN中采用vector-wise fashion对特征向量进行交叉。将embedding矩阵表示为，第i行表示第i个Field对应的特征embedding 向量，共有m个D维的embeddding vector。

CIN是多层输出结构，每层都会产出一个中间结果矩阵，将第k层产出的矩阵表示为，其中表示第k层（embedding）特征向量的数量，且对应的。

对于每一层的向量矩阵是通过下面的方式计算的：



其中，，表示filed的数量，是超参数，需要自定义，表示 中第h行特征向量，表示CIN第k层输出矩阵的第i行特征向量，表示原始特征矩阵第j行特征向量。是用于计算第(k+1)层输出中的第h行向量的参数矩阵，因此是一个标量值，表示第i行第j列位置的参数，即对哈达玛积计算标量倍数。运算∘表示哈达玛积，即两个相同维度的向量的相同位置的元素相乘的值，作为结果中相同位置处的输出元素，结果仍为向量，例如⟨a1, a2, a3⟩◦⟨b1,b2,b3⟩ = ⟨a1b1, a2b2, a3b3⟩。最终。也就是说第(k+1)层的第h个特征向量是由第k层的每一个特征向量与第0层的每一个特征向量进行Hadamard乘积然后乘上一个系数矩阵，最后全部相加得到的。所以说特征交叉是显示的。如果用T表示网络深度，那么CIN最高可以实现(T+1)阶特征交叉。

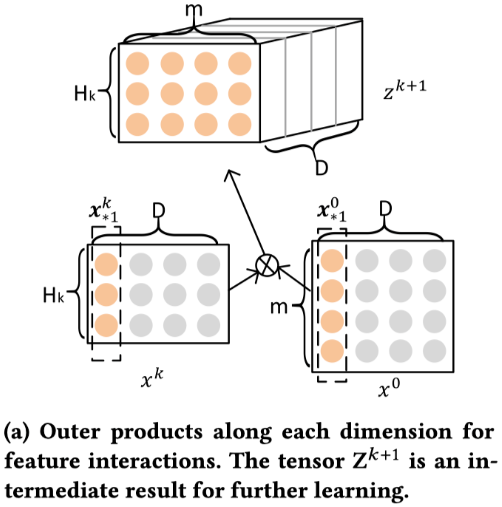
从计算原理上进行理解，上面的公式首先将上一层（第k层）输出的特征矩阵的特征向量（），与原始特征矩阵的特征向量（），两两之间进行哈达玛积。因为上一层有个特征向量，原始特征矩阵有m个特征向量，所以运算完成会得到个向量。将得到的全部向量结果根据参数进行加权求和，得到第(k+1)层的结果。

我们换个视角观察上述计算过程，中每一行特征向量都是与特征交叉之后加权求和得到的。中任意两个特征向量的不同点在于，计算过程中的加权求和参数不同。所以可以将的计算拆成2个步骤：

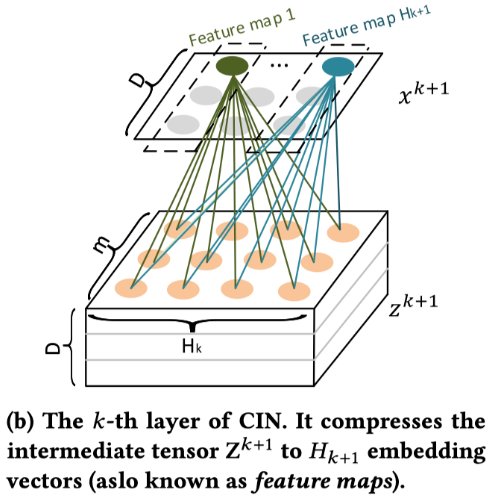
1）将与中的特征向量两两交叉计算，得到中间结果，其中包含个D维特征向量；

2）使用不同的加权参数对中每个特征向量进行加权求和，得到；

步骤1）中，计算的过程可以看做是矩阵与的向量外积计算，如下图（a）所示，得到的。这种网络结构非常类似于RNN结构，即下一层输出的结果取决于上一层输出的结果和一个额外的输入，而且我们在每层中都是用这样的结构，因此特征交互关系就是在vector-wise水平上得到的。



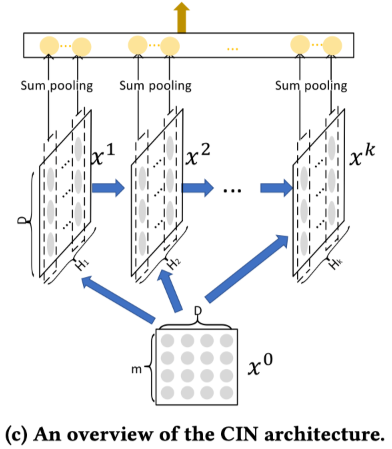
步骤2）通过不同的参数矩阵对中的向量进行加权求和，这个过程类似于CNN中的卷积操作。具体的，可以将整个看做图片，视为不同的过滤器（filter，卷积核），每个过滤器沿着Embedding维度（D）滑动，并与进行一次卷积计算得到一个D维的 feature map 向量，其中D为的通道数。最终通过次不同的卷积之后，便得到隐藏层向量，这被称为一个特征映射（feature map）。通过卷积操作对上层特征矩阵进行压缩编码，这也是CIN中“compress”的由来。计算过程如图（b）所示：



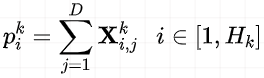
得到每层的输出之后，通过sum pooling操作将所有的特征矩阵进一步压缩为向量，作为最终的输出。

|  |
| --- |
| 为什么这么设计呢，好处是什么？CIN与DCN中Cross层的设计动机是相似的， Cross层的input也是前一层与输出层。这种做法的优点是：有限高阶、自动叉乘、参数共享。  CIN与DCN中Cross的几个主要差异：   1. DCN中的Cross是bit-wise的，而CIN是vector-wise的； 2. 在第k层，DCN中的Cross包含从1阶～（k+1）阶的所有组合特征，而CIN只包含（k+1）阶的组合特征。相应的DCN中Cross在输出层输出全部结果，而CIN在每层都输出中间结果。   其中造成差异2的原因是，DCN中的Cross层计算公式中除了与CIN一样包含“上一层与输入层的x乘”外，会额外“+输入层”，这是涵盖所有阶特征的不同策略。 |

|  |
| --- |
| 注意：  X\_k是通过X\_{k-1}和X0计算得到的，因此特征交互关系是通过显性的计算的并且特征交互的阶数随着层数的增加也在加深。 |



整个CIN的结构如上图c所示，为了能显示利用到每一层的交叉特征,最后需要将每一层的交叉特征输出，但是如果我们直接全部输出的话，可能会带来一个较大的问题，就是特征太多了，后面再接入全连接层的话会占据更多的内存和计算资源。如上图c所示，表达CIN的整体结构，用T表示CIN的深度，每一个隐藏层和输出单元都具有关联，中含有个D维的feature map，在第k个隐层，对于每个feature map进行sum pooling操作之后得到一个标量值，即：



因此，对于第k个隐层，可以得到poolng之后的长度为的向量：



同理，对于CIN中的每一个中间层都产出一个向量，将k层所有向量进行拼接，得到：



即CIN的输出是除了X0以外每一层的feature map的sum pooling横向拼接的结果。

如果我们直接使用CIN用于二分类，那么输出单元就是一个在上的sigmoid节点：

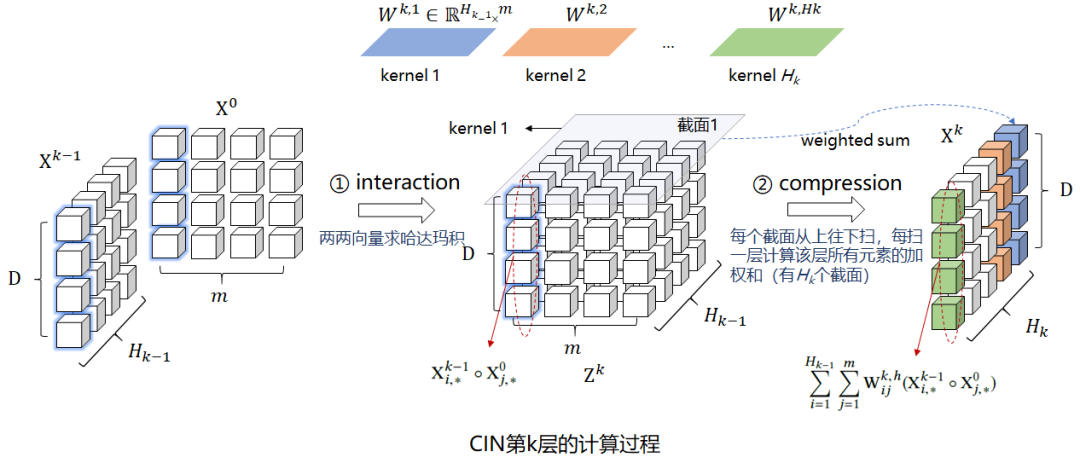


其中，是长度为的参数向量。

从上述分析可以看出，xDeepFM的CIN模块在k层仅包含了(k+1)阶交叉特征。对比DCN模型中，在k层包含了从1到k阶全部交叉特征。这两种方式殊途同归，都能够实现有限阶特征交叉。

|  |
| --- |
| 注：文中没有提到如何来确定的大小，其实是一组超参数。 |
| 由CIN的结构，可以思考两个问题：   1. 每层通过sum pooling对vector的元素加和输出，这么做的意义与合理性是什么？可以设想，如果CIN只有1层，只有m个vector，即H1=m，且加和的权重矩阵恒等于1，即，那么sum pooling的输出结果就是一系列的两两向量内积之和，即标准的FM（不考虑一阶偏置） 2. 除了第一层，中间层的这种基于vector的高阶组合有什么物理意义？回顾FM，虽然是二阶的，但是可以扩展到多阶。例如考虑三阶FM，是对三个嵌入向量做Hadamard乘积，再对得到的vector做sum，CIN基于vector-wise的高阶组合，再做sum pooling与之是类似的，这也是模型名字（xDeepFM）的由来。 |

为什么说CIN特征交叉建模是vector-wise的呢？个人认为论文里的图片和公式，并没有很直观的体现出vector-wise。所以，如果我们把图a和b所示的张量在三维空间里旋转一下，用另一种角度来看，或许能帮助更直观的理解。



上面这张图我们也把CIN第k层的计算拆解成两步，第一步是对和中的两两embedding向量做哈达玛积，得到一个中间张量，这一步实际就是对两两特征做交互，所以称为interaction。第二步是使用个截面（类似CNN中的“kernel”），每个截面对中间张量从上往下扫，每扫一层计算该层所有元素的加权和，一个截面扫完D层得到的一个embedding向量（类似CNN中的“feature map”），个截面扫完得到个embedding向量。这一步实际是用kernel将中间张量压缩成一个embedding向量，所以称此过程为compression。这也是整个子网络被命名为Compressed Interaction Network（CIN）的原因。从这两步计算过程可以看到，特征交互是在vector-wise级别上进行的（哈达玛积），尽管中间经过压缩，每一个隐层仍然保留着embedding vector的形式，所以说CIN实现的是vector-wise的特征交叉。

### CIN与RNN和CNN的关系

1. CIN与RNN的关系

CIN中下一层的输出都依赖于当前最后一个隐藏层输出的结果和一个额外的输入（embedding），而且每层都是用这样的结构，和RNN是非常相似的。

2. CIN与CNN的关系

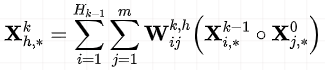
我们发现第(k+1)层的每一个新的向量都是由第k层的所有向量以及第0层的所有向量分别进行element-wise 相乘得到的，因此可以将看做一幅图像，图像的大小为(H\_k \* H\_0 \* D)，把权重矩阵看做filter，使用filter与其进行操作得到下一层的新向量，最终将(H\_k \* H\_0)压缩为了Hk个向量，这也是compressed名字的由来。

|  |
| --- |
| 注意：  filter size：可以将m和H\_k看做filter的宽和高，D看做filter的层数（深度）;  filter的元素：每一层中的每个元素为，然后将其复制成D层的深度即可。 |

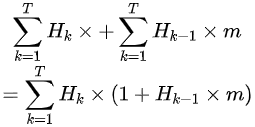
### CIN复杂度分析

#### 空间复杂度

CIN部分主要的学习参数就是：



计算第k层的第h（）个embedding需要个参数，而第k层共有个embedding vec（即个），所以第k层参数量为。假设CIN有T层，那么共有个参数。最后将各层输出池化拼接，需要个参数进行加权求和。所以，CIN部分所有参数和为：



因此CIN的空间复杂度与每个特征的embedding向量的维度D的大小无关。

对于普通的DNN，第一层参数量为，后续层参数为，最后将所有输出进行加权求和，需要参数为。所以全部参数量为。

不妨假设每一层的，那么CIN的空间复杂度为，普通DNN的空间复杂度为。因此，DNN的参数会随着D的增加而增加。

一般来说，m（特征Field个数）与（每层feature map数目）都是较小的数值，所以在一个可接受的范围之内（这也说明了CIN的可行性）。

但如果在一些极端情况，m与都比较大的时候，为了降低参数数量，作者提到可以使用矩阵L阶分解的思想，将分解为和（矩阵分解）【类似于FM】：





在这种情况下，假设每层的特征映射数量相同，即，那么的参数为。所以CIN的空间复杂度从变为，而普通DNN的空间复杂度变为。

#### 时间复杂度

在CIN中，每一层都需要计算(feature map)，每个中间张量的计算时间为O(mHD)，而后每层都有H个feature map，计算H个feature map，所以T层的CIN计算时间复杂度为，时间复杂度为，而普通DNN的计算时间复杂度为，因此相比之下，CIN的主要缺点在于时间复杂度比DNN高。

### CIN的优点

（1）交互是向量（vector-wise）的交互，不是（bit-wise）的交互；

（2）高阶特征交互是显示的；

（3）网络的复杂性（参数容量）不会随着交互程度（隐藏层的数量）的增加而呈指数增长。

## DNN

在Embedding Layer之后，除了连接CIN模块，同时并行的会接入到简单的多层感知机。与Wide&Deep、DeepFM类似，设置这个模块的目的在于，隐式交叉编码可以作为显式交叉（CIN）的补充，进一步的提高模型表征能力。

## 模型组合

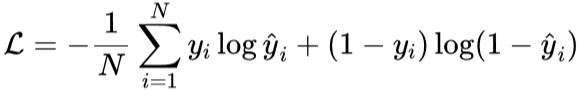
各个模块分析完毕，将三个输出模块（Linear、CIN、DNN）统一到一起，最后输出单元为：



其中a表示原生特征，分别表示DNN、CIN的输出向量，分别是对应模块的可训练参数，b为全局偏置项，为激活函数。

从这个形式上来看，xDeepFM既有低阶项（linear部分）又有高阶项（DNN和CIN部分），既有显式交叉（CIN部分）又有隐式组合（DNN部分），并且是基于vector-wise级别的交叉，会使模型更加健壮。

对于二元分类，损失函数为对数损失：



其中，N为总样本数量，优化目标是最小化目标函数（即是损失函数+正则化项）：



# xDeepFM与FM、DeepFM的关联

假设输入的所有特征Field都是单值的，CIN模块的层数（深度）只有1层，并且其中的feature map只有1个，此时的xDeepFM模型等价于DeepFM模型。因为CIN模块只对所有的特征embedding vector进行两两哈达玛积，将得到的结果进行加权求和（DeepFM是权重恒定为1进行求和），然后再对结果进行sum pooling作为输出。结合一阶项与DNN部分，模型基本等价于DeepFM。因此DeepFM是xDeepFM的特殊形式，而xDeepFM是DeepFM的一般形式。

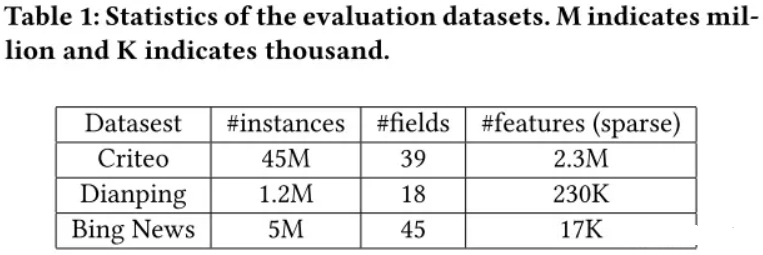
如果更进一步，将DNN部分去除，并对feature map（特征映射）使用一个常数形式的sum filter（权重恒定为1），那么此时xDeepFM与FM模型完全等价。

# 实验

实验部分主要解决下面几个问题：

* CIN模块是否真的做到了高阶的交叉？
* 是否有必要将Explicit和Implicit的网络结合？
* 网络的参数设置对于模型最终的影响是什么样的？

实验使用了三个数据集：Criteo Dataset、Dianping Dataset、Bing News Dataset。



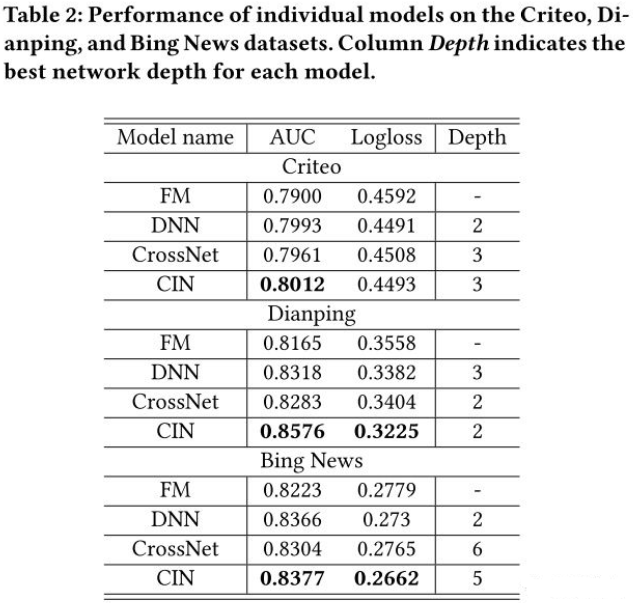
评测指标使用AUC与Logloss。其中AUC主要关注的是正负样本的相对顺序，并且对非平衡数据不敏感。而Logloss关注预测值与真实值之间的相关程度，在计算广告中因为涉及到广告竞价策略，一般使用Logloss比较多。

对比模型：LR、FM、DNN、PNN（IPNN与OPNN择其优）、Wide&Deep、DCN和DeepFM。

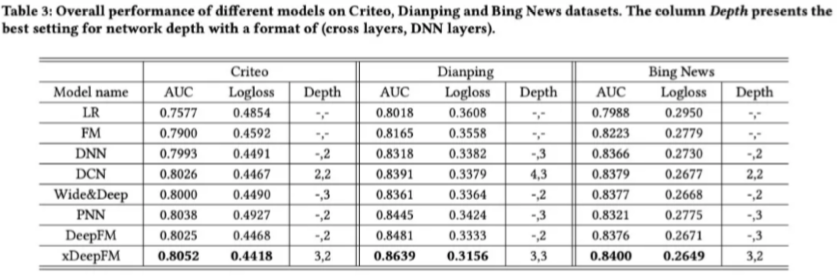
参数设置：learning rate：0.001，优化器：Adam ，batch size：4096，对于DNN、DCN、Wide&Deep、DeepFM与xDeepFM使用L2正则，其中。对于PNN使用dropout rate=0.5。模型每层的默认节点数:（1）DNN Layer节点为400；（2）在 Criteo 中，CIN Layer节点200。而 Dianping 和 Bing News 中，CIN Layer节点数为100；在实验过程中，Field embedding维度默认为10。

## CIN模块是否真的做到了高阶的交叉(单个神经模块之间的对比)

在三个数据集上，这里对比了FM、DNN、CrossNet、CIN四个单独模型的效果，实验证明CIN均取得了最好的效果，凸显出了CIN结构（显式高阶交叉）的优越性，并且在DNN、CrossNet、CIN均优于FM的结果，也证明了在sparse feature上的高阶交互特征是有正向收益的。



## 是否有必要将Explicit和Implicit的网络结合（组合模型的效果）



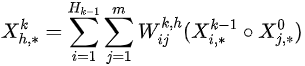
从LR模型一路对比到xDeepFM 模型，验证了同时组合了显式高阶交互特征和隐式高阶交互特征的xDeepFM 模型可以去的最好的实验效果，xDeepFM相较于 DNN有很大的提升。

并且还看到了一个有意思的现象：实验中所有的神经网络模型，在最好的参数设置中，都不需要太深的网络层数，一般只需要2～3层的网络深度即可。

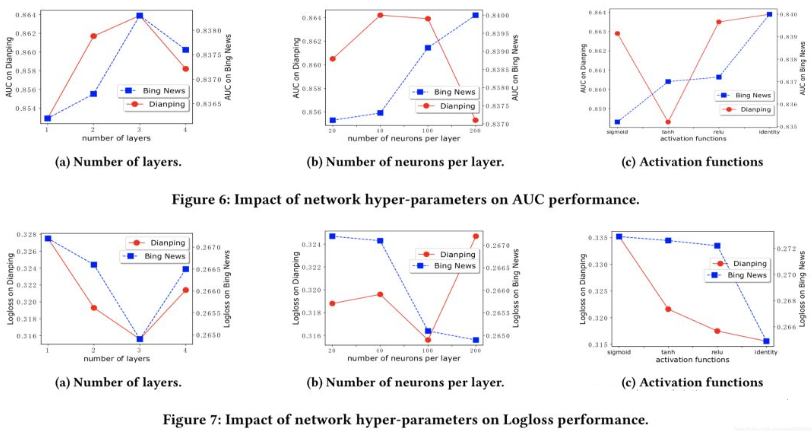
## 网络的设置对于模型最终的影响是什么样的

主要是隐层的深度、每层神经元的数量、激活函数。

其中隐层的深度、每层神经元的数量，没有什么特别。值得一提的是，在激活函数的选取上，paper中在公式：



中使用的激活函数是恒等式（就是没有激活函数），作者这里也让实验了其他的例如sigmoid、tanh、relu等常用的非线性激活函数，但是结果显示在上面公式上的恒等式变换的效果反而是最好的，即可以不用其他的非线性激活函数效果就很好，用了的话效果反而会下降。



从实验结果可以看出：

* 增加网络层的深度可以提升效果, 把网络层数设置为3在数据集上的效果是最好的；
* 增加CIN中feature maps的个数早期可以提升效果的，太大可能会带来过拟合(例如Dianping数据集，100 就可以了，200的时候效果会下降);
* 激活函数使用identity效果是最好的

# 实践

参考：

https://github.com/jpegbert/code\_study/tree/master/xDeepFM/

https://github.com/Leavingseason/xDeepFM

# 总结

模型主要优势：

* CIN可以高效的学习有界的高阶特征；
* xDeepFM模型可以同时显示和隐式的学习高阶交互特征；
* CIN模块可以显示的控制特征交叉的阶数(通过vector-wise的形式)，在大量数据集上的结果也显示了xDeepFM的卓越效果；
* 以vector-wise方式而不是bit-wise方式学习特征交互关系。

模型可改进方面：

* 目前对于multivalent fields的特征是使用sum pooling的方式处理的，未来可以借鉴DIN的思想进行改进（引入attention对不同向量加权，使模型可以注意到哪些特征更重要）；
* CIN的时间复杂度比较高，paper中打算未来在GPU集群上使用分布式的方式来训练模型。

这种用于高维稀疏特征的混合模型一般在比较浅层的比如2-3层的网络结构下会取得最好的效果。

参考：

<https://mp.weixin.qq.com/s/l9syRZ95aQA2G_pkV3RC3A>

<https://mp.weixin.qq.com/s/8gwjxgvW6lETGe_lGabaKg>

<https://mp.weixin.qq.com/s/cQACkJVM4uCEfgwa8c-IvA>

<https://mp.weixin.qq.com/s/_a_8BtJyKA2Yvu7thVk1jg>

<https://mp.weixin.qq.com/s/-GBnnWMjRjvoqOrv6QbsVg>

<https://mp.weixin.qq.com/s/raU2uIMsnY14qbYoPZ73Pg>

<https://mp.weixin.qq.com/s/5xLymgknLBuDBqWny3X3sw>