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，热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。

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。

# 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。该模型提出了CIN（Compressed Interaction Network）网络结构。该网络提出的目的是：提高特征交互的能力。

在xDeepFM中，作者设计了一种新型的网络Compressed Interaction Network(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。xDeepFM最出名的在于它的特征交叉学习部分,也就是CIN层，可谓是一种艺术般的交叉。其也在海量的数据竞赛中展现了不俗的成绩。

因此本文主要是针对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。

xDeepFM是19年之前所有竞赛中排名非常靠前的一种方案，而xDeepFM最出名的在于它的特征交叉学习部 分,也就是CIN层，可谓是一种艺术般的交叉。其也在海量的数据竞赛中展现了不俗的成绩。

文章提出了一个新颖的模型结构-压缩交互网络CIN，CIN可以显式的生成特征交互，并且是在vector-wise水平生成的。paper中证明了CIN与CNN和RNN结构具有一些共性，并将CIN与DNN结合，形成了paper的模型结构-xDeepFM，xDeepFM模型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。

该模型的主要贡献在于，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。

微软亚洲研究院社会计算组提出了一种极深因子分解机模型（xDeepFM），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，还兼具记忆与泛化的学习能力。

1.现在诸如DeepFM和Deep&Wide等模型都可以自动学习隐式的高维交互特征，并结合了低维特征，但是有一个缺点就是它们的高维特征都是在bite-wise的层面上进行交互的。本篇论文提出了一种压缩交互网络(Compressed Interaction Network(CIN))，能够学习显式的交互特征并且是在vector-wise的级别，CIN带有一 些CNN和RNN的特点，最终作者将整个模型命名为"eXtreme Deep Factorization Machine(xDeepFM)"。

首先还是强调特征变换的广泛使用和重要作用，而对于类别特征的计算交叉积的变换方式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。

自动学习交互特征的方法，主要是针对传统的生成交互特征的特征工程工作：

1. 获取到有用的特征需要耗费的成本比较高；
2. 对于具有大量原始特征的任务，想要提取到所有的交叉特征是不可行的；
3. 人工设计特征无法泛化到训练样本中未出现的特征。

下面是作者通过对比前人的工作的优缺点，引出了自己的xDeepFM模型结构。

1. FM：由于paper介绍的是xDeepFM模型，所以在说到FM模型时，对于FM模型中的每个特征的embedding向量中的每一个元素，paper中称之为bit。经典的FM模型可以引入任意高阶的交互特征；
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。
6. DeepFM：通过引入浅层模块和深层模块的组合结构构，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；
7. DCN：捕捉有界阶数的交互特征；
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、隐式高阶交互模块和传统的FM模块。

可以看到xDeepFM相比于DeepFM模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。

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。

注：paper中一个有意思的论述式：DNN模型多是在bit-wise层面建模特征交互关系，而FM模型则是在vector-wise层面建模特征交互关系。所以还很难说在推荐领域，DNN模型就是用来建模高阶交互关系的最有效的模型。

关于xDeepFM的另一个贡献点“有限阶数特征交叉”，所具备的优点之前也提到过。简单DNN结构虽然说能够隐式交叉特征，学习任意函数的表示，但是我们并不清楚其最大特征交叉阶数为多少，简单来说就是无法得知哪些特征交叉可以得到最佳效果，而显式指定阶数交叉特征能够在一定程度缓解这些问题。

在论文原文中，作者对DCN模型的Cross Network进行分析，并从数学层面证明了Cross Network最终学习到的是一种特定形式的表示，认为该结构不利于模型充分表征交叉特征。个人理解，Cross Network之所以采用该形式，是因为其精简参数，在实现显式特征交叉的同时控制计算复杂度。二者孰优孰劣，仍需要在实际业务场景中对比试验方能判定。

# 特征交互

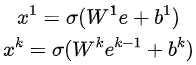
文章提出两个概念的特征交叉：bit-wise level和vector-wise level。bit-wise level指的是神经网络中节点之间的交互（DCN模型的Cross Network交互类型就是bit-wise level）；vector-wise level指的是embedding向量之间的特征交叉（例如FM、DeepFM、PNN的特征交叉）。

因此本文主要是针对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。

在进行模型分析与对比之前，首先为大家介绍一下什么是vector-wise模式。与vector-wise概念相对应的是bit-wise，在最开始的FM模型当中，通过特征隐向量之间的点积来表征特征之间的交叉组合。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，这种方式称为vector-wise。后续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，这种方式称为bit-wise。将常见的bit-wise方式改为vector-wise，使模型与FM思想更贴切，这也是xDeepFM的Motivation之一。这里需要提醒的是，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。

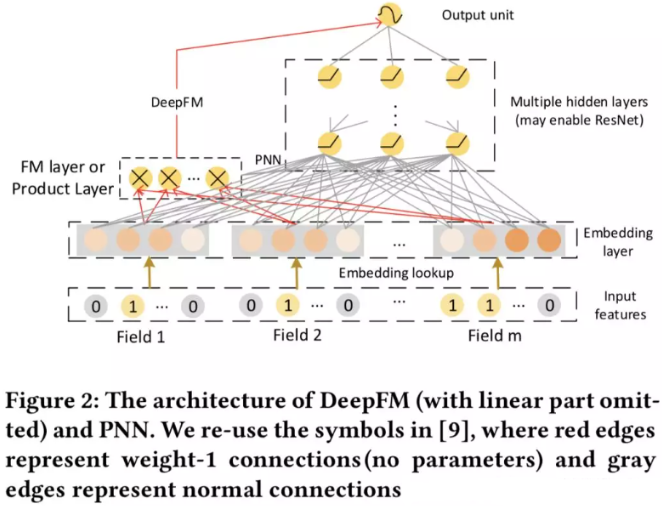
## 隐式特征交互

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：

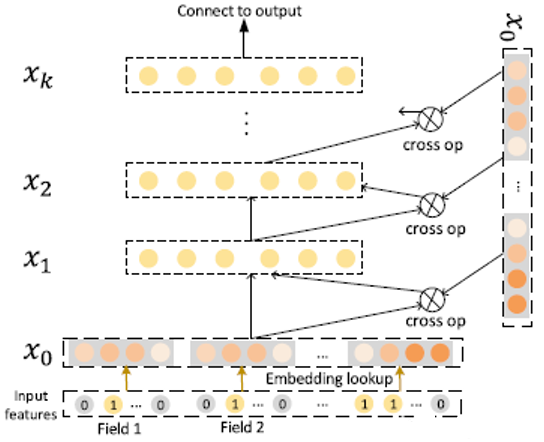


其中，k表示神经网络层的深度，σ表示激活函数，x^k表示第k层神经网络的输出，这与图2中的深度网络部分比较相似（当然不包括FM层或者Product层），这种前向神经网络的模型架构是以bit-wise的方式来建模交互关系的，也就是说，即使是同一个field embedding向量内的bit元素也会互相影响。

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：

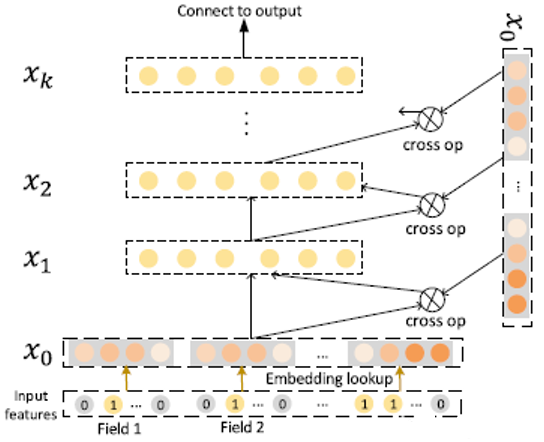


其中CrossNet中的每一个层都是x0的标量倍数，同时paper中也证明了这一理论：

Theory: 一个k层交叉的网络的第(i+1)层计算公式为：，那么交叉网络xk的输出为x0的标量倍数。

# DCN的局限性

DCN的Cross Network架构如下所示（只是cross network部分，不是全部DCN）：

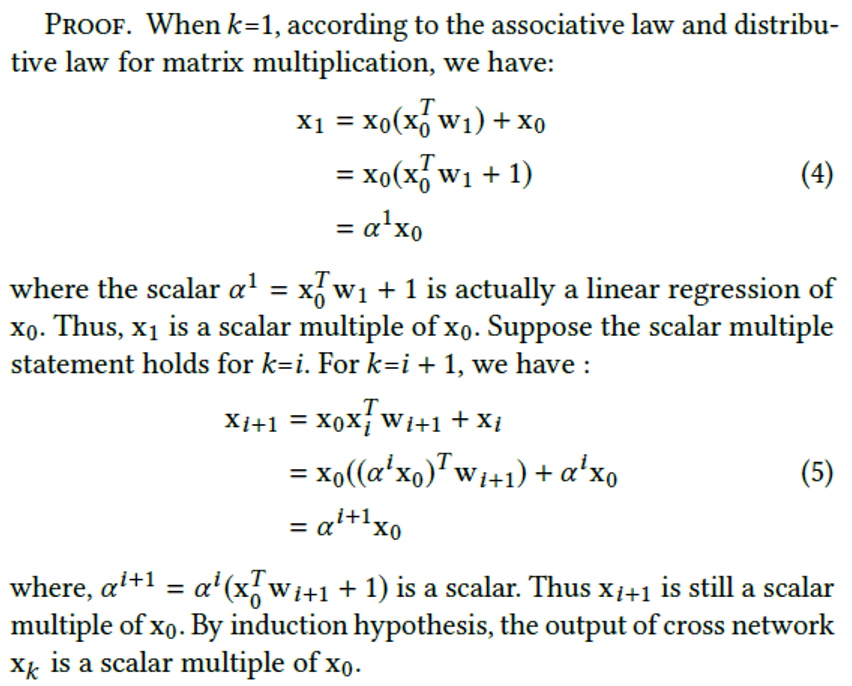


公式如下：



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也有局限性：

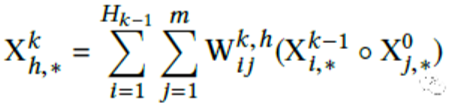
* CrossNet的输出只能是指定的形式，即是关于x0的标量倍数；
* 特征交叉也是也是以bit-wise形式得到的。

# Compressed Interaction Network(CIN)

## CIN

了解Cross Network的局限性后，paper作者提出了Compressed Interaction Network(CIN)。

假设输入一个有m个field(raw feature)，首先将这m个field都转换成都是D维的 embedding。则可以得到一个m\*D维的embedding矩阵，用X0来表示。 因此对于第k层Xk，使用下面的公式进行计算

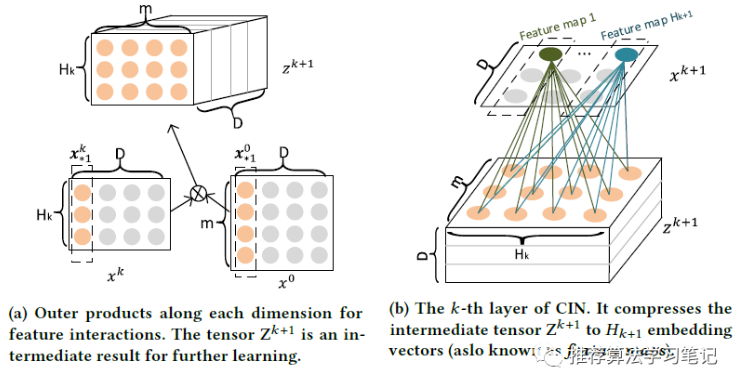


表示第k层第h个field对应的embedding，是一个scalar，空心圆表示Hadamard product，例如⟨a1, a2, a3⟩◦⟨b1,b2,b3⟩ = ⟨a1b1, a2b2, a3b3⟩

这个公式就是将第k-1层的embeding和第0层的embedding两两做Hadamard product，然后weight sum后输出一个embedding。同时，我们可以决定第k层有多少个field，使用Hk表示field的个数。

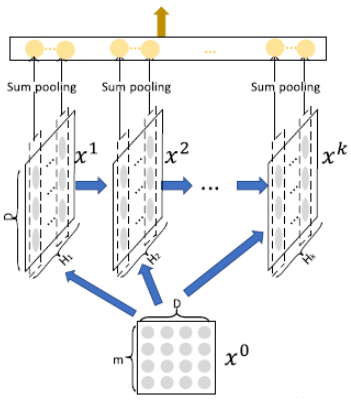
## CIN和CNN的关系

CIN其实是可以转换成CNN的，如下图所示。具体细节可参见paper。

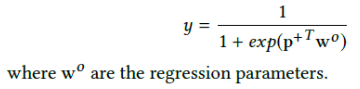


## CIN输出

在得到每一层的Xk输出后，将每个field（D维向量）的元素相加起来，得到一个Hk维的向量作为输出。所有层的输出构成一个sum pooling层，如下图所示



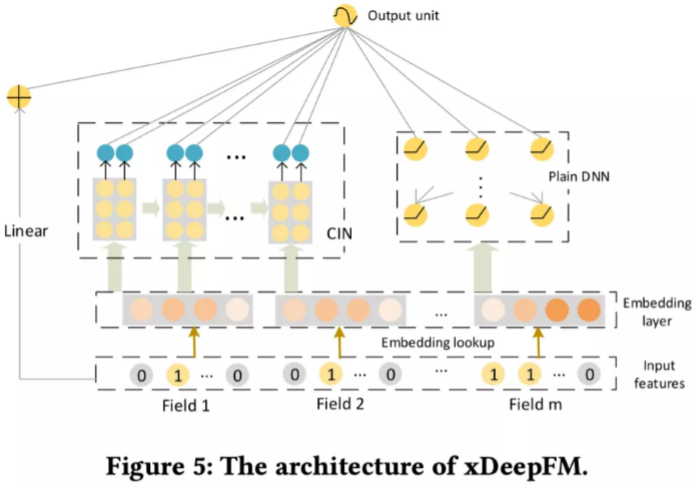
最后sum pooling层使用sigmoid函数作为最终输出，公式如下所示



# 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)以及CIN部分(Explicit feature interactions)。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。我们按照该图看看模型每一步都在做什么，尤其是CIN层做了哪些操作，为什么能在诸多数据竞赛中拿到相较于DeepFM等模型那么大的优势。

## 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

paper中也提到了在推荐系统中，不像cv和nlp领域的数据具有时空关联的特性，通常对于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特征的的情况也是比较常见的。

Embedding层其实是非常常见的，它的作用就是将原始特征压缩成低维连续特征。通常有两种情况：

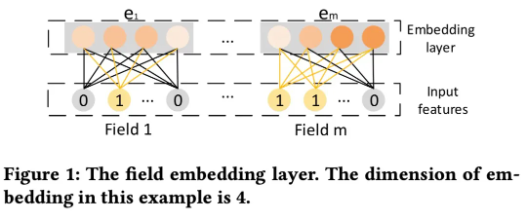
* 单个特征，例如特征名为‘性别’，该特征对应的embedding可以直接用于DNN的输入；
* 复合特征，或者说交叉特征，例如[user\_id=s02,gender=male, organization=msra,interests=comedy&rock]这种交叉特征，对于每个特征user\_id、gender、organization、interests都有他们自身对应的embedding，一般是计算交叉特征中单个特征embedding的sum pooling结果用于DNN的输入。

Embedding层的输出结果一般是表示成如下的wide concatenated vector形式：



其中m表示特征的数量，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。

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。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：

paper中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。

此处embedding做的事情就是将传统的单个特征映射到一个D维的dense特征上,假设有m个field,最终我们将得到：





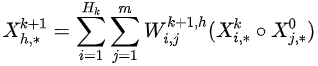
其中，是所有embedding向量组成的矩阵，，m表示field的数量， 表示embedding特征，D为embedding维度。

## CIN(Compression Interaction Network)

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，令，其中表示第k层特征向量数。

接下来，让我们详细的分析下计算过程，公式如下：



其中，表示 中第h行特征向量，同理，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。是参数矩阵，是一个标量值，表示第i行第j列位置的参数 。运算∘表示哈达玛积，即向量对应元素相乘，结果仍为向量。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。

让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。

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，不同的参数矩阵便是不同的卷积核，每个卷积核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。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：

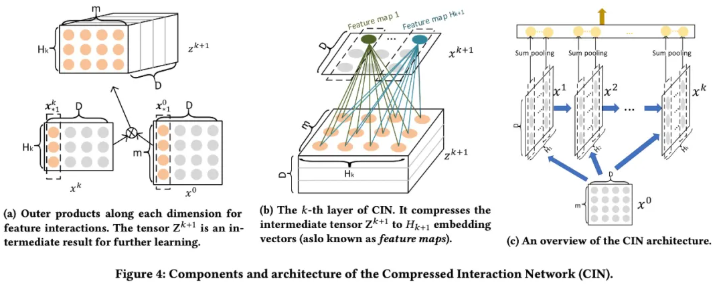
得到每层的输出之后，通过sum pooling操作将所有的特征矩阵进一步压缩为向量，作为最终的输出。

中含有个D维feature map，对于每个feature map进行求和池化之后得到一个标量值，其中，所以产出长度为的向量。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。最终 CIN 的输出为，，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。

整体如下图所示：



图(a)和(b)表示了如何从这一层的隐藏层（）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。

### CIN中的符号

embedding向量是看做vector-wise形式的特征交互，我们之后将多个field embedding表示成矩阵：



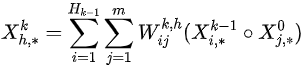
其中，的第i行向量表示第i个field特征的embedding向量，将其表示成：，

D 表示field embedding向量的维度。

对应的CIN的第k层的输出也是一个矩阵，表示成：



其中，表示第k层的（embedding）特征向量的数量，且对应的，对于每一层，是通过下面的方式计算的：



其中，，表示用于计算第k层输出中的第h行向量的一个参数矩阵，因此是一个标量数值，即对哈达玛积计算标量倍数。

计算符号∘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。

这个公式到底干了什么？（1）取前一层中的个vector，与输入层中的m个vector进行两两Hadamard乘积运算，得到个vector，然后加权求和；（2）第k层的不同vector的区别在于，这个vector求和的权重矩阵不同，对应有多少个不同的权重矩阵，是一个可以调整的超参。

|  |
| --- |
| 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l层，DCN中的Cross包含从1阶～（l+1）阶的所有组合特征，而CIN只包含（l+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。 |

需要注意的是，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啊的阶数随着层数的增加也在加深。

paper中说到，上面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。

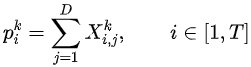
CIN结构除了与RNN结构相似性之外，CIN还与CNN有一定的可类比性，如图4a所示，引入一个中间张量(intermediate tensor) ，它是关于隐藏层X\_k和X0的outer products（沿着embedding维度），因此可以将看做一幅图像，并且是一个filter，

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

如图4b所示，将filter沿着的embedding维度方向做平滑操作，即可得到隐向量，类比到CV领域中，我们将其称为一个feature map，因此就是个不同feature map的集合，就是个不同feature map的集合。CIN中的名词compressed是指第k个隐层将个向量的潜在空间压缩至个向量（对于个向量这样的结构，在潜在空间中一共有个，每一个这样的结构只能压缩成为输出中的一个向量）。

|  |
| --- |
| 当CIN的深度为1，隐层的feature map也只有1个，那么其就完全等价于FM模型了（可能会多了特征的平方项，少了特征的非交叉项，这些先不管了，你懂！） |

如图4c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，在第k个隐层，在其中每个feature map上，我们做sum pooling操作，即：



因此，对于第k个隐层，可以得到poolng之后的长度为的向量：



对于所有隐层的pooling之后的向量，我们将它们concat之后作为输出单元，



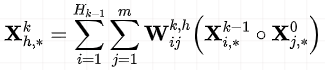
到这里，如果我们直接使用CIN用于二分类，那么输出单元就是一个在上的sigmoid节点：



其中，表示回归参数。

需要注意的是，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。然后根据任务的需要套一个激活函数就行了，比如sigmoid。

xDeepFM在CIN层实现了特征的显示交叉，究竟是如何做到的呢？就是M层 =(M-1)层 + 0层的思路，即第M阶的交叉特征是由(M-1)层的特征和第0层(原始特征)交叉得到的。具体地，假设第0层的原始特征为, ,同时假设第k层的特征为,其中Hk为第k层的网络的特征向量个数,所以H0=m,那么要想得到第k层的特征,就可以通过下面的式子进行计算：





其中, , 为第h层特征向量的参数矩阵，表示哈达玛积（Hadamard product），即两个向量对应元素相乘。最终。也就是说第k层的第h个特征向量是由第(k-1)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。

由于是通过与的交互得到，因此特征交互是一种显示的方式，且交互的程度随着层的深度而增加。这里作者指出，CIN与RNN非常类似：下一个输出的隐藏层都依赖于当前最后一个隐藏层和一个额外的输入（embedding），且整个结构与CNN有着密切的联系。

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。所以我们使用sum pooling。这样第k层第i个向量的输出为：



那么第k层的输出为：

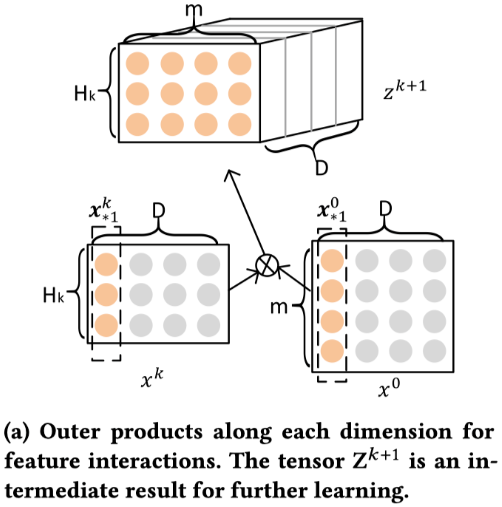


最终CIN的输出为：

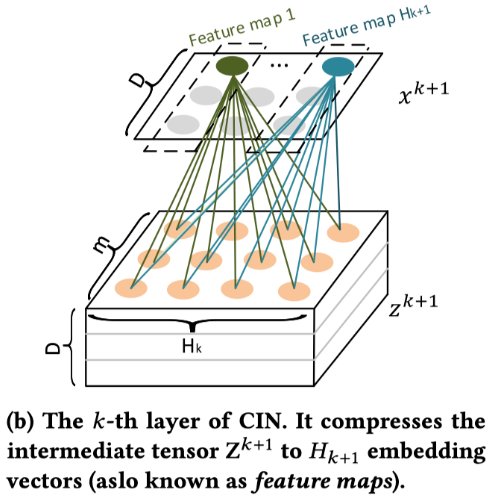


对于CIN的计算方式，可以从CNN的角度进行分析：以下得到第(k+1)层的输出

1. 外积操作：引入一个过渡张量，它是隐藏层与输入的外积，如下图所示：

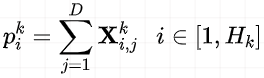


1. 然后我们可以将整个可以视为图片，【注：原文写的是k，但这里得到的是下一层，所以应该是(k+1)】视为过滤器（filter）【重点：这里，因此这是单个过滤器，对于所有的过滤器应该有个，所以才会得到下图的】。如下图所示，过滤器沿着Embedding维度（D）滑动。然后得到隐藏向量，这被称为一个特征映射（feature map）。



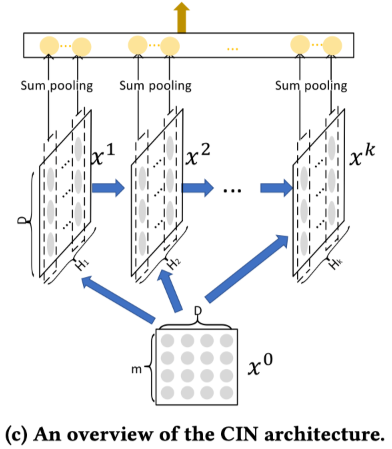
因此是个不同特征映射的集合。CIN网络中的压缩（compressed）指的便是第个隐藏层将向量空间压缩至向量。

1. 下图是整个CIN的结构，定义T为整个网络的深度。每个隐藏层为，对于第k层，将所有的特征映射进行一个池化操作（sum pooling）【例如对上图Feature map 1向量进行一个累加】：



因此便得到一个池化向量对于第k隐藏层。

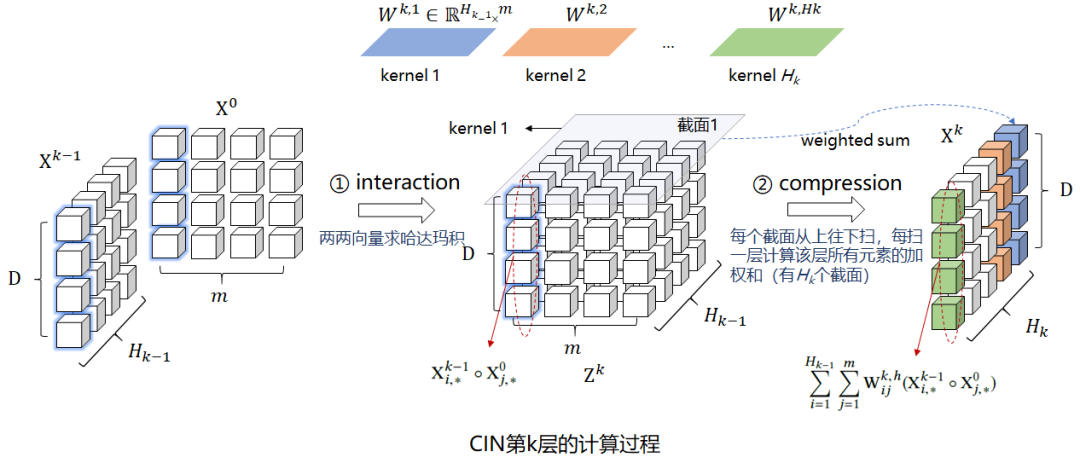
最后在对于所有的隐藏层的池化向量进行一个拼接：



【注】：此文章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？不知道大家是否会有同样的疑惑，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，所以我们称此过程为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复杂度分析

#### 空间复杂度

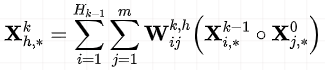
第k层有个，所以第k层参数量为，假设CIN有T层，那么共有个参数。最后将各层输出池化拼接，需要个参数进行加权求和。所以，共有个参数。

对比普通的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。

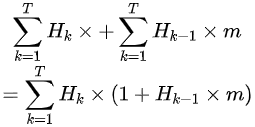
不妨假设每一层的，那么CIN的空间复杂度为，普通DNN的空间复杂度为。

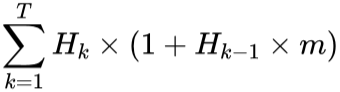
一般来说，m（特征Field个数）与（每层feature map数目）都是较小的数值，所以在一个可接受的范围之内。但如果在一些极端情况，m与都比较大的时候，可以使用矩阵L阶分解的思想，将表示为两个小矩阵的乘积，，其中，，并且同时。在这种情况下，假设，那么的参数为。所以CIN的空间复杂度从变为，而普通DNN的空间复杂度变为。

主要的学习参数就是：



计算第k层的第h（）个emb需要，而第k层共有个emb vec，因此，第k层的共有个参数。假设包括 CIN经过一个二元分类任务，那么CIN总共的学习参数为（分为一共T层的CIN的参数和最后一层回归层的参数）：





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。

为了和DNN做对比， T层的DNN的学习参数应该为，参数会随着D的增加而增加。对于CIN，通常m与不会很大，因此的参数量是可以接受的。【这是为了说明CIN的可行性】

当m和H\_k较大时，为了降低参数数量，作者还提到可以将分解为和（矩阵分解）【类似于FM】：





如果假设每层的特征映射数量相同，为H，那么这样就可以将空间复杂度从降低到。

#### 时间复杂度

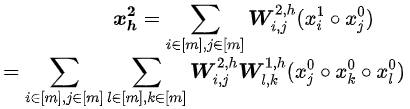
在CIN中，每一层都需要计算(feature map)，每个中间张量的计算时间为O(mHD)，而后每层都有H个feature map，计算H个feature map，所以T层的CIN计算时间复杂度为，时间复杂度为，而普通DNN的计算时间复杂度为，因此相比之下，CIN的主要缺点在于时间复杂度比DNN高。

#### 多项式逼近

通过对问题进行简化，即假设CIN中不同层的feature map的数量全部一致，均为fields的数量m，并且用[m]表示小于等于m的正整数。CIN中的第一层的第h个feature map表示为，即：



因此，在第一层中是通O(m^2)个参数来建模成对的特征交互关系，相似的，第二层的第h个feature map表示为：



上面公式中有关下标为k和l的相关计算都已经在前面一层计算完成了，这里只是为了方便观察才又写出来的，知道就好。这里可以看到第二层中feature map的只用了个参数就建模出了3-way的特征交互关系。

我们知道一个经典的k阶多项式一般是需要个参数的，而我们展示了CIN在一系列feature map中只需要个参数就可以近似此类多项式。

### 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，将所有模块组合到一块，三个输出模块（Linear、CIN、DNN）统一到一起，作为模型的最终输出，公式如下：



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。

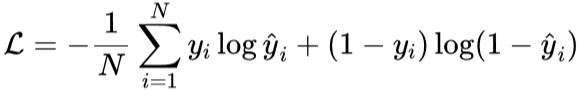
从这个形式上来看，xDeepFM既有低阶项又有高阶项，既有显式交叉又有隐式组合，并且是基于vector-wise级别的交叉，可谓是应有尽有，算是FM类模型的完备实现了。

由于CIN学习的是高阶显示特征，DNN学习高阶隐式特征。因此将两者结合，会使模型更加健壮。该模型被称为xDeepFM，因为模型：（1）包含了低阶和高阶特征的交互；（2）包含了隐式特征交互和显示特征交互。最后输出单元为：



其中a表示原生特征，表示DNN、CIN的输出。

对于二元分类，损失函数为对数损失：



其中，N为总样本数量，优化目标来最小化目标函数（目标函数即是损失函数+正则化）



1. CIN与RNN的关系

CIN中下一层的输出都依赖于上一层的输入以及额外的输入，和RNN是非常相似的。

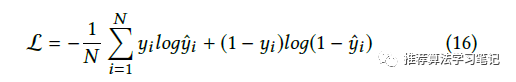
2. CIN与CNN的关系

我们发现第k层的每一个新的向量都是由第(k-1)层的所有向量以及第0层的所有向量分别进行element-wise 相乘, 然后形成“图像”,我们再使用filter - 与其进行操作得到下一层的新向量，最终将压缩为了Hk个向量,这也是compressed名字的由来。

公式如下所示

https://mmbiz.qpic.cn/mmbiz_png/OibBxV7NSmubUC4g8bm420ia0V4uXfc3pWD0By8kzB3oNiaoPiceAh62OJfvb8dsnY0tBaAicic0Erf9qhDoXSo7WNdQ/640?wx_fmt=png&tp=webp&wxfrom=5&wx_lazy=1&wx_co=1

损失函数为



加上L2正则化

https://mmbiz.qpic.cn/mmbiz_png/OibBxV7NSmubUC4g8bm420ia0V4uXfc3pWXm2EFQMklaa0rc2LULBK2lv7a8Uibtn9hUZgDhCJicgEABdPSgkIPP2Q/640?wx_fmt=png&tp=webp&wxfrom=5&wx_lazy=1&wx_co=1

# xDeepFM与FM、DeepFM的关联

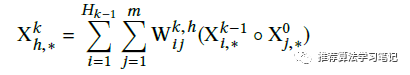
1. 对于xDeepFM，将CIN模块的层数（深度）设置为1，feature map（特征映射）数量也为1时，其实就是DeepFM的结构，因此DeepFM是xDeepFM的特殊形式，而xDeepFM是DeepFM的一般形式；
2. 在1中的基础上，当再将xDeepFM中的DNN去除，并对feature map（特征映射）使用一个常数形式的sum filter，那么xDeepFM就退化成了FM形式了。

假设输入的所有特征Field都是单值的，CIN模块只有1层，并且其中的feature map只有1个，此时的xDeepFM模型基本等价于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。

如果更进一步，将DNN部分去除，同时在加权求和部分固定权重为1，那么此时xDeepFM与FM模型完全等价。

对于xDeepFM模型，假设CIN的深度与特征映射的数量都为1，则xDeepFM就相当于DeepFM的泛化。当进一步删除DNN部分，并且对于特征映射使用一个sum filter，那xDeepFM就将是传统的FM模型。【联想之前的分解】

根据公式



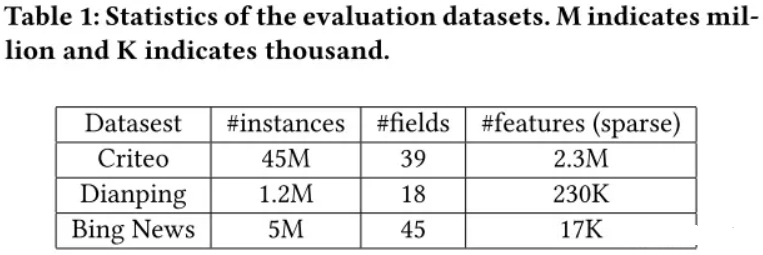
将k设为1并且第1层field的个数H1=1，然后sum pooling后，就是DeepFM。可见xDeepFM是DeepFM的泛化版本。

# 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。



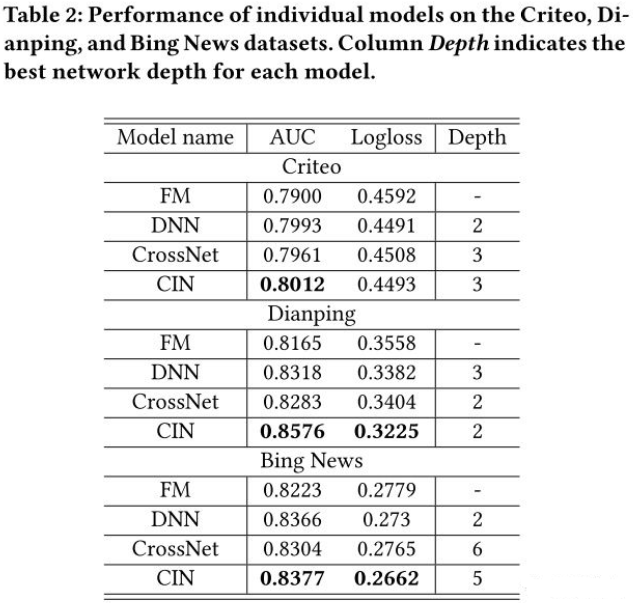
评测指标使用AUC与Logloss。其中AUC主要关注的是正负样本的相对顺序，并且对非平衡数据不敏感。而Logloss关注预测值与真实值之间的相关程度，在计算广告中因为涉及到广告竞价策略，一般使用Logloss比较多。

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

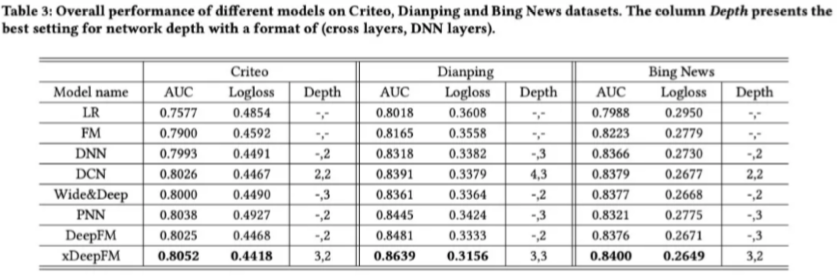
参数设置：learning rate：0.001，优化器：Adam ，batch size：4096，对于DNN、DCN、Wide&Deep、DeepFM与xDeepFM使用L2正则，其中。对于PNN使用dropout rate=0.5。模型每层的默认节点数:（1）DNN Layer节点为400；（2）在 Criteo 中，CIN Layer节点200。而 Dianping 和 Bing News 中，CIN Layer节点数为100；在实验过程中，Field embedding维度默认为10。

## 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）



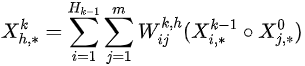
从LR模型一路对比到xDeepFM 模型，验证了同时组合了显式高阶交互特征和隐式高阶交互特征的xDeepFM 模型可以去的最好的实验效果，xDeepFM相较于 DNN有很大的提升。

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。

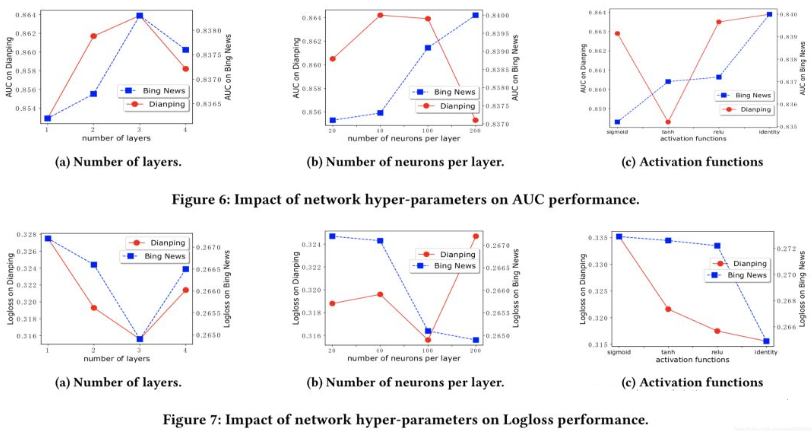
## 网络的设置对于模型最终的影响是什么样的

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，就不再说了。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。对于这种现象，有什么合理的解释吗，还是就只能理解为这事deep learning中玄学问题。



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

# 总结

模型主要优势：

* CIN可以学习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；
* 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；
* 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。

xDeepFM算法的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。

对于这种高维稀疏特征来说，基于FM思想的模型例如DeepFM,Deep&Wide,PNN 等比LR不知道高到哪里去了

并不是混合模型就一定好，但是单用DNN component一般效果比较差

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

# 参考代码

https://github.com/Leavingseason/xDeepFM

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>