深度排序模型概述（一）Wide&Deep/xDeepFM

xDeepFM 机器AI学习 数据AI挖掘 2019-12-25

更多机器学习关注公众号。

本文记录几个在广告和推荐里面rank阶段常用的模型。广告领域机器学习问题的输入其实很大程度了影响了模型的选择，因为输入一般维度非常高，稀疏，同时包含连续性特征和离散型特征。模型即使到现在DeepFM类的方法，其实也都很简单。模型的发展主要体现于对特征的充分挖掘上，比如利用低阶和高阶特征、尝试自动学习交叉特征而非手动、尝试更精准地实现高阶特征(bounded-degree)。

广告相关的领域最早大行其道的模型当属LR模型，原因就是LR模型简单，可解释性好，拓展性高，精心细调之后模型效果也会非常好。人工特征工程的初期，加些交叉特征，无论是离线评估指标还是线上的业务指标都会很容易提高，当然大家都会很开心。吃掉容易收割的80%之后，基本上就到了瓶颈。这时候就会发现即使深挖业务场景貌似也找不到比较好的特征了，加了一堆特征搞不好效果还降了。继续做得话，除了特征工程，还可以尝试特征选择，改进优化算法(比如加快收敛)，在线学习之类的。随着业务的持续迭代，手动交叉特征被视为脏活累活，那肯定要想如何做自动交叉特征。这方面的工作同时也有进展。Rendle在2010年提出的FM模型。FM模型利用特征的隐向量做内积来实现特征的交叉。后续阮毓钦提出了Field-aware FM模型，一个特征对应多个隐向量，在criteo举办的ctr比赛上崭露头角。这些其实都已经算是ctr领域的常用套路。FM类模型由于复杂度的原因一般都只能实现二阶交叉。所以不能利用高阶交叉总会让人感觉缺少点意思，毕竟很多场景高阶交叉的确有意义。另外不得不提的特征组合的工作就是Facebook提出的gbdt+lr的方法，通过gbdt来实现监督式的特征组合。浅层模型尝试之后势必要引入深层模型，毕竟深度学习很大作用就是作为特征表示学习。大厂引入肯定要更早一些，毕竟核心业务都要长时间持续投入人力进行模型优化。至于思路，我觉得可能跟现有市面上看到的模型差不多。一般多层感知机mlp认为可以实现特征高阶交叉(high-order feature interactions)。2016年出现wide&deep、fnn和pnn等工作。wide&deep算影响力比较大的工作了。wide部分是手动特征交叉(负责memorization)，deep部分利用mlp来实现高阶特征交叉(负责generalization)，wide部分和deep部分joint train。fnn比较暴力，直接使用预训练的fm隐向量传给mlp。PNN则是先做特征交叉然后给mlp处理。fnn和pnn这两种方法的缺点就是忽视了低阶特征交叉。2017年出现DeepFM、Deep&Cross和NFM等工作。DeepFM模型和Deep&Cross(包含下面要介绍的xDeepFM)都可以认为是Wide&Deep架构输入和wide部分进行改进。DeepFM和之前模型相比优势在于两点，一个是相对于Wide&Deep不再需要手工构建wide部分，另一个相对于FNN把FM的隐向量

参数直接作为网络参数学习。DeepFM将embedding层结果输入给FM和MLP，两者输出叠加，达到捕捉了低阶和高阶特征交叉的目的。Deep&Cross和DeepFM类似，cross network模块可以实现bounded-degree feature interactions。总结来说，涌现了这么模型，到底哪个好，我觉得很难一锤定音。每个模型都有存在的价值和合适的应用场景。有时候也不一定非得效果提示才行，有时候效果没提升但是可以大大减少特征工程的工作也算有收益。根据实际的场景、业务需求和迭代阶段选择合适的模型，花费合理的成本，取得最大的业务价值才是最重要的。

深度排序模型的结构可以分为两种，一种是并行结构，分别做低阶和高阶特征组合；一种是串行结构,如PNN、NFM、AFM等。这也构成了深度排序模型的两条演进路线，如何更有效地捕获特征组合是核心，沿着以下两个演进路线发展：一：更有效地捕获二阶特征：Wide&Deep → DeepFM → NeuralFFM → DeepFFM 二、显式建模2阶/3阶/4阶...K阶特征组合：DeepCross → xDeepFM

Wide&Deep

Wide&Deep 模型的核心思想是结合线性模型的记忆能力（memorization）和 DNN 模型的泛化能力（generalization），在训练过程中同时优化 2 个模型的参数，从而达到整体模型的预测能力最优。记忆（memorization）即从历史数据中发现item或者特征之间的相关性。泛化（generalization）即相关性的传递，发现在历史数据中很少或者没有出现的新的特征组合。

网络结构

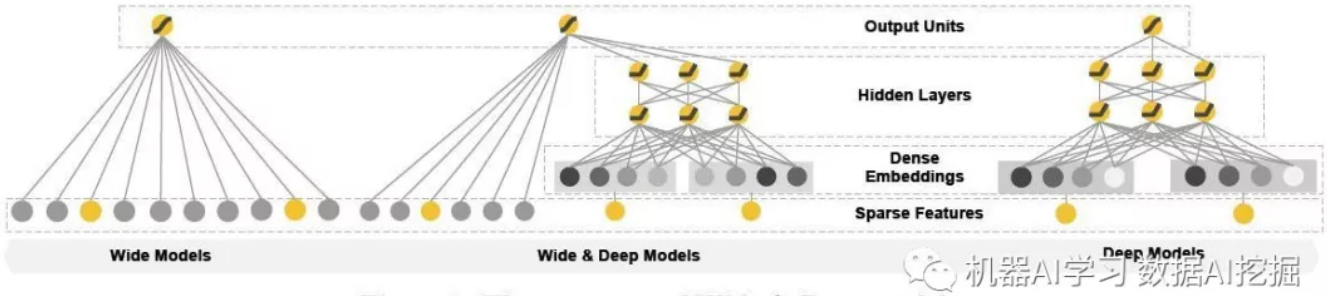


Figure 1: The spectrum of Wide & Deep models.

可以认为：Wide&Deep = LR + DNN

> Wide部分 实际上，Wide模型就是一个广义线性模型：

$$y = w^T x + b \quad y=wTx+b$$

最终在y的基础上增加Sigmoid，Sigmoid函数作为最终的输出。

> Deep部分 实际上，Deep模型是一个前馈神经网络。深度神经网络模型通常需要的输入是连续的稠密特征，对于稀疏，高维的类别特征，通常首先将其转换为低维的向量，这个过程也称为embedding。

在训练的时候，首先随机初始化embedding向量，并在模型的训练过程中逐渐修改该向量的值，即将向量作为参数参与模型的训练。

Wide & Deep模型的联合训练（joint training） 联合训练是指同时训练Wide模型和Deep模型，并将两个模型的结果的加权和作为最终的预测结果：

$$P(Y = 1|x) = \sigma(w_{wide}^T[x, \Phi(x)] + w_{deep}^T a^{(lf)} + b)$$

P(Y=1|x)=σ(wwideT[x,Φ(x)]+wdeepTaa(lf)+b)

训练的方法：

Wide模型：FTRL（Follow-the-regularized-leader）
Deep模型：AdaGrad

在推荐的场景中,最重要的行为数据是点击(下载/购买)数据,传统的Wide类型的特征可以对确定性的推荐很好的建模, 如果考虑到多样性,就需要对稀疏数据和"没有直接观测到"的行为数据建模。deep部分可以对多阶的行为传导更好的表达。值得注意的是,最后一层中Deep和Wide部分是如何合并到一起的. 文中的做法是将Wide部分的特征和Deep部分的最后一层的特征对齐,然后统一送入到一个逻辑回归模型中。

应用

对于推荐系统，其最一般的结构如下图所示：

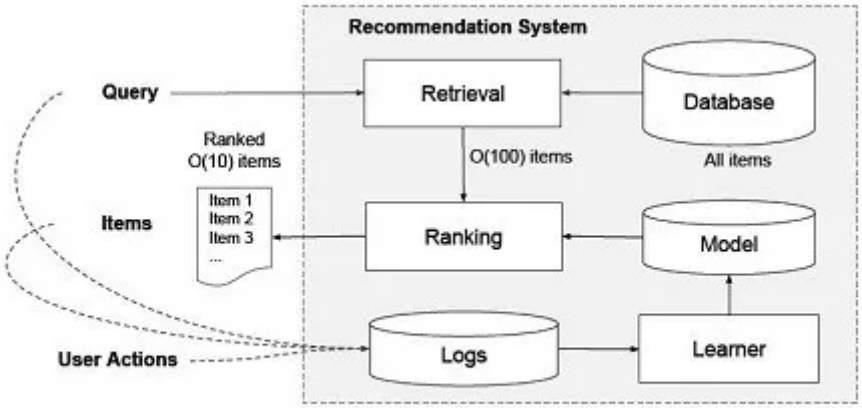


Figure 2: Overview of the recom...

当一个用户访问app商店时，此时会产生一个请求，请求到达推荐系统后，推荐系统为该用户返回推荐的apps列表。

在实际的推荐系统中，通常将推荐的过程分为两个部分，即上图中的Retrieval和Ranking，Retrieval负责从数据库中检索出与用户相关的一些apps，Ranking负责对这些检索出的apps打分，最终，按照分数的高低返回相应的列表给用户。

Deep & Cross Network(DCN)

对比同样来自 google 的工作 Wide & Deep ，DCN 不需要特征工程来获得高阶的交叉特征，对比 FM 系列的模型，DCN 拥有更高的计算效率并且能够提取到更高阶的交叉特征。

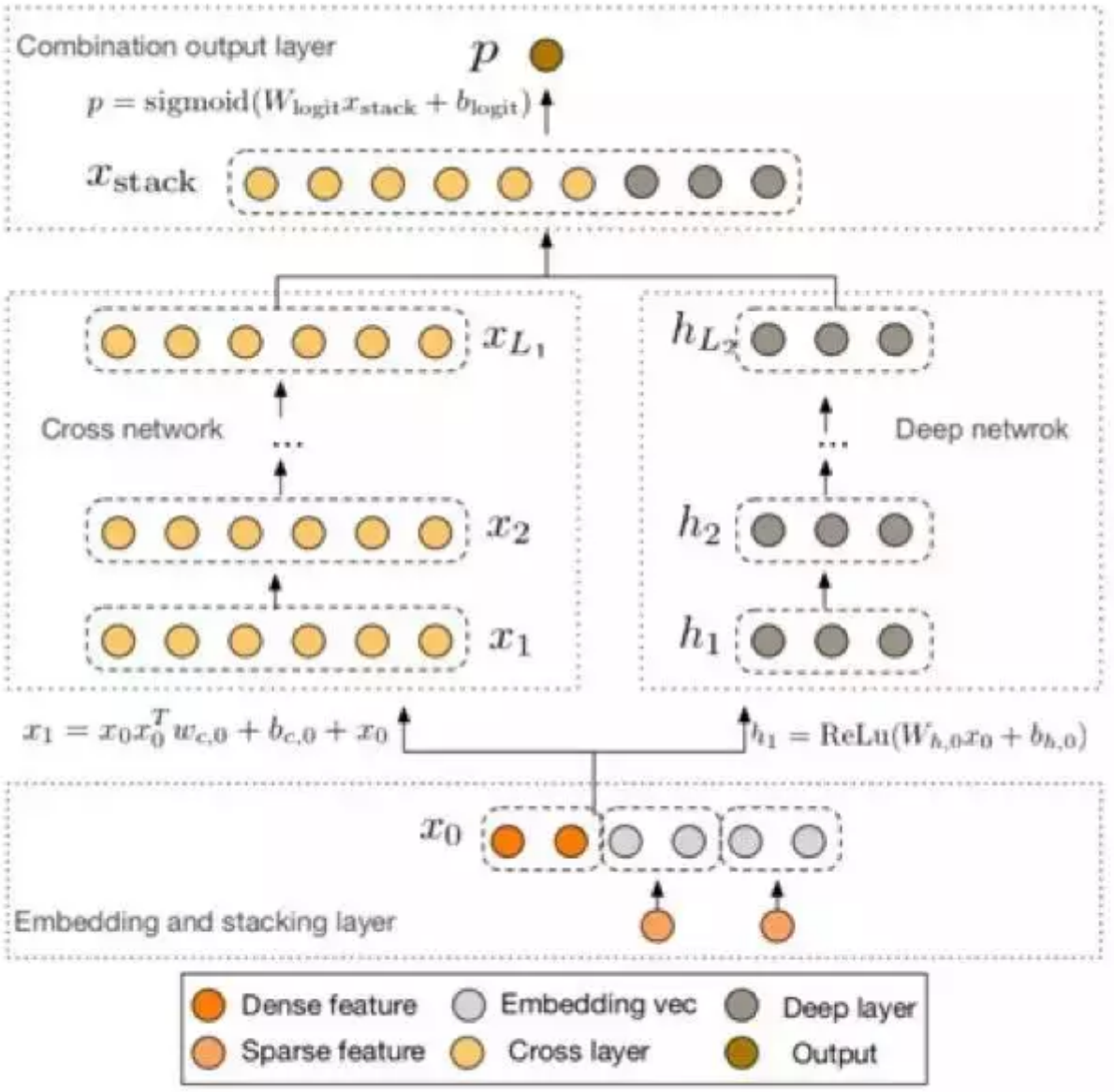


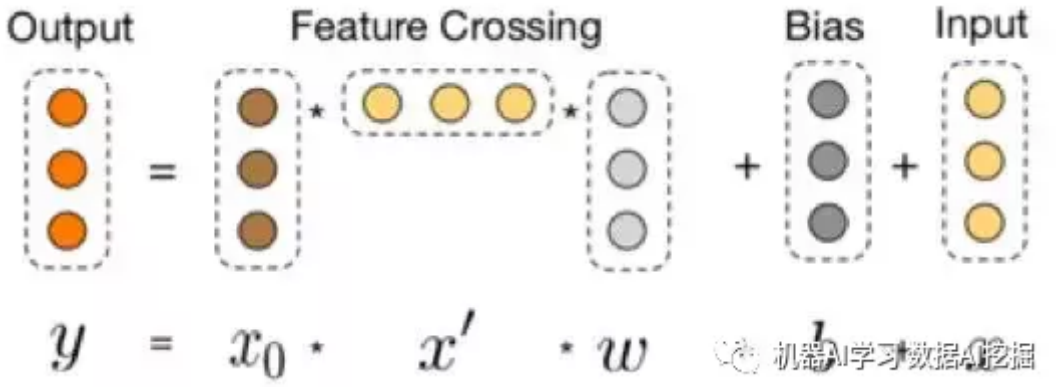
Figure 1: The Deep & Cross Network

从网络结构上面来看，该模型是非常简单明了的，特征分为类别型与数值型，类别型特征经过 embedding 之后与数值型特征直接拼接作为模型的输入。所有的特征分别经过 cross 和 deep 网络，如果把这两个网络看作特征提取的话，经过提取后的特征向量拼接之后是常规的二分类，如果训练数据是曝光和点击，最后输出的就可以看作点击率了。

离散特征嵌入 离散特征嵌入这个想法最初来自于 Mikolov 的 word2vec 系列文章。最初解决的问题是词的独热表示过于稀疏，并且不同词之间的向量形式表示完全没有联系。具体思路在此不赘述，最终的实现是将一个上万维的词独热表示嵌入到了只有几百维的稠密向量中。而嵌入的本质其实是构建一张随机初始化的向量查找表，通过我们的训练目标做有监督学习来得到不同词在特定目标下，处于向量空间中的位置。将词嵌入的思路推广到其它的离散特征处理中，我们可以用同样的方法将各种类别特征如“用户性别”、“城市”、“日期”嵌入

到稠密的向量空间中。经过这样处理之后，自然就解决了原本 FM 遇到的特征稀疏问题。

高阶交叉特征 在广告场景下，特征交叉的组合与点击率是有显著相关的，例如，“USA”与“Thanksgiving”、“China”与“Chinese New Year”这样的关联特征，对用户的点击有着正向的影响。而本文开发了一个新的算子，来得到交叉



特征：
即，

$x_{l+1} = x_0 x_T w_l + b_l + x_l = f(x_l, w_l, b_l) + x_l$ $x_{l+1} = x_0 x_l^T w_l + b_l + x_l = f(x_l, w_l, b_l) + x_l$

考虑 x_0 为输入的特征及第一层的输入，x 为 第 L 层的输入，我们可以看到它的基本思路还是用矩阵乘法来实现特征的组合。

这是个递推形式算子，所以使用它很容易能得到高于二阶的交叉特征；并且该模型还用了残差的思想，解决网络性能退化的问题；此公式还有一个小的优化技巧，三矩阵相乘那个算子，用乘法结合律先计算后面两个矩阵的积，这样可以减少三分之一的计算复杂度。

DCN和同场景模型对比

在deepFM中， 进行了离散特征嵌入的操作，并且还将嵌入前的离散特征加入到了 FM 层；所以该网络可以看作是传统的 FM 、离散特征嵌入之后的 FM 和基本 DNN 三个模型融合的结果。wide & deep 的思路中，deep 部分的做法和 deepFM 是大相径庭的，关键的 wide 部分其实是离线的特征工程，根据业务场景提前完成了特征交叉等处理，该模型可以看作是 DNN 与离线特征模型的融合结果。而从 DCN 的网络中我们可以发现，deep 部分网络除了使用离散嵌入特征外，还拼接了数值型特征；cross 部分网络直接完成了特征组合，对比 FM 层它可以学到更高阶的组合特征，对比 wide 网络它不需要做线下的特征工程。

xDeepFm

传统的推荐系统中，挖掘交叉特征主要依靠人工提取，这种做法主要有以下三种缺点：1）重要的特征都是与应用场景息息相关的，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；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。目前大部分相关的研究工作是基于因子分解机的框架，利用多层全连接神经网络去自动学习特征间的高阶交互关系。xDeepFM主要是针对DeepFM和DCN的改进。例如FNN、PNN和DeepFM，其缺点是模型学习出的是隐式的交互特征，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。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。

> bit-wise VS vector-wise 假设隐向量的维度为3维，如果两个特征(对应的向量分别为(a1,b1,c1)和(a2,b2,c2)的话）在进行交互时，交互的形式类似于
>
> $f(w1 * a1 * a2, w2 * b1 * b2, w3 * c1 * c2)$ f(w1∗a1∗a2,w2∗b1∗b2,w3∗c1∗c2) 的话，此时我们认为特征交互是发生在元素级（bit-wise）上。如果特征交互形式类似于
>
> $f(w * (a1 * a2, b1 * b2, c1 * c2))$ f(w∗(a1∗a2,b1∗b2,c1∗c2))的话，那么我们认为特征交互是发生在特征向量级（vector-wise）。
> explicitly VS implicitly 显式的特征交互和隐式的特征交互。以两个特征为例
>
> x_i xi 和 x_j xj，在经过一系列变换后，我们可以表示成
>
> $w_{ij} * (x_i * x_j)$ wij∗(xi∗xj) 的形式，就可以认为是显式特征交互，否则的话，是隐式的特征交互。

为了实现自动学习显式的高阶特征交互，同时使得交互发生在向量级上，xDeepFm提出了一种新的名为压缩交互网络（Compressed Interaction Network，简称CIN）的神经模型。在CIN中，隐向量是一个单元对象，因此我们将输入的原特征和神经网络中的隐层都分别组织成一个矩阵，记为X0和Xk。CIN中每一层的神经元都是根据前一层的隐层以及原特征向量推算而来，其计算公式如下：

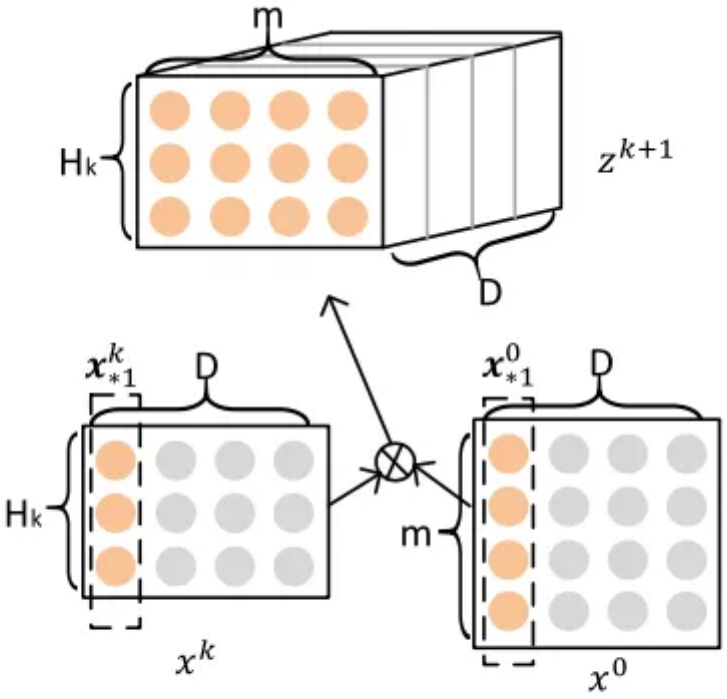
$$X^k_{h,*} = \sum_{i=1}^{H_{k-1}} \sum_{j=1}^{m} W^{k,h}_{ij} (X^{k-1}_{i,*} \circ X^0_{j,*}) \qquad X^k_{h,*} = \sum_{i=1}^{H_{k-1}} \sum_{j=1}^{m} W^{k,h}_{ij} (X^{k-1}_{i,*} \circ X^0_{j,*})$$

其中点乘的部分计算如下：

$$< a_1, a_2, a_3 > \cdot < b_1, b_2, b_3 > = < a_1 b_1, a_2 b_2, a_3 b_3 > \qquad <a1,a2,a3>\cdot<b1,b2,b3> =<a1b1,a2b2,a3b3>$$

第k层隐层含有Hk+1Hk+1条神经元向量。隐层的计算可以分成两个步骤：（1）根据前一层隐层的状态Xk和原

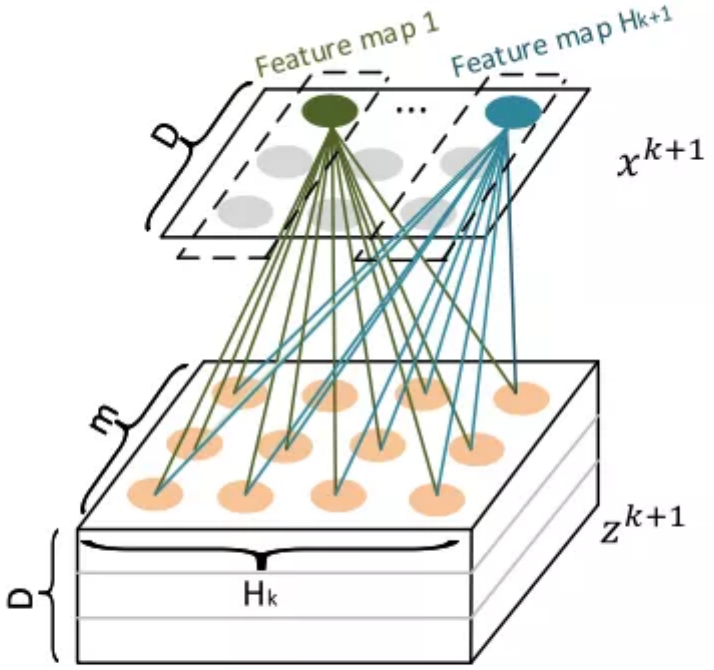
特征矩阵X0，计算出一个中间结果Zk+1Zk+1，它是一个三维的张量，如下图所示：



(a) Outer products along each dimension for feature interactions. The tensor Z^{k+1} is an intermediate result for further learning.

CIN的宏观框架可以总结为下图：

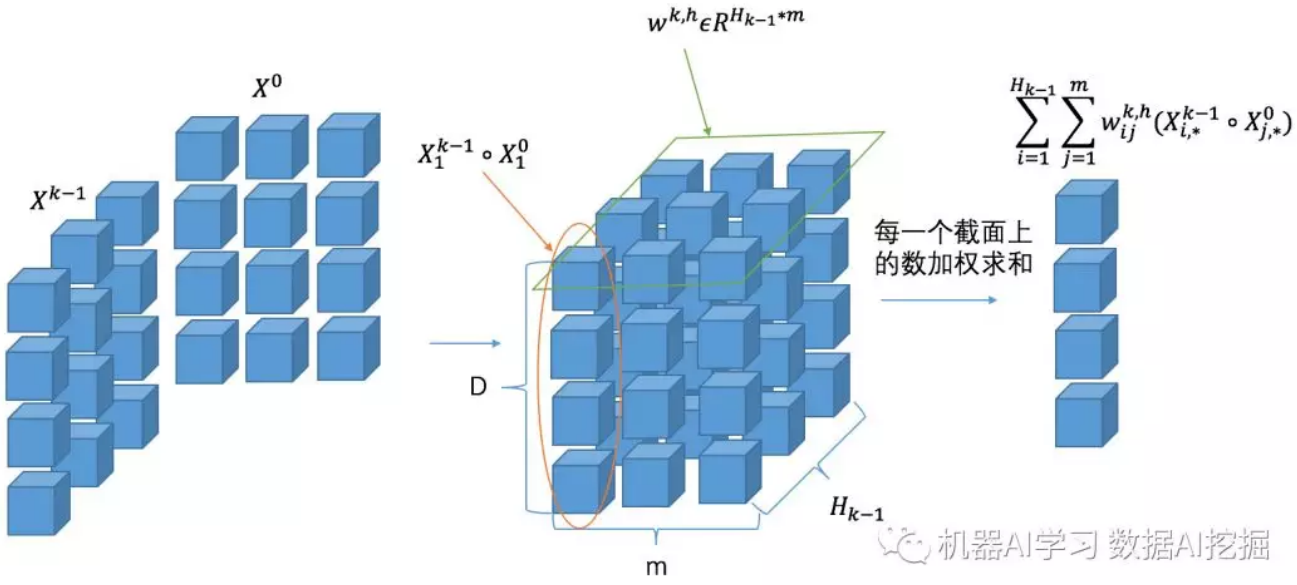
在这个中间结果上，我们用Hk+1Hk+1个尺寸为m * Hkm∗Hk的卷积核生成下一层隐层的状态，该过程如图2所示。这一操作与计算机视觉中最流行的卷积神经网络大体是一致的，唯一的区别在于卷积核的设计。CIN中一个神经元相关的接受域是垂直于特征维度D的整个平面，而CNN中的接受域是当前神经元周围的局部小范围区域，因此CIN中经过卷积操作得到的特征图（Feature Map）是一个向量，而不是一个矩



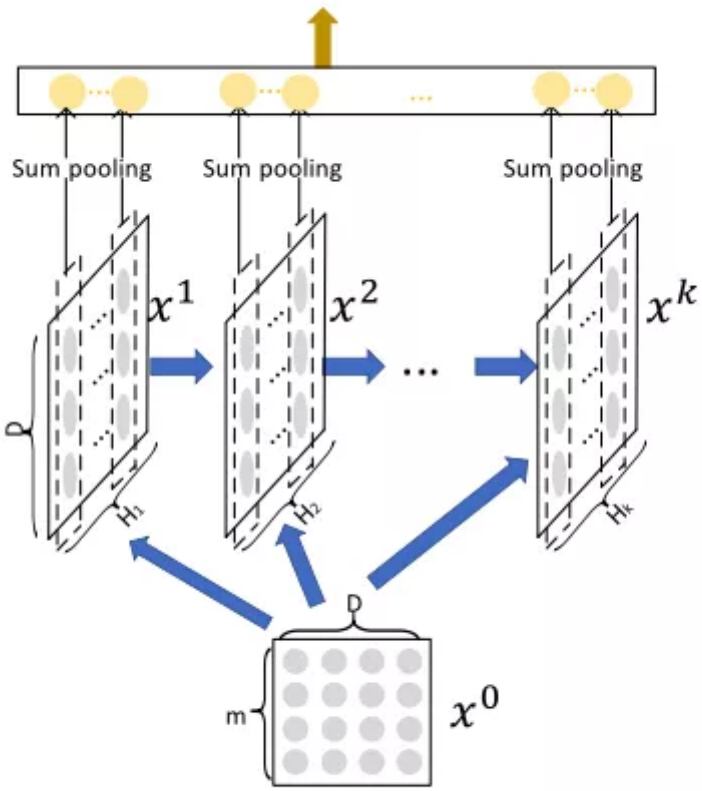
(b) The k-th layer of CIN. It compresses the intermediate tensor $\mathbf{Z}^{k+1}$ to H_{k+1} embedding vectors (aslo known as *feature maps*).

阵。

得到第k层其中一个向量的过程：



CIN的宏观框架可以总结为下图：



(c) An overview of the CIN architecture.

可以看出，它的特点是，最终学习出的特征交互的阶数是由网络的层数决定的，每一层隐层都通过一个池化操作连接到输出层，从而保证了输出单元可以见到不同阶数的特征交互模式。同时不难看出，CIN的结构与循环神经网络RNN是很类似的，即每一层的状态是由前一层隐层的值与一个额外的输入数据计算所得。不同的是，CIN中不同层的参数是不一样的，而在RNN中是相同的；RNN中每次额外的输入数据是不一样的，而CIN中额外的输入数据是固定的，始终是$X^0

。可以看到，CIN是通过（vector−wise）来学习特征之间的交互的，还有一个问题，就是它为什么是显式的进行学习

。可以看到，CIN是通过（vector−wise）来学习特征之间的交互的，还有一个问题，就是它为什么是显式的进行学习？我

们先从X1来开始看，来开始看，X^1的第的第h$个神经元向量可以表示成：

$$X_{1h} = \sum_{i\in[m]} \sum_{j\in[m]} W_{ij1,h}(X_{0i} \cdot X_{0j}) \qquad Xh1 = \sum i\in[m] \ \ j\in[m] Wi,j1,h(Xi0 \cdot Xj0)$$

进一步，X2 X2的第hh个神经元向量可以表示成：

$$\sum_{k=1}^{H_{k-1}} \sum_{j=1}^{H_0} W^{k,h}_{i,j} \cdot (X^{k-1}_h \circ X^0_j) \quad \in \mathbb{R}^D$$

$$X^k_h = \sum_{i=1}^{H_{k-1}} \sum_{j=1}^{H_0} W^{k,h}_{i,j} \cdot (X^{k-1}_i \circ X^0_j) \quad \in \mathbb{R}^{H_k \times D}$$

$$W^{k,h} \in \mathbb{R}^{H_{k-1} \times H_0}, \quad 1 \le h \le H_k$$

$$(X^{k-1} \circ X^0) \in \mathbb{R}^{H_{k-1} \times H_0 \times D}$$

$$X^0 \in \mathbb{R}^{m \times D}$$

因此，我们能够通过上面的式子对特征交互的形式进行一个很好的表示，它是显式的学习特征交叉。

将CIN与线性回归单元、全连接神经网络单元组合在一起，得到最终的模型并命名为极深因子分解机xDeepFM，其结构如下图：

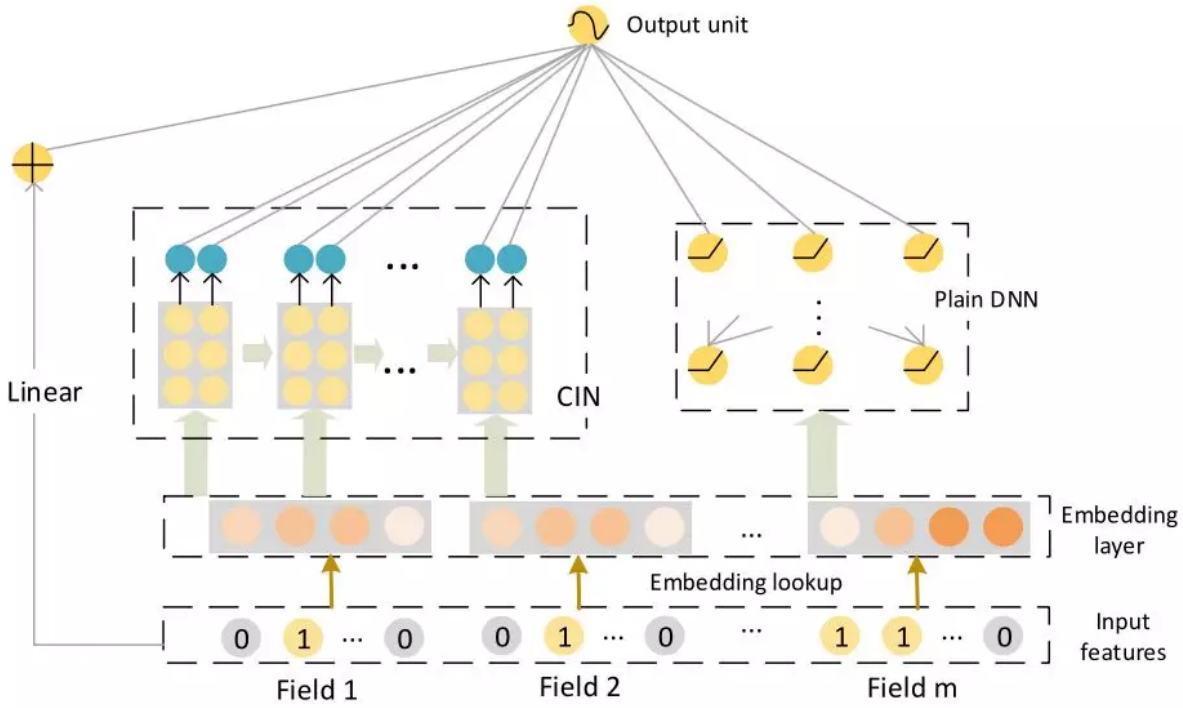


Figure 5: The architecture of xDeepFM.

集成的CIN和DNN两个模块能够帮助模型同时以显式和隐式的方式学习高阶的特征交互，而集成的线性模块和深度神经模块也让模型兼具记忆与泛化的学习能力。值得一提的是，为了提高模型的通用性，xDeepFM中不同的模块共享相同的输入数据。而在具体的应用场景下，不同的模块也可以接入各自不同的输入数据，例如，线性模块中依旧可以接入很多根据先验知识提取的交叉特征来提高记忆能力，而在CIN或者DNN中，为了减少模型的计算复杂度，可以只导入一部分稀疏的特征子集。