CTR竞赛近年所有神经网络方案原理解析

kaggle竞赛宝典 6天前

以下文章来源于炼丹笔记 ，作者二品炼丹师一元

CTR竞赛经典案例

1. FFM冠军:Display Advertising Challenge,kaggle,2014
2. FFM冠军: Click-Through Rate Prediction,kaggle,2015
3. DeepFM Top4: Mercari Price Suggestion Challenge,kaggle,2018
4. ONN冠军：腾讯赛第一届广告大赛，腾讯竞赛平台,2017
5. NFFM/ONN Top10：腾讯赛第二届第三届大赛，腾讯竞赛平台,2018，2019
6. xDeepFM冠军: Categorical Feature Encoding Challenge II, kaggle,2020

CTR神经网络特征交叉汇总

本篇文章把之前一个月学习的网络特征交叉的文章结合自己平时实践的经验梳理一遍,方便今后学习回顾。如果对于CTR/CVR/NLP等最新算法和原理感兴趣的欢迎关注公众号"**炼丹笔记**"。

1. LR
2. Poly2
3. FM
4. FFM
5. MLP
6. WDL(DLRS16)
7. DeepFM(IJCAI17)
8. NFM(IJCAI17)
9. AFM(IJCAI17)
10. xDeepFM(KDD19)
11. TFNET(SIGIR20)
12. ONN/NFFM(Arxiv19)
13. AoAFFM(AAAI20)

14. AutoFIS(KDD20)

上面所有的模型可以大致分为四个部分，特征embedding，特征低阶高阶显示交叉，特征隐式交叉，特征筛选。

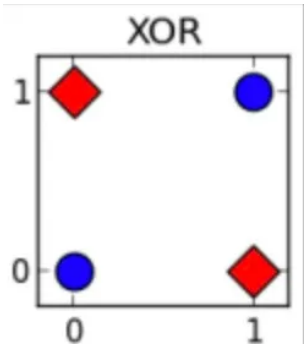
特征Embedding(LR,Poly2,FM,FFM,ONN/NFFM)

LR -> Poly2(特征交叉,基于点)

早期最早我们最为熟知的模型是LR,它的数学形式为：

- **LR**:$g(x) = w_0 + \sum_i w_i x_i$

我们知道LR其实是线性模型，所以很多非线性关系我们的LR模型是没法捕捉的，最典型的就是XOR，单条直线是没法直接分割的。



如果不做任何处理，LR是没法做到很好的区分的。那么怎么做呢？做些交叉特征。

- **Poly2**:$q(x) = w_0 + \sum_i w_i x_i + \sum_i \sum_j$

这么看之前(0,0),(0,1),(1,0),(1,1)就被分配到了四个不同的bin中,对应位置的bin设置高的权重就可以解决上面的问题了。

Poly2 -> FM(升级特征交叉)

下面我们看一个特殊的场景,假设在我们的训练数据集中只有一个负样本(ESPN,Adidas),那么对于Poly2,(ESPN,Adidas)的组合特征对应的权重就会非常大，因为我们只能从该组合中学习它们的关系,这可能不是非常理想，那么如何缓解这样的问题呢?

- **FM**:$g(x) = w_0 + \sum_i w_i x_i + \sum_i \sum_{j=i+1} \langle v_i$

我们将 表示为两个向量的内积,每个向量的维度 ,这样做有什么好处呢？最明显的一点就是：我们可以只通过一个特征的出现去更新另外一个向量了，对应到上面的例子，我们可能还会有其他的组合(ESPN, Nike), (NBC, Adidas),而这些组合都可以用来更新我们的向量，所以我们的预测将会更加精确。我们再看一个看电影的例子,

User/ Movie	Just My Luck	Lady in the Water	Snakes on a Plane	Superman Returns	The Night Listener	You Me and Dupree
Claudia Puig	3		3.5	4	4.5	2.5
Gene Seymour	1.5	3	3.5	5	3	3.5
Jack Matthews		3	4	5	3	3.5
Lisa Rose	3	2.5	3.5	3.5	3	2.5
Mick LaSalle	2	3	4	3	3	2
Toby			4.5	4		1

对于Poly2算法，如果出现了某个用户A没有看过的电影B，那么对应的(UserA,MovieB)的组合就是0,但是如果使用FM的话,结果就不一样了，因为UserA和MovieB的向量可以从其他的特征对中进行学习,所以(UserA,MovieB)的组合可能会非常有意义。

FM -> FFM

FM已经缓解了非常多的问题，那么哪里还可以做出改进呢?

我们看下面几个例子:

- 解释1: 一词多意，一个单词在不同的上下文中的意思是不一样的，**对应过来就是一个特征对应多个隐向量**，在不同的场景下，我们需要不同的隐向量来表示。

- 解释2: 增强模型表示能力，原来我们一个原始特征x_i对应1个$v_i \in R^k$,但是现在我们可以对应多个不同的$v_{i,f_{1,...,F}} \in R^k$,我们一个$x_i$对应$F$个不同的向量,模型的表示能力提升了。

- 解释3: Operation-aware, 任意两个不同field的向量做特征交叉需要在某个特定的领域，也就是说我们需要先将其投影到一个新的交叉空间，例如<地域,品类>交叉,我们需要将地域A投影到品类得到$v_{地域A,品类}$,同时将品类B投影到地域$v_{地域,品类B}$,这样二者都到了同一个交叉空间,然后再在此空间进行交叉(内积等操作)。

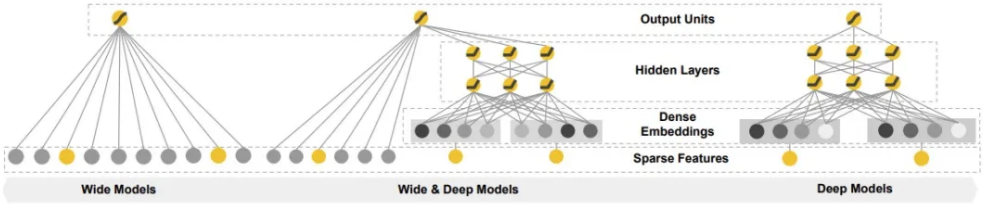
于是我们就可以得到:

- **FFM**: $g(x) = w_0 + \sum_i w_i x_i + \sum_i \sum_{j>i} \langle v_{i,f_j}, v_{j,f_i} \rangle x_i x_j$

隐式&显示特征交叉

不管是Poly2,FM和FFM本质还是只学习了一阶和二阶的特征,那么高阶的特征怎么办？随着Deep模型的发展,我们尝试使用Deep模型去挖掘高阶交互特征,

传统 -> WDL



我们将Deep模型加入到原先的模型中来学习高阶的交叉,

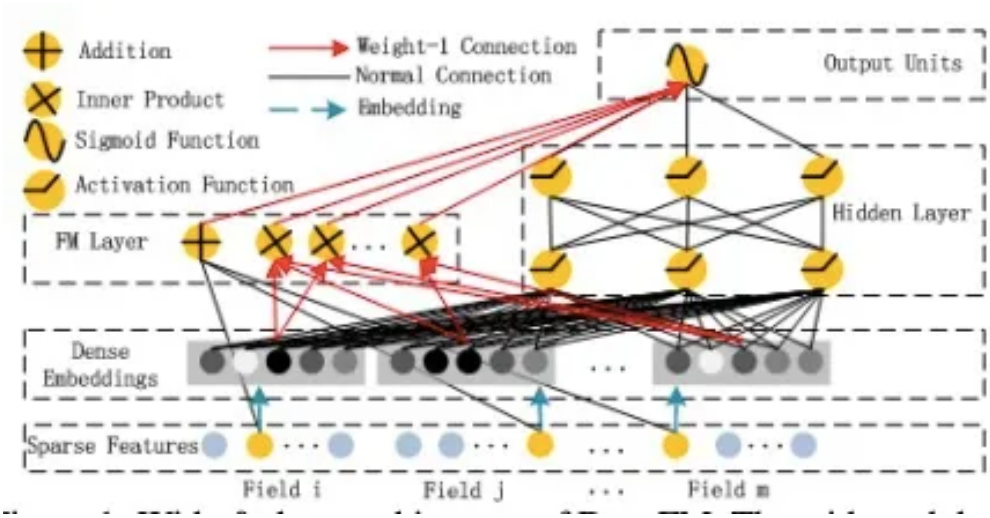
- **WDL**:$\sigma(w_{wide}^T[x,\phi(x)]+w_{deep}^T a^{(l_f)}+b)$

其中:

- $x=[x_1,x_2,\ldots,x_d]$ 是一个d维度的特征;
- $\phi(x)=\prod_{i=1}^d x_i^{c_{ki}}, c_{ki}\in\{0,1\}$
- a^{l_f}为对类别特征embedding之后MLP之后的结果;

WDS -> DeepFM(显示FM+隐式)

起初我们人为把特征embedding之后扔到MLP之中就可以很好地学习到高阶的交叉信息,但其实不然, MLP在特征组合的时候学习的并不是非常好,这个时候我们就需要找到之前我们的一些组合方法。将第一部分的内容加入进来。



- **DeepFM**:$\sigma(y_{FM}+y_{DNN})$

其中:

- $y_{FM}=\langle w,x\rangle+\sum_{j_1=1}^{d}\sum_{j_2=j_1+1}^{d}$
- y_{DNN}:embedding之后concat加入MLP

WDS -> NFM(显示二阶+隐式)

做二阶特征,除了内积的形式外,我们还可以做element-wise的乘法, NFM提出了Bi-Interaction pooling,

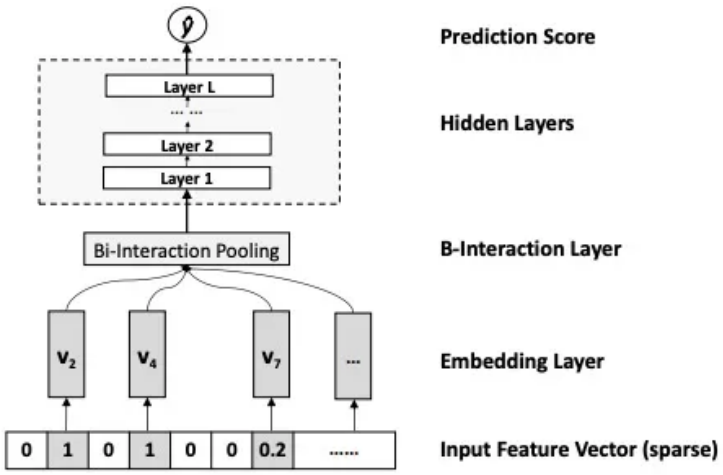


Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

在Bi-Interaction pooling之后连接上一层MLP学习高阶的特征交叉就可以得到:

- **NFM:**

$$y_{NFM} = w_0 + \sum_{i=1}^{n} w_i x_i + f(x) \rightarrow y_{NFM} = w_0 + \sum_{i=1}^{n} w_i x_i + h^T \sigma_L(W_L(\dots\sigma_1(W_1 f_{BI}(V_x) + b_1)\dots) + b_L)$$

其中:

- $f_{BI}(V_x) = \sum_{i=1}^{n}\sum_{j=i+1}^{n} x_i v_i \odot x_j v_j,$

NFM算另外一种结合两种将MLP与显示交叉结合的方式(之前是分两个分支),而实验中,我们发现后者可能效果更好一些。在模型的早期,加入Bi-interaction pooling并且拼接在之前的模型中一般是可以提升模型的效果的。

DeepFM/NFM -> TFNET(显示高级二阶+隐式)

之前的显示二阶特征交叉的形式都相对简单, 例如:

- **内积交叉**(FM,DeepFM的思想):

这么看我们其实每个元素在做内积的时候都乘上了一个1，我们是不是可以把这个1换成更加通用的t_{ij}呢,答案当然是可以的。

- **加权交叉**(NFM的element-wise乘法):

我们这么做忽略了两个向量不同元素之间的交叉,例如v_{i1}和v_{j2}此类的交叉，于是我们就想着能不能再扩展一下，所以我们就得到:

- **混合加权交叉**:

能不能再扩展一下(张量的思想),

- **加入多层语义信息**:

最终我们得到$S \in R^{q*m}, q = n*(n-1)/2$，$n$为我们模型的特征个数,这基本就是TFNET的核心。

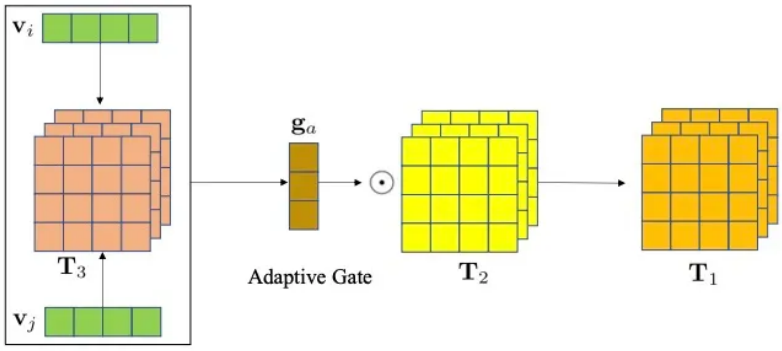


Figure 2: The architecture of how to learn the weighted operating tensor $\mathbf{T}_1$.

此处,作者认为不同层的语义应该是不一样的,所以又加入了attention 来学习T_1。

- $T_1 = g_a \odot T_2$,

其中,

- $T_2 \in R^{d*m*d}$， $g_a \in R^m$,
- g_a为不同语义层的权重；,

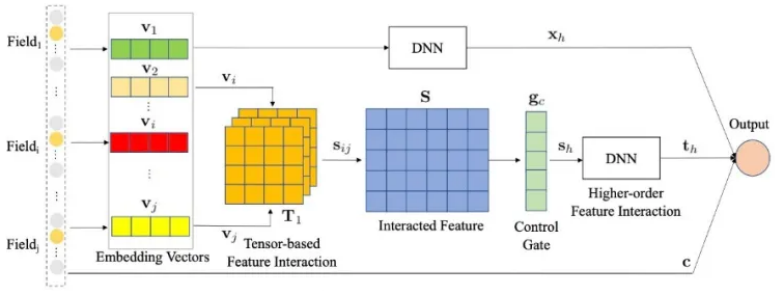


Figure 1: The architecture of our proposed TFNet model, which consists of the following parts: sparse input layer, embedding layer, tensor-based feature interaction layer, higher-order feature interaction layer and the final output.

TFNET的输出为:

其中:

- c:原始特征,
- x_h:embedding加入MLP层的输出,隐式高阶信息
- t_h:高级显示二阶的交叉(这边最后还用g_c进行了压缩)

DeepFM -> ONN/NFFM(显示二阶+隐式)

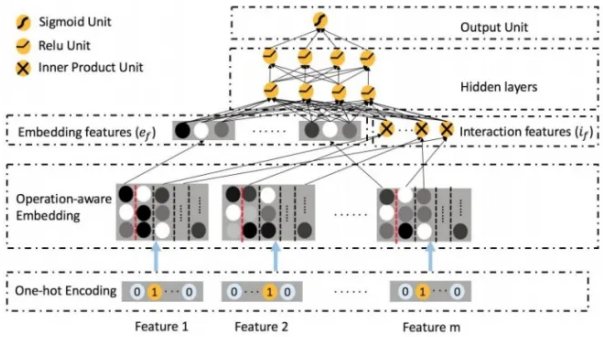


Figure 1: The architecture of the ONN model. From the bottom-up perspective, the model first uses an operation-aware embedding layer to map high-dimensional sparse features into low-dimension representations. After that, "copy" operations and "inner-product" operations are performed on corresponding features respectively. Finally, an MLP learns high-order feature interactions and produce the output.

在FM和FFM中,我们发现了FFM的表示相较于FM的几种好处,而且早期实验中也显示了FFM比FM却是要好很多。

所以类似的，既然FM和FFM都可以认为是Embedding，一个映射为单个dense向量，一个映射为多个dense向量，所以ONN就是将Embedding部分的特征修改，从FM的形式转化为FFM，后面仍然是一样的做cross的内积。

其中:

- $\bar{y} = \sigma(MLP([e_f, i_f]))$, 这边的$\sigma(x) = \frac{1}{1+exp(-x)}$

其中:

- e_f为FFM的Embedding，
- i_f为cross的内积, ,

ONN是第一届腾讯赛冠军提出来的，在后两届的腾讯赛中，第一名也都有用到该模型，所以FFM的embedding是非常值得借鉴的。

此处还有一个小细节就是作者将embedding的特征和cross的特征进行concat输入MLP。

DeepFM -> xDeepFM(显示高阶+ 隐式)

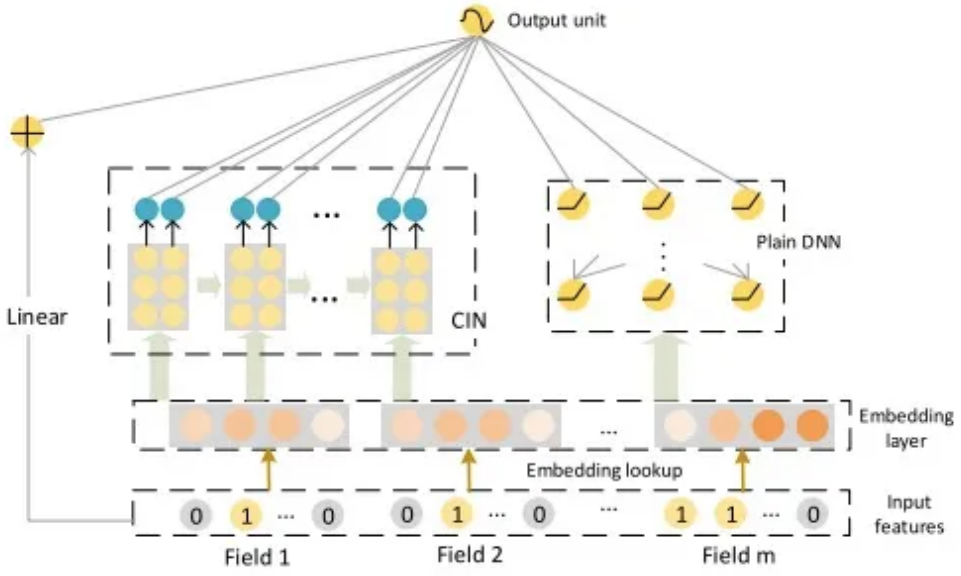


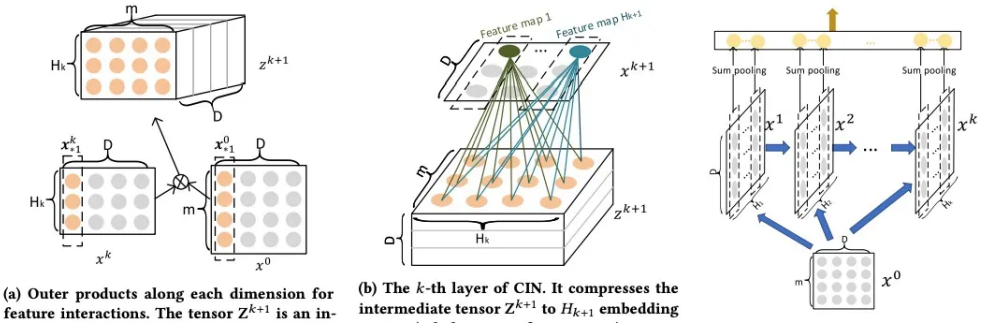
Figure 5: The architecture of xDeepFM.

上面所有的模型要么是直接用embedding之后用MLP模型进行隐式高阶特征在于显示的二阶交叉相加，要么就是二阶显示交叉之后再与一阶的embedding进行concat后加入MLP进行隐式学习。却还没有一个是显式学习高阶特征交叉的，那么这么做会有提升吗？xDeepFM告诉我们是的！！！

- **xDeepFM**:

其中p^+是CIN层的输出(显示特征交叉)，

- $p_i^k = \sum_{j=1}^{D} X_{i,j}^k$,



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

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

(c) An overview of the CIN architecture.

Figure 4: Components and architecture of the Compressed Interaction Network (CIN).

注意：xDeepFM还多来一层原始层的信息$w_{linear}^T a$,这边一般也是可以带来提升的。剩下的就是显示的高阶特征交叉与隐式的相加，和之前的类似。

ONN/NFFM VS xDeepFM

ONN和xDeepFM是目前在数据竞赛中经常用到的模型(据本人所知,2020年之前的CTR竞赛最多的处理categorical的特征还是这两类模型)，19年的腾讯赛冠军也是用的NFFM模型,二者的对比如下(18年的腾讯数据),整体来看，NFFM的效果要略好于xDeepFM，但二者融合效果还是极棒的!

	复赛数据	复赛+初赛数据	ensemble
nffm(单模型)	0.764	0.770	0.7734
xdeepfm(单模型)	0.763	0.766	0.770
nffm+xdeepfm	0.768	-	0.7740

无效特征&冗余特征筛选

FM -> AFM(噪音信息处理)

枚举式特征交叉的问题1：从上面所有模型的构建我们可以看到,所有的模型都是枚举式的二阶交叉,枚举的话毫无疑问就会带来非常大的问题，特征冗余，会带出非常多的无用的特征(Noise),这在实践中也是类似的，随机加入多个高斯噪音，模型的效果可能会下降（虽然很多时候下降不是非常多，但基本都是下降的），那怎么办呢?

- 解法1: 降低不重要特征的权重。

所以我们就看到了最早的AFM模型。

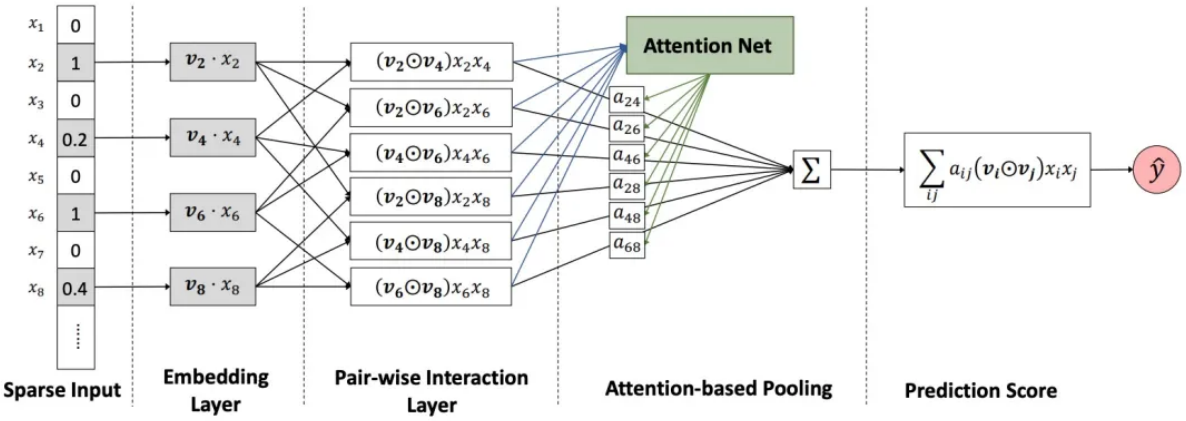


Figure 1: The neural network architecture of our proposed Attentional Factorization Machine model.

AFM的数学表示形式为:

- $y_{AFM}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + p^T \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij}(v_i \odot v_j) x_i x_j$

其中,

- $a_{ij} = \frac{exp(a'_{ij})}{\sum_{(i,j)\in R_x} exp(a'_{ij})}$, $a'_{ij} = h^T Relu(W(v_i \odot v_j)x_i x_j + b)$,

其中a_{ij}就是我们特征i和特征j交叉的attention分数，用来评估每个交叉特征的重要性。

注意如果p全部为1,删去a_{ij}我们就还原得到了FM,所以此处认为是FM到AFM的衍生。

FM/DeepFM/PNN -> AutoFIS

- 解法2: 先找到不重要的特征交叉(权重直接置为0),然后固定重新训练。

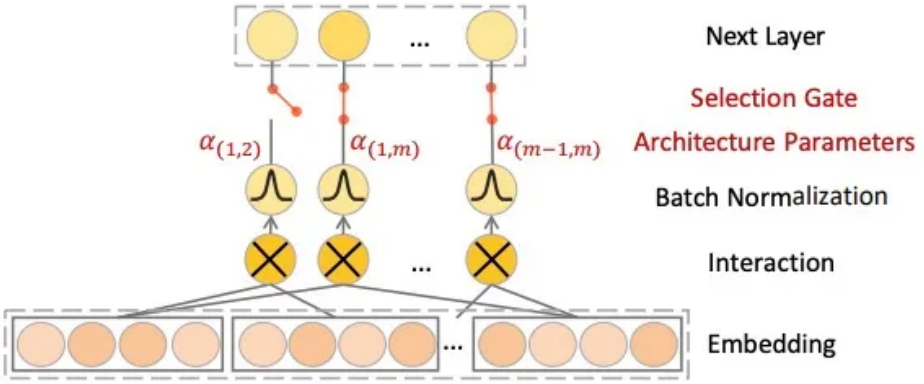


Figure 2: Overview of AutoFIS

Step1:寻找到不重要的特征交叉,

Step2:重新训练:,

- 当$\alpha^*_{(i,j)} = 0$时,我们有$\mathcal{G}_{(i,j)} = 0$, 否则$\mathcal{G}_{(i,j)} = 1$,此时,$\alpha_{(i,j)}$就是我们的attention score, 由$\mathcal{G}_{(i,j)}$来做交叉特征的筛选。

我们发现和AFM的方案不同之处是，我们$\alpha_{(i,j)}$不需要通过attention的方式计算得到的,而且AutoFIS是两步训练,在第一部分先过滤,第二步重新训练,个人感觉效果会更好一点。AFM的思想和AotuFIS的思想都非常通过都可以直接嵌入到存在二阶交叉的模块中。

AFM/FFM -> AoAFFM(噪音+冗余信息处理)

枚举式特征交叉的问题2：除了枚举带来的噪音数据会导致我们的模型的效果下降之外,还有一类信息,就是冗余信息也会带来模型效果的下降，此处的冗余我们指：比如有一个交叉特征A非常重要，是非常强的特征，但是又有一个交叉特征B也非常重要，但是A特征和B特征的相关性几乎为1，也就是说A特征基本上已经包含了B特征的信息，其实只保留一个特征就好了，保留两个不仅会带来内存的消耗，还会导致我们的模型效果下降。实践中,最为常见，而且很多时候冗余信息带来的模型性能下降的程度比噪音数据会大。

- 解法: 降低不重要特征的权重 & 希望将冗余信息的特征的权重调低。

所以我们就看到了最早的AoAFFM模型中的AoA希望处理的事情,当然此处作者不是基于FM做的,是基于FFM做的。

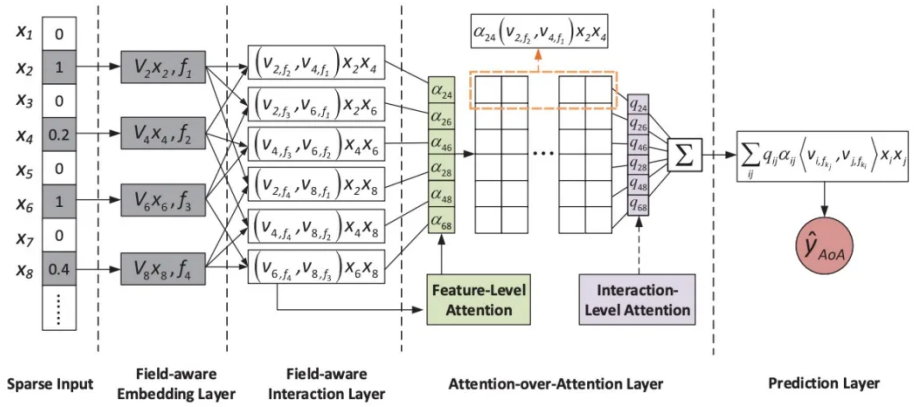


Figure 1: The neural network architecture of Attention-over-Attention Field-aware Factorization Machine

于是我们有：

其中α_{ij}为特征层的attention, q_{ij}为交互层的attention权重。

如果忽略embedding层的操作，我们发现相较于AFM,AoAFFM多了一层Interaction-Level的Attention。而它的作用就是希望降低冗余特征的影响：

> 在Feature-level的Attention网络中，我们可以学习每个特征的权重，**但是却没法降低我们冗余特征带来的影响。**所以两个类似的特征最终会有类似的attention，为了判别这种冗余的交互特征，我们使用interaction-level的attenion来处理，$Q \in R^|\mathcal{F||}
> ,从上面的网络框架图中，我们发现因为我们的Q$是不依赖于前面的信息，所以即使对于相似的embedding向量，我们也有可能会有不同的权重，所以这个方法可以缓解特征冗余的问题。

小结

从这些文章看，目前神经网络的特征交叉研究的方向主要是：①.深化二阶特征的交叉(内积,element-wise,加入attention系数,加入张量细化);②.探索显示的高阶交叉; ③.探讨噪音&冗余特征的处理; ④.探索embedding的合理性;

参考文献

1. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems:https://arxiv.org/abs/1803.05170
2. AoAFFM:Attention-over-Attention Field-Aware Factorization Machine：https://www.researchgate.net/publication/342540145_Attention-over-Attention_Field-Aware_Factorization_Machine
3. AFM:Attentional Factorization Machines_Learning the Weight of Feature Interactions via Attention Networks：https://arxiv.org/abs/1708.04617

4. ONN:Operation-aware Neural Network for User Response Prediction：https://arxiv.org/abs/1904.12579
5. NFM:Neural Factorization Machines for Sparse Predictive Analytics：https://dl.acm.org/doi/10.1145/3077136.3080777
6. AutoFIS:Automatic Feature Interaction Selection in Factorization Models for CTR Prediction：https://arxiv.org/abs/2003.11235
7. TFNET:Multi-Semantic Feature Interaction for CTR Prediction：https://arxiv.org/abs/2006.15939
8. DeepFM：https://arxiv.org/abs/1703.04247
9. Field-aware Factorization Machines for CTR Prediction：https://www.csie.ntu.edu.tw/~cjlin/papers/ffm.pdf
10. Factorization Machines:https://www.csie.ntu.edu.tw/~b97053/paper/Rendle2010FM.pdf
11. Wide & Deep Learning for Recommender Systems:https://arxiv.org/abs/1606.07792
12. Training and testing low-degree polynomial data mappings via linear SVM:https://www.jmlr.org/papers/volume11/chang10a/chang10a.pdf