DNN的不可解释性是经常被诟病的一点。在CTR预估中，我们常常需要分析特征的重要程度，而DNN直接把所有特征都concat起来，其特征交互都是bit-wise的，根本无法判断每个特征的重要性。今天介绍的两个模型 -- InterHAt和FiBiNET都是使用了attention机制来得到特征的重要性的。

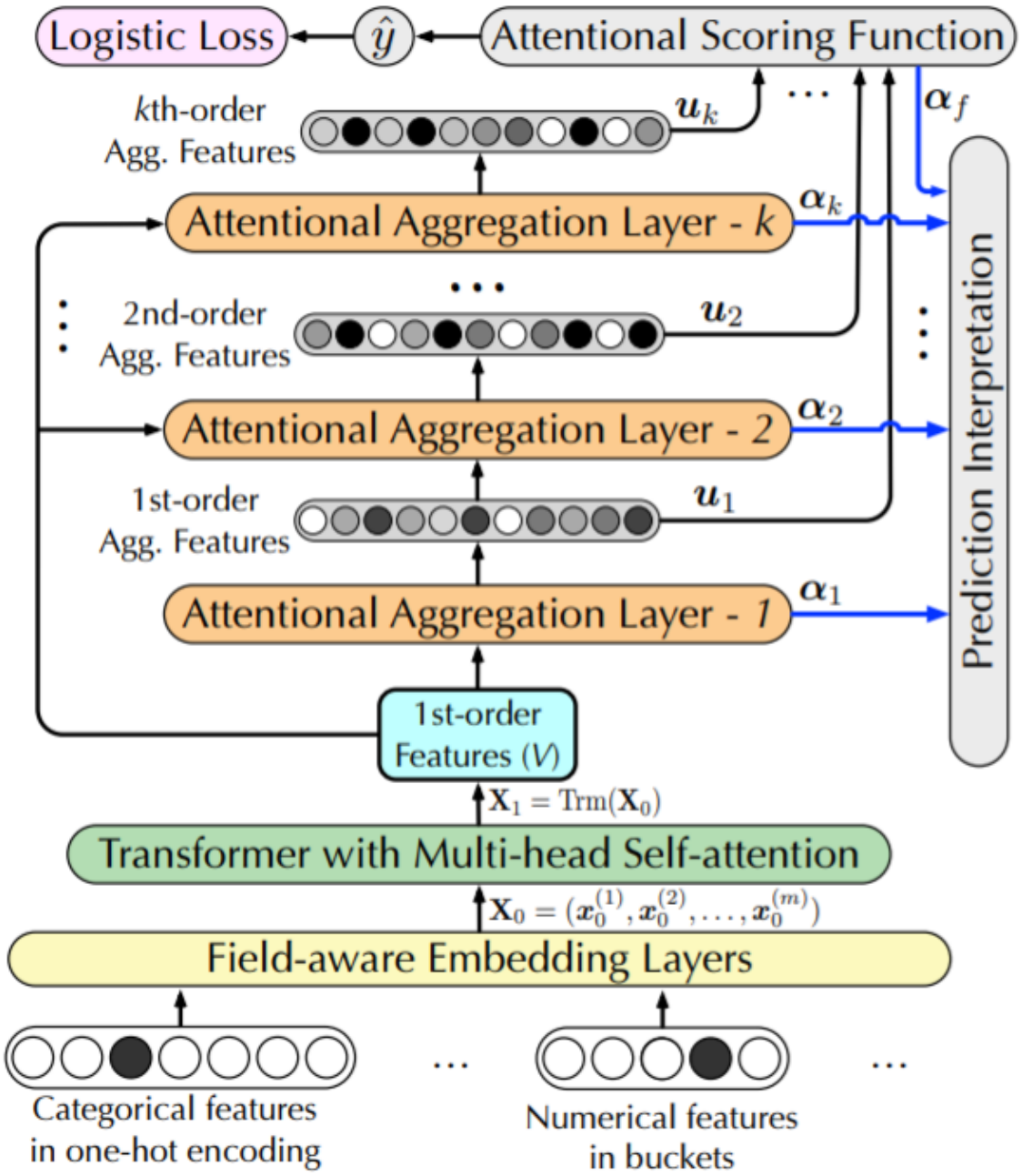
深度学习的可解释性，虽然不能够带来直接的收益，但是其重要性不言而喻。尤其是在诸如医疗、金融推荐领域，一些不可靠的推荐算法会推荐热门但是并不有用的链接，造成健康和财产损失（如魏则西事件）。

这两个模型本身并不像之前介绍的那些模型一样那么有名，只不过因为我曾经用过，所以记录下来以免忘记。用**attention机制对不同特征进行重要性加权**、同时获得模型的ante-hoc可解释性也是很值得借鉴的一种方法。

0x01. InterHAt [WSDM, 2020]

出自论文 Interpretable Click-Through Rate Prediction through Hierarchical Attention. 主要的卖点有两个：良好的**可解释性**，以及训练速度提升(high **efficiency**). 可解释性是针对DNN来说的，DNN中的网络权重和激活值这些都是难以解释的。训练速度上是跟其他显式捕捉高阶交叉特征的模型相比，例如xDeepFM中的CIN（立方级别复杂度）。

1. 网络结构



1.1 Multi-head Self-Attention

embedding之后，首先经过multi-head self-attention层，使用多头来捕捉在不同子空间的特征交互。经过多头注意力之后得到的是向量 X_1，是"polysemy-augmented feature".（这个思想和AutoInt类似，但是要知道，经过self-attention之后，每个位置上就已经是**二阶特征交互**了，这就已经损失了很多可解释性，不过好在self-attention的权重也是可以拿到的。但是文中把这个 X_1 当成一阶特征，这个我认为有不妥。）

1.2 Hierarchical Attention

通过枚举不同的高阶特征组合来得到特征交互是复杂度很高的（组合爆炸），例如xDeepFM中的CIN模块复杂度是**立方**级别。而InterHAt中使用了一种复杂度较低的方法：计算出第i层特征embedding的一个"**聚合**" u_i，然后再让 u_i 去和 X_1 做交互，得到i+1阶特征 X_{i+1} .

那么，如何得到第i层的聚合呢？也是使用attention的方法，对第i层交叉特征做加权求和。假设第i层总共有m个embedding，每个embedding的size 为d，那么我们用下述方法计算出第i层的一个"聚合"u_i：

Mathematically, given the i-th feature matrix $\mathbf{X}_i = \left(\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(m)}\right)$, its attentional aggregation representation $\boldsymbol{u}_i$ is

$$\boldsymbol{u}_i = \text{AttentionalAgg}(\mathbf{X}_i) = \sum_{j=1}^{m} \alpha_i^{(j)} \boldsymbol{x}_i^{(j)}, \qquad (1)$$

即，计算第i层所有特征embedding的加权平均。最后输出的$u_i \in R^d$, 其中d是embedding size。这样，就将m个d维特征embedding"压缩"表示成了1个d维特征embedding。

权重计算公式如下，$\alpha_i^{(j)}$表示第i层的第j个特征权重。

$$\alpha_i^{(j)} = \frac{\exp\left(\boldsymbol{c}_i^T ReLU(\mathbf{W}_i \boldsymbol{x}_i^{(j)})\right)}{\sum_{j' \in F} \exp\left(\boldsymbol{c}_i^T ReLU(\mathbf{W}_i \boldsymbol{x}_i^{(j')})\right)},$$

这就是一个注意力计算方法，相当于先把第i层的第j个特征embedding $x_i^{(j)}$经过一层MLP，然后再过Relu激活函数，然后再过一层MLP转换成一个数；然后对所有的特征embedding都做同样的操作，最后计算softmax值。其中$\boldsymbol{W_i}, \boldsymbol{c_i}$是第i层的可学习参数，这个参数量比较小，说明attention net是比较轻量级的。

我们把 $\boldsymbol{u_i}$ 当成第i层交叉特征的一个"**代表**"，因为u_i就是第i层所有特征embedding的加权平均，可以代表**第i阶**特征。让u_i代表所有的第i层交叉特征去和 $\boldsymbol{X_1}$ 做交互，得到第 i+1 层交叉特征 $\boldsymbol{X_{i+1}}$：

$$\boldsymbol{x}_{i+1}^{(j)} = \boldsymbol{u}_i \circ \boldsymbol{x}_1^{(j)} + \boldsymbol{x}_i^{(j)}, \quad j \in \{1, \ldots, m\},$$

其实可以看到，这里的第i+1层其实包含的是第i层特征和i+1阶交叉特征，因为还有一个残差连接，所以第k层并只是第k阶交叉特征，因此可解释性并不那么好。

1.3 Output

最后的输出层是对所有阶的聚合 $\boldsymbol{u_1, \ldots u_k}$再做一个**attention**：

$$\boldsymbol{u}_f = \text{AttentionalAgg}(\mathbf{U}) = \sum_{j=1}^{k} \alpha_f^{(j)} \boldsymbol{u}_j,$$

$$\alpha_f^{(j)} = \frac{\exp\left(\boldsymbol{c}_f^T ReLU(\mathbf{W}_f \boldsymbol{u}_j)\right)}{\sum_{j' \in \{1,\ldots,k\}} \exp\left(\boldsymbol{c}_f^T ReLU(\mathbf{W}_f \boldsymbol{u}_{j'})\right)},$$

以得到不同阶特征的重要程度。最后的输出logit就是

$$\boldsymbol{u}_f$$

过一层MLP。

我认为这样做的一个好处是可以判断**到底要用多少阶的交叉特征**。之前说过，交叉特征的阶数过高会导致模型复杂度太高，那么如果我们知道不同阶特征的重要程度，就可以比较好的做**阶数的剪枝**。

2. 可解释性分析

这个模型可以说是将attention用到了极致 -- 第一层的self-attention先做了一次交互，得到了融合其他特征信息的表示；然后每层的feature都计算了attention进行融合；最后，不同阶的特征还计算了attention，得到不同阶特征的重要程度。这样的话，理论上就可以进行case study，来判断模型在做出预估的时候更加关注什么特征、第几阶的特征。

我们通过给第k阶特征"聚合" u_k 的权重$\alpha_f^{(k)}$得到第k阶特征的重要程度；

在聚合第k阶特征的时候，我们也给不同的特征embedding 以不同的权重 $\alpha_k^{(j)}$，所以这样就可以得到第k阶的哪个特征更为重要了。

例如文中给出MovieLens上的一个case study：

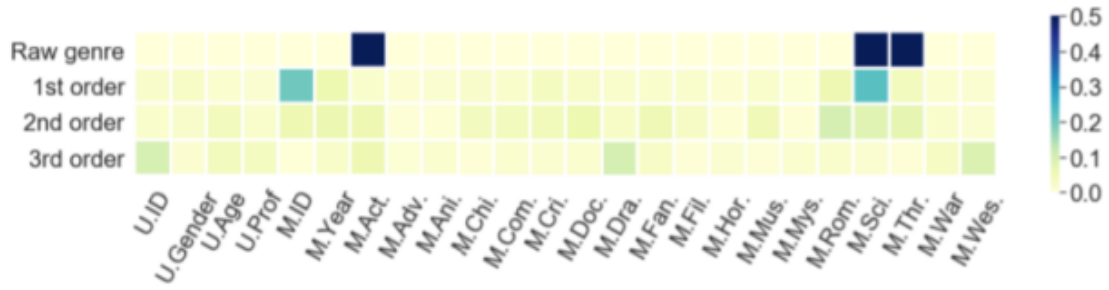


Figure 5: Attention weights of a first-order salient feature example (*The Terminator*, 1984)

预测《终结者》电影是否被一名用户点击。发现一阶特征M.ID和M.Sci最为重要，可能是这部电影本身知名度够高、而且是科幻电影，这是两个最重要的特征。高阶特征并不是很明显，可能是因为用户根据电影本身、和它是科幻电影这一个特征就能够判断是否要点击了。

文中还举了一个三阶特征最重要的case：1999年的《玩具总动员2》.

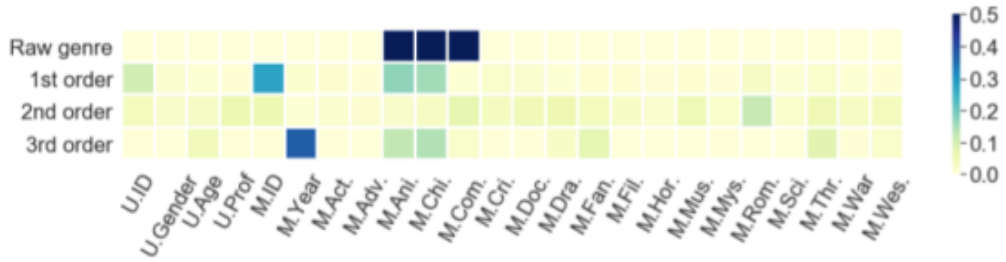


Figure 7: Attention weights of a third-order salient feature example (*Toy story 2*, 1999)

发现year, animation, children是最重要的三阶交叉特征。作者的解释是，1999年是动画电影蓬勃发展的一年。

虽然我们还是不能准确的判断到底每个交叉特征都有多重要（因为multi-head self-attention层已经做了个二阶交叉），但是有了这么多的attention，我们还是能解释很多东西了，比DNN这个黑盒要强许多。

3. 实际应用结果

在实际应用上，**InterHAt的AUC和xDeepFM**差不多，而且提供了良好的可解释性，复杂度也降低了不少。

0x02. FiBiNET [Recsys, 2019]

出自论文 FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction

这篇文章的两个卖点是：通过**SE-block**来获得特征重要度；以及**用双线性特征交互**(Bilinear feature Interaction)来获得比内积、哈达玛积更精细的二阶特征交互。

2.1 模型结构

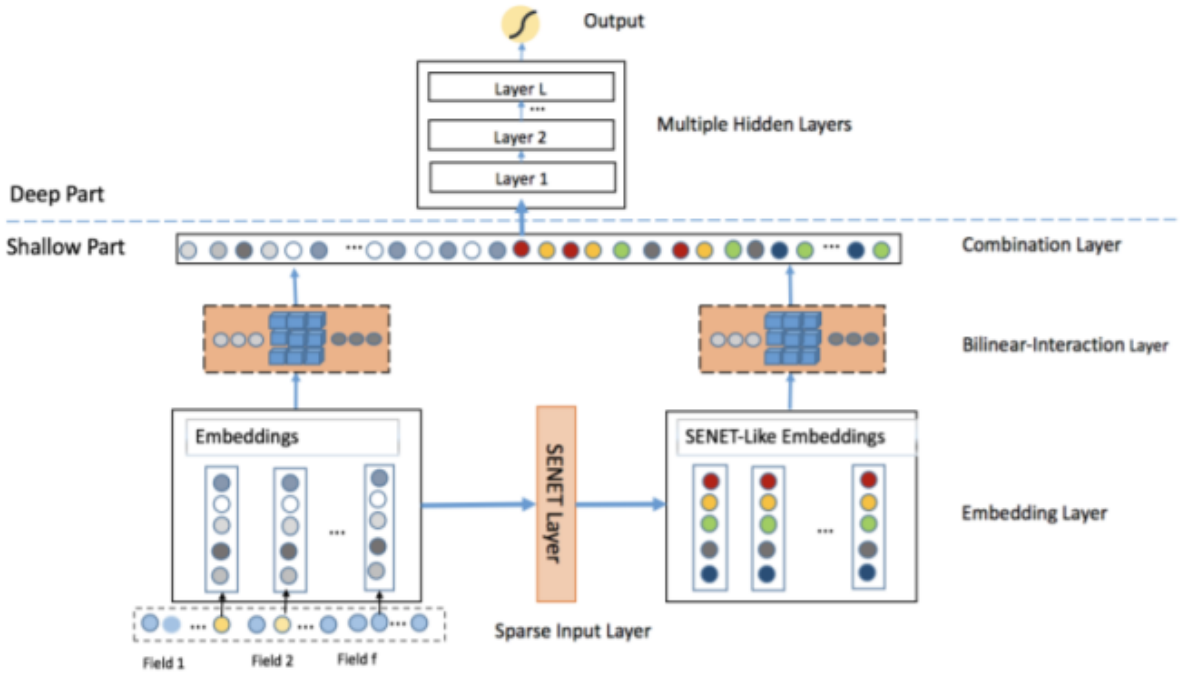


Figure 1: The architecture of our proposed FiBiNET

图中可以看到相比于我们熟悉的基于深度学习的CTR预估模型，主要增加了SENET Layer和Bilinear-Interaction Layer两个结构。从图中可以看出，我们先对embedding之后的特征经过SENET求出每个特征的重要程度，然后对特征进行加权。之后，用双线性交互层来进行比点积、哈达玛积更复杂的二阶特征交叉。分别对原始的特征embedding和经过SENET后的embedding求二阶特征交叉，然后把结果拼接起来输入DNN，试图捕捉隐式高阶特征交叉。

下面就针对这SENET和双线性交互层这两个结构进行简单的说明。

2.1.1 SENET Layer

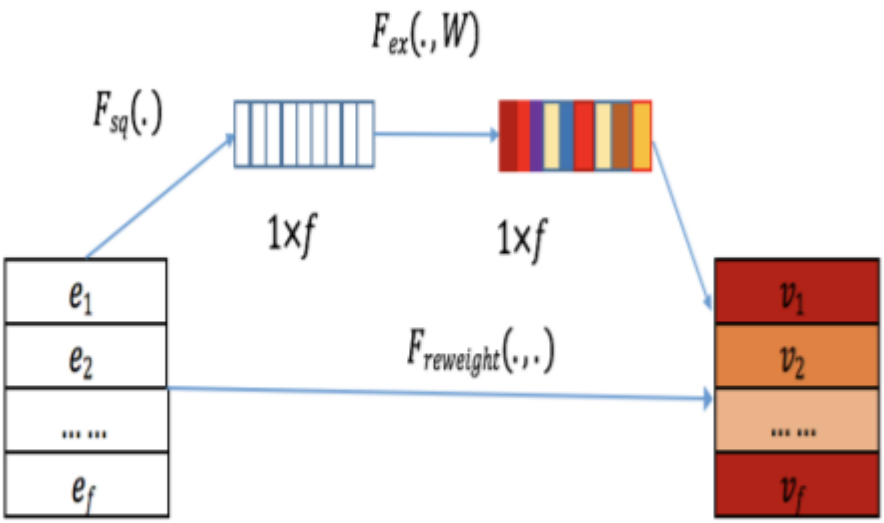


Figure 2: The SENET Layer

使用特征的embedding向量作为输入，计算特征**权重**向量$A = [a_1, \ldots, a_i, \ldots a_f]$,最后将原始特征组embedding向量$E$乘上$A$得到一组新的embedding向量$V = [v_1, \ldots, v_i, \ldots v_f]$具体来说，分为3个步骤：

- squeeze: **平均池化**的方式计算得到：$z_i = F_{sq}(e_i) = \frac{1}{k}\sum_{t=1}^{k} e_i^{(t)}$。当然，也可以使用最大池化的方式。
- excitation: 使用两层的神经网络来学习。第一层为一个维度缩减层，第二层为维度提升层。形式化表示为：$A = F_{ex}(Z) = \sigma_2(W_2\sigma_1(W_1 Z))$，其中$A \in R^f$是一个向量，$\sigma_1$和$\sigma_2$是激活函数，需要学习的参数为$W_1 \in R^{f \times \frac{f}{r}}$，$W_2 \in R^{\frac{f}{r} \times f}$，$r$r为缩减比例参数。

$$A = F_{ex}(Z) = \sigma_2(W_2\sigma_1(W_1 Z))$$

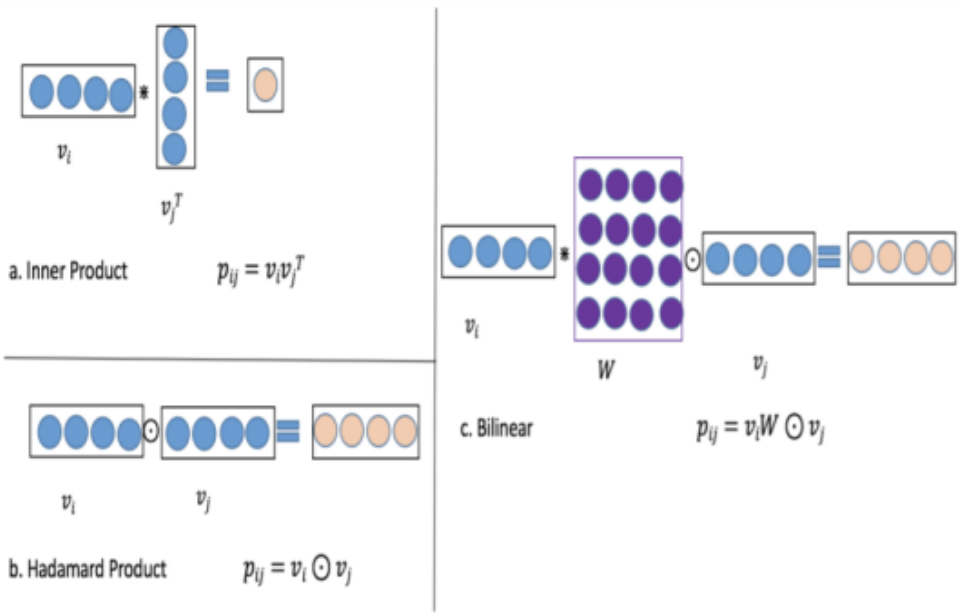
- reweight: 根据excitation层得到的权重对原始特征进行加权。

整个过程和原始的SENET论文并无差别，只不过原论文是针对通道加权，这里是对特征加权。

2.1.2 双线性交互层

内积生成一个scalar，往往用在浅层网络中，如FM和FFM；哈达玛积生成一个向量，往往用在深层网络，如FNN，AFM等。而本文认为，内积和哈达玛积的表达能力（model capacity）仍然不够强。

文章提出结合内积和哈达玛积并引入一个额外的参数矩阵W来学习特征交叉：



交叉向量

$$p_{ij}$$

可以通过一下三种方式计算得到：

- Field-All Type：$p_{ij} = v_i \cdot W \odot v_j$。这种情况下，所有特征组交叉时共享一个参数矩阵 W ，额外参数量为 $k \times k$。

- Field-Each Type：$p_{ij} = v_i \cdot W_i \odot v_j$ 这种情况下，每个field i 维护一个参数矩阵 W_i ,额外参数量为 $(f-1) \times k \times k$
- Field-Interaction Type: $p_{ij} = v_i \cdot W_{ij} \odot v_j$ 。每对交互特征 p_{ij} 都有一个参数矩阵 W_{ij} ，额外参数量为 $\frac{f(f-1)}{2} \times k \times k$

2.1.3 Output层

最终，交叉层由**原始**的特征组embedding向量 E 以及SENET层输出的embedding向量 V 分别得到交叉向量 $p = [p_1, \ldots, p_i, \ldots p_n]$ 和 $q = [q_1, \ldots, q_i, \ldots q_n]$ ，其中 $p_i, q_i \in R^k$ 为向量。对二者进行拼接操作，得到结果向量，输入到DNN中，得到输出logit。

2.2 实际应用

其实，在其他模型上也都可以借鉴SE-layer的思想，在输入到下一层网络之前先对特征进行注意力加权，一般都会有一个稳定的提升。同时还能够获得特征的可解释性 -- 究竟是哪些特征最为重要。