结构化数据预测问题

【数据集样例】

Data fields

- id: ad identifier
- click: 0/1 for non-click/click
- hour: format is YYMMDDHH, so 14091123 means - 23:00 on Sept. 11, 2014 UTC.
- C1 -- anonymized categorical variable
- banner_pos
- site_id
- site_domain
- site_category
- app_id
- app_domain
- app_category
- device_id
- device_ip
- device_model
- device_type
- device_conn_type
- C14-C21 -- anonymized categorical variables

	id	click	hour	C1	banner_pos	site_id	site_domain	site_category	app_id	app_do
0	1000009418151094273	0	14102100	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8
1	10000169349117863715	0	14102100	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8
2	10000371904215119486	0	14102100	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8
3	10000640724480838376	0	14102100	1005	0	1fbe01fe	f3845767	28905ebd	ecad2386	7801e8
4	10000679056417042096	0	14102100	1005	1	fe8cc448	9166c161	0569f928	ecad2386	7801e8

难点

- 传统模型学习能力有限
- 类别特征多，矩阵稀疏，深度模型很难更新
- 交叉特征对预测很有帮助，但手工设计交叉特征耗时费力

传统模型

- 逻辑回归
  - 线性模型，模型能力有限

    - 难以利用特征之间的关系
- 矩阵分解(Matrix Factorization)
    - 对每个实体进行embedding
- 因子分解机(Factorization Machines)
    - 更好的获得特征对之间的交叉特征信息
- 序列模型(RNN)
    - 将一个用户的浏览/点解行为看作序列数据
    - 利用历史行为的依赖信息
    - 训练好的模型，W是不变的，但用户的行为偏好会变化

深度模型

优点 模型能力强大，可学习特征间的低、高阶交叉特征

缺点 可解释性差(通过手工设计网络结构、加入注意力模块缓解)

1. A Convolutional Click Prediction Model(CCPM, 2015ACM)

- CNN可以获得不同元素之间的交叉信息
- 可以提取序列中的局部-整体特征

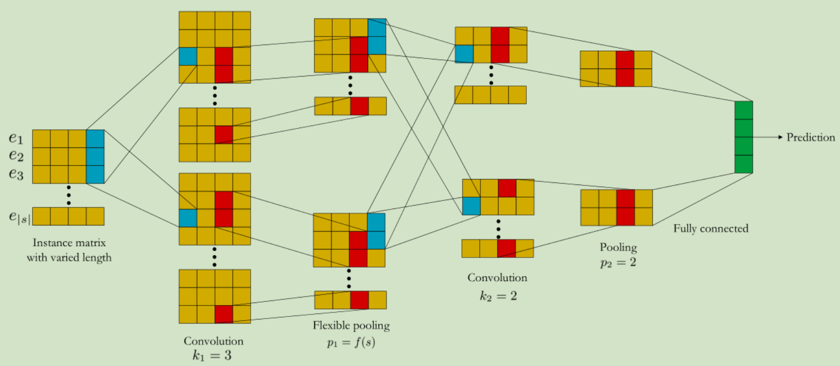

Figure 1: The framework of CCPM. The left part is an input instance (single ad impression or sequential ad impression) with varied elements and the length of element embedding is $d = 4$. The architecture has two convolutional layers with two feature maps each. The widths of filters at two layers are three and two respectively. The flexible pooling layer p_1 changes with length of instance and the last pooling layer $p_2 = 2$.

1.1 卷积层

每个样本有 n 个离散特征，每个离散特征都是用 d 维的嵌入向量 $\mathbf{e}_i \in R^d$ 表示, 则一个样本可以表示为矩阵 $\mathbf{s} \in R^{d \times n}$:

$$\mathbf{s} = \begin{bmatrix} \vdots & \vdots & \vdots \\ \mathbf{e}_1 & \cdots & \mathbf{e}_n \\ \vdots & \vdots & \vdots \end{bmatrix}_{d \times n} . \tag{1}$$

卷积层是通过卷积核 $\mathbf{w} \in R^{d \times w}$ 沿着行方向移动。这样 $\mathbf{s}$ 经过卷积后得到矩阵 $\mathbf{r}$，其维度为 $d \times (n + w - 1)$。可以记为：$\mathbf{r} = \mathbf{F}(\mathbf{s}, \mathbf{w})$（$\mathbf{F}$ 表示卷积函数）。

从行的角度看，给定 $\mathbf{w}_i \in R^w, \mathbf{s}_i \in R^n$，经过卷积可以得到：

$$\mathbf{r}_i = \mathbf{w}_i^T \mathbf{s}_{i, j-w+1:j}, (j \in [1, n + w - 1]) \tag{2}$$

1.2 灵活的p-Max池化

对于卷积后的矩阵 $\mathbf{r}$ 的第 i 行向量 $\mathbf{r}_i \in R^n$，选择前 p 个最大的值作为池化后的向量 $\mathbf{s}_i^p \in R^p$。

对于序列的 $\mathbf{s}$ 是变长的，因此池化层应该具有灵活性，对于较长的序列，抽取多个最大值作为池化结果；对于较短序列只需要抽取较少的最大值作为池化结果。本文使用下面的函数作为p值的大小：

$$p_i = \{ \begin{matrix} (1 - (i/l)^{l-i}) n, & i = 1, \cdots, l - 1 \\ 3, & i = l \end{matrix} , \tag{3}$$

其中，l 是网络中卷积层的总层数，n 是输入样例的长度，p_i 表示第 i 层池化层。

该函数的优点：

- 最后一层池化层是固定的，保证最后输出的向量是定长的
- 幂指函数在开始时变化较慢，避免在前几层丧失重要特征

激活函数： 池化层后使用tanh激活函数

2. Deep Learning over Multi-field Categorical Data(2016)

本文使用三种特征转换方法：因子分解机(FM)、受限玻尔兹曼机(RBM)、去噪自编码器，将类别特征转化为稠密的向量，再利用深度网络有效提取高阶交叉特征。

2.1 Factorisation-machine supported Neural Networks (FNN)

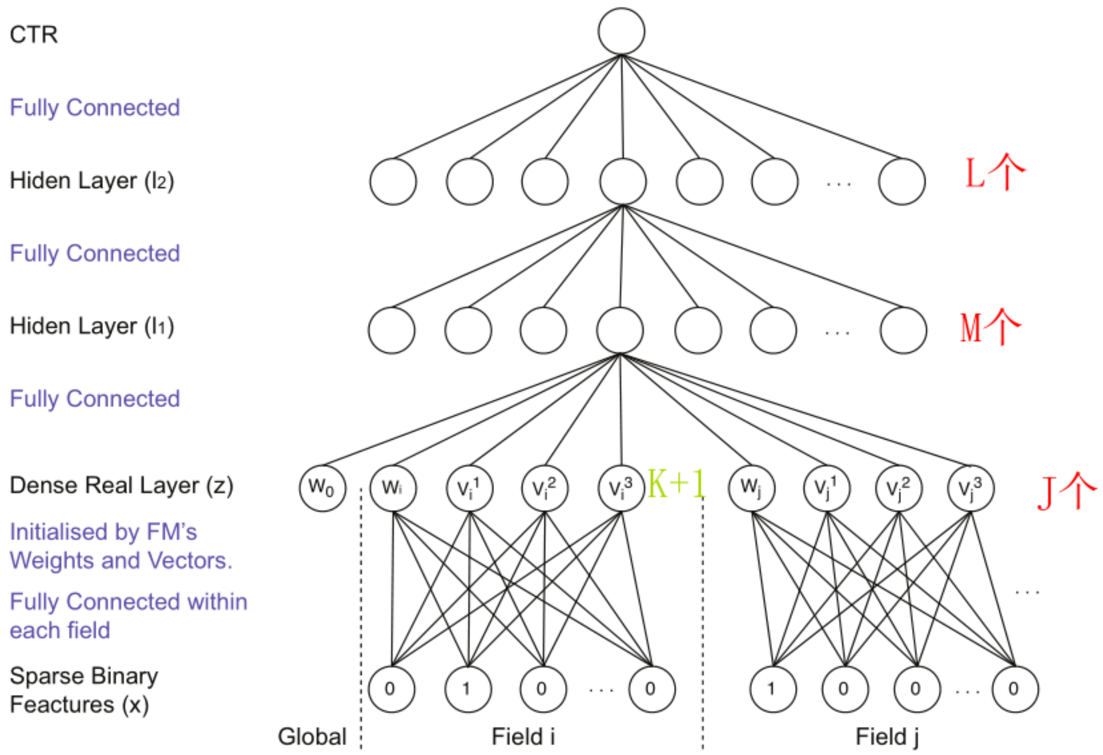

Fig. 1. A 4-layer FNN model structure.

从上到下依次是输出层、隐藏层、Dense Real(z)层、稀疏表示层，Dense Real layer实际上相当于嵌入层，只是嵌入矩阵使用因子分解机训练得到的向量初始化。因子分解机：

$$y_{FM} = sigmoid\left(w_0 + \sum_{i=1}^{N} w_i x_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j\right), \tag{1}$$

2.2 Sampling-based Neural Networks (SNN)

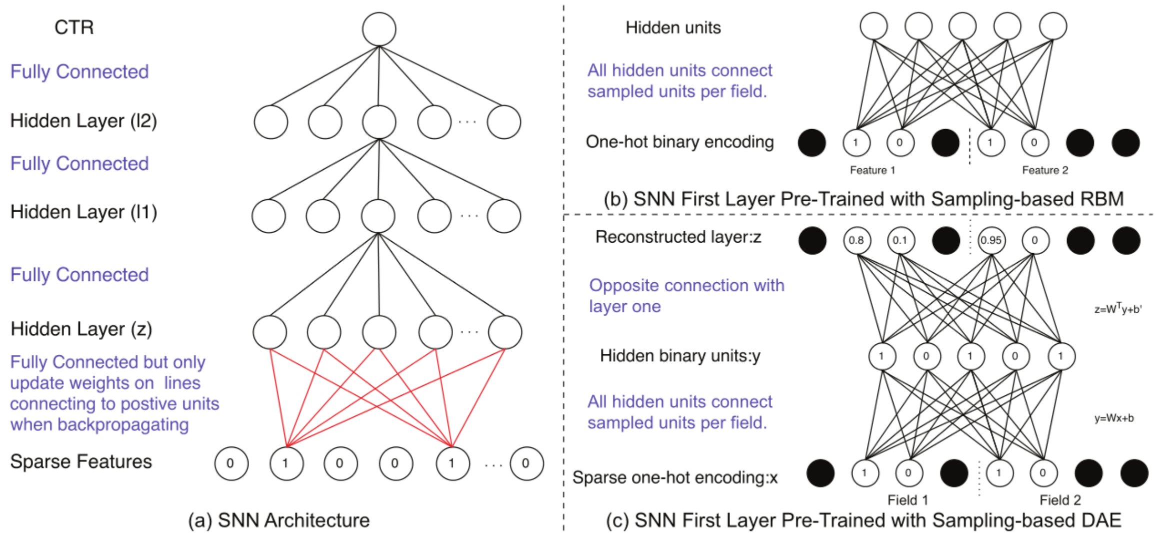

Fig. 2. A 4-layer SNN architecture and two first-layer pre-training methods.

最底层使用sigmoid函数的全连接层，为了在稀疏one-hot编码情况下有效学习权重W，本文使用受限玻尔兹曼机和降噪自编码器预训练得到W的初始值，在用于网络的更新。

3. Product-based Neural Networks for User Response Prediction(2016)

本文使用embedding层建立基于乘积的神经网络(PNN)，用于捕捉两个类别特征之间的交叉模式，后接深度神经网络捕获高阶交叉特征。

3.1 Product-based Neural Network

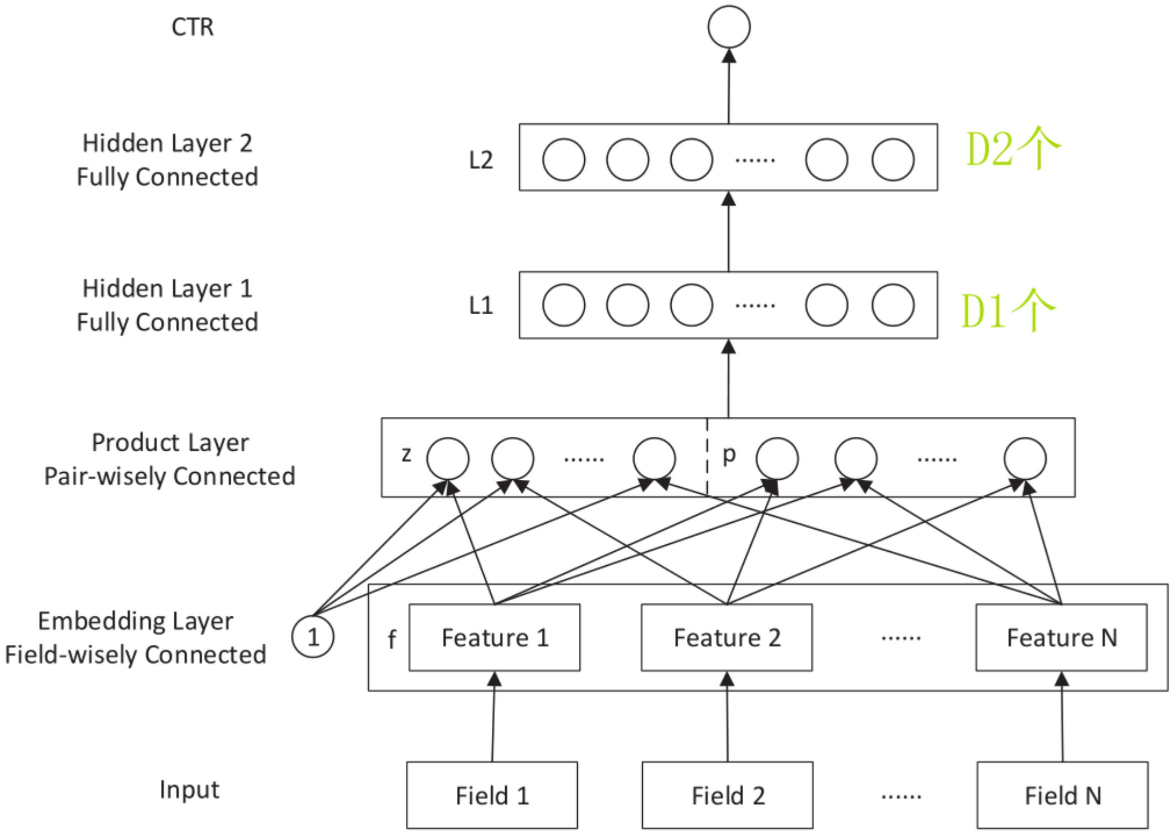

Fig. 1: Product-based Neural Network Architecture.

其中，第一个隐藏层和Product层是全连接的，其输出是 $l_1 \in R^{D_1}$ ，输入由线性信号 l_z 和二阶交叉信号 l_p 组成：

$$l_1 = relu(l_z + l_p + b_1) , \tag{1}$$

其中， $l_z, l_p, b_1 \in R^{D_1}$ 。

$$\begin{aligned} l_z &= (l_z^1, l_z^2, \cdots, l_z^n, \cdots, l_z^{D_1}), \qquad l_z^n = W_z^n z \\ l_p &= (l_p^1, l_p^2, \cdots, l_p^n, \cdots, l_p^{D_1}), \qquad l_p^n = W_p^n p \end{aligned} \tag{2}$$

第一个隐藏层本质上是分别对 z 和 p 线性转换后进行相加的操作。

而 z 是 1 和各个特征的embedding的concat：

$$z = (z_1, z_2, \cdots, z_N) \triangleq (f_1, f_2, \cdots, f_N) \tag{3}$$

$$p = \{p_{i,j} = g(f_i, f_j)\}, i = 1, \cdots, N, j = 1, \cdots, N \tag{4}$$

4. Wide & Deep Learning for Recommender Systems(2016)

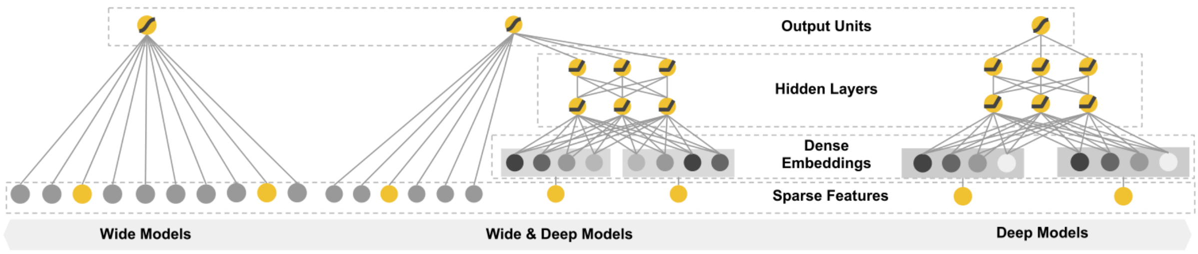

Figure 1: The spectrum of Wide & Deep models.

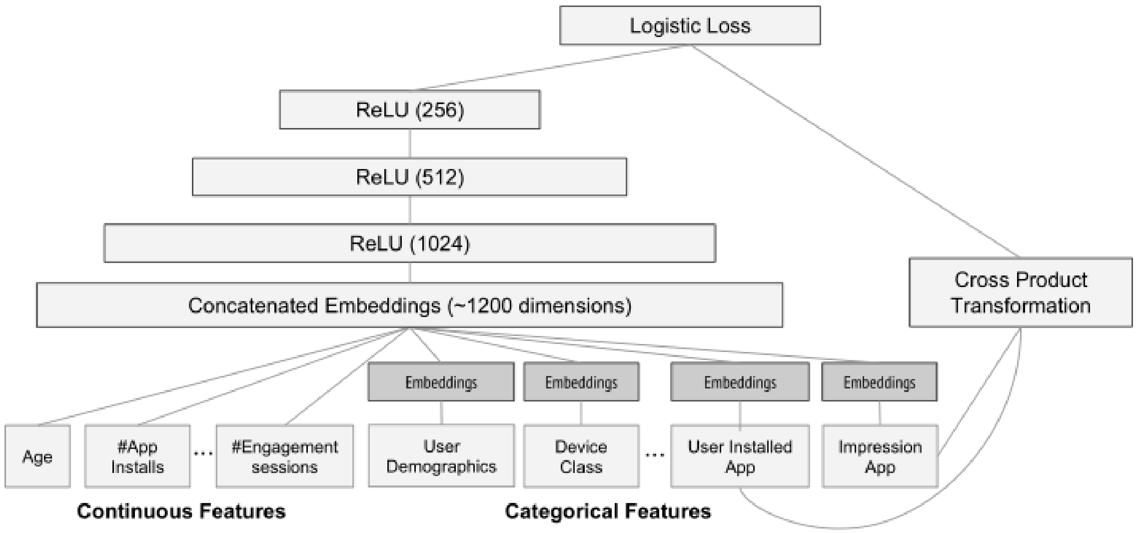

Figure 4: Wide & Deep model structure for apps recommendation.

5. Factorization-Machine based Neural Network for CTR Prediction(2017)

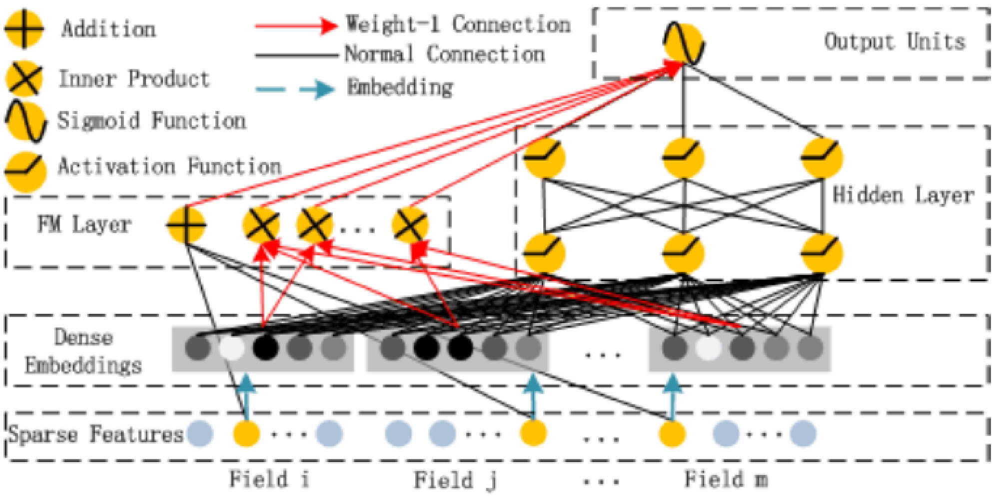

Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

6. Deep & Cross Network for Ad Click Predictions(2017)

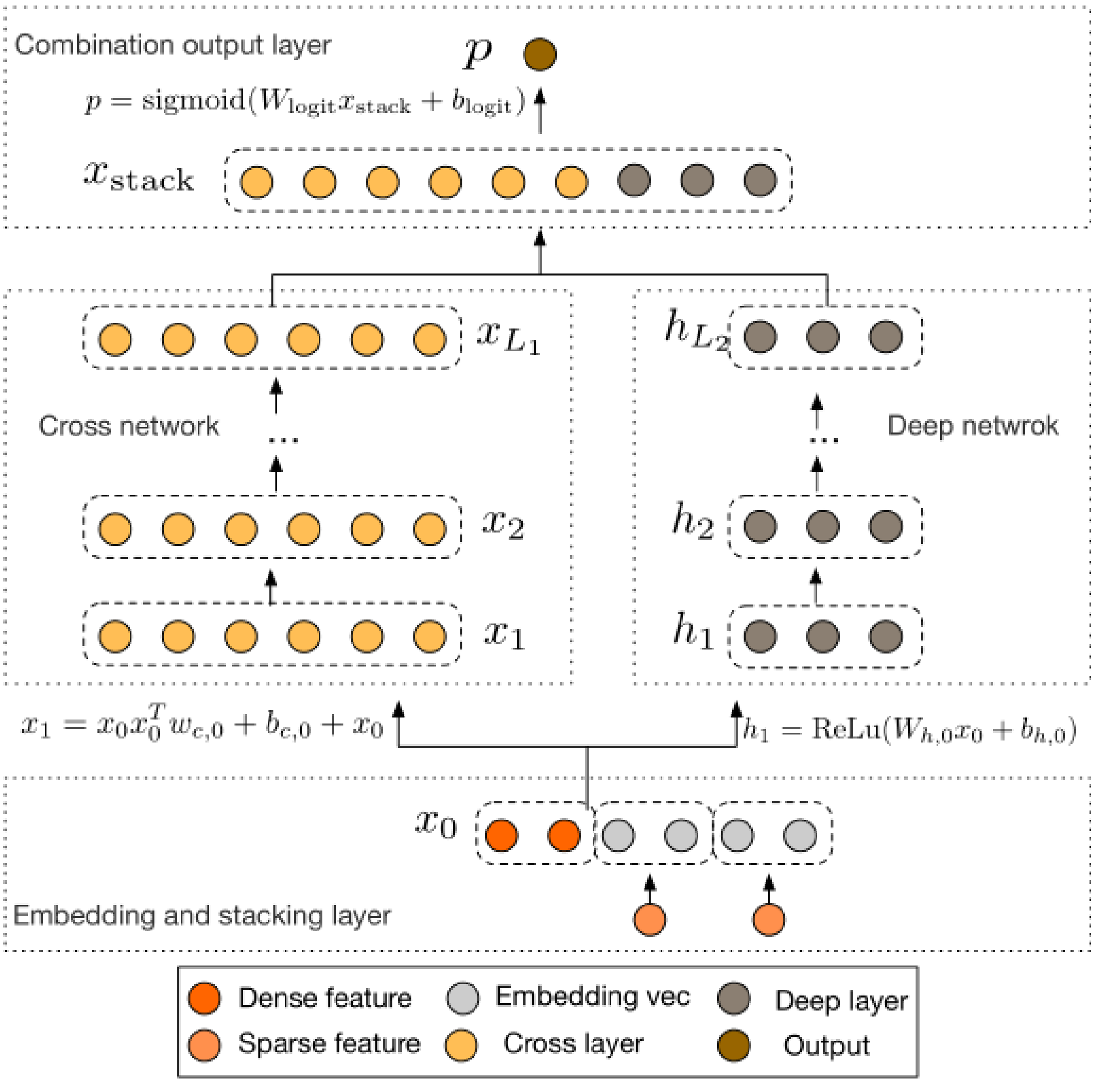

Figure 1: The Deep & Cross Network

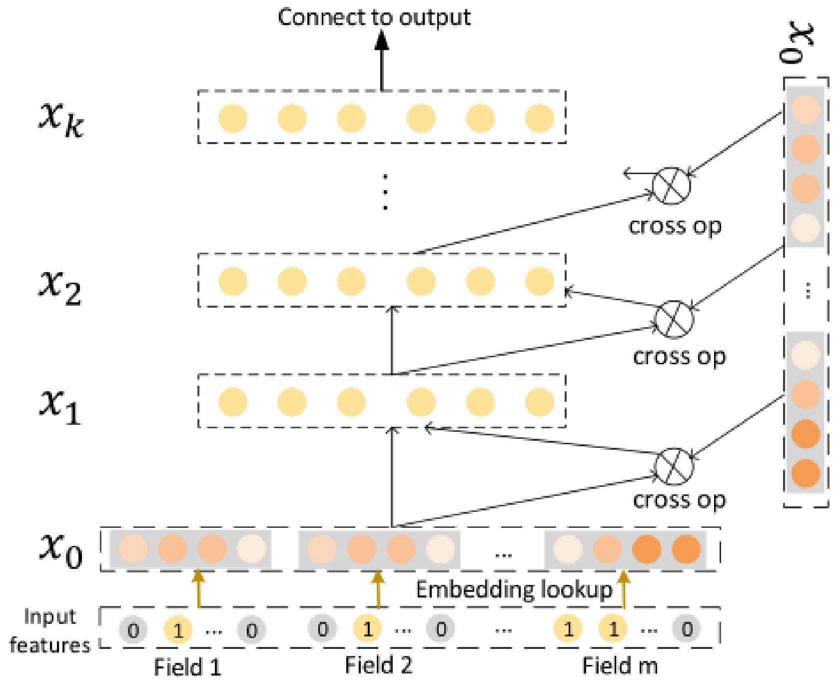

Figure 3: The architecture of the Cross Network.

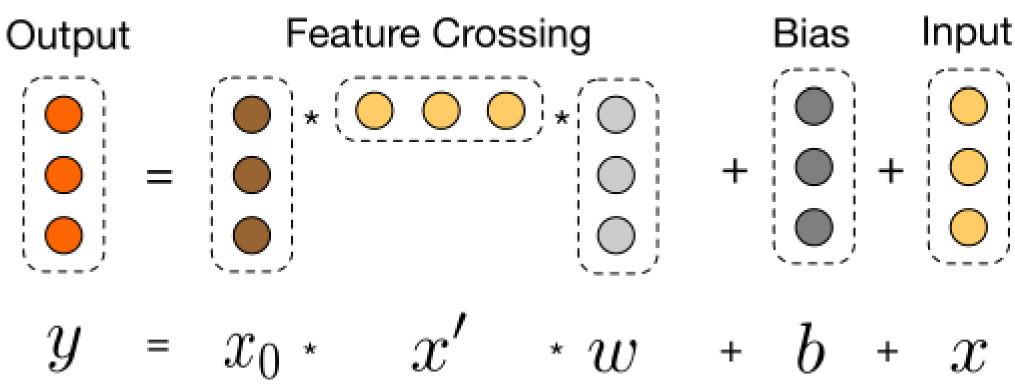

Figure 2: Visualization of a cross layer.

7. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks

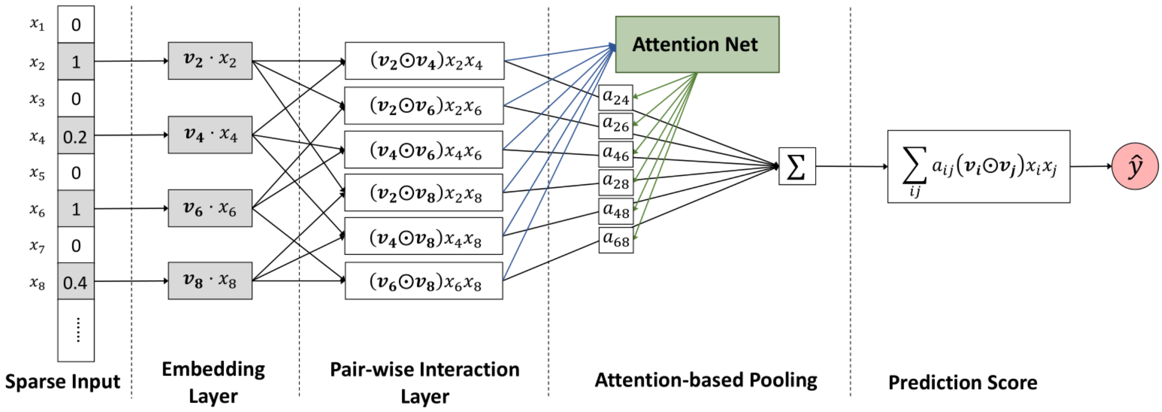

Figure 1: The neural network architecture of our proposed Attentional Factorization Machine model.

8. Neural Factorization Machines for Sparse Predictive Analytics

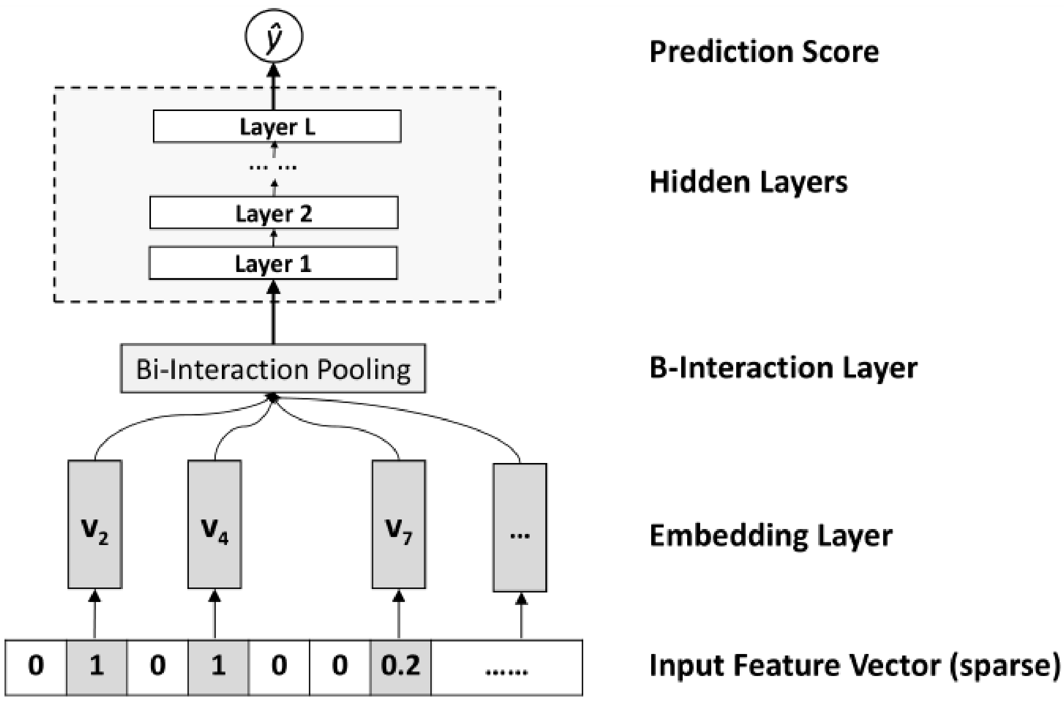

Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j,$$