Play with DIN

# 模型结构

* baseModel

· Github (written in tensorflow)

Input:

1. hyper para

(user\_count(maybe include user\_category and related list), item\_count, item\_category\_count, item\_category\_list\_all)

1. train

(user, item, hist, label)

1. eval(used for AUC calculate)

(user,hist,(pos\_item, neg\_item))

1. test(but in train.py, this isn’t used, including train in base and din)

(user,hist) and calculate all the item score…..

Output:

1. train

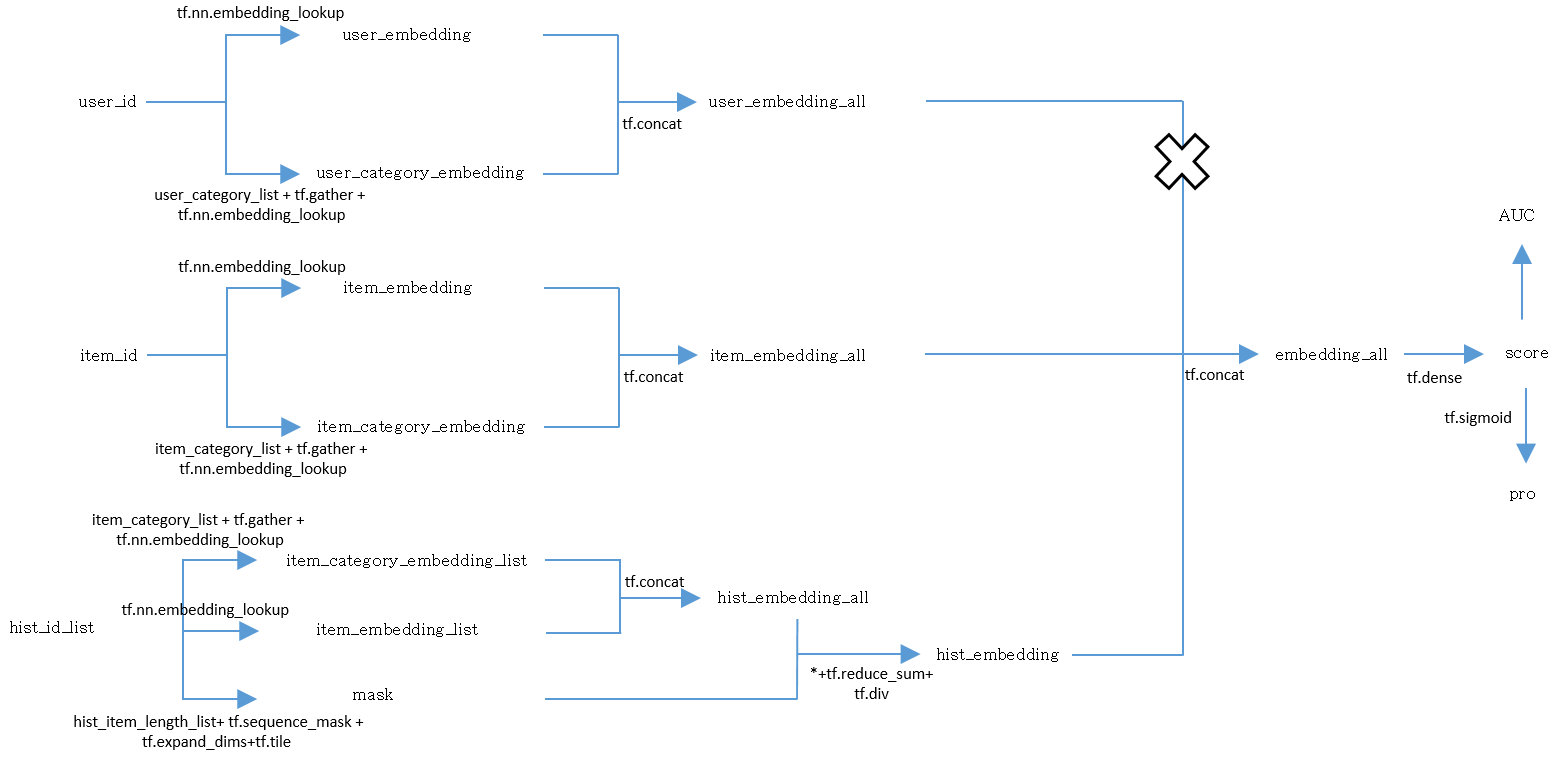
binary cross entropy loss:-(ylogp+(1-y)log(1-p))

1. eval

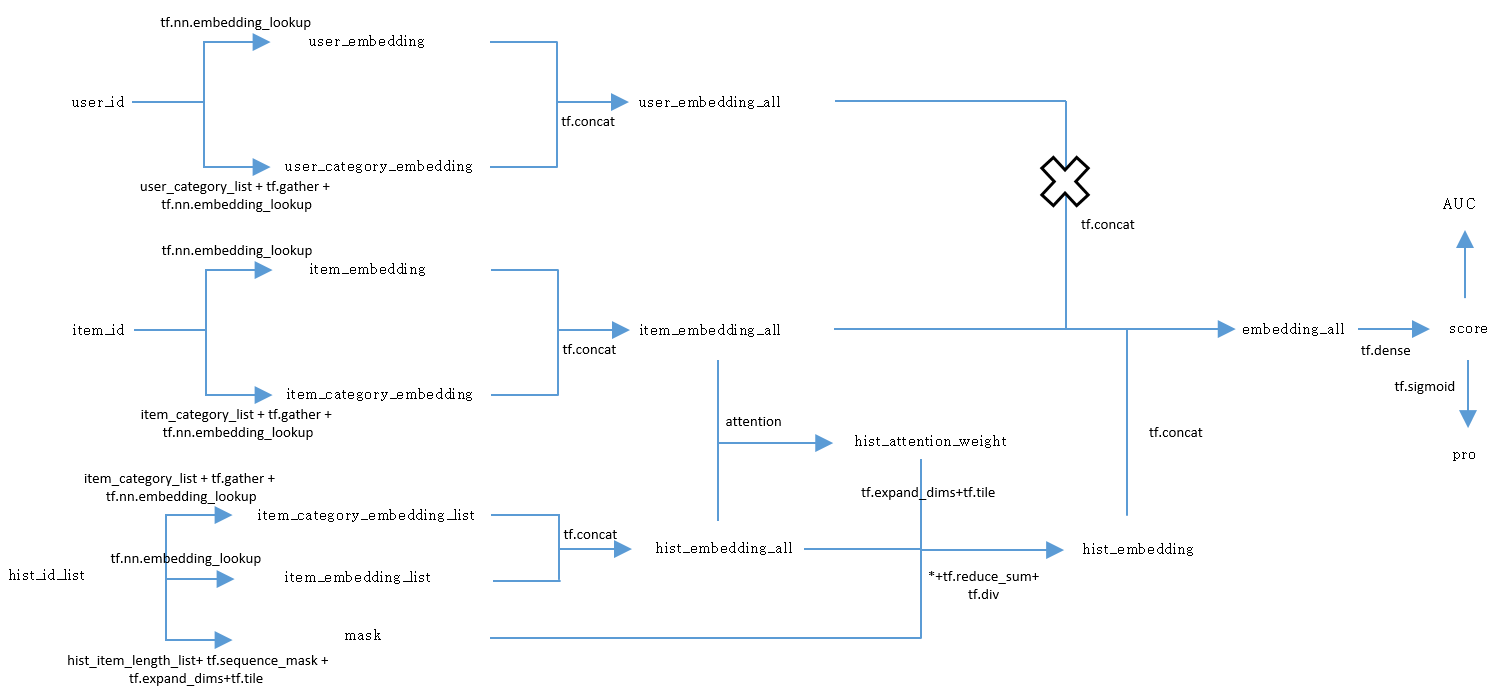
AUC:the probability that the score of the pos\_item bigger than the neg\_item

1. test

the probability of every item related to the user



* DIN model



基于上面的结构，本地的模型将×号的部分添加到模型中，并且在item或者user的multi\_hot特征的处理方式中，采用求和的方式整合特征。

# DIN实现技术难点及细节

* 输入变长的序列：

可以在keras下的Input层中设置shape为（None,）这里的第二维度相当于数据量的维度。这个是不work的，应该输入等长的序列，在预处理时进行处理。但是对应地，在Embedding层中，如果对不同长度的序列进行取嵌入向量的操作，可以先把E层中的input\_length设置为None，再结合下面的操作进行后续的建模，而且，None下可以根据输入的维度进行后续序列的第二个对应的维度的规模进行调整，如E层为None，若某个输入的长度为1，则对应得到的嵌入维度为(Batch\_dim,1,Embedding\_dim)，输入长度改变，对应的第二维度规模改变。注意，keras与keras.backen（K）或tf下的维度不一样，后两者在设置维度的时候需要考虑数据规模的维度，即第一维，而Keras下的函数，如Reshape，Concatenate等，第一维度均为某一样本下的第一个维度。

* 对变长的hist序列进行建模：

使用最长的hist作为整个数据集下的hist序列数组的长度，即固定了长度。由于不同长度下的hist下的item进行嵌入向量的求和的时候，0代表嵌入向量是无需参与计算的，因此需要使用mask将对应的嵌入向量进行掩埋（乘0处理），然后再除以对应的序列长度以求均值。在keras中，可以使用keras.backen（K）作为调用tf函数，以做更高级的操作。

* 调用自定义函数：

Keras下的Lambda函数可以在里面填入lambda自定义函数作为参数，然后再调用该Lambda函数对Keras层进行建模，如果需要K里面的函数则需要该方法去自定义函数，然后调用。不然直接使用K下的函数作为某一Keras层会报错（即没有输入正确的Keras层）。Ps，特别需要与第一点中的维度相结合，不然维度会发生混乱。

* 对hist下的category进行建模：

由于有时候，category是多样的，即多热的。因此需要类似于对hist物品嵌入向量的处理类似，进行均值合并，但是由于直接进行gather抓取category类型后维度发生较大的变化，假设历史序列长度为L，各个物品的category序列长度为C，对应的属性嵌入向量长度为H，则原本一个样本下的hist规模为L，对应到category后为L×C，再映射到嵌入向量为L×C×H，然后再对后面C×H进行类似的mask合并。想法简单，但是难于操作，首先是L×C -> L×C×H维度的变化，用gather只可以进行解决L -> L×C的映射，然后的三维是需要Reshape进行扁平化；然后是mask的合并，因为不能直接对C×H进行操作（原本的想法是利用K.map\_fn进行逐个样本映射合并，但是在Lambda里面不能同时调用keras内置模块的函数以及K函数，不然会发生维度混乱，失败），这里只能在扁平化后进行合并，后发现直接调用gather或者Embedding层即可把对应索引的词向量输出，即可结束。

* 维度对应：

[数据规模None,输入规模,Embedding规模]或者更高维一般输入规模为1，在Input层里面限定。

* Dense层（tf.layers.dense和keras.layers.Dense）：

可以接收秩或维度大于等于2（不算batch维度）的输入，然后传入到对应的dense层，即前馈网络。其具体的操作，相当于把最后一个维度下的数据传入到dense层中进行处理，输入对应的隐层节点数目的数据然后其他维度保持不变。这个特性对于一个样本下有多个输入需要dense处理的情况尤其有用。

* 梯度消失问题：

在base和baseDIN（只是添加了Dense attenation层）的训练里，发现若是对历史行为进行最后的合并求和操作后，采取取平均，则会导致train出现loss-nan，acc-0.00的情况，初步认定为取平均后值过小，达到0.001的量级，导致传入后续的网络结构时对结果影响很小，即反向传播会出现梯度消失的情况，更好的办法是对参数的初始化方法进行修改，如果是正态分布，可以加大标准差使得参数初始化的数据范围更大。

# 实验结果

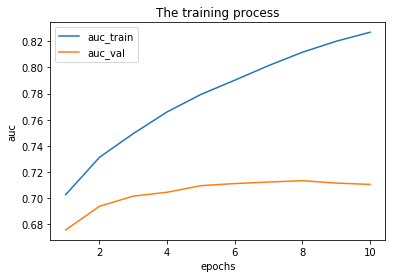
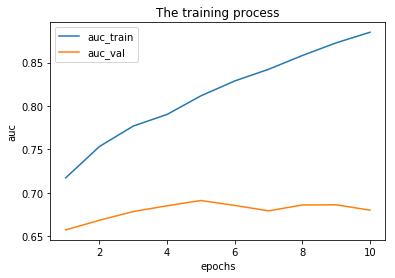
使用movieLen数据集

构造数据集为[user,movie,movie\_rate\_hist,label]

The training rate is 0.85

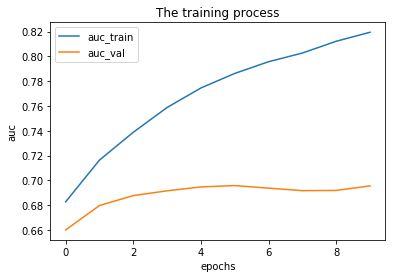
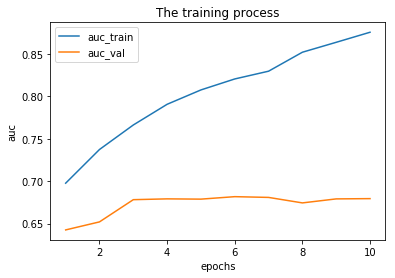
(only 10000 samples analysis, 32 hidden)

* baseModel

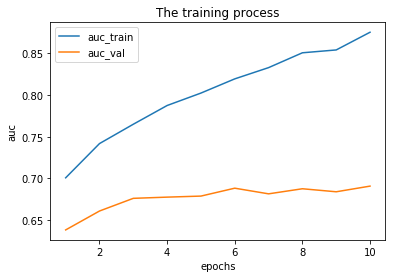
 

baseModel applied sigmoid baseModel applied PReLU

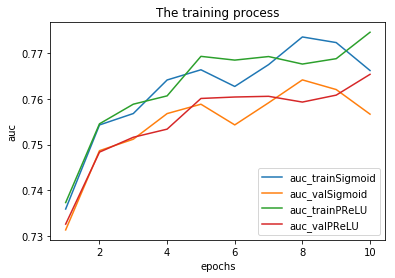
* DIN

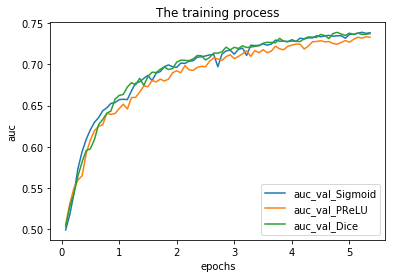
DINModel applied sigmoid DINModel applied PReLU



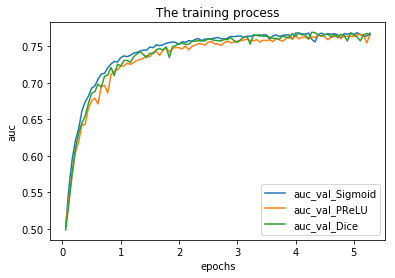
DINModel applied Dice



0.85 : rate for train, eval for every epoch, 100w



0.95 : rate for train, eval for every 100 batchs, 10w



0.99 : rate for train, eval for every 500 batchs, 60w