“万物皆可embedding”

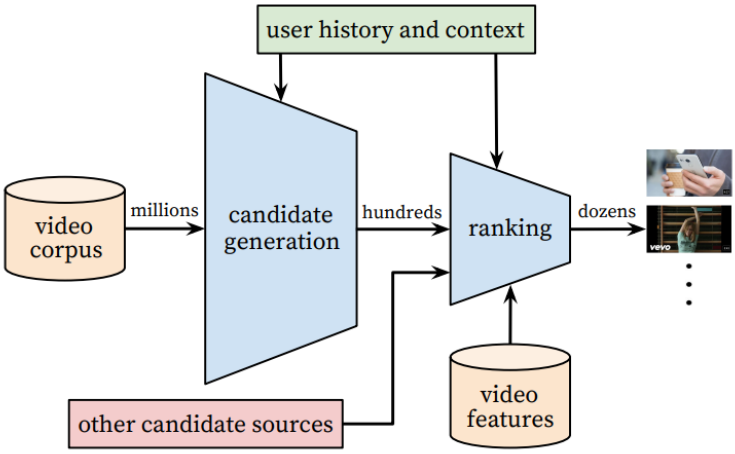
原创 DeePR 深度学习与推荐系统 2月29日

不知道大家有没有这种感受，在学习推荐系统算法模型时，少不了embedding的应用，有的推荐算法模型甚至可以说就是在做embedding的过程，可见embedding在推荐系统中的重要性。

这篇文章就专门把embedding单独提出来，梳理一下embedding在推荐系统中的应用。以下内容主要从深度学习方法和传统的协同过滤方法两个方面加深和理解在推荐系统领域对embedding的认识，感受下“embedding”这一重要思想。

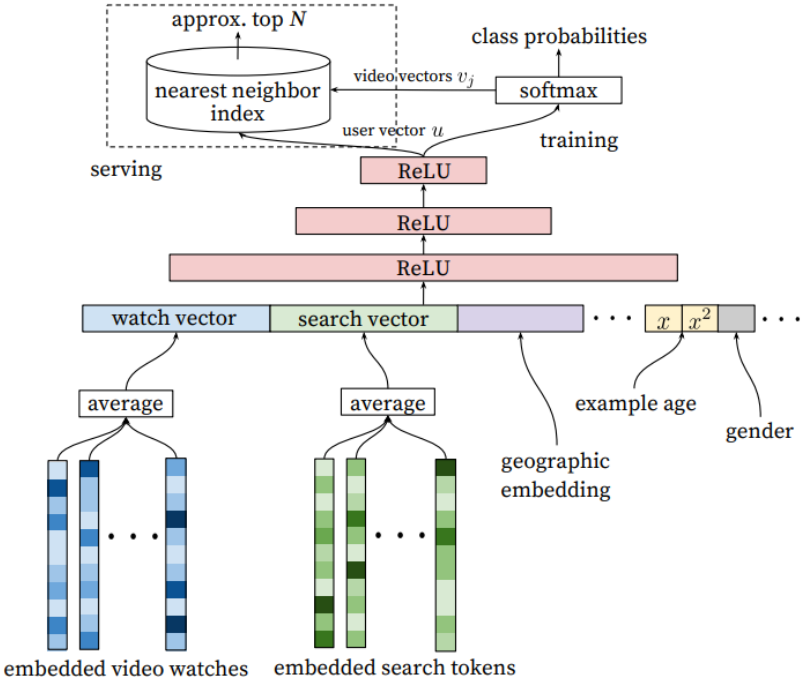
深度学习方法

先拿一篇推荐系统领域中最为经典的论文之一“Deep Neural Networks for YouTubeRecommendations”来讲，Youtube的这篇视频推荐模型框架基本上奠定了后面推荐系统的主要步骤：召回和排序，如下图所示：



其中召回阶段（candidate generation）就是要从推荐的百万级别的视频库中筛选出用户可能感兴趣的视频，把推荐的视频库量级从百万级降到几百个。但是到底怎样才能快速高效的完成筛选呢？

要知道youtube是非常活跃的视频网站，每秒同时访问的人数成千上万，要同时实现每秒对每个用户都个性化的从百万视频候选集中挑出几百个用户感兴趣的视频，想想都不容易，因此每次做用户召回都跑一遍模型是不可能的，其解决方法就和接下来要介绍的embedding应用相关。如下图 1 所示为youtube召回阶段的模型：

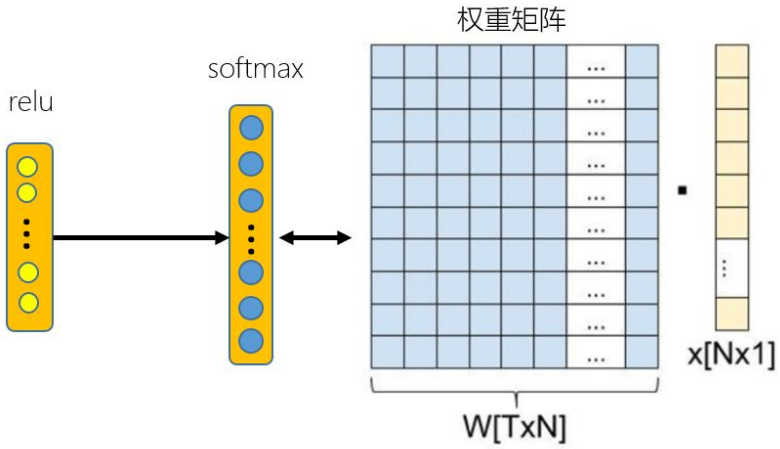


▲ 图1. YouTube召回模型

在离线的模型训练阶段，采用的是简单粗暴的softmax来对视频库中的所有视频进行分类，label为用户Next watched video的那个视频，因此整个召回阶段构建模型的思想就是预测某一时刻某一用户在对百万级以上的视频库哪个视频感兴趣，它最想点击和观看的视频是哪一个。

这里有一个很重要的问题就是百万级别以上的视频做softmax是很费计算资源和时间的，因此在召回模型的离线训练阶段采用了word2vec中的Negative Sample思想。可是即便是这样，面对百万级别的视频库，以及每秒成千上万的召回请求，在线上还是无法满足需求，不能直接使用模型去对视频做softmax，按照概率大小选取TopN来做召回，因此就有了embedding做召回的方法。

如上图 1 所示主要思想是把softmax的前一层的输出作为user的embedding，而softmax层中的权重矩阵的每一行向量作为video的embedding，这样在模型的部署过程中，就没有必要部署整个神经网络来完成从原始特征向量到最终输出的预测过程，只需要将User Embedding和video Embedding存储到线上内存数据库，通过内积运算再排序的方法就可以得到video的排名，这大大加快了召回效率。



这里再用图示的方法具体介绍一下哪个是user embedding，哪些是video embedding，为什么可以向量内积召回，为什么内积得到的值大小就可以反映出用户对视频的兴趣程度。如上图所示，把论文中的倒数第二层和最后一层softmax单独拿出来分析，实际上论文中最后一个softmax是一个加了softmax函数的全连接层，该全连接层（图中softmax）的神经元个数和需要分类的视频个数相同，都为T，softmax函数对该全连接层（图中softmax）的输出再进行归一化得到每个视频对应的概率值。

这个全连接层（图中softmax）的每一个神经元都对应有上一层全连接层（图中relu）输出的权重向量，为上图中W[T×N]权重矩阵的每一行，而这每一行权重向量就是video embedding，个数和神经元的个数相同都是视频库中视频的个数T，上一层全连接层（图中relu）的输出为user embedding，是上图中的X[N×1]。因此通过内积user embedding和video embedding其实

得到的正是未经过softmax函数归一化的每个视频的预测值，从一定程度上反映了预测概率的大小。

因此user embedding和video embedding内积值越大，则反应该用户对该视频感兴趣的概率值大，所以可以提前将User Embedding和video Embedding存储到线上内存数据库，通过内积运算再排序的方法得到video的排名，以此来提高召回速度和效率。

总结一句话就是，为了提高召回速度和效率，相当于把召回模型的inference转换成了通过embedding向量内积方法快速得到用户对视频得分的方法。从这个例子可以看出embedding的应用之一：**通过计算用户和物品的Embedding相似度，Embedding可以直接作为推荐系统的召回层或者召回方法之一。**

其实像embedding的这种应用非常多，像微软的一篇论文Item2Vec: Neural Item Embeddingfor Collaborative Filtering，专门训练item的embedding，然后通过计算item之间的embedding相似度来做基于物品的协同过滤，实际上通过这种方式也可以缩小推荐候选物品的范围，用来直接做物品的相似推荐。

而专门学习item embedding的方法还有很多，比如通过graph embedding方式的deepwalk，LINE，Node2vec和SDNE等等。因此再结合这些embedding的应用例子，再对embedding在召回方面的应用浓缩总结一下就是：**通过计算用户和物品或物品和物品的Embedding相似度，来缩小推荐候选库的范围。**

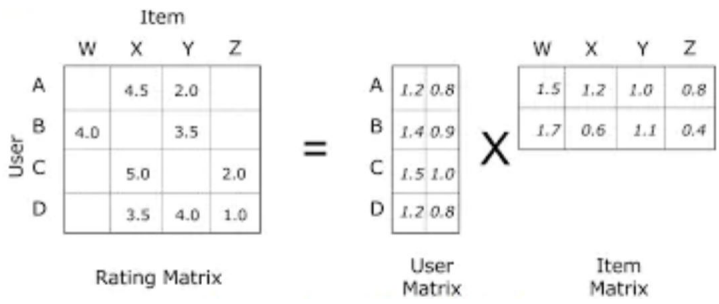
除此之外，通过总结目前主流的ctr预估模型比如wide&deep，deepFM，PNN和DCN等等可以发现，embedding还有一个非常普遍的应用就是**实现高维稀疏特征向量向低维稠密特征向量的转换**，通俗来讲就是把离散特征经过独热编码后的稀疏向量表达转化成稠密的特征向量表达。

或者从另一个角度看，embedding本身就是对事物的多维度特征表示，因此在ctr预估模型中，**训练好的embedding可以当作输入深度学习模型的特征**，比如FNN利用FM预训练好的embedding来当作模型的输入，然后通过特征交叉操作比如多层感知机得到这些embedding的交叉特征。具体的交叉特征分析以及有关推荐系统中对于特征的处理可以查看公众号中的**特征处理**系列文章。

有关深度学习方法中的embedding应用就介绍到这，这里再着重介绍一下最近学习的有关协同过滤的传统推荐方法中embedding思想的体现。

传统方法

虽然目前有关推荐系统的模型中深度学习越来越占据重要地位，但是embedding的重要思想却贯穿始终，在传统的推荐方法中依然可以看到embedding的影子。例如基于矩阵分解的协同过滤模型，如下图所示：



通过将用户对物品的打分矩阵Rating Matrix分解成User Matrix和Item Matrix两个矩阵相乘，我们可以把User Matrix和Item Matrix相乘分别看作是A，B，C，D四个user embedding和W，X，Y，Z四个item embedding相乘。

可以看到打分矩阵比较稀疏，说明有的user没有给item打分，因此矩阵分解的目的就是通过分解的user embedding和item embedding相乘来填充user没有打分的item，从而可以进行推荐。而模型通过user embedding和item embedding相乘拟合user已给item的打分来学习embedding参数。

如上图所示，分解的user embedding B（[1.4,0.9]）和item embedding W（[1.5,1.7]）相乘得到3.63要尽量接近Rating Matrix中的B行W列也就是4.0，根据这种拟合学习得到embedding，最终再根据模型学习的user embedding和item embedding相乘得到user没有给item的打分，比如Rating Matrix中的A行W列的得分为user embedding A（[1.2,0.8]）和item embedding W（[1.5,1.7]）相乘得到3.16.

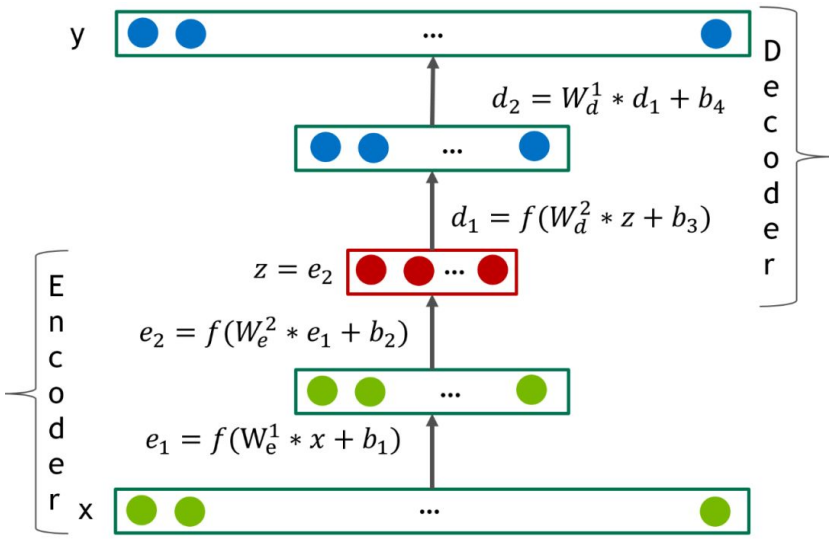
因为模型比较简单，这里直接给出利用tensorflow实现的模型的代码：

```
def build_model(user_indices, item_indices, rank, ratings, user_cnt, item_cnt

    W_user = tf.Variable(tf.truncated_normal([user_cnt, rank], stddev=init_va
                         name = 'user_embedding', dtype=tf.float32)
    W_item = tf.Variable(tf.truncated_normal([item_cnt, rank], stddev=init_va
                         name = 'item_embedding', dtype=tf.float32)

    W_user_bias = tf.concat([W_user, tf.ones((user_cnt,1), dtype=tf.float32)]
    W_item_bias = tf.concat([tf.ones((item_cnt,1), dtype=tf.float32), W_item]

```

```
    user_feature = tf.nn.embedding_lookup(W_user_bias, user_indices, name = '
    item_feature = tf.nn.embedding_lookup(W_item_bias, item_indices, name = '

    preds = tf.add(tf.reduce_sum( tf.multiply(user_feature , item_feature) ,

    square_error = tf.sqrt(tf.reduce_mean( tf.squared_difference(preds, ratin
    loss = square_error + lamb*(tf.reduce_mean(tf.nn.l2_loss(W_user)) + tf.re

    tf.summary.scalar('square_error', square_error)
    tf.summary.scalar('loss', loss)
    merged_summary = tf.summary.merge_all()
    #tf.global_variables_initializer()
    train_step = tf.train.GradientDescentOptimizer(lr).minimize(loss)   # tf.

    return train_step, square_error, loss, merged_summary
```

虽然模型比较简单，但是可以发现embedding的思想其实贯穿在整个模型当中。

除了基于矩阵分解的协同过滤，还有基于自编码器的协同过滤，自编码器做协同过滤的思想主要是把用户对所有物品的打分组成一个固定维度的向量（没有打分的填充为0）然后通过自编码器对该打分向量进行编码解码然后得到和该打分向量维度相同的向量，该自编码器模型的输出向量就是为了拟合输入向量。如图所示为自编码器模型：



先对输入向量通过两个全连接层进行编码，然后再通过两个全连接层进行解码最后得到拟合输入向量的输出向量。该模型同样也是通过拟合已经有的打分来学习整个模型的参数，模型学习好之后，之前填充为0的物品打分就可以通过模型输出向量的对应位置得到。

虽然这个模型没有很明显的embedding思想的体现，但是以我个人对embedding的理解，在模型最终输出的用户对所有物品的打分稠密向量是否可以用来当作表示用户兴趣的user embedding向量呢?

总结

通过这篇文章对embedding的分析，总结embedding有以下三个作用：

- 通过计算用户和物品或物品和物品的Embedding相似度，来缩小推荐候选库的范围。
- 实现高维稀疏特征向量向低维稠密特征向量的转换。
- 训练好的embedding可以当作输入深度学习模型的特征。

无论embedding在模型中最终起到哪些重要的作用，在对embedding的本质理解上，它自始至终都是用一个多维稠密向量来对事物从多维度进行的特征刻画。

• END •

点击以下标题查看更多往期内容：

- 序列特征的处理方法之一：基于注意力机制方法
- 序列特征的处理方法之二：基于卷积神经网络方法
- 多值类别特征加入CTR预估模型的方法
- 稠密特征加入CTR预估模型的方法
- CTR预估模型的发展有这样的规律

深度学习与推荐系统

微信扫码关注，获取更多精彩