19｜NeuralCF：如何用深度学习改造协同过滤？

你好，我是王喆，今天，我们来学习协同过滤的深度学习进化版本，NeuralCF。

在第 15 节课里，我们学习了最经典的推荐算法，协同过滤。在前深度学习的时代，协同过滤曾经大放异彩，但随着技术的发展，协同过滤相比深度学习模型的弊端就日益显现出来了，因为它是通过直接利用非常稀疏的共现矩阵进行预测的，所以模型的泛化能力非常弱，遇到历史行为非常少的用户，就没法产生准确的推荐结果了。

虽然，我们可以通过矩阵分解算法增强它的泛化能力，但因为矩阵分解是利用非常简单的内积方式来处理用户向量和物品向量的交叉问题的，所以，它的拟合能力也比较弱。这该怎么办呢？不是说深度学习模型的拟合能力都很强吗？我们能不能利用深度学习来改进协同过滤算法呢？

当然是可以的。2017 年，新加坡国立的研究者就使用深度学习网络来改进了传统的协同过滤算法，取名 NeuralCF（神经网络协同过滤）。NeuralCF 大大提高了协同过滤算法的泛化能力和拟合能力，让这个经典的推荐算法又重新在深度学习时代焕发生机。这节课，我们就一起来学习并实现 NeuralCF！

NeuralCF 模型的结构

在学习 NeuralCF 之前，我们先来简单回顾一下协同过滤和矩阵分解的原理。协同过滤是利用用户和物品之间的交互行为历史，构建出一个像图 1 左一样的共现矩阵。在共现矩阵的基础上，利用每一行的用

户向量相似性，找到相似用户，再利用相似用户喜欢的物品进行推荐。

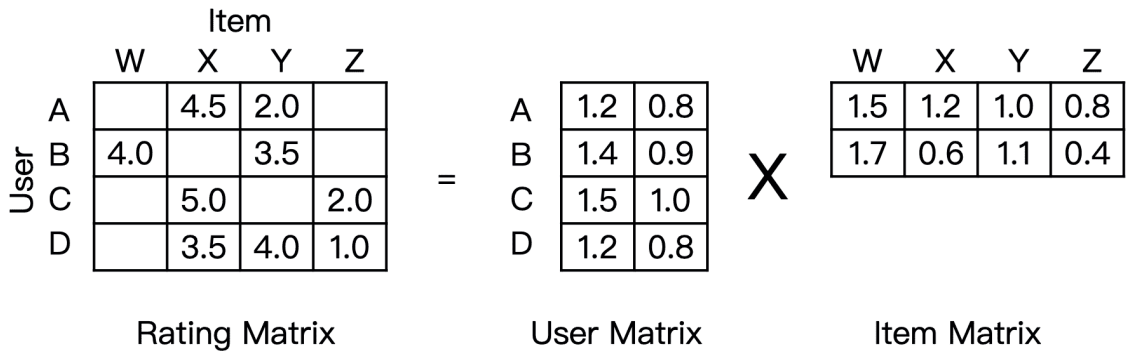

图1 矩阵分解算法的原理

矩阵分解则进一步加强了协同过滤的泛化能力，它把协同过滤中的共现矩阵分解成了用户矩阵和物品矩阵，从用户矩阵中提取出用户隐向量，从物品矩阵中提取出物品隐向量，再利用它们之间的内积相似性，进行推荐排序。如果用神经网络的思路来理解矩阵分解，它的结构图就是图 2 这样的。

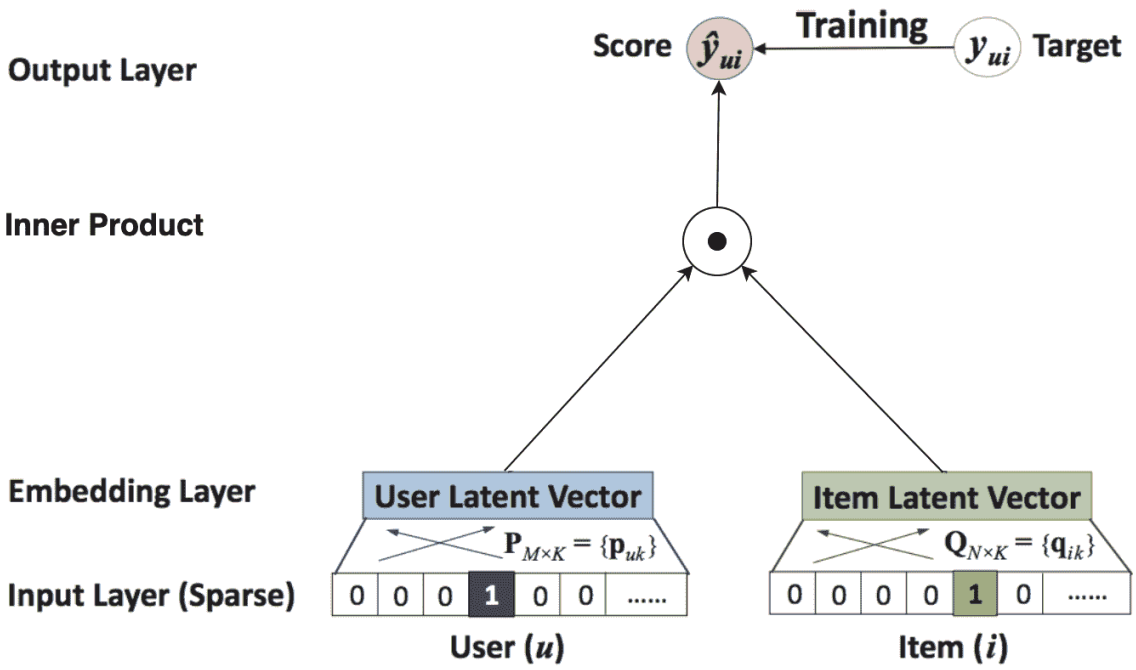

图2 矩阵分解的神经网络化示意图

图 2 中的输入层是由用户 ID 和物品 ID 生成的 One-hot 向量，Embedding 层是把 One-hot 向量转化成稠密的 Embedding 向量表达，这部分就是矩阵分解中的用户隐向量和物品隐向量。输出层使用了用户隐向量和物品隐向量的内积作为最终预测得分，之后通过跟目标得分对比，进行反向梯度传播，更新整个网络。

把矩阵分解神经网络化之后，把它跟 Embedding+MLP 以及 Wide&Deep 模型做对比，我们可以一眼看出网络中的薄弱环节：矩阵分解在 Embedding 层之上的操作好像过于简单了，就是直接利用内积得出最终结果。这会导致特征之间还没有充分交叉就直接输出结果，模型会有欠拟合的风险。针对这一弱点，NeuralCF 对矩阵分解进行了改进，它的结构图是图 3 这样的。

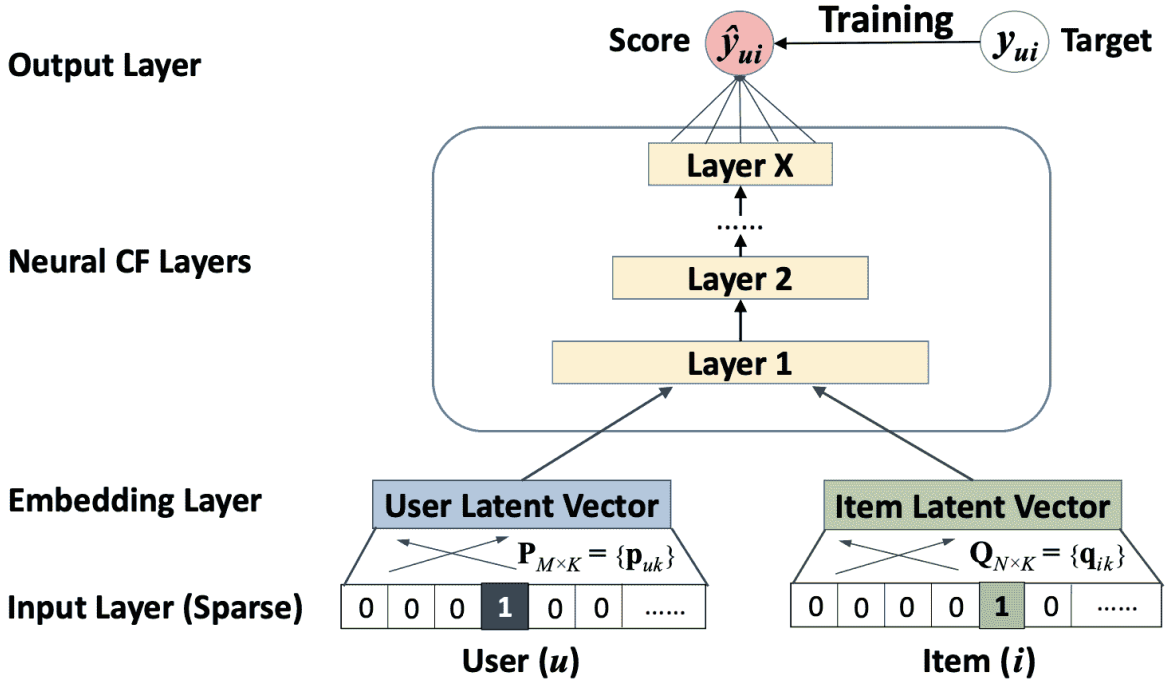

图3 NeuralCF的模型结构图（出自论文Neural Collaborative Filtering）

我想你一定可以一眼看出它们的区别，那就是 NeuralCF 用一个多层的神经网络替代掉了原来简单的点积操作。这样就可以让用户和物品隐向量之间进行充分的交叉，提高模型整体的拟合能力。

NeuralCF 模型的扩展，双塔模型

有了之前实现矩阵分解和深度学习模型的经验，我想你理解起来 NeuralCF 肯定不会有困难。事实上，NeuralCF 的模型结构之中，蕴含了一个非常有价值的思想，就是我们可以把模型分成用户侧模型和物品侧模型两部分，然后用互操作层把这两部分联合起来，产生最后的预测得分。

这里的用户侧模型结构和物品侧模型结构，可以是简单的 Embedding 层，也可以是复杂的神经网络结构，最后的互操作层可以是简单的点

积操作，也可以是比较复杂的 MLP 结构。但只要是这种物品侧模型 + 用户侧模型 + 互操作层的模型结构，我们把它统称为“双塔模型”结构。

图 4 就是一个典型“双塔模型”的抽象结构。它的名字形象地解释了它的结构组成，两侧的模型结构就像两个高塔一样，而最上面的互操作层则像两个塔尖搭建起的空中走廊，负责两侧信息的沟通。

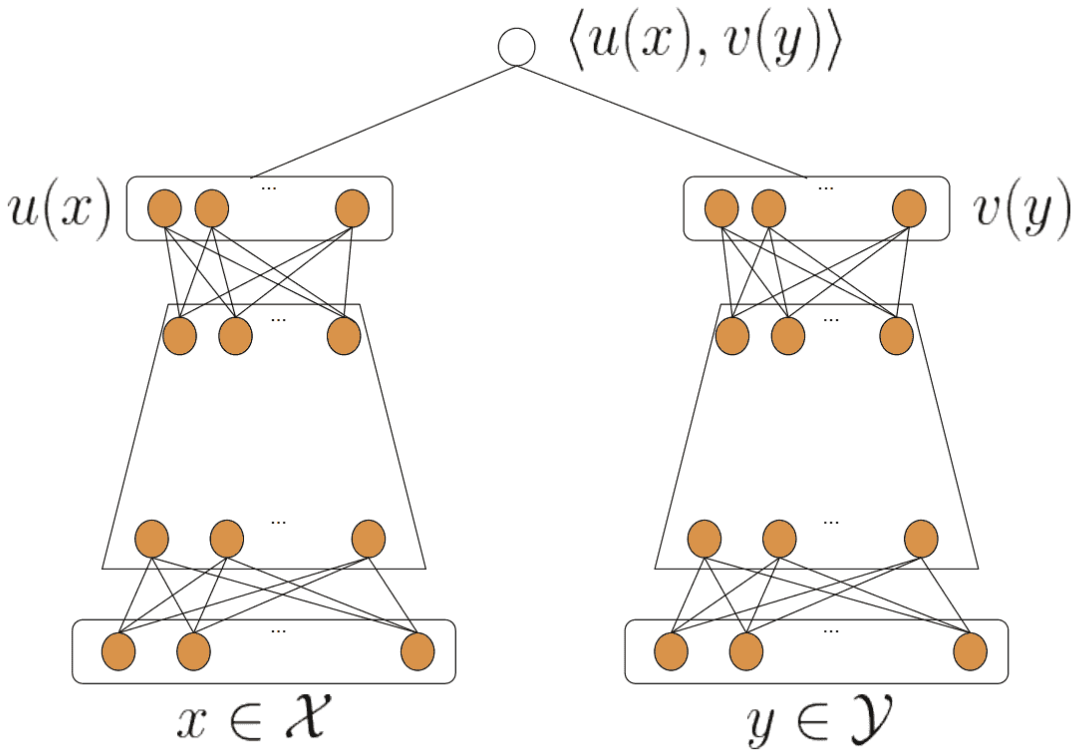

图4 双塔模型结构
（出自论文 Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations）

对于 NerualCF 来说，它只利用了用户 ID 作为“用户塔”的输入特征，用物品 ID 作为“物品塔”的输入特征。事实上，我们完全可以把其他用

户和物品相关的特征也分别放入用户塔和物品塔，让模型能够学到的信息更全面。比如说，YouTube 在构建用于召回层的双塔模型时，就分别在用户侧和物品侧输入了多种不同的特征，如图 5 所示。

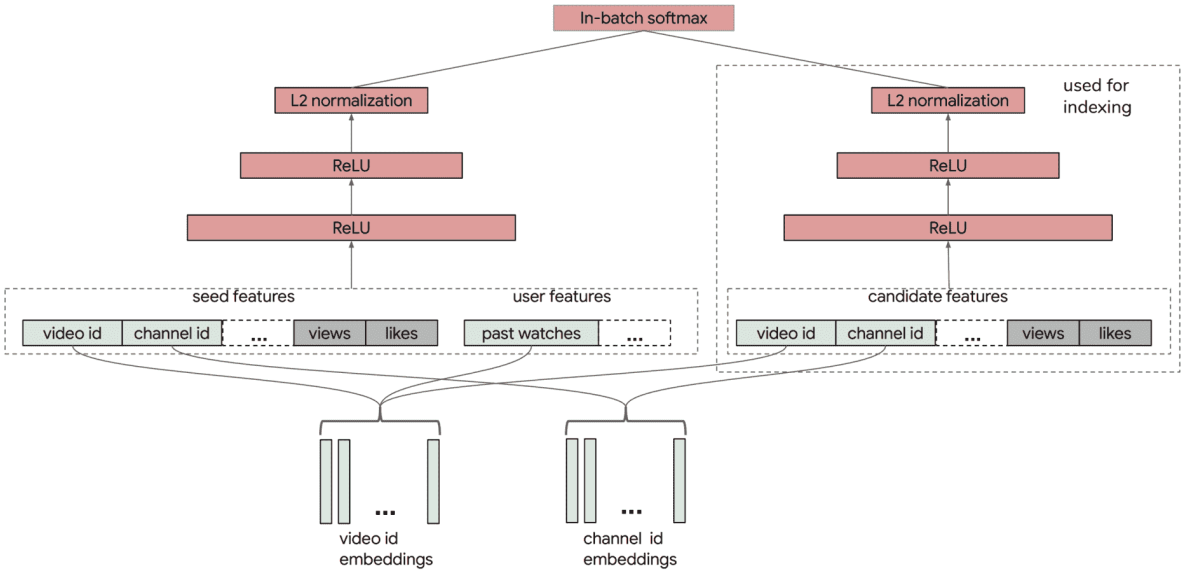

图5 YouTube双塔召回模型的架构
（出自论文 Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations）

我们看到，YouTube 召回双塔模型的用户侧特征包括了用户正在观看的视频 ID、频道 ID（图中的 seed features）、该视频的观看数、被喜欢的次数，以及用户历史观看过的视频 ID 等等。物品侧的特征包括了候选视频的 ID、频道 ID、被观看次数、被喜欢次数等等。在经过了多层 ReLU 神经网络的学习之后，双塔模型最终通过 softmax 输出层连接两部分，输出最终预测分数。

看到这里，你可能会有疑问，这个双塔模型相比我们之前学过的 Embedding MLP 和 Wide&Deep 有什么优势呢？其实在实际工作

中，双塔模型最重要的优势就在于它易上线、易服务。为什么这么说呢？

你注意看一下物品塔和用户塔最顶端的那层神经元，那层神经元的输出其实就是一个全新的物品 Embedding 和用户 Embedding。拿图 4 来说，物品塔的输入特征向量是 x，经过物品塔的一系列变换，生成了向量 u(x)，那么这个 u(x) 就是这个物品的 Embedding 向量。同理，v(y) 是用户 y 的 Embedding 向量，这时，我们就可以把 u(x) 和 v(y) 存入特征数据库，这样一来，线上服务的时候，我们只要把 u(x) 和 v(y) 取出来，再对它们做简单的互操作层运算就可以得出最后的模型预估结果了！

所以使用双塔模型，我们不用把整个模型都部署上线，只需要预存物品塔和用户塔的输出，以及在线上实现互操作层就可以了。如果这个互操作层是点积操作，那么这个实现可以说没有任何难度，这是实际应用中非常容易落地的，也是工程师们喜闻乐见的，这也正是双塔模型在业界巨大的优势所在。

正是因为这样的优势，双塔模型被广泛地应用在 YouTube、Facebook、百度等各大公司的推荐场景中，持续发挥着它的能量。

NeuralCF 的 TensorFlow 实现

熟悉了 NerualCF 和双塔模型的结构之后，我们就可以使用 TensorFlow 来实现它们了。通过之前 Embedding+MLP 模型以及 Wide&Deep 模型的实现，我想你对 TensorFlow 中读取数据，定义特征，训练模型的过程肯定已经驾轻就熟了。我们只要更改之前代码中模型定义的部分，就可以实现 NeuralCF。具体的代码你可以参考

SparrowRecsys 项目中的 NeuralCF.py，我只贴出了 NeuralCF 模型部分的实现。下面，我们重点讲解一下它们的实现思路。

```
# neural cf model arch two. only embedding in each
def neural_cf_model_1(feature_inputs, item_feature
    # 物品侧特征层
    item_tower = tf.keras.layers.DenseFeatures(ite
    # 用户侧特征层
    user_tower = tf.keras.layers.DenseFeatures(use
    # 连接层及后续多层神经网络
    interact_layer = tf.keras.layers.concatenate([
    for num_nodes in hidden_units:
        interact_layer = tf.keras.layers.Dense(num
    # sigmoid单神经元输出层
    output_layer = tf.keras.layers.Dense(1, activa
    # 定义keras模型
    neural_cf_model = tf.keras.Model(feature_input
    return neural_cf_model
```

你可以看到代码中定义的生成 NeuralCF 模型的函数，它接收了四个输入变量。其中 `feature_inputs` 代表着所有的模型输入，`item_feature_columns` 和 `user_feature_columns` 分别包含了物品侧和用户侧的特征。在训练时，如果我们只在 `item_feature_columns` 中放入 `movie_id` ，在 `user_feature_columns` 放入 `user_id，` 就是 NeuralCF 的经典实现了。

通过 DenseFeatures 层创建好用户侧和物品侧输入层之后，我们会再利用 concatenate 层将二者连接起来，然后输入多层神经网络进行训练。如果想要定义多层神经网络的层数和神经元数量，我们可以通过设置 `hidden_units` 数组来实现。

除了经典的 NeuralCF 实现，我还基于双塔模型的原理实现了一个 NeuralCF 的双塔版本。你可以参考下面的模型定义。与上面的经典 NerualCF 实现不同，我把多层神经网络操作放到了物品塔和用户塔内部，让塔内的特征进行充分交叉，最后使用内积层作为物品塔和用户塔的交互层。具体的步骤你可以参考下面代码中的注释，实现过程很好理解，我就不再赘述了。

```
# neural cf model arch one. embedding+MLP in each
def neural_cf_model_2(feature_inputs, item_feature
    # 物品侧输入特征层
    item_tower = tf.keras.layers.DenseFeatures(ite
    # 物品塔结构
    for num_nodes in hidden_units:
        item_tower = tf.keras.layers.Dense(num_nod
    # 用户侧输入特征层
    user_tower = tf.keras.layers.DenseFeatures(use
    # 用户塔结构
    for num_nodes in hidden_units:
        user_tower = tf.keras.layers.Dense(num_nod
    # 使用内积操作交互物品塔和用户塔，产生最后输出
    output = tf.keras.layers.Dot(axes=1)([item_tow
    # 定义keras模型
    neural_cf_model = tf.keras.Model(feature_input
    return neural_cf_model
```

在实现了 Embedding MLP、Wide&Deep 和 NeuralCF 之后，相信你可以感觉到，实现甚至创造一个深度学习模型并不难，基于 TensorFlow 提供的 Keras 接口，我们可以根据我们的设计思路，像搭积木一样实现模型的不同结构，以此来验证我们的想法，这也正是深度推荐模型的魅力和优势。相信随着课程的进展，你不仅对这一点

能够有更深刻的感受，同时，你设计和实现模型的能力也会进一步加强。

小结

这节课，我们首先学习了经典推荐算法协同过滤的深度学习进化版本NerualCF。相比于矩阵分解算法，NeuralCF 用一个多层的神经网络，替代了矩阵分解算法中简单的点积操作，让用户和物品隐向量之间进行充分的交叉。这种通过改进物品隐向量和用户隐向量互操作层的方法，大大增加了模型的拟合能力。

利用 NerualCF 的思想，我们进一步学习了双塔模型。它通过丰富物品侧和用户侧的特征，让模型能够融入除了用户 ID 和物品 ID 外更丰富的信息。除此之外，双塔模型最大的优势在于模型服务的便捷性，由于最终的互操作层是简单的内积操作或浅层神经网络。因此，我们可以把物品塔的输出当作物品 Embedding，用户塔的输出当作用户 Embedding 存入特征数据库，在线上只要实现简单的互操作过程就可以了。

最后，我们继续使用 TensorFlow 实现了 NerualCF 和双塔模型，相信你能进一步感受到利用 TensorFlow 构建深度学习模型的便捷性，以及它和传统推荐模型相比，在模型结构灵活性上的巨大优势。

为了帮助你复习，我把刚才说的这些重点内容总结在了一张图里，你可以看看。

知识点	关键描述
NerualCF模型的结构	用多层神经网络替代矩阵分解模型中的点积操作
双塔模型	物品侧和用户侧融入更多特征，使用多层网络， 在互操作层使用点击操作或浅层神经网络
双塔模型的优势	保存物品塔和用户塔Embedding结果到数据库，线上实现简单的互操作进行预测，易实现、易上线
NerualCF实现流程	Item DenseFeatures+User DenseFeatures –> concatenate layer –> 多层Dense layers –> Sigmoid Dense layer
双塔模型实现流程	Item 多层Dense layers + User 多层Dense layers –> Dot layer

课后思考

对于我们这节课学习的双塔模型来说，把物品侧的 Embedding 和用户侧的 Embedding 存起来，就可以进行线上服务了。但如果我们把一些场景特征，比如当前时间、当前地点加到用户侧或者物品侧，那我们还能用这种方式进行模型服务吗？为什么？

欢迎把你的思考和疑惑写在留言区，也欢迎你把这节课转发出去，我们下节课见！