# 18｜Wide&Deep：怎样让你的模型既有想象力又有记忆力？

你好，我是王喆。

今天，我们来学习一个在业界有着巨大影响力的推荐模型，Google 的 Wide&Deep。可以说，只要掌握了 Wide&Deep，我们就抓住了深度推荐模型这几年发展的一个主要方向。那 Wide&Deep 模型是什么意思呢？我们把它翻译成中文就是“宽并且深的模型”。

这个名字看起来好像很通俗易懂，但你真的理解其中“宽”和“深”的含义吗？上一节课我们讲过 Embedding+MLP 的经典结构，因为 MLP 可以有多层神经网络的结构，所以它是一个比较“深”的模型，但 Wide&Deep 这个模型说的“深”和 MLP 是同样的意思吗？“宽”的部分又是什么样的呢？宽和深分别有什么不同的作用呢？以及我们为什么要把它们结合在一起形成一个模型呢？

这节课，就让我们就带着这诸多疑问，从模型的结构开始学起，再深入到 Wide&Deep 在 Google 的应用细节中去，最后亲自动手实现这个模型。

## Wide&Deep 模型的结构

首先，我们来看看 Wide&Deep 模型的结构，理解了结构再深入去学习细节，我们才能掌握得更好。

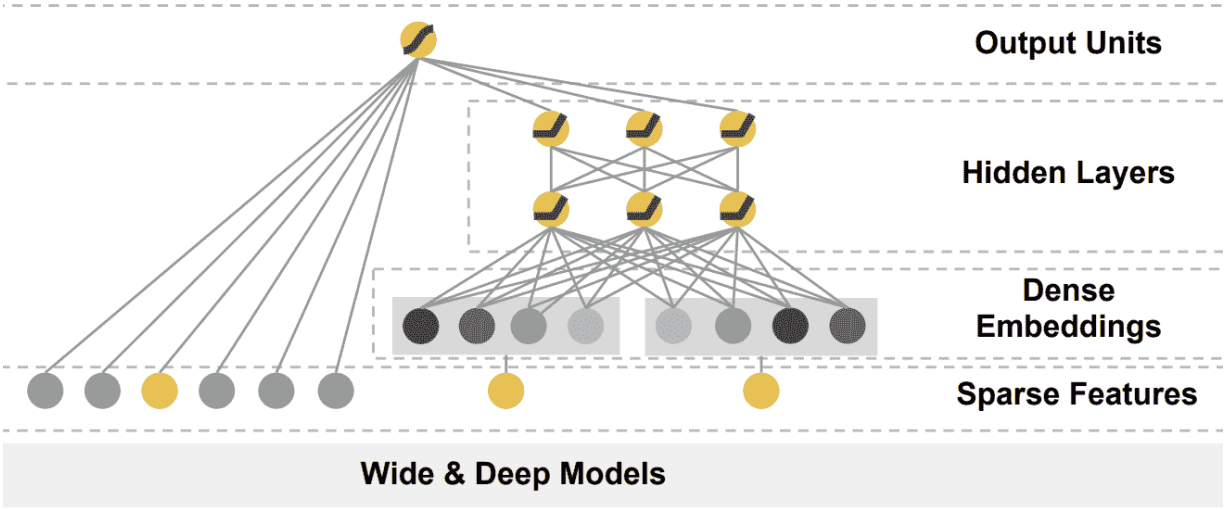


图1 Wide&Deep模型结构
（出自Wide & Deep Learning for Recommender Systems ）

上图就是 Wide&Deep 模型的结构图了，它是由左侧的 Wide 部分和右侧的 Deep 部分组成的。Wide 部分的结构太简单了，就是把输入层直接连接到输出层，中间没有做任何处理。Deep 层的结构稍复杂，但我相信你也不会陌生，因为它就是我们上节课学习的 Embedding+MLP 的模型结构。

知道了 Wide&Deep 模型的结构之后，我们先来解决第一个问题，那就是 Google 为什么要创造这样一个混合式的模型结构呢？这我们还得从 Wide 部分和 Deep 部分的不同作用说起。

简单来说，Wide 部分的主要作用是让模型具有较强的“记忆能力”（Memorization），而 Deep 部分的主要作用是让模型具有“泛化能力”（Generalization），因为只有这样的结构特点，才能让模型兼具逻辑回归和深度神经网络的优点，也就是既能快速处理和记忆大量历史行为特征，又具有强大的表达能力，这就是 Google 提出这个模型的动机。

那么问题又来了，所谓的“记忆能力”和“泛化能力”到底指什么呢？这我们就得好好聊一聊了，因为理解这两种能力是彻底理解 Wide&Deep 模型的关键。

**模型的记忆能力**

**所谓的 “记忆能力”，可以被宽泛地理解为模型直接学习历史数据中物品或者特征的“共现频率”，并且把它们直接作为推荐依据的能力 。**就像我们在电影推荐中可以发现一系列的规则，比如，看了 A 电影的用户经常喜欢看电影 B，这种“因为 A 所以 B”式的规则，非常直接也非常有价值。

但这类规则有两个特点：一是数量非常多，一个“记性不好”的推荐模型很难把它们都记住；二是没办法推而广之，因为这类规则非常具体，没办法或者说也没必要跟其他特征做进一步的组合。就像看了电影 A 的用户 80% 都喜欢看电影 B，这个特征已经非常强了，我们就没必要把它跟其他特征再组合在一起。

现在，我们就可以回答开头的问题了，为什么模型要有 Wide 部分？就是因为 Wide 部分可以增强模型的记忆能力，让模型记住大量的直接且重要的规则，这正是单层的线性模型所擅长的。

**模型的泛化能力**

接下来，我们来谈谈模型的“泛化能力”。**“泛化能力”指的是模型对于新鲜样本、以及从未出现过的特征组合的预测能力。** 这怎么理解呢？我们还是来看一个例子。假设，我们知道 25 岁的男性用户喜欢看电影 A，35 岁的女性用户也喜欢看电影 A。如果我们想让一个只有记忆能力的模型回答，“35 岁的男性喜不喜欢看电影 A”这样的问题，这个模型就会“说”，我从来没学过这样的知识啊，没法回答你。

这就体现出泛化能力的重要性了。模型有了很强的泛化能力之后，才能够对一些非常稀疏的，甚至从未出现过的情况作出尽量“靠谱”的预测。

回到刚才的例子，有泛化能力的模型回答“35 岁的男性喜不喜欢看电影 A”这个问题，它思考的逻辑可能是这样的：从第一条知识，“25 岁的男性用户喜欢看电影 A“中，我们可以学到男性用户是喜欢看电影 A 的。从第二条知识，“35 岁的女性用户也喜欢看电影 A”中，我们可以学到 35 岁的用户是喜欢看电影 A 的。那在没有其他知识的前提下，35 岁的男性同时包含了合适的年龄和性别这两个特征，所以他大概率也是喜欢电影 A 的。这就是模型的泛化能力。

事实上，我们学过的矩阵分解算法，就是为了解决协同过滤“泛化能力”不强而诞生的。因为协同过滤只会“死板”地使用用户的原始行为特征，而矩阵分解因为生成了用户和物品的隐向量，所以就可以计算任意两个用户和物品之间的相似度了。这就是泛化能力强的另一个例子。

从上节课中我们学过深度学习模型有很强的数据拟合能力，在多层神经网络之中，特征可以得到充分的交叉，让模型学习到新的知识。因此，Wide&Deep 模型的 Deep 部分，就沿用了上节课介绍的 Embedding+MLP 的模型结构，来增强模型的泛化能力。

好，清楚了记忆能力和泛化能力是什么之后，让我们再回到 Wide&Deep 模型提出时的业务场景上，去理解 Wide&Deep 模型是怎么综合性地学习到记忆能力和泛化能力的。

## Wide&Deep 模型的应用场景

Wide&Deep 模型是由 Google 的应用商店团队 Google Play 提出的，在 Google Play 为用户推荐 APP 这样的应用场景下，Wide&Deep 模型的推荐目标就显而易见了，就是应该尽量推荐那些用户可能喜欢，愿意安装的应用。那具体到 Wide&Deep 模型中，Google Play 团队是如何为 Wide 部分和 Deep 部分挑选特征的呢？下面，我们就一起来看看。

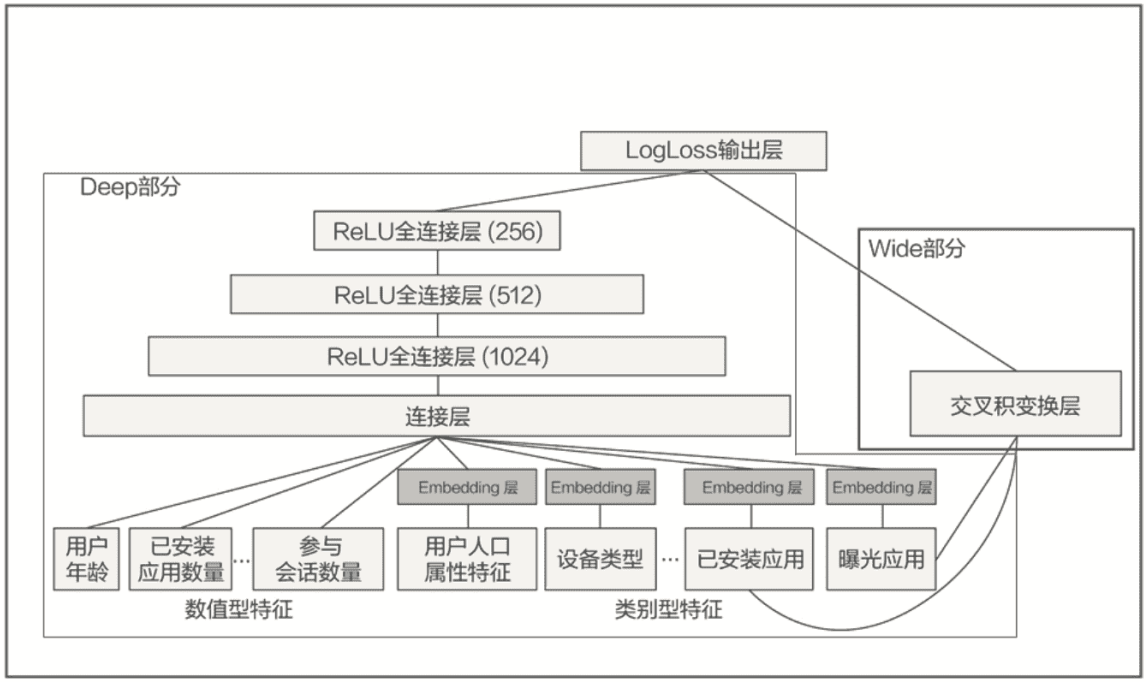


图2 Google Play Wide&Deep模型的细节
（出自Wide & Deep Learning for Recommender Systems ）

我们先来看看图 2，它补充了 Google Play Wide&Deep 模型的细节，让我们可以清楚地看到各部分用到的特征是什么。我们先从右边 Wide 部分的特征看起。这部分很简单，只利用了两个特征的交叉，这两个特征是“已安装应用”和“当前曝光应用”。这样一来，Wide 部分想学到的知识就非常直观啦，就是希望记忆好“如果 A 所以 B”这样的

简单规则。在 Google Play 的场景下，就是希望记住“如果用户已经安装了应用 A，是否会安装 B”这样的规则。

接着，我们再来看看左边的 Deep 部分，它就是一个非常典型的 Embedding+MLP 结构了。我们看到其中的输入特征很多，有用户年龄、属性特征、设备类型，还有已安装应用的 Embedding 等等。我们把这些特征一股脑地放进多层神经网络里面去学习之后，它们互相之间会发生多重的交叉组合，这最终会让模型具备很强的泛化能力。

比如说，我们把用户年龄、人口属性和已安装应用组合起来。假设，样本中有 25 岁男性安装抖音的记录，也有 35 岁女性安装抖音的记录，那我们该怎么预测 25 岁女性安装抖音的概率呢？这就需要用到已有特征的交叉来实现了。虽然我们没有 25 岁女性安装抖音的样本，但模型也能通过对已有知识的泛化，经过多层神经网络的学习，来推测出这个概率。

总的来说，Wide&Deep 通过组合 Wide 部分的线性模型和 Deep 部分的深度网络，取各自所长，就能得到一个综合能力更强的组合模型。

## Wide&Deep 模型的 TensorFlow 实现

在理解了 Wide&Deep 模型的原理和技术细节之后，就又到了“show me the code”的环节了。接下来，我们就动手在 SparrowRecsys 上实现 Wide&Deep 模型吧！

通过上节课的实战，我相信你已经熟悉了 TensorFlow 的大部分操作，也清楚了载入训练数据，创建 TensorFlow 所需的 Feature column 的方法。因此，Wide&Deep 模型的实践过程中，我们会重点关注定义模型的部分。

这里，我们也会像上节课一样，继续使用 TensorFlow 的 Keras 接口来构建 Wide&Deep 模型。具体的代码如下：

```
# wide and deep model architecture
# deep part for all input features
deep = tf.keras.layers.DenseFeatures(numerical_col
deep = tf.keras.layers.Dense(128, activation='relu
deep = tf.keras.layers.Dense(128, activation='relu
# wide part for cross feature
wide = tf.keras.layers.DenseFeatures(crossed_featu
both = tf.keras.layers.concatenate([deep, wide])
output_layer = tf.keras.layers.Dense(1, activation
model = tf.keras.Model(inputs, output_layer)
```

从代码中我们可以看到，在创建模型的时候，我们依次配置了模型的 Deep 部分和 Wide 部分。我们先来看 Deep 部分，它是输入层加两层 128 维隐层的结构，它的输入是类别型 Embedding 向量和数值型特征，实际上这跟上节课 Embedding+MLP 模型所用的特征是一样的。

Wide 部分其实不需要有什么特殊操作，我们直接把输入特征连接到了输出层就可以了。但是，这里我们要重点关注一下 Wide 部分所用的特征 crossed_feature。

```
movie_feature = tf.feature_column.categorical_colu
rated_movie_feature = tf.feature_column.categorica
crossed_feature = tf.feature_column.crossed_column
```

在生成 crossed_feature 的过程中，我其实仿照了 Google Play 的应用方式，生成了一个由“用户已好评电影”和“当前评价电影”组成的一个交叉特征，就是代码中的 crossed_feature，设置这个特征的目的在于

让模型记住好评电影之间的相关规则，更具体点来说就是，就是让模型记住“一个喜欢电影 A 的用户，也会喜欢电影 B”这样的规则。

当然，这样的规则不是唯一的，需要你根据自己的业务特点来设计，比如在电商网站中，这样的规则可以是，购买了键盘的用户也会购买鼠标。在新闻网站中，可以是打开过足球新闻的用户，也会点击 NBA 新闻等等。

在 Deep 部分和 Wide 部分都构建完后，我们要使用 `concatenate layer` 把两部分连接起来，形成一个完整的特征向量，输入到最终的 sigmoid 神经元中，产生推荐分数。

总的来说，在我们上一节的 Embedding MLP 模型基础上实现 Wide&Deep 是非常方便的，Deep 部分基本没有变化，我们只需要加上 Wide 部分的特征和设置就可以了。Wide&Deep 的全部相关代码，我都实现在了 SparrowRecsys 的 WideNDeep.py 文件中，你可以直接参考源代码。但我更希望，你能尝试设置不同的特征，以及不同的参数组合，来真实地体验一下深度学习模型的调参过程。

## 小结

这节课，我们一起实现了业界影响力非常大的深度学习模型 Wide&Deep，它是由 Wide 部分和 Deep 部分组成的。其中，Wide 部分主要是为了增强模型的“记忆能力”，让模型记住“如果 A，那么 B”这样的简单但数量非常多的规则。Deep 部分是为了增强模型的“泛化能力”，让模型具备对于稀缺样本、以及从未出现过的特征组合的预测能力。Wide&Deep 正是通过这样取长补短的方式，让模型的综合能力提升。

在具体实践的时候，我们继续使用 TensorFlow 的 Keras 接口实现了 Wide&Deep 模型。相比上节课 Embedding MLP 模型的实现，我们新加入了“用户已好评电影”和“当前评价电影”组成的交叉特征 crossed_feature，让 Wide 部分学习“一个喜欢电影 A 的用户，也会喜欢电影 B”这样的规则。

好了，这就是我们这节课的主要内容，同样，我也把重要的知识点总结在了表格里，你可以利用它来巩固复习。

| 知识点 | 关键描述 |
| --- | --- |
| Wide&Deep模型的结构 | 由单层的Wide部分和具备多层结构的Deep部分组成 |
| 模型的记忆能力 | 模型直接学习历史数据中物品或者特征的“共现频率”，并以此直接作为推荐依据的能力 |
| 模型的泛化能力 | 模型对于新鲜样本、以及从未出现过的特征组合的预测能力 |
| TensorFlow的Wide&Deep模型实现 | Wide部分+Deep部分 –> concatenate –> sigmoid输出神经元 |
| TensorFlow生成交叉特征的方法 | tf.feature_column.crossed_column |



## 课后思考

对于 Deep 部分来说，你觉得我们一股脑地把所有特征都扔进 MLP 中去训练，这样的方式有没有什么改进的空间？比如说，“用户喜欢的电影风格”和“电影本身的风格”这两个特征，我们能不能进一步挖掘出它们之间的相关性，而不是简单粗暴地扔给神经网络去处理呢？

欢迎把你的思考和疑问写在留言区，如果的你朋友也正在为 Wide&Deep 模型的实现而困扰，欢迎你把这节课转发给他，我们下节课见！