运用多输入模型优化不同维度特征

背景介绍

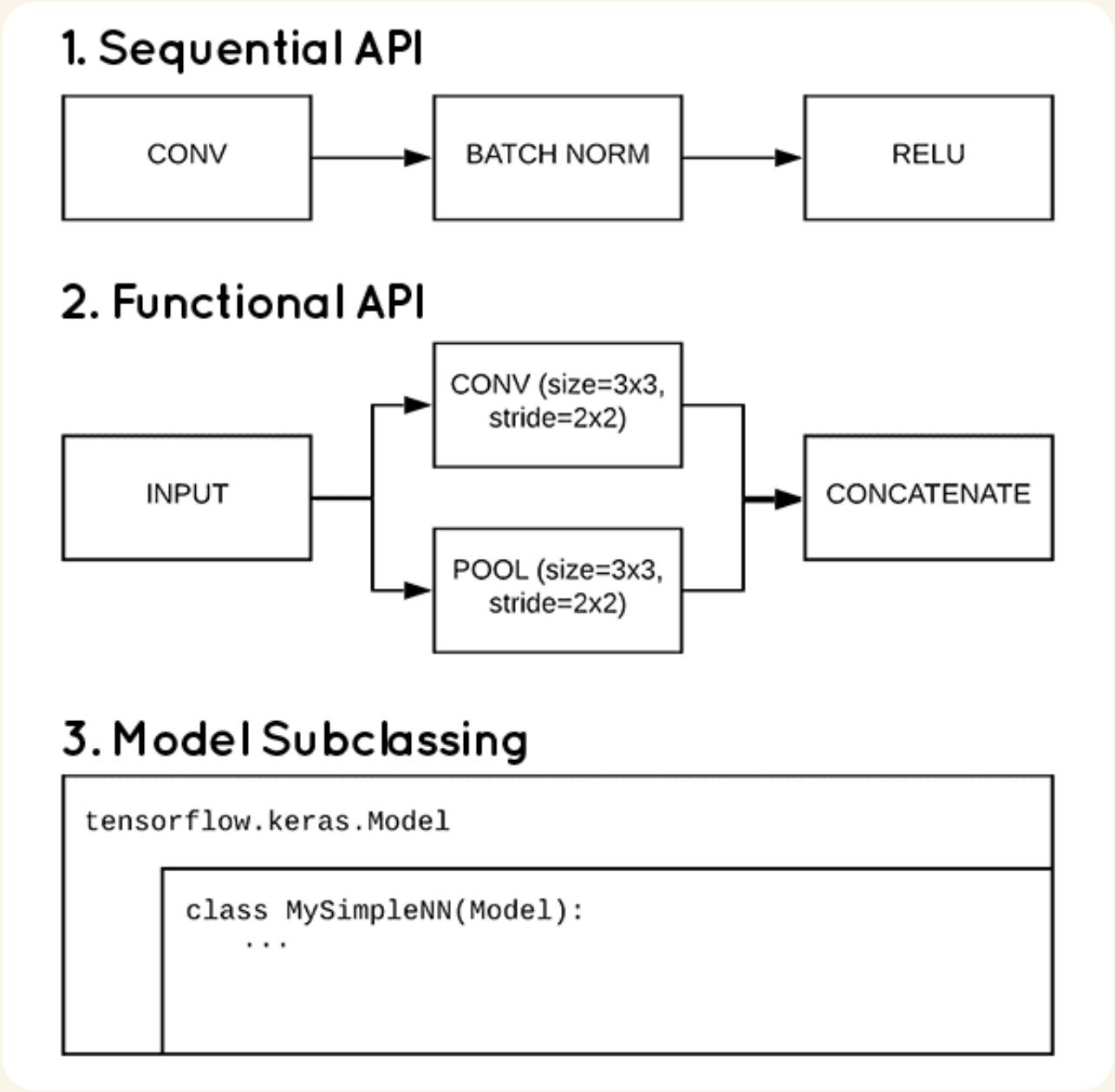
使用神经网络模型做用户付费金额预测，一种常见的特征工程场景，是把某个特定付费区间，比如付费金额大于10小于等于12的付费用户信息处理成特征，既可以将这部分用户的付费人数作为特征，也可以将这部分用户的付费金额总和作为特征。

对于上述的两种不同维度的特征，发现一个有意思的现象，按用户属性，比如国家，平台等属性对用户样本分组，部分样本在有付费人数特征的模型（简称模型A）上表现更好，即预测结果的误差更小，另一部分样本在有付费金额特征的模型（简称模型B）上表现更好。

如果能把这两个模型合并成一个模型，理论上可以让不同分组的样本都取得最好的效果。尝试把付费人数特征和付费金额特征同时放入同一个模型（简称模型C）里，但由于付费金额特征的值比付费人数的值要大得多，整体模型表现会更接近只有付费金额特征的模型B，达不到理想效果。

优化方案

针对这种多组不同维度的特征输入同一个模型（简称模型D），keras支持通过functional API构建多输入模型，能很好地同时处理多组特征，不会因为一组特征在数值上更明显而忽略另一组特征。（需要TensorFlow 2.0以上版本）



Talk is cheap. Show me the code.

废话少说，上代码。

1. 准备数据特征

   x_user_train是付费人数特征，x_price_train是付费金额特征。

2. 构建输入

   分别使用付费人数特征和付费金额特征构建不同的输入输出。

```
    # 构建付费人数特征部分的输入输出
    in_user = layers.Input(shape = [len(x_user_train.columns)],
name='user')
    out_user = layers.Dense(len(x_user_train.columns),
                            activation='linear',

 kernel_constraint=keras.constraints.NonNeg())(in_user)

    # 构建付费金额特征部分的输入输出
    in_price = layers.Input(shape = [len(x_price_train.columns)],
name='price')
    out_price = layers.Dense(len(x_price_train.columns),
                             activation='linear',

kernel_constraint=keras.constraints.NonNeg())(in_price)
```

3. 组合输出
   将上述两个输出组合成一个输出。

```
    # 将两个输出组合成一个输出
    out = layers.concatenate([out_user, out_price])
    out = layers.Dense(len(x_user_train.columns) +
len(x_price_train.columns),
                       activation="linear",

kernel_constraint=keras.constraints.NonNeg())(out)
    out = layers.Dense(1,
                       activation="linear",

kernel_constraint=keras.constraints.NonNeg())(out)
```

4. 构建模型

   构建一个两个输入一个输出的模型。

```
    # 构建一个两个输入一个输出的模型
    model = models.Model(inputs=[in_user, in_price], outputs=out)
```

5. 运行结果

1. 在所有测试样本量的总体预测误差上，模型D明显优于前面三个模型，在误差指标上有3%到6%的显著提升。

	模型A	模型B	模型C	模型D
总体预测误差	20.591%	17.791%	17.656%	14.531%

2. 模型D同时保留了部分样本在模型A上表现更好，另一部分样本在模型B上表现更好的特点，达到理想效果。

结论

构建多输入模型处理多组不同维度的特征，对比单组特征模型，能显著提升模型技术指标和业务效果。同时，多输入模型能很好保留不同类型样本适应不同维度特征的特点，堪称神器。

附录

建立模型部分的完整代码如下：

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import models


def build_model(x_user_train, x_price_train):
    # 构建付费人数特征部分的输入
    in_user = layers.Input(shape=[len(x_user_train.columns)],
name='user')
    out_user = layers.Dense(len(x_user_train.columns),
                            activation='linear',

 kernel_constraint=keras.constraints.NonNeg())(in_user)

    # 构建付费金额特征部分的输入
    in_price = layers.Input(shape=[len(x_price_train.columns)],
name='price')
```

```
    out_price = layers.Dense(len(x_price_train.columns),
                             activation='linear',

kernel_constraint=keras.constraints.NonNeg())(in_price)

    # 将两个输入组合成一个输出
    out = layers.concatenate([out_user, out_price])
    out = layers.Dense(len(x_user_train.columns) +
len(x_price_train.columns),
                       activation="linear",
                       kernel_constraint=keras.constraints.NonNeg())
(out)
    out = layers.Dense(1,
                       activation="linear",
                       kernel_constraint=keras.constraints.NonNeg())
(out)

    # 构建两个输入一个输出的模型
    model = models.Model(inputs=[in_user, in_price], outputs=out)

    optimizer = tf.keras.optimizers.RMSprop(0.001)

    model.compile(loss='mse',
                  optimizer=optimizer,
                  metrics=['mse', 'mape'])
    return model
```