**《深度学习-keras的Sequential与Functional模型》**

大家好，上期推文介绍了Keras的一些特点和一些基本的知识点，不知道大家在平时的时间有没有自己学习一下深度学习相关的知识，或者机器学习相关的知识呢？有这些的预备知识对于学这个专题还是有帮助的。

本期内容我们先来聊一聊Keras中模型的种类，也就是来聊聊Sequential与Functional模型，即序贯模型和函数式模型，我们一个个来看。

**Sequential模型**

神经网络模型是一种将信息朝着某一个方向进行传递的模型，方向性的传递形式就很适合以一种顺序(序贯)的数据结构来进行表示，有一种“一条路走到黑”的感觉。在工程项目中使用序贯模型可以解决很多的实际需求,它的主要特点如下：

1. 简单的线性结构
2. 从开始到结束的结构顺序
3. 没有分叉结构
4. 多个网络层(输入层，隐层，激活层等等)的线性堆叠

不要觉得它简单功能就不强大，要知道全连接神经网络、卷积神经网络(CNN)、循环神经网络(RNN)等网络都可以使用Sequential Model来进行构建。其构建

模型通常分为五个步骤：  
1.定义模型  
2.定义优化目标  
3.输入数据  
4.训练模型  
5.评估模型的性能

这里要提醒的是，大家在学习Keras中的时候要有层(layers)的概念。如卷积层，池化层，全连接层，LSTM层等等。在Keras中我们构建模型可以直接使用这些层来构建一个目标神经网络。接下来我们简单的来说说这几个步骤：

**定义模型**

可以通过向Sequential模型传递一个层的列表来构造该模型：

from keras.models import Sequential  
from keras.layers import Dense, Activation  
model = Sequential([  
 Dense(32, input\_shape=(784,)),  
 Activation(**'relu'**),  
 Dense(10),  
 Activation(**'softmax'**),  
 ])

上述代码是使用层的列表来构建Sequential模型的，其中堆叠了四层：

第一层是全连接层，神经元个数为32，input\_shape=(784,)；

第二层是激活层，激活函数为relu；

第三层是全连接层，神经元个数为10；

第二层是激活层，激活函数为softmax；

当然了，我们也可以使用add()方法进行层的堆叠：

model = Sequential()  
model.add(Dense(32, input\_shape=(784,)))  
model.add(Activation(**'relu'**))  
model.add(Dense(10))  
model.add(Activation(**'softmax'**))

个人比较喜欢第二种方式，十分简单

大家可能有一个疑问，在第一层使用了input\_shape，其他层却没有传入这个参数，这是为什么呢？这是因为：后面的各个层可以自动的推导出中间数据的shape，因此就不用传参了。

**定义优化目标**

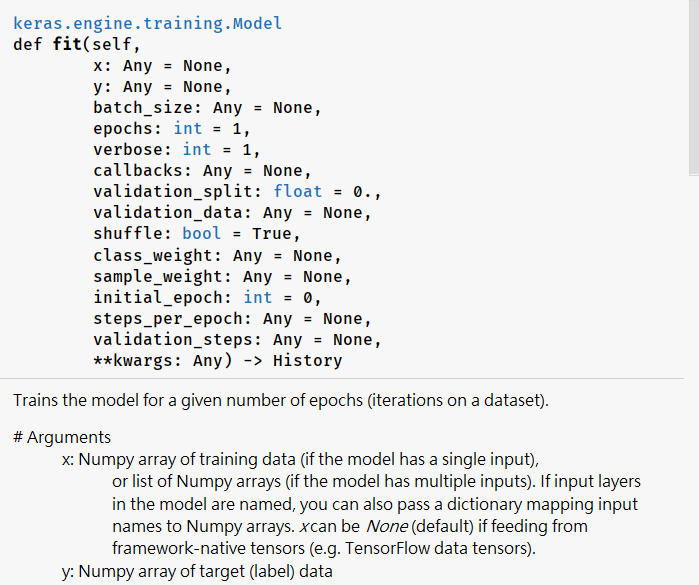
我们知道不同种类的任务有不同的优化目标，也就存在要使用不同的优化器(Optimizer)、损失函数(Loss)和评估标准(Metrics)。在实际的项目工程中，多分类的损失函数我们一般选择categorical\_crossentropy，回归问题常用MSE损失函数。

其实这个定义优化目标的过程是整个构建模型中最重要的部分，因为在这部分我们要选择优化器和损失函数。通过选择的优化器和损失函数来对模型中的参数进行计算和更新。在Keras中我们是使用compile()方法来完成：

*# 多分类问题*model.compile(optimizer=**'rmsprop'**,  
 loss=**'categorical\_crossentropy'**,  
 metrics=[**'accuracy'**])  
*# 回归问题*model.compile(optimizer=**'rmsprop'**,  
 loss=**'mse'**)

**输入数据和训练模型**

定义优化目标以后就需要进行数据的输入了，在Keras中是使用fit()方法将数据传递到构建的模型当中的。fit()方法具有较多的参数，其返回值是一个History类对象，这个对象包含两个属性，分别为epoch和history，epoch为训练轮数，history字典类型，包含val\_loss,val\_acc,loss,acc四个key值。



大家可以查看到这些参数的具体含义。一般情况下我们使用的参数主要有：x, y, batch\_size，epochs，verbose，validation\_split, validation\_data等。如：

model.fit(X\_train, y\_train, epochs=10, batch\_size=64, verbose=1, validation\_split=0.05)

上述代码表示我们输入了X\_train, y\_train训练集数据，迭代次数为10，数据批次大小为64，verbose = 1表示输出进度条记录，validation\_split=0.05表示从训练集中取出0.05比例的数据作为验证集。

**评估模型性能**

fit()之后就是在测试数据集上对我们的数据进行验证了，在Keras中我们可以这样来进行模型的评估：

score = model.evaluate(X\_test, y\_test, batch\_size = 64)

这个模型的评估方法使用很简单。大家可以看上篇推文，推文中的手写字体识别模型就是使用Sequential模型来搭建的，代码我也贴一下：

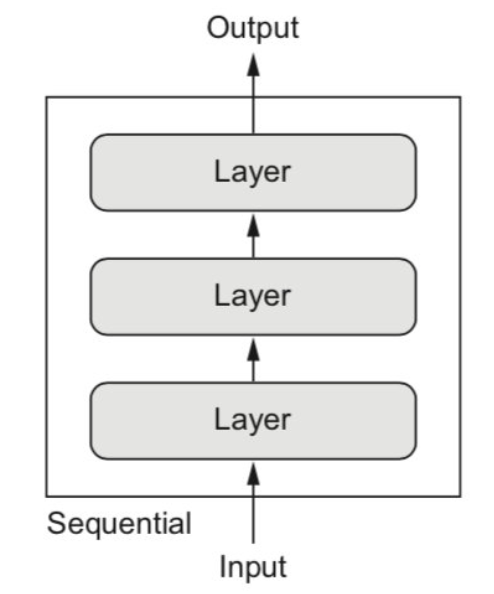
from keras.datasets import mnist  
from keras.utils import np\_utils  
from keras.models import Sequential  
from keras.layers import Dense,Activation  
path = **r'C:\Users\LEGION\Desktop\datasets\mnist.npz'**(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data(path)  
  
X\_train = X\_train.reshape(len(X\_train),-1)  
X\_test = X\_test.reshape(len(X\_test), -1)  
X\_train = X\_train.astype(**'float32'**)/255  
X\_test = X\_test.astype(**'float32'**)/255  
  
y\_train = np\_utils.to\_categorical(y\_train)  
y\_test = np\_utils.to\_categorical(y\_test)  
  
model = Sequential()  
model.add(Dense(512, input\_shape=(28\*28,),activation=**'relu'**))  
model.add(Dense(10,activation=**'softmax'**))  
model.compile(optimizer=**'adam'**, loss=**'categorical\_crossentropy'**, metrics=[**'accuracy'**])  
model.fit(X\_train, y\_train, epochs=10, batch\_size=64, verbose=1, validation\_split=0.05)  
loss, accuracy = model.evaluate(X\_test, y\_test)  
Testloss, Testaccuracy = model.evaluate(X\_test, y\_test)  
print(**'Testloss:'**, Testloss)  
print(**'Testaccuracy:'**, Testaccuracy)

**Functional模型**

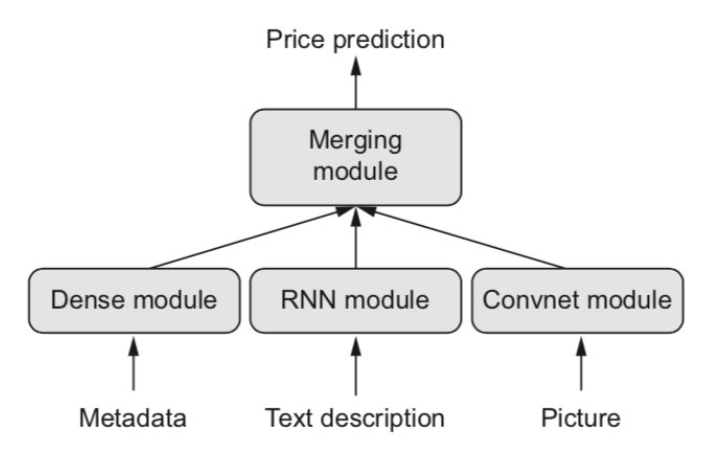
接下来我们来看一下Functional模型。函数式模型称作Functional，仔细研究发现其类名是Model，因此有时候也用Model来代表函数式模型，即Model模型。

我们知道Sequential模型类似一条路走到㡳的结构，如VGG也是一条路走到㡳的模型。那么在实际的工程项目中，如果我们的目标是多个输出或者非循环有向模型，此时我们就应该选择使用函数式模型来构建。

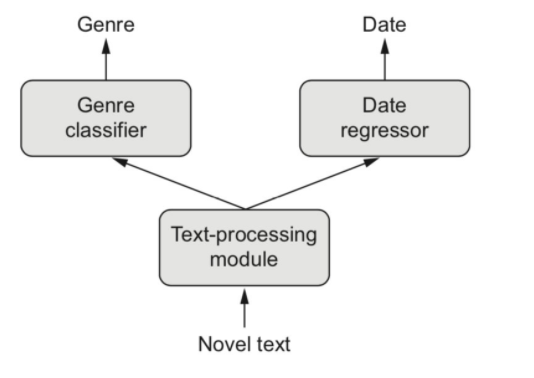
函数式模型是最广泛的一类模型，Sequential模型是它的特例，下图给出了几种模型的简单示例：



Sequential模型示例



多输入单输出Functional模型的示例



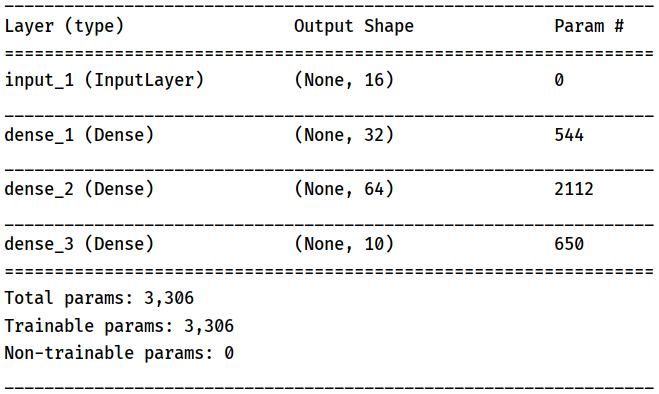
单输入多输出Functional模型的示例

Keras 的函数式模型把层（layers）当作函数来使用，接收张量并返回张量。

我们先使用函数式模型来实现一个序贯模型（全连接网络模型）：

from keras import Input  
from keras import layers  
from keras import models  
  
*# 构建输入层的张量*input\_tensor = Input(shape=(16,))  
*# 将输入层的返回的input\_tensor作为参数输入到第一层网络的Dense中*dense\_1 = layers.Dense(32, activation=**'relu'**)(input\_tensor)  
*# 将第一层的返回的dense\_1作为参数输入到第二层的Dense中*dense\_2 = layers.Dense(64, activation=**'relu'**)(dense\_1)  
*# 最后一层来构建输出层，并使用10个神经元来进行分类*output\_tensor = layers.Dense(10, activation=**'softmax'**)(dense\_2)  
*# 将输入层和输出层的张量输入模型中*model = models.Model(input\_tensor, output\_tensor)  
  
*# 显示模型的相关信息*model.summary()

上述模型的网络结构示意图如下：



我们从图中可以了解到每一层的输出形状，参数情况。

那么基于Functional模型构建过程一般的步骤如下：

1. 首先使用Keras的Input()方法来构建输入层，该层一般不含可输入的参数。
2. 将输入层返回的张量继续输入到下一层，如果还有下一层，继续此操作
3. 将输入层和输出层的张量输入模型中

上述代码只是使用函数式模型来构建一个非常简单的全连接网络。**我们还未实现多输入单输出，单输入多输出和多输入多输出的模型，后期打算一个种类使用一期推文来进行详细的介绍**。大家可以先学一下机器学习中的多模型融合相关的技术，这对于后期的学习十分有利。

**总结**

想必大家也已经知道了一些[Sequential顺序模型](https://blog.csdn.net/weixin_42886817/article/details/99831718" \l "Sequential__16" \t "_self)和[Functional模型](https://blog.csdn.net/weixin_42886817/article/details/99831718#Model__162)相关的知识。对于Sequential而言掌握它是很容易的，大家或许对于函数式模型还不是很熟悉，但是没关系，后续我们会详细的探究不同输入和输出形式的函数式模型，到时候也会结合实例来进行阐述。

由于本人接下来一段时间可能出差，所以下篇的推文内容的还没固定，不过有两种情况：

1. 讲述Keras实现机器学习的线性回归和非线性回归模型。
2. 新专题爬虫的相关知识。