1. 模型的构建
2. keras中的Model

Model用于构建模型，即将各种层进行组织和连接，并封装成一个整体，即模型，将输入数据通过各种层得到输出。

我们可以将模型通过继承Model，这个类类定义自己的模型，在继承类中，我们需要重写构造函数\_\_init\_\_()和模型调用函数call(input)，还可以添加自定义的方法。

class MyModel(tf.keras.Model):

2. def \_\_init\_\_(self):

3. super().\_\_init\_\_() # Python 2 下使用 super(MyModel, self).\_\_init\_\_()

4. # 此处添加初始化代码（包含 call 方法中会用到的层），例如

5. # layer1 = tf.keras.layers.BuiltInLayer(...)

6. # layer2 = MyCustomLayer(...)

7.

8. def call(self, input):

9. # 此处添加模型调用的代码（处理输入并返回输出），例如

10. # x = layer1(input)

11. # output = layer2(x)

12. return output

13.

14. # 还可以添加自定义的方法

1. keras中的layers

layers将各种计算流程和变量进行了封装，内置了大量常用的预定义层，如卷积层，池化层，全连接层等等。

1. 构建模型类

将线性回归的模型封装到一个类中。

（一）代码

import tensorflow as tf  
  
x = tf.constant([[1., 2., 3.], [4., 5., 6.]])  
y = tf.constant([[10.], [20.]])  
  
class Linear(tf.keras.Model):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.dense = tf.keras.layers.Dense(  
 units =1, ***#输出维度*** activation=None, ***#激活函数*** kernel\_initializer=tf.zeros\_initializer(), ***#权重参数*** bias\_initializer=tf.zeros\_initializer ***#偏置参数*** )  
  
 def call(self, input):  
 output = self.dense(input)  
 return output  
  
model = Linear()  
opt = tf.keras.optimizers.SGD(1e-3)  
for i in range(10000):  
 with tf.GradientTape() as tape:  
 ***#调用封装的模型类实例*** y\_pred = model(x)   
 loss = tf.reduce\_mean(tf.square(y\_pred - y))  
 grads = tape.gradient(loss, model.variables) ***#调用model.variables直接得到模型中的所有变量*** opt.apply\_gradients(zip(grads, model.variables))  
print(model.variables)

（二）代码分析

1. 封装模型类的\_\_init()\_\_函数

定义一个全连接层，全连接层的参数为：

Units:输出张量的维数

Activation:激活函数，一般有tf.nn.relu, tf.nn.tanh, tf.nn.sigmoid

Kernel\_initializer, bias\_initializer：权重矩阵kernel和偏置向量bias两个变量的初始化器。

2. 模型类的call(input)函数

使用\_\_init\_\_函数中定义的层，对输入数据进行计算，得到输出。

3. model.variables

Model是我们的模型类的实例，model.variables可以得出该模型类中的所有变量。