1、前言

上一节学习了 Keras 的初步用法，这次将使用 Keras 构建神经网络层结构，进而实现线性回归模型、分类模型等。没有隐藏层的模型称之为简单模型，含有隐藏层的模型称之为神经网络，因为 keras 的高度抽象 API，因此使用 Sequential 顺序模型就可以构建出各种各样的模型，因此，直接从神经网络的模型搭建开始。

A、线性回归

```
import numpy as np#生成训练集、测试集
from keras.layers import Dense
from keras import Sequential
import matplotlib.pyplot as plt

X = np.linspace(-1, 1, 200)
np.random.shuffle(X)
Y = 0.5 * X + 2 + np.random.normal(0, 0.05, (200,))
# plt.scatter(X, Y)
# plt.show()
X_train, Y_train = X[:160], Y[:160]#训练集
X_test, Y_test = X[160:], Y[160:]#测试集

model = Sequential(
    [
        Dense(1, input_dim=1),
    ]
)#搭建神经网络层结构，只有输入层和输出层，特征单元 1，输出单元 1

model.compile(loss='mse', optimizer='sgd')#损失函数：mse 二次函数，优化器：sgd 顺序梯度下降算法

print("Training")
for step in range(301):
    cost = model.train_on_batch(X_train, Y_train)
    if step % 100 == 0:
        print('cost:', cost)

cost = model.evaluate(X_test, Y_test, batch_size=40)
print("Test")
print('cost:', cost)
# W, b = model.layers[0].get_weights()
# print("weight:", W, "bias", b)
print(model.layers[0].get_weights())
```

```
Y_pred = model.predict(X_test)#预测
plt.scatter(X_test, Y_test)#测试机散点图
plt.plot(X_test, Y_pred)#测试预测集曲线
plt.show()
```

B、logistic 回归多分类神经网络

```
from keras.datasets import mnist
from keras.utils import np_utils
from keras.layers import Dense, Activation
from keras.optimizers import RMSprop
from keras import Sequential
import numpy as np

(X_train, y_train), (X_test, y_test) =
mnist.load_data()
X_train = X_train.reshape(X_train.shape[0], -1) / 255
X_test = X_test.reshape(X_test.shape[0], -1) / 255
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
#下载训练数据及测试数据
model = Sequential(
    [
        Dense(32, input_dim=784),
        Activation('relu'),
        Dense(10),
        Activation('softmax')
    ]
)
#搭建神经网络层结构
rmsprop = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08,
decay=0.0)
model.compile(
    optimizer=rmsprop,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)#使用 RMSprop 优化器，分类损失函数，正确率检验

print("Training")
model.fit(X_train, y_train, epochs=2, batch_size=32)

print("Testing")
```

```
loss, accuracy = model.evaluate(X_test, y_test)

print("loss:", loss)
print("accurracy", accuracy)
```