# 使用Keras解决线性回归问题

## 1.使用Keras的一般步骤

1) 准备数据

2）搭建网络模型

3）编译模型

4）训练

5）测试

## 2.代码

|  |
| --- |
| *#coding=utf-8* **import** numpy **as** np **from** keras.models **import** Sequential **from** keras.layers **import** Dense **import** matplotlib.pyplot **as** plt  X = np.linspace(-1, 1, 200) np.random.seed(100) np.random.shuffle(X) Y = 0.5 \* X + 2 + np.random.normal(0, 0.05, (200,)) *#mean,stdev,size* plt.scatter(X, Y) plt.show()  *#准备数据* X\_train, Y\_train = X[:160], Y[:160] X\_test, Y\_test = X[160:], Y[160:]  *#搭建模型* model = Sequential() model.add(Dense(output\_dim=1, input\_dim=1))  *#编译* model.compile(loss=**'mse'**, optimizer=**'sgd'**)  *#训练* **for** step **in** range(500):  cost = model.train\_on\_batch(X\_train, Y\_train)  **if** step % 100 == 0:  print(**"train cost:"**, cost)  *#测试* cost = model.evaluate(x = X\_test, y = Y\_test, batch\_size = 40) print(**"test cost:"**, cost)  *#输出训练权重* W, b = model.layers[0].get\_weights() print(**"W = "**, W, **", b = "**, b) *#绘制结果* Y\_pred = model.predict(X\_test) plt.scatter(X\_test, Y\_test) plt.plot(X\_test, Y\_pred) plt.show() |

输出：



