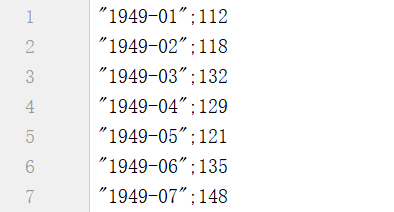
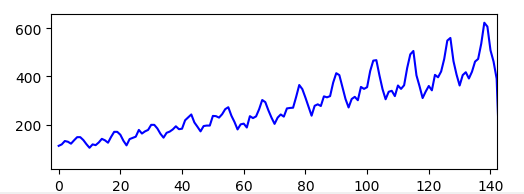
# 使用LSTM做1d序列预测

## 数据准备

本实验使用的数据集是国际航空月度乘客数数据集，这里我将数据放到一个data.txt文件中。





归一化， 使用MinMaxScaler函数：

|  |
| --- |
| scaler = MinMaxScaler(feature\_range=(0, 1)) data\_set = scaler.fit\_transform(data\_set) |

我们使用前两个乘客数来预测下一时刻的乘客数（look\_back = 2），例如，我们使用[112, 118]来预测下一时刻的乘客数，标签为132，我们需要将上面的数据生成我们的训练数据，代码如下：

|  |
| --- |
| **def** create\_dataset(dataset, look\_back):  dataX, dataY = [], []  **for** i **in** range(len(dataset) - look\_back - 1):  a = dataset[i:(i+look\_back),0]  b = dataset[i+look\_back,0]  dataX.append(a)  dataY.append(b)  **return** np.array(dataX), np.array(dataY) |

生成的dataX的shape为(sample\_num, feature\_num)，dataY的shape为(sample\_num,)，但是我们后面使用的LSTM需要输入的数据shape为(sample\_num, time\_step, feature\_num)，所以这里我们还要做一个reshape，代码如下：

|  |
| --- |
| trainX, trainY = create\_dataset(train\_data, look\_back) trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1])) testX, testY = create\_dataset(test\_data, look\_back) testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1])) |

## 搭建并训练网络

|  |
| --- |
| **def** trainNet(trainX, trainY, epochs, batch\_size):  *#create network* model = Sequential()  model.add(LSTM(4, input\_shape=(1, look\_back)))  model.add(Dense(1))  *#train network* model.compile(loss=**'mean\_squared\_error'**, optimizer=**'adam'**)  model.fit(trainX, trainY, epochs=epochs, batch\_size=batch\_size, verbose=2)  model.save(**'rnn.h5'**)  **return** model |

## 

## 预测

|  |
| --- |
| *#predict* trainPredict = model.predict(trainX) testPredict = model.predict(testX) |

我们把数据再转化到原数据的尺度：

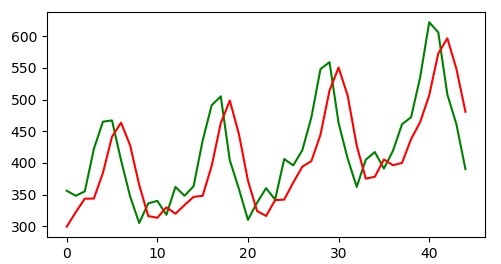
|  |
| --- |
| *#inverse transform* trainPredict = scaler.inverse\_transform(trainPredict) trainY = scaler.inverse\_transform([trainY]) testPredict = scaler.inverse\_transform(testPredict) testY = scaler.inverse\_transform([testY]) |

误差评价：

|  |
| --- |
| *#mean squared error* trainScore = sqrt(mean\_squared\_error(trainY[0], trainPredict[:, 0])) print(**'Train Score: %0.2f RMSE'** % (trainScore)) testScore = sqrt(mean\_squared\_error(testY[0], testPredict[:, 0])) print(**'Test Score: %0.2f RMSE'** % (testScore)) |

我们可视化测试结果：

|  |
| --- |
| *#draw Result* plt.plot(testY[0], color=**'green'**) plt.plot(testPredict, color=**'red'**) plt.show() |



## 4.代码

|  |
| --- |
| **import** numpy **as** np **import** matplotlib.pyplot **as** plt **from** keras **import** Sequential **from** keras.layers **import** Dense **from** keras.layers **import** LSTM **from** sklearn.preprocessing **import** MinMaxScaler **from** keras.models **import** load\_model **from** math **import** \* **from** sklearn.metrics **import** mean\_squared\_error  TRAIN = 1  **def** LoadData(filename):  data = []  **try**:  **with** open(filename) **as** f:  line = f.readline()  **while** line:  line = line[:-1]  x = line.split(**';'**)[1]  data.append(float(x))  line = f.readline()  **except** FileNotFoundError:  print(**'file not found, please check!'**)  **return** np.array(data).astype(**'float32'**)   **def** create\_dataset(dataset, look\_back):  dataX, dataY = [], []  **for** i **in** range(len(dataset) - look\_back - 1):  a = dataset[i:(i+look\_back),0]  b = dataset[i+look\_back,0]  dataX.append(a)  dataY.append(b)  **return** np.array(dataX), np.array(dataY)  **def** trainNet(trainX, trainY, epochs, batch\_size):  *#create network* model = Sequential()  model.add(LSTM(4, input\_shape=(1, look\_back)))  model.add(Dense(1))  *#train network* model.compile(loss=**'mean\_squared\_error'**, optimizer=**'adam'**)  model.fit(trainX, trainY, epochs=epochs, batch\_size=batch\_size, verbose=2)  model.save(**'rnn.h5'**)  **return** model  **def** loadNet():  model = load\_model(**'rnn.h5'**)  **return** model  *#build train data* data\_set = LoadData(**'data.txt'**) data\_set = data\_set.reshape(-1, 1) scaler = MinMaxScaler(feature\_range=(0, 1)) data\_set = scaler.fit\_transform(data\_set) train\_size = int(len(data\_set) \* 0.67) test\_size = len(data\_set) - train\_size train\_data, test\_data = data\_set[0:train\_size, :], data\_set[train\_size:, :] look\_back = 2 batch\_size = 5 trainX, trainY = create\_dataset(train\_data, look\_back) trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1])) testX, testY = create\_dataset(test\_data, look\_back) testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))   **if** TRAIN:  model = trainNet(trainX, trainY, 100, batch\_size) **else**:  model = loadNet()  *#predict* trainPredict = model.predict(trainX) testPredict = model.predict(testX)  *#inverse transform* trainPredict = scaler.inverse\_transform(trainPredict) trainY = scaler.inverse\_transform([trainY]) testPredict = scaler.inverse\_transform(testPredict) testY = scaler.inverse\_transform([testY])  *#mean squared error* trainScore = sqrt(mean\_squared\_error(trainY[0], trainPredict[:, 0])) print(**'Train Score: %0.2f RMSE'** % (trainScore)) testScore = sqrt(mean\_squared\_error(testY[0], testPredict[:, 0])) print(**'Test Score: %0.2f RMSE'** % (testScore))  *#draw Result* plt.plot(testY[0], color=**'green'**) plt.plot(testPredict, color=**'red'**) plt.show() |