## 使用LSTM进行股票价格预测实验

|  |  |
| --- | --- |
| 【实验名称】 | 使用LSTM进行股票价格预测 |
| 【知识点准备】 | |
| Python基础，keras，科学计算包基础（pandas，numpy） | |
| 【实验目的】 | |
| 1. 强化对循环神经网络的理解  2. 学习使用LSTM解决实际问题 | |
| 【实验内容】 | |
| 仅使用大盘指数单一特征进行股票价格预测  使用大盘指数及500支个股指数共同预测大盘指数 | |
| 【实验要求】 | |
| MSE Test小于0.1 | |
| 【实验步骤】 | |
| 1. 单特征回归   1.1 导入必要模块并定义超参数  from keras.layers import Input, Dense, LSTM  from keras.models import Model import pandas as pd  import numpy as np import matplotlib.pyplot as plt %matplotlib inline from sklearn.preprocessing import MinMaxScaler import time  output\_dim = 1 batch\_size = 256 epochs = 10 seq\_len = 5 hidden\_size = 128   * 1. 加载数据集   #panda加载csv文件  data = pd.read\_csv('data\_stocks.csv')  #删除时间戳列  data.drop('DATE',axis=1,inplace=True)  #将所有数据分成训练集和测试集  data\_train = data.iloc[:int(data.shape[0]\*0.8),:] data\_test = data.iloc[int(data.shape[0]\*0.8):,:] print(data\_train.shape,data\_test.shape)  #进行归一化处理  scaler = MinMaxScaler(feature\_range=(-1,1)) scaler.fit(data\_train) data\_train = scaler.transform(data\_train) data\_test = scaler.transform(data\_test)  X\_train = np.array([data\_train[i : i + seq\_len, 0] for i in  range(data\_train.shape[0] - seq\_len)])[:, :, np.newaxis] y\_train = np.array([data\_train[i + seq\_len, 0] for i in range(data\_train.shape[0] - seq\_len)]) X\_test = np.array([data\_test[i : i + seq\_len, 0] for i in range(data\_test.shape[0] - seq\_len)])[:, :, np.newaxis] y\_test = np.array([data\_test[i + seq\_len, 0] for i in range(data\_test.shape[0] - seq\_len)])  print(X\_train.shape, y\_train.shape, X\_test.shape, y\_test.shape)   * 1. 定义网络   X = Input(shape=[X\_train.shape[1], X\_train.shape[2],]) h = LSTM(hidden\_size, activation='relu')(X) Y = Dense(output\_dim, activation='sigmoid')(h)  model = Model(X, Y) model.compile(loss='mean\_squared\_error', optimizer='adam')  1.4 训练网络并查看网络结果  model.fit(X\_train, y\_train, epochs=epochs, batch\_size=batch\_size,  shuffle=False) y\_pred = model.predict(X\_test) print('MSE Train:', model.evaluate(X\_train, y\_train, batch\_size=batch\_size)) print('MSE Test:', model.evaluate(X\_test, y\_test, batch\_size=batch\_size)) plt.plot(y\_test, label='test') plt.plot(y\_pred, label='pred') plt.legend() plt.show()    2 .多特征回归  2.1导入必要模块并定义超参数  #使用多种特征进行预测  from keras.layers import Input, Dense, LSTM  from keras.models import Model import pandas as pd  import numpy as np import matplotlib.pyplot as plt %matplotlib inline from sklearn.preprocessing import MinMaxScaler import time  output\_dim = 1 batch\_size = 256 epochs = 10 seq\_len = 5 hidden\_size = 128  2.2加载数据集  #panda加载csv文件  data = pd.read\_csv('data\_stocks.csv')  #删除时间戳列  data.drop('DATE',axis=1,inplace=True)  #将所有数据分成训练集和测试集  data\_train = data.iloc[:int(data.shape[0]\*0.8),:] data\_test = data.iloc[int(data.shape[0]\*0.8):,:] print(data\_train.shape,data\_test.shape)  #进行归一化处理  scaler = MinMaxScaler(feature\_range=(-1,1)) scaler.fit(data\_train) data\_train = scaler.transform(data\_train) data\_test = scaler.transform(data\_test)  X\_train = np.array([data\_train[i:i + seq\_len,:] for i in range(data\_train.shape[0]-seq\_len)]) y\_train = np.array([data\_train[i + seq\_len,0] for i in range(data\_train.shape[0]-seq\_len)])  X\_test = np.array([data\_test[i:i + seq\_len,:] for i in range(data\_test.shape[0]-seq\_len)]) y\_test = np.array([data\_test[i + seq\_len,0] for i in range(data\_test.shape[0]-seq\_len)])  print(X\_train.shape,y\_train.shape,X\_test.shape,y\_test.shape)  2.3定义网络  X = Input(shape=[X\_train.shape[1],X\_train.shape[2],]) h = LSTM(hidden\_size,activation='relu')(X) Y = Dense(output\_dim,activation='sigmoid')(h)  model = Model(X,Y) model.compile(loss='mean\_squared\_error',optimizer='adam')  2.4训练网络并查看网络结果  model.fit(X\_train,y\_train,epochs=  epochs,batch\_size=batch\_size,shuffle=False) y\_pred = model.predict(X\_test)  print('训练集上的损失',model.evaluate(X\_train,y\_train,batch\_size=batch\_size)) print('测试集上的损失',model.evaluate(X\_test,y\_test,batch\_size=batch\_size)) plt.plot(y\_test,label='test') plt.plot(y\_pred,label='pred') plt.legend() plt.show() | |