9.29 9.30

CNN+LSTM神经网络学习

首先来看一下代码：

import pandas as pd

import time, datetime

df\_data\_5minute= get\_price('AG9999.XSGE', start\_date='2016-01-01', end\_date='2018-01-01', frequency='5m')

#df\_data\_5minute=pd.read\_csv('黄金主力5分钟数据.csv')

'''

或者使用JQdata

from jqdatasdk import \*

#jqdata的账号密码

auth('邮箱：', 'jiaohiabin@ruc.edu.cn')

df\_data\_5minute= get\_price('AU9999.XSGE', start\_date='2016-01-01', end\_date='2018-01-01', frequency='5m')

'''

首先从数据库获取一下过往交易的数据，从2016年到2018年进行拟合

我们看一下get\_price 函数

get\_price(security, start\_date=None, end\_date=None, frequency='daily', fields=None, skip\_paused=False, fq='pre', count=None)

security：一只股票的代码或一组股票代码的 list

start\_date: 开始时间，与参数 count 二选一，不可同时使用。如果两者都没有设置，则start\_date生效，默认时间为’2015-01-01’。如果取分钟数据，时间可以精确到分钟

end\_date: 结束时间，默认是’2015-12-31’，包含此日期。当取分钟数据时，如果只有日期，则日内时间等同于’00:00:00’，所以返回的数据不包括 end\_date 这一天

frequency: 单位时间长度，几天或者几分钟，默认是 daily ，即表示1天。现在支持

* Xd 几天
* Xm 几分钟
* daily 1天
* minute 1分钟

需要注意的是，当X>1时，fields 只支持 [‘open’, ‘close’, ‘high’, ‘low’, ‘volume’, ‘money’] 这几个标准字段

fields: 字符串 list，选择要获取的行情数据字段，默认是 None（表示 [‘open’, ‘close’, ‘high’, ‘low’, ‘volume’, ‘money’] 这几个标准字段）。参数 felids 支持 SecurityUnitData 里面的所有基本属性，包含：[‘open’, ‘close’, ‘high’, ‘low’, ‘volume’, ‘money’, ‘factor’, ‘high\_limit’, ‘low\_limit’, ‘avg’, ‘pre\_close’, ‘paused’]

sikp\_paused:是否跳过不易日期（包括停牌、未上市或者退市后数据者为 nan。需要注意的是，该参数默认为 False ，即不跳过不交易日期。如果当该参数是 True 时，只能选取一只股票的信息

fq: 复权选项。参数值设为 ‘pre’，表示前复权，为默认设置；参数值为 None，表示不复权，返回实际价格；参数值设置为 ‘post’，表示后复权

count:与 start\_date 二选一，不可同时使用。参数 count 表示数量，返回结果集的行数，表示获取 end\_date 之前几个 frequency 的数据

df\_data\_5minute.head()

#head函数只读取数据的前5行

df\_data\_5minute

df\_data\_5minute.drop('Unnamed: 0', axis=1, inplace=True)

df\_data\_5minute

（2）drop函数的使用：inplace参数

采用drop方法，有下面三种等价的表达式：

1. DF= DF.drop('column\_name', axis=1)；

2. DF.drop('column\_name',axis=1, inplace=True)

3. DF.drop([DF.columns[[0,1, 3]]], axis=1, inplace=True) # Note: zero indexed

目的去掉这个数据库中的所有名称是0的列

df=df\_data\_5minute

close = df['close']

df.drop(labels=['close'], axis=1,inplace = True)

df.insert(0, 'close', close)

Df

#目的是为了吧close的价格放在表格的最前面

data\_train =df.iloc[:int(df.shape[0] \* 0.7), :]

data\_test = df.iloc[int(df.shape[0] \* 0.7):, :]

print(data\_train.shape, data\_test.shape)

#然后把数据集分为训练集和测试集7:3

#并打印出来这两个数据集的维度

(37317, 6) (15993, 6)

#准备训练模型，初始化模型

# -\*- coding: utf-8 -\*-

import pandas as pd

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.preprocessing import MinMaxScaler

import time

scaler = MinMaxScaler(feature\_range=(-1, 1))

scaler.fit(data\_train)

#这一段是在做包的标准化

#[https://](https://www.jianshu.com/p/2a635d9e894d)[www.jianshu.com/p/2a635d9e894d](https://www.jianshu.com/p/2a635d9e894d)

#对所有的列数据做minimax标准化

from keras.layers import Input, Dense, LSTM

from keras.models import Model

from keras.layers import \*

from keras.models import \*

from keras.optimizers import Adam

#写一下需要的keras的 TensorFlow包

output\_dim = 1 #因为是regression问题

batch\_size = 256 #包的大小

epochs = 60 #迭代次数

seq\_len = 5 #选取的RNN的历史数据卷积的长度

hidden\_size = 128 #隐含层一层的宽度

输出大小

TIME\_STEPS = 5 #步长

INPUT\_DIM = 6 #输入的维度，因为有6行的数据要做LSTM卷积

lstm\_units = 64 #

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)