**上海电力大学**

**大作业**



**题　　目:**  基于Tensorflow的工商银行股票收盘价预测

**姓 名：** 谢扬筱

**学 号：** y21208026

**院　　系：**　　 计算机科学与技术学院

**专业年级：**　 电子信息（计算机技术）2021级

**2021年　 12 月　 26 日**

1. **整体设计目标**
2. 本次大作业的整体目标是爬取工商银行股票数据，保存为excel文件。
3. 对爬取到的数据进行数据预处理以满足训练数据的要求。使用tensorflow创建简单的循环神经网络对开盘价进行预测。训练完成之后计算预测的准确率及画出相关的图。
4. **代码总体框架**

1、爬虫

从网址为“http://quotes.money.163.com/trade/lsjysj\_601398.html”上爬取工商银行股票自上市以来的每天的交易数据。并把爬取的数据保存为excel文件格式。

1. 预测
2. 得到爬取的数据，并把数据处理。提取的数据中的收盘价作为预测使用的数据集。
3. 把收盘价数据分为训练集和测试集。
4. 对训练的数据进行归一化。
5. 使用tensorflow进行循环神经网络搭建及配置网络参数。
6. 训练模型，得到训练结果。
7. 使用神经网络模型对测试数据进行预测。
8. 进行预测准确率的计算。
9. 使用matplotlin画出损失函数和预测曲线图。
10. 计算评价指标的计算。
11. **第三方库介绍**
12. urllib库

urllib库是python的内置库，无需安装，可直接使用，是http请求库。在爬虫环节使用了urllib.request 请求模块。

1. Beautiful Soup库

Beautiful Soup 库一般称为bs4。使用之前需要进行安装，安装命令为“pip install bs4”。这是一个是一个可以从HTML或XML文件中提取数据的Python库。在爬虫中使用bs4提取网页中的数据。

1. xlwt库

xlwt库是python操作excel的常用库。使用之前需要进行安装，安装命令为“pip install xlwt”。在爬虫中使用xlwt把爬取的数据保存为excel格式。

1. math库

math库是python内置的数学相关计算的库。

1. numpy库

numpy是python用来进行矩阵运算、高维数组运算的数学计算库。使用之前需要进行安装，安装命令为“pip install numpy”。在导入numpy库一般用别名np代替numpy。所以导入numpy代码为“import numpy as np”。

1. pandas库

Pandas 是一个开放的库，提供高性能、易于使用的数据结构和数据分析工具。pandas 名字衍生自术语 "panel data"（面板数据）和 "Python data analysis"（Python 数据分析）。pandas 一个强大的分析结构化数据的工具集，基础是[Numpy](https://www.runoob.com/numpy/numpy-tutorial.html" \t "https://www.runoob.com/pandas/_blank)（提供高性能的矩阵运算）。Pandas 可以从各种文件格式比如 CSV、JSON、SQL、Microsoft Excel 导入数据。Pandas 可以对各种数据进行运算操作，比如归并、再成形、选择，还有数据清洗和数据加工特征。Pandas 广泛应用在学术、金融、统计学等各个数据分析领域。使用之前需要进行安装，安装命令为“pip install pandas”。

1. matplotlib库

matplotlib是Python绘图领域使用最广泛的套件。它能让使用者很轻松地将数据图形化，并且提供多样化的输出格式。使用之前需要进行安装，安装命令为“pip install matplotlib”。

1. tensorflow

TensorFlow是Google于2015年发布的深度学习框架，最初版本只支持符号式编程。得益于发布时间较早，以及 Google 在深度学习领域的影响力，TensorFlow 很快成为最流行的深度学习框架。但是由于TensorFlow接口设计频繁变动，功能设计重复冗余，符号式编程开发和调试非常困难等问题，TensorFlow 1.x 版本一度被业界诟病。2019年，Google推出TensorFlow 2 正式版本，将以动态图优先模式运行，从而能够避免TensorFlow 1.x版本的诸多缺陷，已获得业界的广泛认可。使用之前需要进行安装，安装命令为“pip install tensorflow”。在预测分析中使用tensorflow搭建神经网络模型。本作业中安装的是最新版tensorflow2.7。使用时一般把tf作为tensorflow的别名。

1. Sklearn

sklearn是一个Python第三方提供的非常强力的机器学习库，它包含了从数据预处理到训练模型的各个方面。在实战使用scikit-learn中可以极大的节省我们编写代码的时间以及减少我们的代码量，使我们有更多的精力去分析数据分布，调整模型和修改超参。

1. **关键代码说明**

1、爬虫设置浏览器代理，将爬虫程序模拟伪装成浏览器进行网页访问。防止网站设置反爬虫技术，禁止爬虫程序直接访问网页。

head = {

# 用户代理信息告诉网页是用浏览器访问

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36"

}

2、爬取数据

dataTable = bs.find("table",{"class":"table\_bg001border\_box limit\_sale"})

# 得到每一行的数据,data是每一个季度的数据

data = dataTable.find\_all("tr")

在上面的代码中使用Beautiful Soup在网页源码中找到需要爬取的数据，进行数据爬取。

3、将爬取的数据保存为excel

while True:

# 按行循环，读取文本文件

line = f.readline()

if not line:

break # 如果没有内容，则退出循环

for i in range(len(line.split("\t"))-1):

# print(line.split("\t"))

item = line.split("\t")[i]

if(i==0 or i==7 or i==8):

sheet.write(x, i, item)

else:

item=float(item)

sheet.write(x,i,item)

x += 1 # excel另起一行

f.close()

workbook.save("data\\bank.xlsx") # 保存xlsx文件

在上面的代码中使用for循环一个一个的把数据写入excel指定的单元格中保存。

4、使用pandas对爬取的数据进行处理，使之满足需要的数据要求和格式。

bank.sort\_values(by="日期",inplace=True)

bank.to\_csv("data//bank.csv")

data=pd.read\_csv("data//bank.csv")

data

以上对数据的排序进行改变，满足按日期的升序排列方式，更好的满足预测任务的要求。

5、划分训练集和测试集

# 前3000天的收盘价作为训练集

training\_set = data.iloc[0:3000, 5:6].values

# 后面天数的收盘价作为测试集

test\_set = data.iloc[3000:, 5:6].values

上面的代码把前3000天的工商银行收盘价作为训练集，后面的天数的收盘价数据作为测试集。

6、对训练集和测试集进行归一化

#归一化到(0，1)之间

sc = MinMaxScaler(feature\_range=(0, 1))

# 求得训练集的最大值，最小值这些训练集固有的属性，并在训练集上进行归一化

training\_set = sc.fit\_transform(training\_set)

# 利用训练集的属性对测试集进行归一化

test\_set = sc.transform(test\_set)

上面的代码作用是对训练集和测试集进行归一化的操作，是所有数据分布在0到1之间的数。

7、使用tensorflow创建神经网络模型

model = tf.keras.Sequential([

    SimpleRNN(60, return\_sequences=True),

    Dropout(0.2),

    SimpleRNN(60),

    Dropout(0.2),

    Dense(1)

])

上面的代码中使用tensorflow创建了循环神经网络模型，有两层SimpleRNN和一层全连接层。

8、对模型配置损失函数和优化器

model.compile(optimizer=tf.keras.optimizers.Adam(0.001),loss="mse")

9、对创建的网络模型进行训练，得到结果

fit\_result=model.fit(x\_train,y\_train,batch\_size=64,epochs=30,validation\_data=(x\_test, y\_test), validation\_freq=1)

10、计算预测准确率

# 测试集输入模型进行预测

pred = model.predict(x\_test)

# 反归一化

pred = sc.inverse\_transform(pred)

real = sc.inverse\_transform(test\_set[60:])

correct, number = 0, len(pred)

#print(len(real))

for i in range(number):

    if(tf.equal(pred[i],real[i])):

        correct = correct+1

print("预测准确率为：%.3f"%(correct/number))

11、计算指标评估模型效果

#均方误差

mse = mean\_squared\_error(pred, real)

#均方根误差

rmse = math.sqrt(mse)

#平均绝对误差

mae = mean\_absolute\_error(pred, real)

print('均方误差: %.6f' % mse)

print('均方根误差: %.6f' % rmse)

print('平均绝对误差: %.6f' % mae)

在上面的代码中计算了均方误差，均方根误差和平均绝对误差。这些数据可以评估模型的效果。

1. **效果和结论**

在本次的作业中成功实现了对网页数据的爬虫，并对爬取的数据成功进行了保存。对爬取的数据进行处理，再使用tensorflow创建简单的循环神经网络进行预测。预测结果与真实数据对比发现预测准确率为0，就是说明预测的数值与真实值存在一定的误差，预测值不可能与真实值一样。本次作为我成功的对神经网络的实例进行了实践，让我增益良多。但本次作业也还有不足之处，需要继续继续思考改进工作。

**六 、附录：程序设计源代码**

1. **爬虫代码**

本爬虫代码使用pycharm编写

# -\*- coding = utf-8 -\*-

# @Time : 2021/12/9 20:39

# @Author : 谢扬筱

# @File : spider.py

import urllib.request

from bs4 import BeautifulSoup

import xlwt

head = {

# 用户代理信息告诉网页是用浏览器访问

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.93 Safari/537.36"

}

def first():

url = "http://quotes.money.163.com/trade/lsjysj\_601398.html"

request = urllib.request.Request(url, headers=head)

try:

response = urllib.request.urlopen(request)

html = response.read().decode()

# 爬取网页源码

# f=open("data\\网页源码.html","a",encoding="utf-8")

# f.write(html)

#print(html)

bs = BeautifulSoup(html, "lxml")

# 查找历史年份

years\_select = bs.find("select", {"name": "year"})

# print(years\_select)

all\_years = years\_select.find\_all("option")

# print(all\_years)

return all\_years

except urllib.error.URLError as e:

if hasattr(e, "code"):

print(e.code)

if hasattr(e, "reason"):

print(e.reason)

def second():

all\_years = first()

txt = ""

for year in all\_years:

year = year.text

print("爬取" + year + "年的数据")

for season in range(4, 0, -1):

url = "http://quotes.money.163.com/trade/lsjysj\_601398.html?year=" +

year + "&season=" + str(season)

request = urllib.request.Request(url, headers=head)

response = urllib.request.urlopen(request)

html = response.read().decode()

bs = BeautifulSoup(html, "lxml")

dataTable = bs.find("table",

{"class":"table\_bg001border\_box limit\_sale"})

# 得到每一行的数据,data是每一个季度的数据

data = dataTable.find\_all("tr")

# print(data)

# print(len(data))

for day in data:

if(len(data) > 1):

day\_data = day.find\_all("td")

# f=open("data\\day\_data.txt","a")

# f.write(str(day\_data))

for i in day\_data:

# 每个格子数据之间空格

txt = txt + i.text + "\t"

if(len(day\_data) > 0):

# 每天的数据分行输出

txt = txt + "\n"

f = open("data\\data.txt", "a")

f.write(txt)

def save():

path = "data\\data.txt"

f = open(path, encoding="utf-8")

workbook = xlwt.Workbook()

# 生成excel的方法，声明excel

sheet = workbook.add\_sheet("历史数据", cell\_overwrite\_ok=True)

col = (

"日期",

"开盘价",

"最高价",

"最低价",

"收盘价",

"涨跌额",

"涨跌幅(%)",

"成交量(手)",

"成交金额(万元)",

"振幅(%)",

"换手率(%)")

# 把列名写入

for i in range(0, 11):

# 设置表格列宽

sheet.col(i).width = 256 \* 12

sheet.write(0, i, col[i])

x = 1

while True:

# 按行循环，读取文本文件

line = f.readline()

if not line:

break # 如果没有内容，则退出循环

for i in range(len(line.split("\t"))-1):

# print(line.split("\t"))

item = line.split("\t")[i]

if(i==0 or i==7 or i==8):

sheet.write(x, i, item)

else:

item=float(item)

sheet.write(x,i,item)

x += 1 # excel另起一行

f.close()

workbook.save("data\\bank.xlsx") # 保存xlsx文件

if \_\_name\_\_ == '\_\_main\_\_':

first()

second()

save()

print("爬取结束!")

1. 预测开盘价代码

预测开盘价代码使用jupyter notebook编写

import math

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from tensorflow.keras.layers import Dropout, Dense, SimpleRNN

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

#设置黑体为默认字体

plt.rcParams["font.sans-serif"] = "SimHei"

#设置正常显示负号

plt.rcParams["axes.unicode\_minus"] = False

# 使用pandas读取文件，处理数据

bank=pd.read\_excel("data//bank.xlsx")

bank

bank.sort\_values(by="日期",inplace=True)

bank.to\_csv("data//bank.csv")

data=pd.read\_csv("data//bank.csv")

data

data.head()

data.describe()

# 前3000天的收盘价作为训练集

training\_set = data.iloc[0:3000, 5:6].values

# 后面天数的收盘价作为测试集

test\_set = data.iloc[3000:, 5:6].values

#归一化到(0，1)之间

sc = MinMaxScaler(feature\_range=(0, 1))

# 求得训练集的最大值，最小值这些训练集固有的属性，并在训练集上进行归一化

training\_set = sc.fit\_transform(training\_set)

# 利用训练集的属性对测试集进行归一化

test\_set = sc.transform(test\_set)

# len(training\_set\_scaled)

# len(test\_set)

#training\_set\_scaled.shape

#training\_set\_scaled

x\_train = []

y\_train = []

x\_test = []

y\_test = []

# 利用for循环，遍历整个训练集，提取训练集中连续60天的收盘价作为输入特征x\_train，

#第61天的数据作为标签。0的是第一天,60是第61天

for i in range(60, len(training\_set)):

    x\_train.append(training\_set[i - 60:i, 0])

y\_train.append(training\_set[i, 0])

# 对训练集进行打乱

np.random.seed(1)

np.random.shuffle(x\_train)

np.random.seed(1)

np.random.shuffle(y\_train)

tf.random.set\_seed(1)

#将训练集由list格式变为array格式

x\_train, y\_train = np.array(x\_train), np.array(y\_train)

x\_train = np.reshape(x\_train, (x\_train.shape[0], 60, 1))

x\_train

#利用for循环，遍历整个测试集，提取测试集中连续60天的开盘价作为输入特征x\_test，

#第61天的数据作为标签，

for i in range(60, len(test\_set)):

    x\_test.append(test\_set[i - 60:i, 0])

    y\_test.append(test\_set[i, 0])

print(len(y\_test))

#测试集变array并reshape为符合RNN输入要求：[送入样本数， 循环核时间展开步数， 每个时间步输入特征个数]

x\_test, y\_test = np.array(x\_test), np.array(y\_test)

x\_test = np.reshape(x\_test, (x\_test.shape[0], 60, 1))

model = tf.keras.Sequential([

    SimpleRNN(60, return\_sequences=True),

    Dropout(0.2),

    SimpleRNN(60),

    Dropout(0.2),

    Dense(1)

])

model.compile(optimizer=tf.keras.optimizers.Adam(0.001),

              loss="mse")

fit\_result=model.fit(x\_train,y\_train,batch\_size=64,epochs=30,validation\_data=(x\_test, y\_test), validation\_freq=1)

loss = fit\_result.history["loss"]

val\_loss = fit\_result.history["val\_loss"]

plt.plot(loss, label="训练损失函数")

plt.plot(val\_loss, label="测试损失函数")

plt.title("训练和测试损失函数曲线图",fontsize=15)

plt.legend()

plt.show()

# 测试集输入模型进行预测

pred = model.predict(x\_test)

# 反归一化

pred = sc.inverse\_transform(pred)

real = sc.inverse\_transform(test\_set[60:])

correct, number = 0, len(pred)

#print(len(real))

for i in range(number):

    if(tf.equal(pred[i],real[i])):

        correct = correct+1

print("预测准确率为：%.3f"%(correct/number))

correct, number = 0, len(pred)

for j in range(number):

    if((real[j]-0.1)<=pred[j] and pred[j]<=(real[j]+0.1)):

        correct = correct+1

print("预测准确率为：%.3f"%(correct/number))

#画出真实数据和预测数据的对比曲线

plt.plot(real, color='red', label='工商银行收盘价价')

plt.plot(pred, color='blue', label='预测的工商银行收盘价')

plt.title('工商银行收盘价预测',fontsize=15)

plt.xlabel('时间',fontsize=10)

plt.ylabel('工商银行收盘价',fontsize=10)

plt.legend()

plt.show()

#均方误差

mse = mean\_squared\_error(pred, real)

#均方根误差

rmse = math.sqrt(mse)

#平均绝对误差

mae = mean\_absolute\_error(pred, real)

print('均方误差: %.6f' % mse)

print('均方根误差: %.6f' % rmse)

print('平均绝对误差: %.6f' % mae)