

组员	论文结构与格式 (25)	论文语言与内容 (25)	课题完成情况 (25)	数据分析与讨论 (25)	总分 (100)
苏日清					
余思贤					
梁宗威					
谭铭濠					

教师评语： _____

基于 GRU 的股市预测

姓名/学号 1、苏日清 2018054439
 2、余思贤 2018054439
 3、梁宗威 2018054439
 4、谭铭濠 2018054439

指导教师：庄师强
电气工程及其自动化

二〇二一年十二月二十六日

1 课程设计的任务与要求

1.1 课程设计的任务

1. 熟悉 MATLAB 中深度学习工具箱的使用方法，喂入数据的方法，配置训练的方法，保存模型的方法和调用模型的方法；
2. 能画出学习模型的基本框架，理解其基本原理；
3. 基于 GRU 对中国石化的开盘股价进行预测。

1.2 课程设计的要求

1. 学会 MATLAB 软件的安装；
2. 熟练掌握 MATLAB 的使用，掌握深度学习工具箱的使用；
3. 能使用深度学习工具箱根据需求搭建神经网络，喂入数据，训练，得出模型并能调用；
4. 通过调参使得模型能更好的贴合实际，打到更好的效果。

2 研究基础

2.1 序列数据神经网络

2.1.1 RNN 网络

循环神经网络 (Recurrent Neural Network, RNN) 是一种用于处理序列数据的神经网络。相比一般的神经网络来说，它能够处理序列变化的数据。

比如某个单词的意思会因为上文提到的内容不同而有不同的含义，RNN 就能够很好地解决这类问题。

一般来说，普通 RNN 网络形式如下：

$$h' = \delta(w_h h + w_i x + b^h) \quad (2.1)$$

$$y = \delta(w_o h' + b^y) \quad (2.2)$$

其中 h' 为传入下一节点的参数， y 常使用对 h' 进行唯独映射，然后使用 *softmax* 进行分类得到所需的参数。

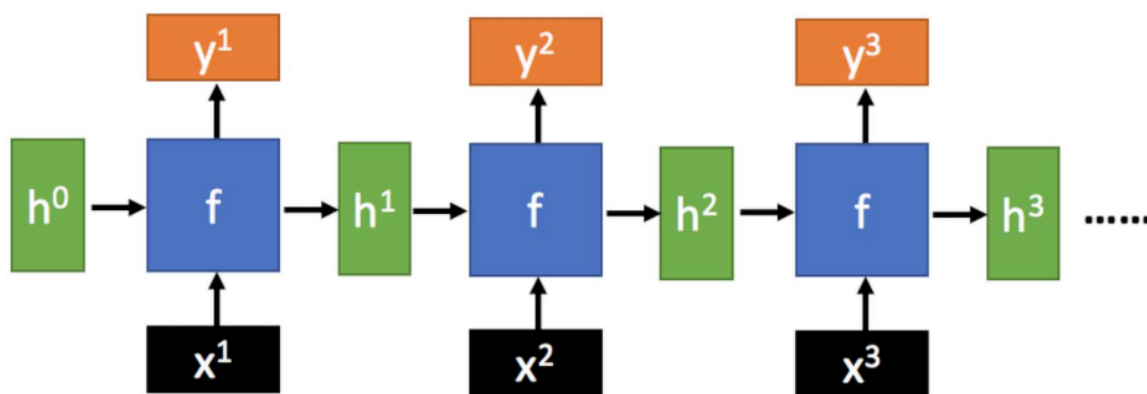


图 2.1: 多级 RNN 网络构成

2.1.2 LSTM 结构

长短期记忆（Long short-term memory, LSTM）是一种特殊的 RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的 RNN，LSTM 能够在更长的序列中有更好的表现。

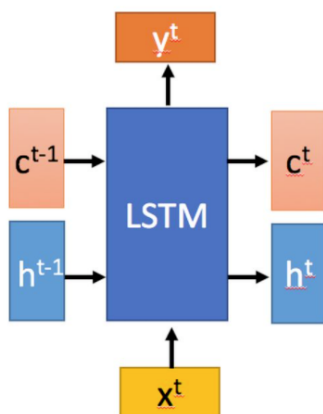


图 2.2: LSTM 网络结构

LSTM 网络的表达式如下：

$$c^t = z^f \odot c^{t-1} + z^i \odot z \quad (2.3)$$

$$h^t = z^o \odot \tanh c^t \quad (2.4)$$

$$y^t = \delta(W'h^t + b^y) \quad (2.5)$$

其中 c^t 可以理解为长期记忆，主要是用来保存节点传递下来的数据的，每次传递会对某些维度进行“忘记”并且会加入当前节点所包含的内容；， h^t 则为短期记忆，仅保存了先前节点的信息。

2.1.3 GRU 网络

GRU (Gate Recurrent Unit) 是循环神经网络 (Recurrent Neural Network, RNN) 的一种。和 LSTM (Long-Short Term Memory) 一样，也是为了解决长期记忆和反向传播中的梯度等问题而提出来的。

GRU 的输入输出结构与普通的 RNN 是一样的。

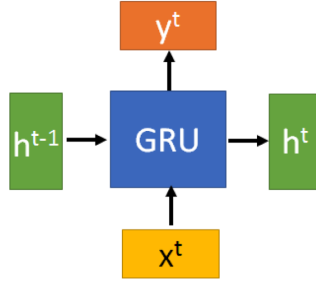


图 2.3: GRU 的输入输出结构

其表达式为：

$$r^t = \delta(x^t w^{xr} + h^{t-1} w^{hr} + b^r) \quad (2.6)$$

$$z^t = \delta(x^t w^{xz} + h^{t-1} w^{hz} + b^z) \quad (2.7)$$

$$h' = \tanh x^t w^{xr} + r^t \odot h^{t-1} w^{hh} + b^h \quad (2.8)$$

$$h^t = (1 - z) \odot h' + z \odot h^{t-1} \quad (2.9)$$

$$y^t = \text{softmax}(h^t w^{hy} + b^y) \quad (2.10)$$

GRU 与 LSTM 相比，需要训练的参数较少，但也能达到与 LSTM 相近的效果。其训练的参数少，对硬件要求要求较低，因此本文采用 GRU 进行实现。

2.2 训练数据的准备

2.2.1 数据采集

这里使用 Python 爬取近 16 年中国石化（600028）的股市信息。
代码如下：

```
1 import tushare as ts
2
3 df1 = ts.get_k_data('600028', ktype='D', start='2005-01-01', end='2021-10-16')
4
5 datapath1 = "./SH600028.csv"
6 df1.to_csv(datapath1)
```

得到了 4032 行数据，包括：时间、开盘价格、收市价、高位、低位、成交量以及股票代码。

2.2.2 数据处理

为使得模型更快收敛，并提高其准确性，对取得的数据进行归一化处理，公式如下：

$$x^* = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.11)$$

核心代码如下：

```
1 from sklearn.preprocessing import MinMaxScaler
2
3 sc = MinMaxScaler(feature_range=(0, 1))
4 training_set_scaled = sc.fit_transform(training_set)
5 test_set = sc.transform(test_set)
```

2.2.3 数据标注

为了实现股票预测，需要对训练用的数据进行标注。

GRU 是根据过去预测未来，因此采用前两个月的数据进行对当日开盘价的预测。

取出后 200 日的数据作为测试数据，前 3832 日数据作为训练数据。

使用 Python 进行数据标注，并保存为 txt 文件，方便在 MATLAB 中进行读取调用。

标注核心代码如下：

```

1 import pandas as pd
2
3 test_num=200
4 day=60
5
6 zhongshihua = pd.read_csv('./SH600028.csv')
7 training_set = zhongshihua.iloc[0:len(zhongshihua) - test_num, 2:3].values
8 test_set = zhongshihua.iloc[len(zhongshihua) - test_num:, 2:3].values
9
10 for i in range(day, len(training_set_scaled)):
11     x_train.append(training_set_scaled[i - day:i, 0])
12     y_train.append(training_set_scaled[i, 0])

```

2.3 股市预测的 Python 实现

先在 Python 中调用 tensorflow 框架进行本项目的实现，验证其可行性。



图 2.4: 股市预测网络结构

核心代码如下（仅展示模型部分）：

```

1 import tensorflow as tf
2 from tensorflow.keras.layers import Dropout, Dense, GRU
3
4 model = tf.keras.Sequential([
5     GRU(512, return_sequences=True),
6     Dropout(0.2),
7     GRU(1024),
8     Dropout(0.2),
9     Dense(1)
10 ])

```

第一层 GRU：512 个单元，每次返回 h^t 参数；令其中 20% 的单元休眠；第二层 GRU：1024 个单元，仅最后一次返回 h^t 参数；令其中 20% 的单元休眠；最后进行全链接输出。

在 $\text{epochs}=100$, $\text{batch size}=32$ 的训练条件下, 预测结果如图2.5:

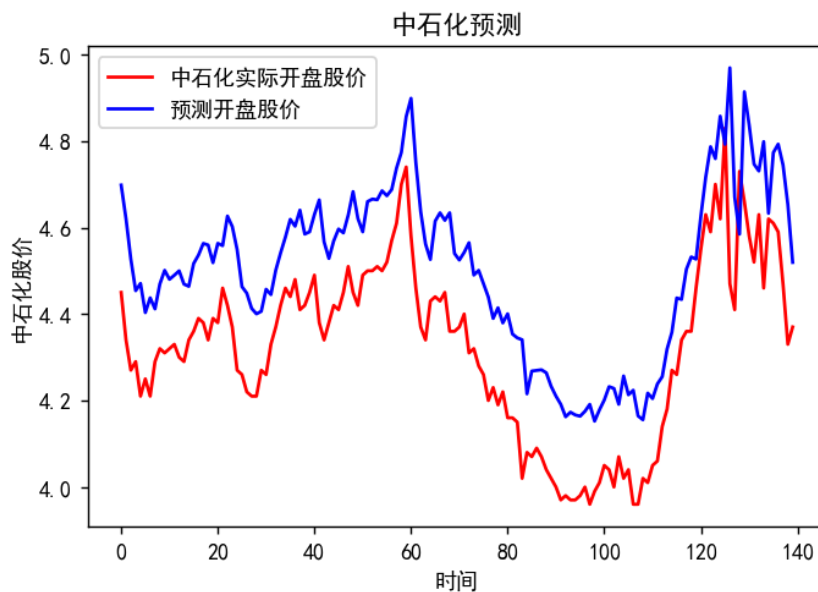


图 2.5: 预测结果

可以看到: 预测出的趋势与实际相比是较为准确的。

3 股市预测的 MATLAB 实现

3.1 深度学习工具箱

深度学习工具箱提供了一个用于通过算法、预训练模型和 App 来设计和实现深度神经网络的框架。在深度学习工具箱中可以使用卷积神经网络 (ConvNet、CNN) 和长短期记忆 (LSTM) 网络对图像、时序和文本数据执行分类和回归。可以使用自动微分、自定义训练循环和共享权重来构建网络架构, 如生成对抗网络 (GAN) 和孪生网络。使用深度网络设计器, 能够以图形方式设计、分析和训练网络。试验管理器可管理多个深度学习试验, 跟踪训练参数, 分析结果, 并比较不同试验的代码。可以可视化层激活, 并以图形方式监控训练进度。^[1]

下面代码架构均使用深度学习工具箱生成。

3.2 在 MATLAB 中搭建 GRU 神经网络

在深度学习工具箱中，可以快速搭建神经网络

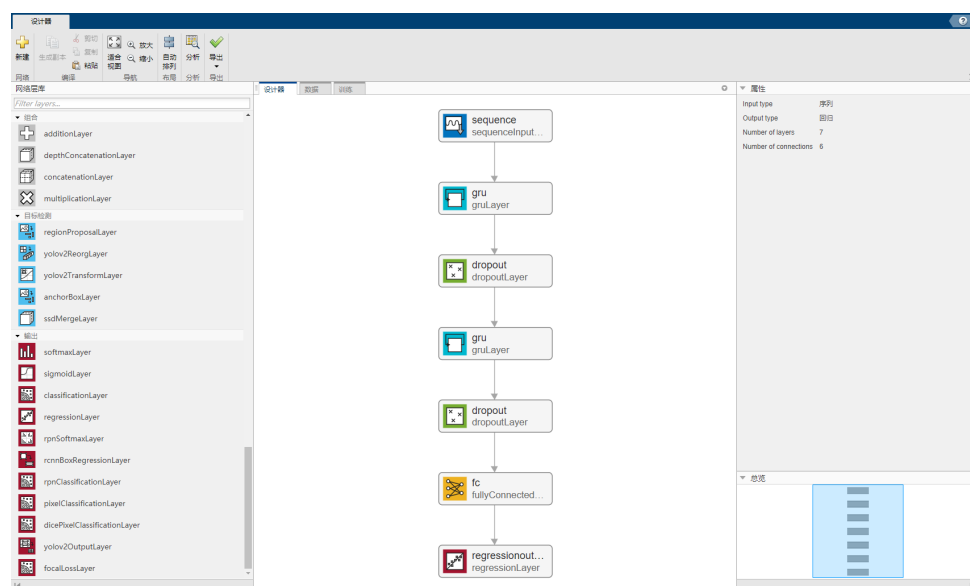


图 3.1: 使用深度学习工具箱中快速搭建基础神经网络

通过深度学习工具箱，我们可以快速导出神经网络模型，得到结果如图3.2所示：



图 3.2: 使用深度学习工具箱创建深度学习网络架构

在 Matlab 中，程序核心代码如下：

1. 网络架构部分：

```
1 function layers=get_gru_net(wd)
2 numFeatures=wd;
```



```

3 numResponses=1;
4 numHiddenUnits=512;
5
6 layers=[sequenceInputLayer(numFeatures)
7         gruLayer(numHiddenUnits)
8         dropoutLayer(0.2)
9         gruLayer(2*numHiddenUnits)
10        dropoutLayer(0.2)
11        fullyConnectedLayer(numResponses)
12        regressionLayer];
13 end
    
```

2. 模型超参数选项设置部分：

```

1 layers=get_gru_net(wd);
2 options=trainingOptions('adam',...
3     'MaxEpochs',100,...
4     'GradientThreshold',1,...
5     'InitialLearnRate',0.001, ...
6     'LearnRateSchedule','piecewise',...
7     'LearnRateDropPeriod',125,...
8     'LearnRateDropFactor',0.2,...
9     'Verbose',0,...
10    'Plots','training-progress');
    
```

3.3 喂入数据并训练

在 Matlab 中，使用函数 *trainNetwork* 将数据喂入神经网络并训练。训练过程如图3.3所示：

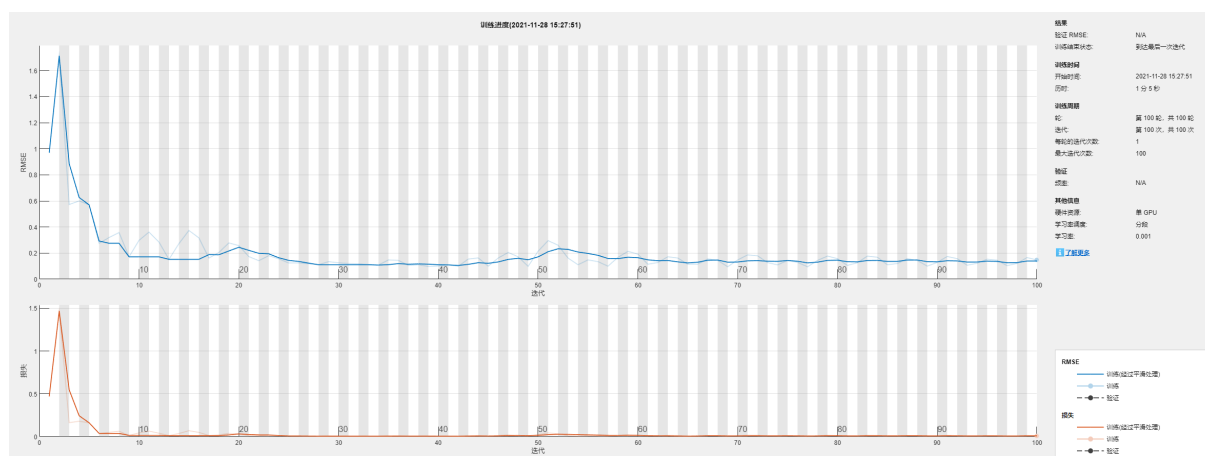


图 3.3: 训练进度展示

使用函数 *predict* 调用训练好的模型进行预测。此神经网络预测结果如图3.4所示：

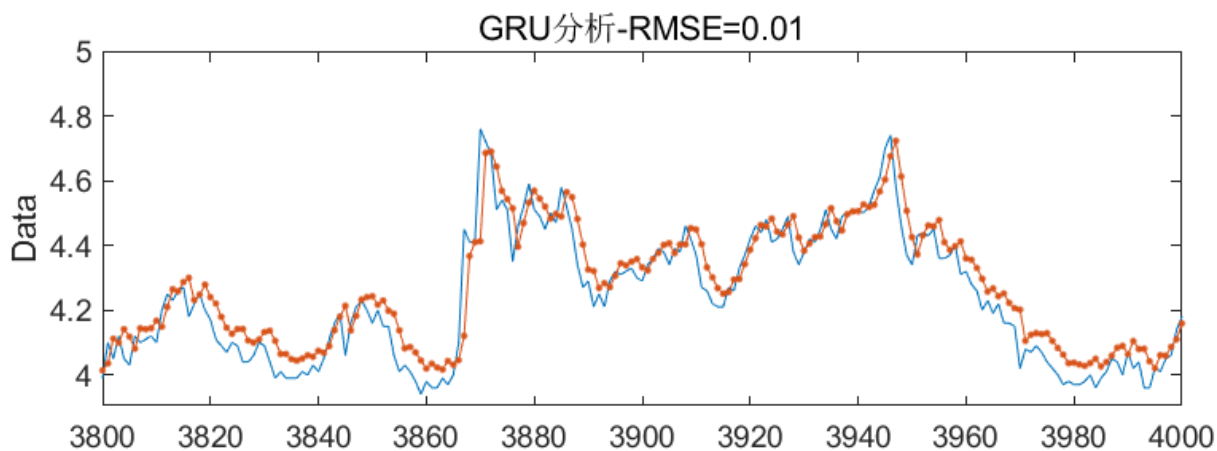


图 3.4: GRU 预测结果：图中橘红色曲线为预测数值，蓝色为实际数值

3.4 模型优化

3.4.1 效果评判

后续测试模型中，我们采用茅台 2005 年 1 月 1 日到 2021 年 3 月 20 日的开盘股价作为训练集，为了节省时间，规定每一种模型均跑 100Epoch。

我们使用如下指标进行模型效果评判：

1. 计算茅台在 2021 年 3 月 30 日到 2021 年 10 月 16 日，共计 200 日的开盘股价预测数据，并与此 200 日的实际开盘股价计算确定系数。

确定系数计算公式如下：

$$r(X, Y) = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}[X]\text{Var}[Y]}} \quad (3.1)$$

式中， $\text{Cov}(X, Y)$ 为 X 与 Y 的协方差， $\text{Var}[X]$ 为 X 的方差， $\text{Var}[Y]$ 为 Y 的方差。

2. 计算模型计算速度，测试数据为：茅台在 2021 年 3 月 30 日到 2021 年 10 月 16 日的股价预测数据。计算速度测试中，时间计算单位为 ms，一共跑 100 次，取平均值。
3. 测试设备为联想拯救者 R9000P（5800H+64G 内存 +RTX3070-8G）

4. 综合评判公式：

$$s = \frac{100r}{T} \quad (3.2)$$

3.4.2 更改模型

在 3.3 中，我们得到了粗略的结果，如图2.5，其模型如图2.4。从结果中可以看出，虽然预测所得的结果趋势大致与实际情况近似，但其偏差还是较大。因此，在这一节中，我们更改了模型。其中更改的方向如下：

1. 增减模型层数
2. 调整 Dropout 单元的数量和 Dropout 的数值
3. 调整 GRU 内神经元个数
4. 在 GRU 中使用 softsign 替换 tanh

我们据此作出了 10 个新的模型，如表3.1所示。

表 3.1: 10 种调参模型

序号	模型结构	备注
1	GRU(20)→Dropout(0.2)→GRU(20)→Dropout(0.2)→Dense	
2	GRU(20)→Dropout(0.2)→GRU(20)→GRU(20)→Dropout(0.2)→Dense	
3	GRU(40)→Dropout(0.2)→GRU(80)→Dropout(0.2)→Dense	
4	GRU(20)→Dropout(0.5)→GRU(20)→Dropout(0.5)→Dense	
5	GRU(80)→Dropout(0.2)→GRU(160)→Dropout(0.2)→Dense	
6	GRU(20)→GRU(20)→Dropout(0.2)→Dense	
7	GRU(20)→Dropout(0.2)→GRU(20)→Dropout(0.2)→Dense	输出使用 softsign
8	GRU(20)→Dropout(0.7)→GRU(20)→Dropout(0.7)→Dense	
9	GRU(20)→Dropout(0.2)→GRU(20)→Dropout(0.2)→GRU(20)→Dropout(0.2)→Dense	
10	GRU(20)→Dropout(0.2)→GRU(20)→GRU(20)→GRU(20)→Dropout(0.2)→Dense	

下面进行模型效果评判，结果如表3.2：

表 3.2: 10 种调参模型的模型效果评判

序号	r	T / ms	s
1	0.9342	58.4025	1.5996
2	0.9354	69.2492	1.3507
3	0.9376	61.2784	1.5301
4	0.9109	61.6864	1.4767
5	0.9382	66.7216	1.4061
6	0.9383	62.9502	1.4905
7	0.5082	257.0746	0.1977
8	0.8806	63.6268	1.3839
9	0.9350	67.8759	1.3776
10	0.9357	79.8768	1.1715

3.4.3 每个 GRU 单元中偏置 b 初始化为 1^[2]

由上一小节, 我们得出: 模型 1, 即 GRU(20)→Dropout(0.2)→GRU(20)→Dropout(0.2)→Dense 的效果最好, 因此此后都在此基础上修改。

此处将所有偏置 b 初始化为 1, 模型代码更改为:

```

1 function layers=get_gru_net(wd)
2     numFeatures=wd;
3     numResponses=1;
4     numHiddenUnits=20;
5
6     layers=[sequenceInputLayer(numFeatures)
7             gruLayer(numHiddenUnits,"BiasInitializer","ones")
8             dropoutLayer(0.2)
9             gruLayer(numHiddenUnits,"BiasInitializer","ones")
10            dropoutLayer(0.2)
11            fullyConnectedLayer(numResponses)
12            regressionLayer];
13 end

```

评判模型效果, 结果如下: $T = 60.573ms$, $r = 0.93623$, $s = \frac{100r}{T} = 1.54562$ 。综合效果不如原本模型, 但其训练收敛速度较原本模型有较大提升 (loss 收敛至 3×10^{-4} 只需要 40 轮)。

3.4.4 全链接层使用 Highway Network 代替^[3]

Highway Network 的灵感来自“解决 RNN 的问题，提出的 LSTM 结构”也就是加入“门”结构。

对于一般网络，其输出可以用如下公式表示：

$$y = H(x, W^H) \quad (3.3)$$

而在 Highway Network 中，我们定义一个新网络 T，此时输出化为：

$$y = H(x, W^H) \cdot T(x, W^T) + x \cdot (1 - T(x, W^T)) \quad (3.4)$$

此时不难发现，对于特殊的 T 值，该输出有如式3.5的表现：

$$y = \begin{cases} x & T = 0 \\ H(x, W^H) & T = 1 \end{cases} \quad (3.5)$$

若假设所有的门 T 的均值为 0.5 的话，就是把所有的原始信息一半激活，一半不变直接输入下一层，这样一来则保留了很多信息。同时反向传播的时候，可以让更多的（梯度）信息直接回流到输入，而不需要经过一个非线性转化。

由于笔者没有找到 Matlab 中如何自定义网络，因此此部分在 Python 中使用 tensorflow 实现。

使用 tensorflow 定义 Highway 单元，其代码如下：

```

1 def highway(x, size, activation, carry_bias=-1.0):
2     W_T = tf.Variable(tf.truncated_normal([size, size], stddev=0.1), name="
        weight_transform")
3     b_T = tf.Variable(tf.constant(carry_bias, shape=[size]), name="bias_transform")
4
5     W = tf.Variable(tf.truncated_normal([size, size], stddev=0.1), name="weight")
6     b = tf.Variable(tf.constant(0.1, shape=[size]), name="bias")
7
8     T = tf.sigmoid(tf.matmul(x, W_T) + b_T, name="transform_gate")
9     H = activation(tf.matmul(x, W) + b, name="activation")
10    C = tf.sub(1.0, T, name="carry_gate")
11
12    y = tf.add(tf.mul(H, T), tf.mul(x, C), "y")
13    return y

```

更改后的神经网络架构为：GRU(20)→Dropout(0.2)→GRU(20)→Dropout(0.2)→Highway。此时对此模型进行评估可以得到如下结果： $T =$

$$50.387ms, r = 0.93382, s = \frac{100r}{T} = 1.8536$$

相较于原本模型： $T = 58.402ms$, $r = 0.93418$, $s = \frac{100r}{T} = 1.5996$ 有较大幅度的提升。

3.5 结果

由于 Matlab 中无法定义 highway 网络，因此 Matlab 使用结果 3.4.3 中模型得出结果。结果如图3.5所示。

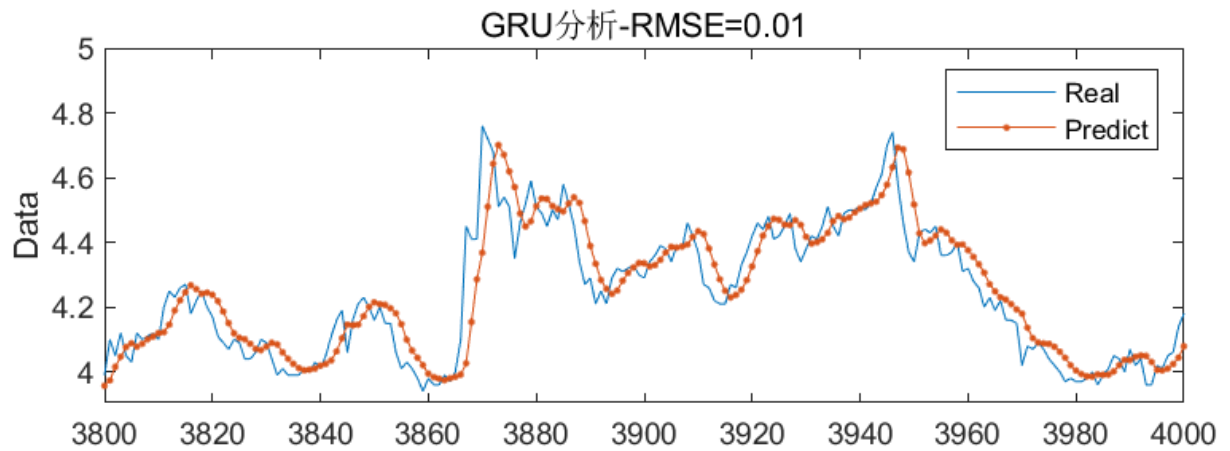


图 3.5: MATLAB 中使用 3.4.3 中模型得出结果

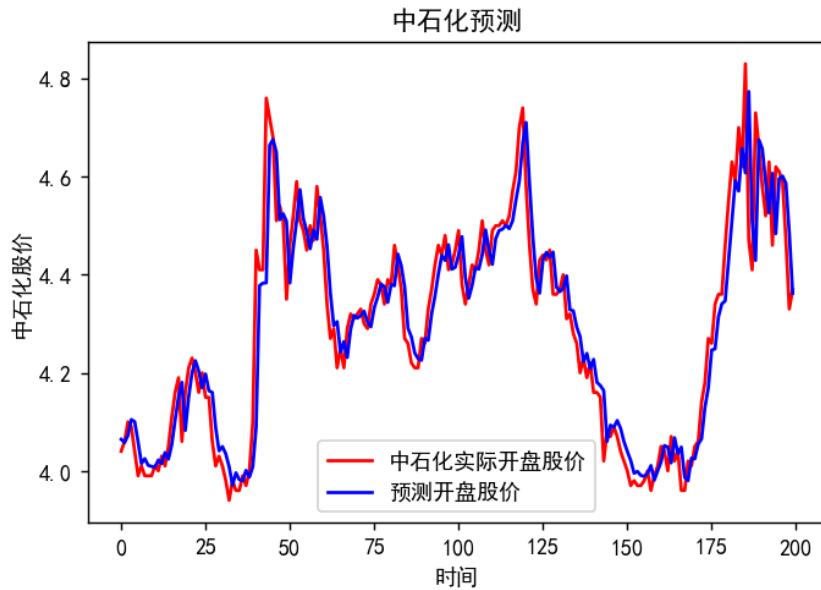


图 3.6: Python 中使用 3.4.4 模型得出的结果

在 Python 中，把全链接层换为 Highway 层，得出结果如图3.6所示。

4 股市 K 线预测

4.1 K 线图的画法

K线图源于日本, 被当时日本米市的商人用来记录米市的行情与价格波动, 后因其细腻独到的标画方式而被引入到股市及期货市场。下面来实现茅台预测日 K 线的描绘。

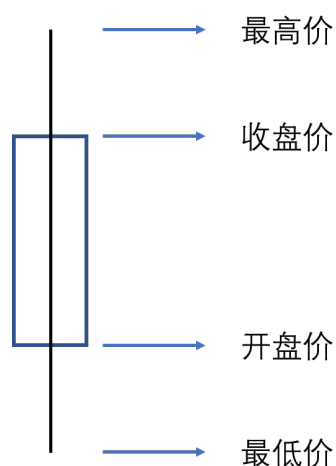


图 4.1: K 线图画法

日 K 线有如下特点^[4]:

1. 日 K 线为股票当日的开盘价、最高价、最低价、收盘价情况
2. 阳线为红色柱体, 表示股票上涨情况
3. 阴线为绿色柱体, 表示股票下跌情况

4.2 K 线数据的准备

K 线数据需要:

1. 开盘价
2. 最高价

3. 最低价

4. 收盘价

使用上一节中的模型对此 4 组数据进行预测并保存到 Xlsx 中，方便在绘制的时候调用。

4.3 Matlab 中 K 线的实现^[5]

Matlab 中有 candle.m 函数可以实现蜡烛图的绘制，但其结果如下：

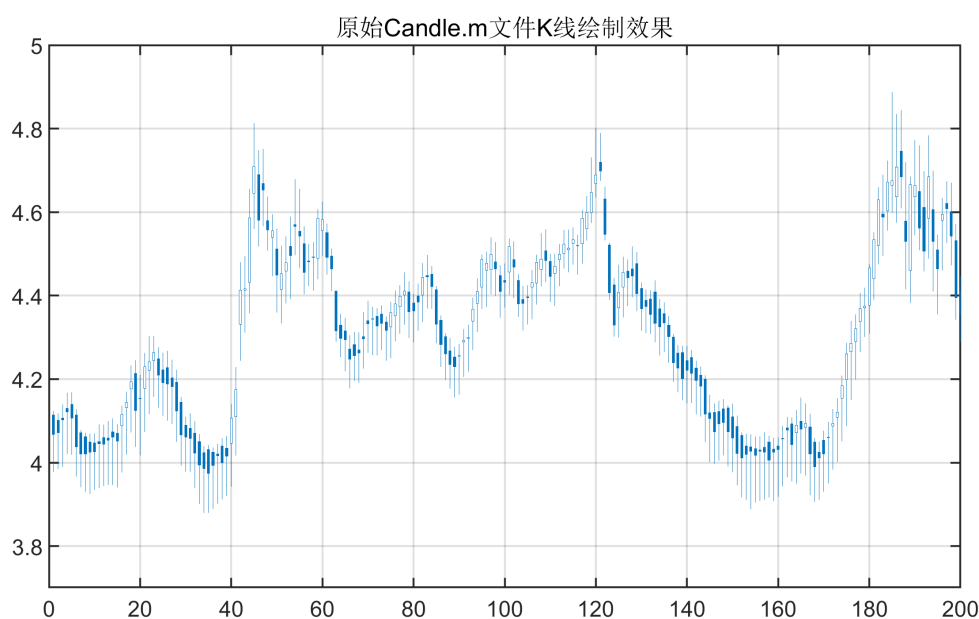


图 4.2: 原始 Candle.m 文件 K 线绘制效果

虽然已经有了大体的框架，但与我们常用的 K 线图还是有些许不同，因此需要对脚本进行修改，加入颜色，使其更加直观。

1. 根据开盘价和收盘价设置响应的颜色，其代码如下：

```
1 kk=indexVertical(:,find(op > cl));
2 kkk=hiloVertical(:,find(op > cl));
3 hh1=plot(ax, kk(:,), kkk(:,), 'color', [0 150/255 0], 'LineWidth', 1, 'AlignVertexCenters',
4         'on');
5 hold on;
6 kk=indexVertical(:,find(op <= cl));
7 kkk=hiloVertical(:,find(op <= cl));
8 hh2=plot(ax, kk(:,), kkk(:,), 'r-', 'LineWidth', 1, 'AlignVertexCenters', 'on');
9 h = [hh1 hh2];
```


2. 判断需要填的颜色，其代码如下：

```
1 filledIndex = ones(numObs, 1);
2 filledIndex(op <= cl) = 2;
```

3. 修改涨跌颜色（涨使用红色，跌使用绿色），其代码如下：

```
1 colorSet = {[0 150/255 0], 'red'};
```

4. 修改边框颜色，其代码修改如下：

```
1 for i = 1 : numObs
2     h(i+2) = fill(ax, ...
3         [indexLeft(i); indexLeft(i); indexRight(i); indexRight(i)], ...
4         [op(i); cl(i); cl(i); op(i)], colorSet{filledIndex(i)}, 'Edgecolor', colorSet{
5             filledIndex(i)}, ...
6         'AlignVertexCenters', 'on');
```

如此修改后即可实现 K 线的绘制。

4.4 结果



图 4.3: 股市 K 线预测

5 实验总结

在这次课程设计中，我们运用课堂上所学习到的知识和课外搜集到的资料，用 MATLAB 实现了对股市的预测。在设计过程中，我们遇到了以下几个问题：

1. 不会使用 MATLAB 的深度学习工具箱。
2. 代码实现上，一开始忘记反归一化，结果中出现负数。
3. 刚开始训练 100 次，用了 78 分钟，效率低。

通过资料查询、代码检查和讨论分析，我们把上述问题都解决了。以下是我们的解决方法：

1. 通过查阅官方文档和网上其他人的使用例子，我们学会了使用 MATLAB 的深度学习工具箱。
2. 通过检查代码，我们发现是忘记反归一化导致负数的出现。反归一化后，不出现负数。
3. 尝试更换电脑进行训练，训练时间缩短，效率提高。

通过这次课程设计，我们学习到了更多有关 MATLAB 的知识，如 RNN 网络、LSTM 结构、GRU 网络的结构和特点；深度学习工具箱的使用；GRU 神经网络在 MATLAB 中的搭建等。同时，这次课程设计也让我们对 MATLAB 的使用更加熟练，提高了我们的动手能力和团队协作能力。

参考文献

- [1] MATHWORKS. Deep learning toolbox[EB/OL]. 2021. https://ww2.mathworks.cn/help/deeplearning/?s_tid=srchbrcm.

- [2] JOZEFOWICZ R, ZAREMBA W, SUTSKEVER I. An empirical exploration of recurrent network architectures[C]//International conference on machine learning. [S.l.]: PMLR, 2015: 2342-2350.
- [3] SRIVASTAVA R K, GREFF K, SCHMIDHUBER J. Highway networks [J]. arXiv preprint arXiv:1505.00387, 2015.
- [4] 李晟. 基于 SVG 技术的 K 线图的设计与实现[D]. 四川: 电子科技大学, 2009.
- [5] 肆拾伍. 如何用 matlab 画股市 K 线, 修改内置 candle 函数[EB/OL]. 2019. https://blog.csdn.net/qq_43157190/article/details/94412482.