

# EI339 Class Project Report

陈子轩 516030910545

2018年12月

## 目录

<b>1</b>	<b>问题描述</b>	<b>1</b>
1.1	股价变动机制 . . . . .	1
1.2	数据描述 . . . . .	2
1.3	任务要求 . . . . .	2
<b>2</b>	<b>算法设计</b>	<b>3</b>
2.1	LSTM . . . . .	3
2.1.1	算法介绍 . . . . .	3
2.1.2	建模方式 . . . . .	5
2.1.3	参数选择依据 . . . . .	6
2.2	DNN . . . . .	6
2.2.1	建模方式 . . . . .	6
2.3	其他的尝试——底层LSTM的实现 . . . . .	6
2.3.1	模型搭建的方法 . . . . .	6
2.3.2	建模方式 . . . . .	6
<b>3</b>	<b>组员分工</b>	<b>8</b>
<b>4</b>	<b>感想</b>	<b>8</b>

## 1 问题描述

### 1.1 股价变动机制

如图所示，影响股价变动的机制有长期因素和短期因素，他们都最终归结为买卖博弈来影响价格。买卖博弈可以通过三个方面体现：已公开的买卖需求、正在实施的买卖动作、

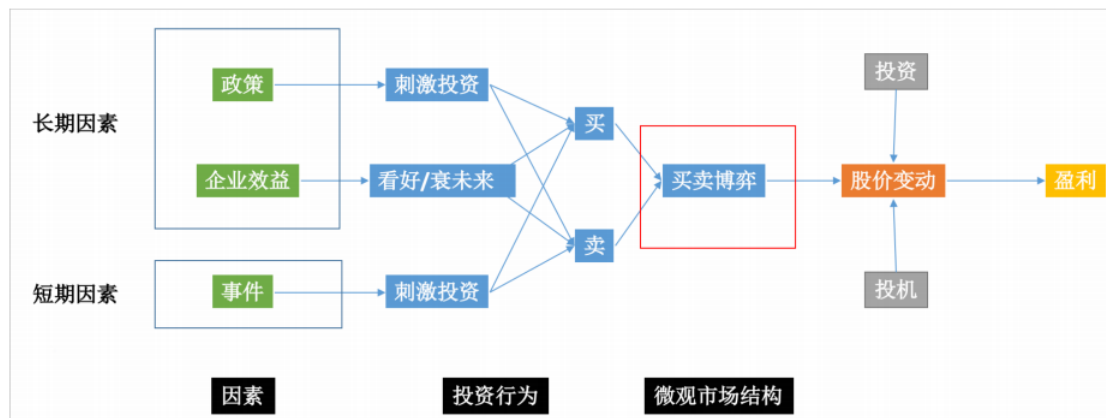


图 1: 股价变动机制

持币观望。已公开的买卖需求可以通过订单簿来体现，正在实施的买卖动作经研究可以看做是随机尝试的实现,而第三类信息是难以获得的。本次大作业主要研究订单簿对价格的影响。

## 1.2 数据描述

订单簿的数据为：

- \* 日期(Date)
- \* 时间(Time)
- \* 申买价(Bid Price)
- \* 申买量(Bid Volume)
- \* 申卖价(Ask Price)
- \* 申卖量(Ask Volume)
- \* 最新成交价>Last Price)
- \* 中间价( $\text{MidPrice} = (\text{Bid Price} + \text{Ask Price}) / 2$ )
- \* 当前累计成交数量(Volume)

## 1.3 任务要求

通过对订单簿中数据的学习，预测下20个时间点中间价（mid price）的均值

## 2 算法设计

### 2.1 LSTM

#### 2.1.1 算法介绍

LSTM(Long-Short-Term-Memory)是RNN的一种变体,要了解LSTM,首先需要了解RNN的工作原理,如下图

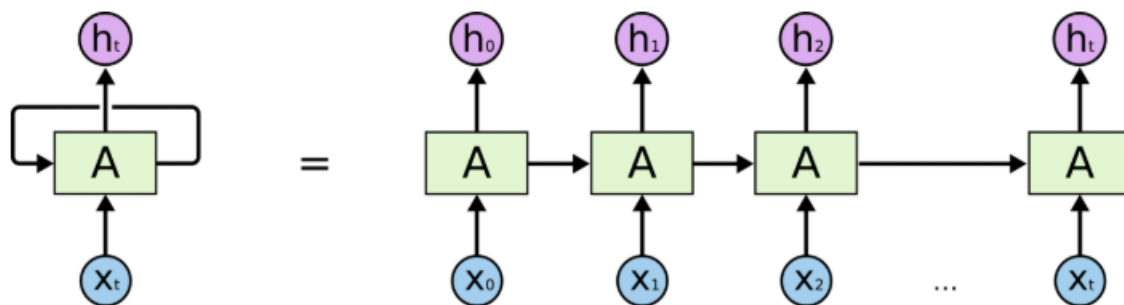


图 2: Recurrent Neural Network(RNN) [1]

**RNN的结构** 传统的RNN包含一个循环态,如图2左边的单元,它可以读入一个输入 $x_t$ ,输出一个 $h_t$ ,从而将状态转移到下一个时间点。如果将这个单元按照时间序列展开则可以得到图二右边的单元序列。由于它的链式的特征,他自然与时间序列和列表产生关联,从而可以解决由于时间的推移所带来的问题。RNN可以利用对先前信息的理解对后续的事件做出预测,例如我们想要预测“the clouds are in the sky”这句话的最后一个单词,由于根据前面的句子很显然最后一个词就是sky,所以通过RNN可以准确的预测出来。

**RNN的缺陷** 但是如果我们想要预测 “I was born in Japan, .... I speak Japanese” 这样需要跨越较长的时间序列才能获得的信息时则显得力不从心,这是基于RNN自身结构的问题,但是幸运的是, LSTM并不存在这样的问题。

**LSTM的结构** LSTM在RNN的基础上,改进了工作单元的机构,传统的RNN模块包含的是重复单一的层,但是LSTM的重复模块包含的是四个交互的层。如下图 我们将结合图5更进一步介绍LSTM的内部信息。

**1. 决定丢弃信息** 对于上一个时间点的信息,首先决定丢弃那些信息,这一步通过遗忘门来决定,该门将读取 $h_{t-1}$ 和 $x_t$ ,输出一个 $f_t$ , $f_t$ 内的值为 $[0, 1]$ ,0表示完全舍弃, 1表示完全保留。

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (1)$$

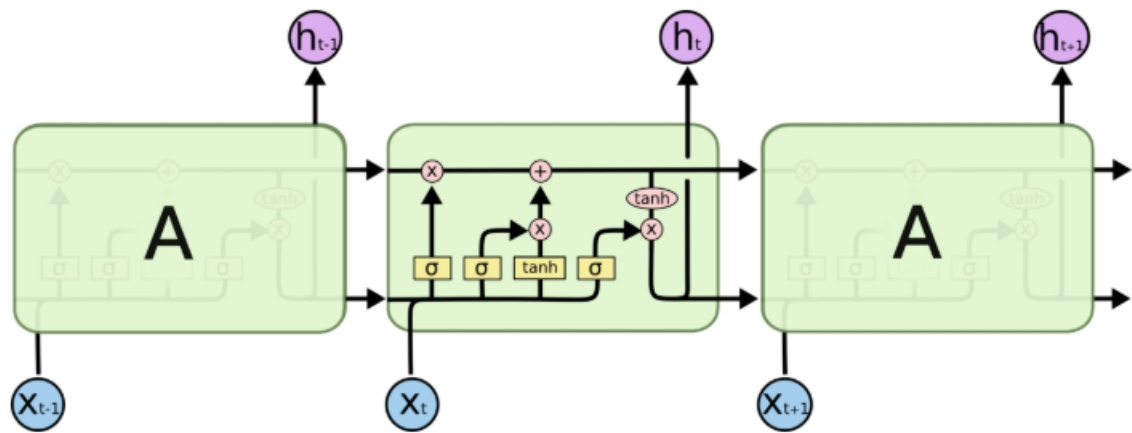


图 3: LSTM的重复模块 [1]

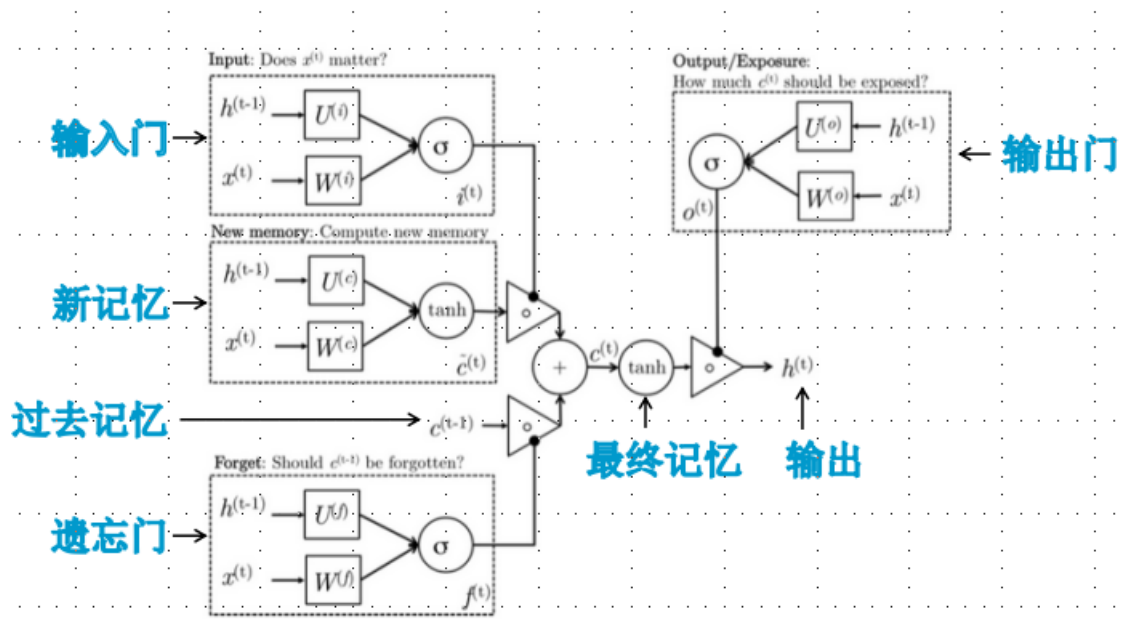


图 4: LSTM的内部结构 [2]

**2. 确定新信息** 下一步将是决定什么样的新信息将被存放在细胞状态中，这一步包含两个门，输入门和新记忆，**新记忆**通过tanh函数构建候选信息向量，**输入门**通过sigmoid函数判定什么值将要更新。

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_f) \quad (2)$$

$$a_t = \sigma(W_a \cdot x_t + U_a \cdot h_{t-1} + b_f) \quad (3)$$

**3. 更新细胞状态** 然后，新记忆将于旧记忆结合构成新的细胞状态

$$C_t = f_t \cdot C_{t-1} + i_t \cdot a_t \quad (4)$$

**4. 确定新输出** 最后，我们通过一个**输出门**决定什么将被输出

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_f) \quad (5)$$

$$h_t = \tanh(C_t) \cdot o_t \quad (6)$$

### 2.1.2 建模方式

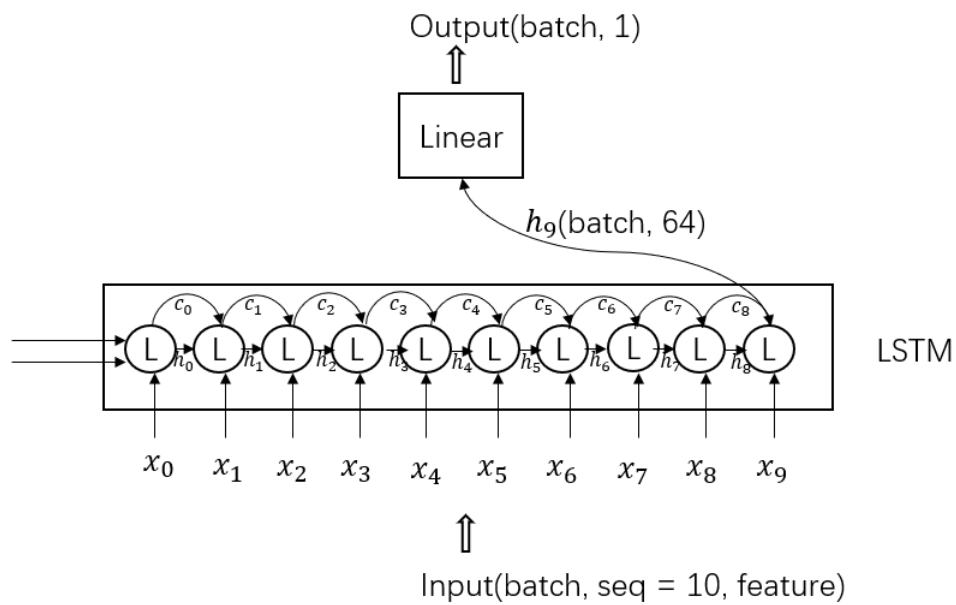


图 5: LSTM建模 [2]

如图5，根据LSTM的模型特征，我们选择通过前10天的订单簿数据预测下20个时间点中间价（mid price）的均值，即时间序列长度 $seq = 10$ 。选择的输入特征为订单簿的各类数据，包括['MidPrice', 'LastPrice', 'Volume', 'BidPrice1', 'BidVolume1', 'AskPrice1', 'AskVolume1', 'VolDiff'],其中VolDiff由相邻两日的Volume之差组成

### 2.1.3 参数选择依据

## 2.2 DNN

### 2.2.1 建模方式

DNN的建模方式比较简单，如图6所示，由三个全连接层构成

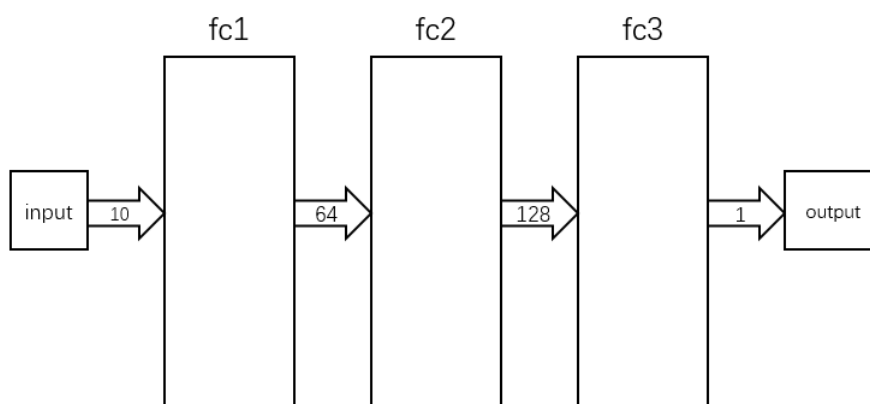


图 6: DNN模型结构 [2]

## 2.3 其他的尝试——底层LSTM的实现

由于前面两种模型都是使用pytorch进行搭建的，很多内部结构以及前向传播和反向传播过程都是由pytorch内部自动完成的，为了进一步探究关于LSTM的内部数据传递和前向和反向传递的原理，我们使用numpy从底层对LSTM进行了实现。

### 2.3.1 模型搭建的方法

如果需要从底层实现LSTM，则需要掌握它的前向传播和反向传播的原理，前向传播参考LSTM模型介绍。一下重点介绍反向传播。

### 2.3.2 建模方式

我们将前向传播的过程倒过来看

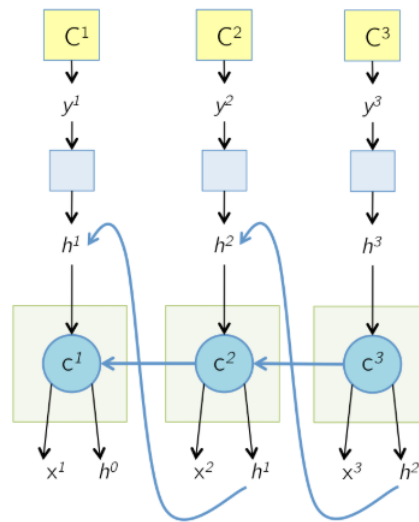


图 7: LSTM反向传播 [3]

**1. 确定新的输出** 由6式可以得出, 如果令  $\delta h_t = \frac{\partial E}{\partial h_t}$ , 则有

$$\frac{\partial E}{\partial o_i^t} = \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial o_i^t} \quad (7)$$

$$\delta o^t = \frac{\partial E}{\partial o^t} = \delta h^t \odot \tanh(c^t) \quad (8)$$

同理

$$\frac{\partial E}{\partial c_i^t} = \frac{\partial E}{\partial h_i^t} \cdot \frac{\partial h_i^t}{\partial c_i^t} \quad (9)$$

$$\delta c^t += \frac{\partial E}{\partial c^t} = \delta h^t \odot o^t \odot (1 - \tanh^2(c^t)) \quad (10)$$

这里用+=的原因是  $c^t$  记忆了所有时刻的细胞状态, 故每个时间点迭代时,  $c^t$  累加

**2. 确定新的细胞状态** 由4式, 可以得到loss对于各个细胞状态的偏导

$$i^t = \delta c^t \odot a^t \quad (11)$$

$$\delta f^t = \delta c^t \odot c^{t-1} \quad (12)$$

$$\delta a^t = \delta c^t \odot i^t \quad (13)$$

$$\delta c^{t-1} = \delta c^t \odot f^t \quad (14)$$

令

$$\hat{i}^t = W_i \cdot x_t + U_i \cdot h_{t-1} + b_f \quad (15)$$

$$\hat{f}^t = W_f \cdot x_t + U_f \cdot h_{t-1} + b_f \quad (16)$$

$$\hat{a}^t = W_a \cdot x_t + U_a \cdot h_{t-1} + b_f \quad (17)$$

$$\hat{c}^t = W_c \cdot x_t + U_c \cdot h_{t-1} + b_f \quad (18)$$

则有

$$\delta \hat{i}^t = \delta i^t \odot \hat{i}^t \odot (1 - \hat{i}^t) \quad (19)$$

$$\delta \hat{f}^t = \delta f^t \odot \hat{f}^t \odot (1 - \hat{f}^t) \quad (20)$$

$$\delta \hat{a}^t = \delta a^t \odot (1 - \tanh^2(\hat{a}^t)) \quad (21)$$

$$\delta \hat{c}^{t-1} = \delta c^t \odot \hat{c}^t \odot (1 - \hat{c}^t) \quad (22)$$

最后，可以求得各个权重矩阵的偏导数

$$\delta W_i + = \delta \hat{i}^t * h_{t-1}^T, \quad \delta U_i + = \delta \hat{i}^t * x_t^T, \quad \delta b_i + = \delta \hat{i}^t \quad (23)$$

$$\delta W_f + = \delta \hat{f}^t * h_{t-1}^T, \quad \delta U_f + = \delta \hat{f}^t * x_t^T, \quad \delta b_f + = \delta \hat{f}^t \quad (24)$$

$$\delta W_a + = \delta \hat{a}^t * h_{t-1}^T, \quad \delta U_a + = \delta \hat{a}^t * x_t^T, \quad \delta b_a + = \delta \hat{a}^t \quad (25)$$

$$\delta W_c + = \delta \hat{c}^t * h_{t-1}^T, \quad \delta U_c + = \delta \hat{c}^t * x_t^T, \quad \delta b_c + = \delta \hat{c}^t \quad (26)$$

至此，反向传播过程完毕，求得对各个权重矩阵的偏导后便可进行随机梯度下降了。

### 3 组员分工

1. 陈子轩 LSTM模型搭建，底层LSTM实现，实验报告撰写
2. 孟令佳原始数据处理，DNN模型搭建与调参，LSTM模型调参，实验报告撰写

### 4 感想

通过本次大作业，我深刻了解了LSTM模型的原理以及用它来解决实际问题的方法，并且根据自己的理解手动实现了LSTM的前向和反向传播，这让我对于深度学习有了更为深刻的认知，同时也让我体会到了数学基础对于高级算法的决定性作用。同时，在利用深度学习解决实际问题的过程中，我掌握了归一化等数据处理的方式，同时在消除过拟合的解决中学会了使用交叉验证集和正则化的方法，这些技巧对我今后的实践都有很大的帮助。此外，大作业还锻炼了我的耐性，在多次调试仍然效果不好的情况下能够继续耐住性子尝试新的方法，这对我以后都有很大的帮助。最后我想感谢我的队友和同学们，他们在我遇



到困难时耐心指导，帮了我很多。最后感谢助教的辛苦工作，让我们能够顺利完成这次作业。

## 参考文献

- [1] <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [2] <https://blog.csdn.net/u012319493/article/details/52802302>
- [3] <https://arunmallya.github.io/writeups/nn/lstm/index.html#/>