

基于人工智能的量化交易 策略设计

刘锦坤

郭锐冰

陈飞扬

Part 1

刘锦坤



模型设计

数据处理

代码实现

Goal&Implementation

Goal: 根据过去一段时间内(days_before)一支股票的交易数据, 判断接下来一段时间(days_after)内, 股票呈现多头趋势还是空头趋势(股价是上涨还是下跌), 根据模型的判断买入和卖出股票, 实现经济收益。

Implementation: 建立一个神经网络模型, 模型输入为过去一段时间的交易数据, 预期输出接下来一段时间内股票上涨和下跌的概率。

神经网络的训练是监督学习的过程, 我们需要过去交易中股票的买卖数据和股价数据作为训练原始数据。

Attention Data is all we need.

Data Acquisition

为了获取过去的股票交易数据作为训练数据，外联了聚宽量化投研平台（JoinQuant平台），对方愿意为我们提供过往股票交易数据。可以通过其提供的Python的API访问获取过往股票交易数据。用于下一步的数据处理。



模块	数据名称	API接口	是否支持试用
通用接口	批量查询股票/期货/基金/期权/债券/宏观数据库	run_offset_query	✓
股票	股票1天/分钟行情数据	get_price(round参数)	✓
基金	基金1天/分钟行情数据	get_price(round参数)	✓
股票单季度	获取多个季度/年度的历史财务数据	get_history_fundamentals	✓
	获取多个标的在指定交易日范围内的市值表数据	get_valuation	✓
期货	获取期货合约的信息	get_futures_info	✓
风险模型	获取因子看板列表数据	get_factor_kanban_values	✗
	获取因子看板分位数历史收益率	get_factor_stats	
	获取风格因子暴露收益率	get_factor_style_returns	
	获取特异收益率（无法被风格因子解释的收益）	get_factor_specific_returns	
alpha因子	批量获取alpha101因子	get_all_alpha_101	
	批量获取alpha191因子	get_all_alpha_191	
债券	可转债交易标的列表	get_all_securities	
	1天/分钟行情数据	get_price	
	指定时间周期的分钟/日行情	get_bars	
	可转债Tick数据	get_ticks	

Attention Model is all we need.

RNN Layer

股票数据属于典型的时间序列数据，查阅相关资料发现，**RNN（Recurrent Neural Network）**类神经网络以其独特的构造适合时间序列的模型。

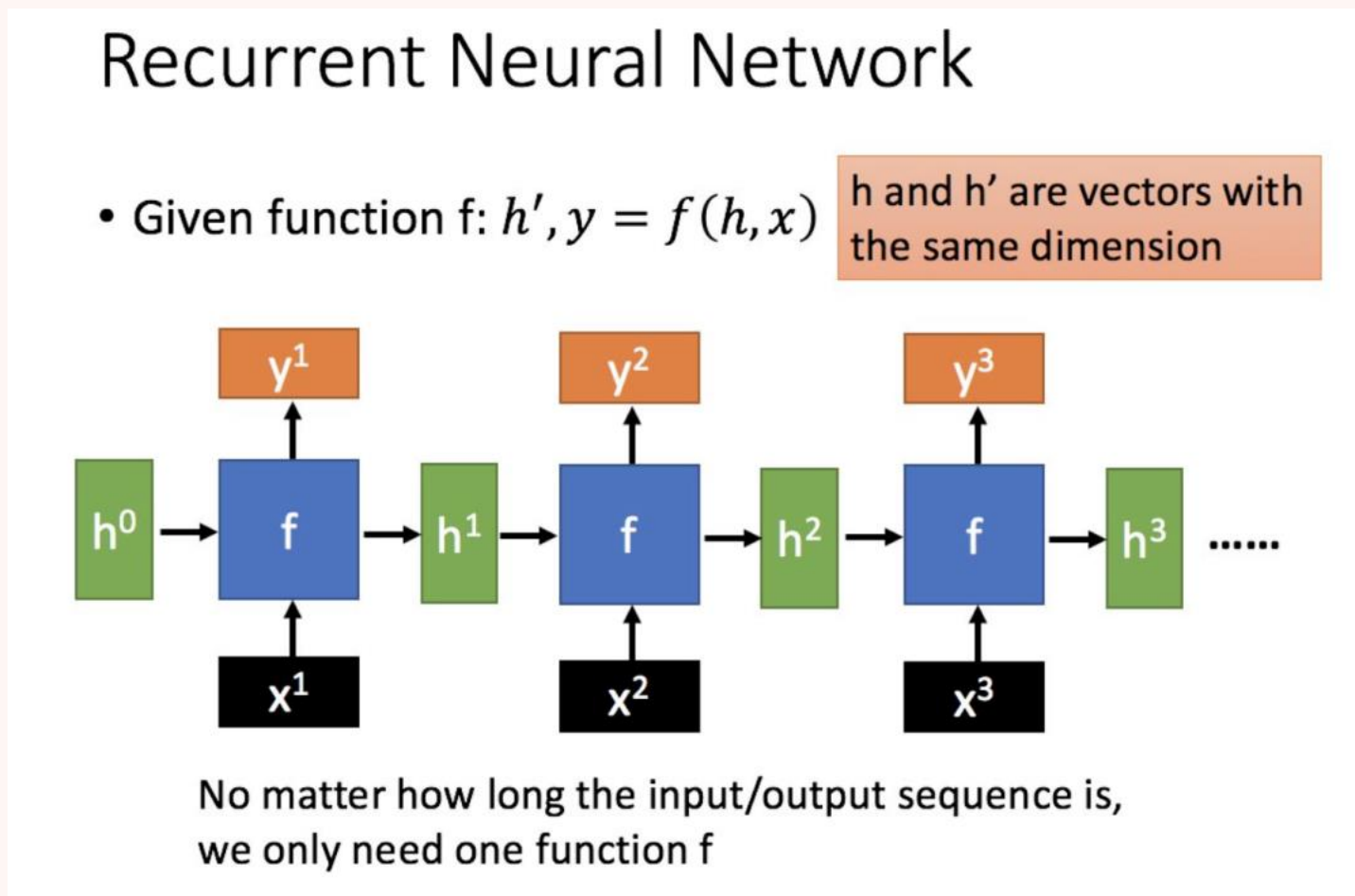


图1. RNN模型示意图

LSTM Layer

但是RNN模型存在对于长距离关系捕捉的不足，序列前端的信息可能在循环传递中被遗忘，对于股票来说，以前的一次暴跌和暴涨很可能就会影响后续投资者的信心，因此我们最终采取了**LSTM（Long Short Term Memory）**网络。

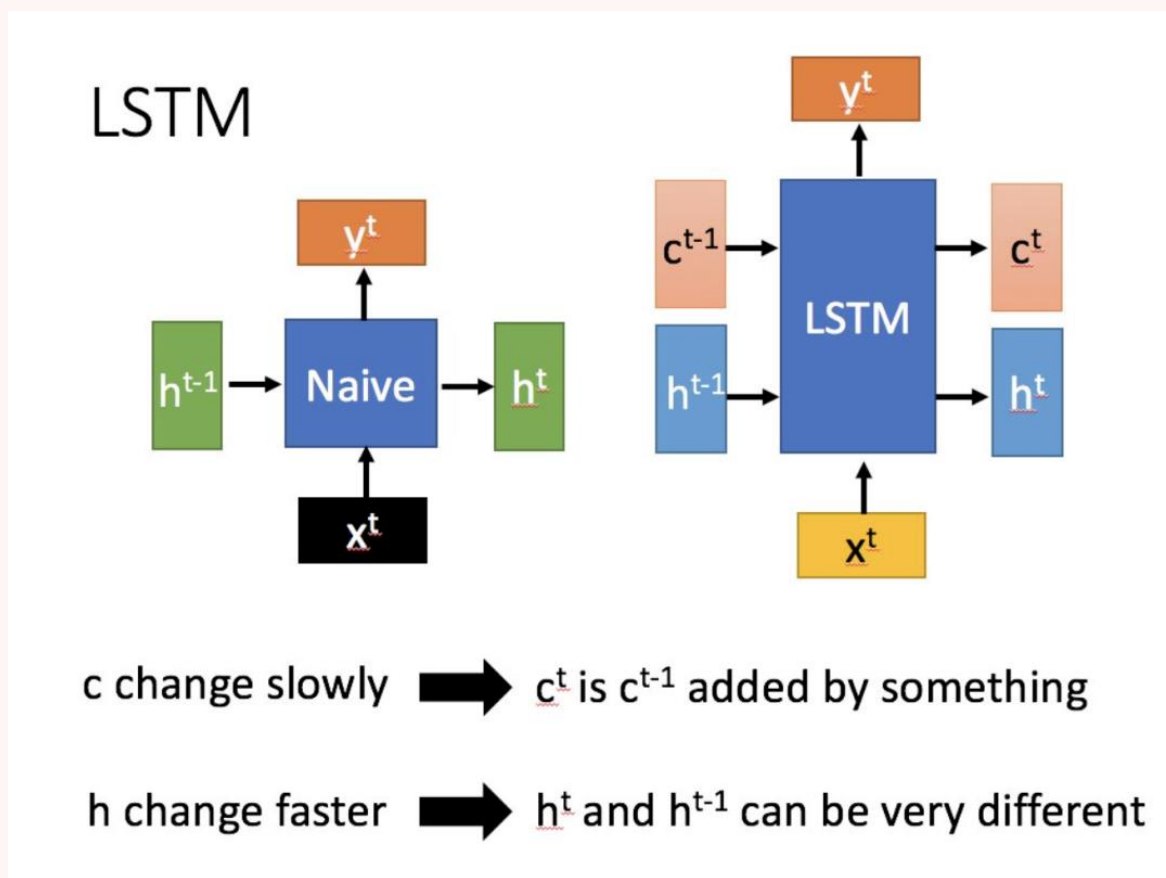


图2. LSTM模型的改进

Network Design

最后设计神经网络结构如下图所示：

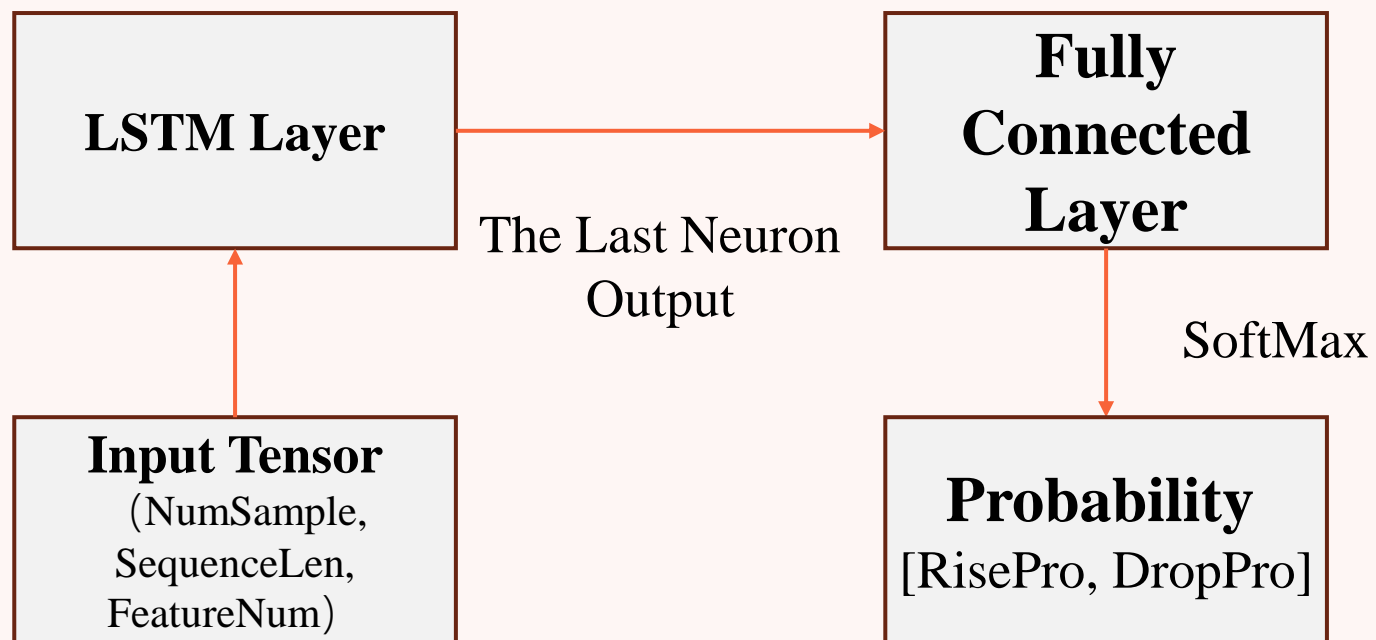


图3. 网络设计图

Data Processing

最终汇集了目标股票在过去每天的开盘价、收盘价、盘中高价、盘中低价、成交量、成交额、前一天收盘价、成交均价八大数据作为原始特征数据。

	A	B	C	D	E	F	G	H	I	
1		open	close	high	low	volume	money	pre_close	avg	
2	2016/1/4	14.56	13.25	14.72	13.25	4849821	67217072	14.72	13.86	
3	2016/1/5	12.52	12.69	13.24	12.07	8199962	1.04E+08	13.25	12.71	
4	2016/1/6	12.7	13.07	13.11	12.7	4843023	62717700	12.69	12.95	
5	2016/1/7	12.82	11.77	12.92	11.77	1594342	19274260	13.07	12.09	
6	2016/1/8	12.05	11.74	12.29	10.9	5921625	70194968	11.77	11.85	

Data Processing

数据预处理：为了使数据中的特征能够更好的被模型捕捉，对数据进行了预处理。借鉴到股票市场常用的一些技术形态（放量上涨，缩量下跌），（尤其是A股市场）交易数据直接的值并不很好的反映多头空头趋势，**其量比更反应市场情绪**（例如今日交易量与昨日交易量的比值），因此将上述八项数据分别与前一天的比值作为每天的处理后数据，得到预处理后的数据。

```
for i in range(num_samples):  
    x[i] = original_data.loc[i+1:i + days_before, 'open':].values/ \...  
           original_data.loc[i:i + days_before-1, 'open':].values
```

而每个样本的训练标签则由days_after后股票涨跌决定，**相对今日上涨则为[0, 1]，下跌则为[1, 0]**，这是模型的训练目标输出。

Implementation&Training

在完成上述设计后，使用Pytorch建立了相应的神经网络模型并进行训练：

- 由于问题本身接近分类问题（看涨或看跌），所以采用了 **CrossEntropyLoss**（交错熵）作为损失函数。
- 通过查阅资料，选择Adam优化器作为训练过程的优化器。
- 为了克服过拟合现象，采取了**dropout**手段，在训练时会随机丢弃部分神经元。
- 为了加速训练，使用了本地的RTX3060和RTX4060，利用CUDA进行了GPU加速计算。

Implementation&Training

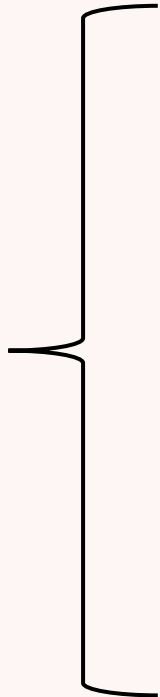
模型成功训练并运行，但是可以调节超参数取得更好的效果

days_before, days_after, batch_size, hidden_size,
num_layer, dropout...

~~Attention~~ Hyperparameters are all we need.

Part 2

郭锐冰



Job1 超参调节

Job2 模型训练

Job3 策略实现

Parameters Optimization

- 优化各种超参数，使得测试集正确率与训练集接近。训练成果

```
Start data processing...
```

```
样本总数：1965
```

```
Using CPU
```

```
训练集准确率：0.6578054298642534
```

```
测试集准确率：0.5329949238578681
```

```
测试集上涨概率：0.4365482233502538
```

```
是否保存模型？(y/n)n
```

days_before = ?

days_after = ?

batch_size = ?

hidden_size = ?

Attention Training is all we need.

Results

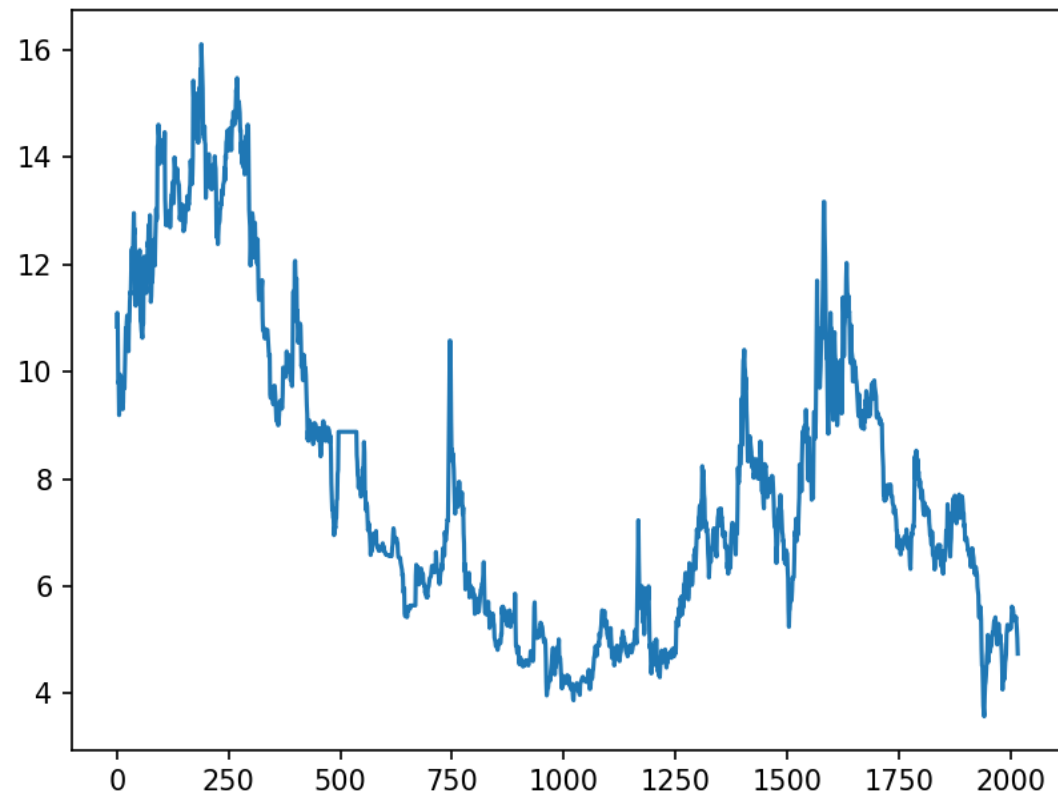


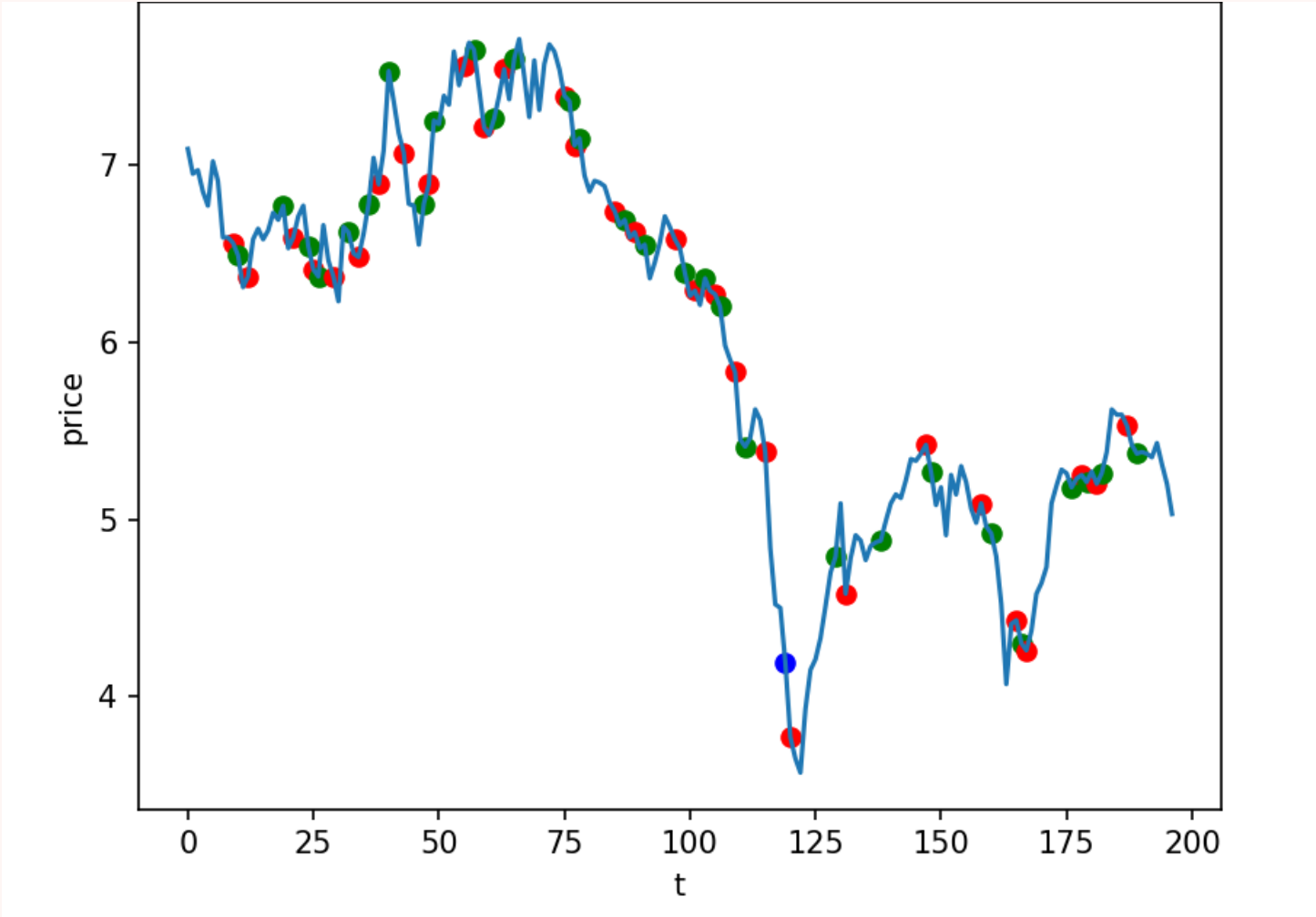
图4. 股票002765.XSHE预测图

Exchange Strategy

- 使用训练的结果，在测试集上尝试建立策略：
- 1) 当上涨概率大于 P_b 时以收盘价**买入**
- 2) 买入后若上涨概率小于 P_s **卖出**，或跌幅大于20%**止损**

```
P_b = 0.7 #上涨概率大于多少买入
P_s = 0.3 #上涨概率低于多少卖出
ans = 0 # 0表示未买入，1表示已买入
trade = np.zeros(len(y_pred))
t = range(len(y_pred))
k = price[0]
for i in range(len(y_pred)):
    if ans == 0 and y_pred[i][1] >= P_b:
        ans = 1
        trade[i] = 1
        k = price[i]
    if ans == 1 and y_pred[i][1] <= P_s:
        ans = 0
        trade[i] = -1
    # 止损策略
    if ans == 1 and ((k - price[i]) / k >= 0.20):
        ans = 0
        trade[i] = -2
```

Strategy Implemented



总收益百分点:
29.03649191324633

卖出, 收益百分点: -0.7587253414264009	卖出, 收益百分点: -1.1164274322168963
卖出, 收益百分点: -0.6240249609984405	卖出, 收益百分点: -7.204116638078901
卖出, 收益百分点: 3.924646781789639	卖出, 收益百分点: -22.11895910780668
卖出, 收益百分点: 4.629629629629626	卖出, 收益百分点: 27.05570291777188
卖出, 收益百分点: 9.288824383164014	卖出, 收益百分点: 6.55021834061135
卖出, 收益百分点: -4.101838755304102	卖出, 收益百分点: -2.7675276752767592
卖出, 收益百分点: 5.224963715529758	卖出, 收益百分点: -3.3398821218074644
卖出, 收益百分点: 1.1904761904762005	卖出, 收益百分点: -2.934537246049659
卖出, 收益百分点: 0.6934812760055454	卖出, 收益百分点: 21.5962441314554
卖出, 收益百分点: 0.7957559681697561	卖出, 收益百分点: -0.7619047619047625
卖出, 收益百分点: -0.4059539918809115	卖出, 收益百分点: 1.1538461538461462
卖出, 收益百分点: 0.5625879043600567	卖出, 收益百分点: -2.8933092224231487
卖出, 收益百分点: -0.7418397626112733	
卖出, 收益百分点: -1.057401812688826	
卖出, 收益百分点: -2.8875379939209784	
卖出, 收益百分点: 1.1128775834658233	
卖出, 收益百分点:	

谢谢！