股市决策模型

马铖迪 PB16000348
程家辉 PB16001131
阚皓玮 PB16011217
完成比例 1：1：1

6 月 15 日

一 摘要

个人认为在股市决策的问题中，最重要的两个方面是选股和决定买入卖出时机。对于 10 年的数据，在我们设定好操作周期的情况下，一方面，piotroski 算法可以很好地选取走势较好的股票；另一方面，由于股市的股价波动可以视为一个时间序列，我们用 FFT 消噪并用 ARIMA 算法预测可以得到未来的股价，并由此可以得到合适得买入卖出时间，由此可以得到一个较好得年化收益率。我们最终实现得操作方法在行情较好时可以达到 53.702%的年化收益率，且风险非常低。

关键词：PIOTROSKI 选股 FFT 消噪 时间序列分析 ARIMA 股市预测

二 模型的建立

1. 股市最基本的规律即为波动性，具有非平稳的波动，在此处运用小波分析的消噪方法较为合适，我们想要提高股市的收益率，主要在于选择情况较好得股票以及一定程度上预测股市的涨跌，故我们一方面需要精确地预测出股票的走势，另一方面得出合理的股票选择，本文应用快速傅里叶变换和时间序列分析的方法，对股市作出预测，以实现较好的年收益率，并总结出可行的操作方案。

2. 在我们已经选出几支合适的股票之后，股票的股价走势可视为一个时间序列模型，股价与时间 t 如下：

$$p = f(t)$$

股市的变化瞬息万变，不是一个平稳的时间序列模型，故我们需要对数据作约化和近似，将其转化为一个平稳的时间序列模型：

$$p = \widetilde{f(t)}$$

此模型相比较于前一组数据，具有较好的平滑性，规律更加明显和平稳，更适合准确预测其走势。

3. 股票的收盘价随着时间的波动被我们视为了一组随时间变化的信号，我们的决策包含买卖时机，周期，投入资金比例。为了使决策的收益率尽可能高，我们需要对数据进行尽可能准确的预测。按照我们建立的模型，我们用 10 年的数据预测 1 年。进行该预测的最大难题在于股市的数据瞬息万变，存在高频振荡噪声。

快速傅里叶变换（FFT）是 20 世纪最著名的十大算法，应用 FFT 可以从给定信号中滤除高频噪声，对离散的点列信号作 FFT，消除噪声频率大于 5 的系数（这个可以按情况调整），再用 IFFT（快速傅里叶逆变换）对数据重构。

4. 在本文的模型中，我们假定即买即卖，不讨论买卖的操作时间及手续费。同时假定股票的真实信息包括股价都是真实可靠的，故可以依次选出走势较好的股票。

5. 我们的每次买入股票，在刚开始时对选出的股票平均分配，而且在每个操作

周期时所用钱全部投入，以实现利滚利，使得收益率最大。

三 正文

基本知识介绍:

1. 自回归移动平均模型(ARMA(p, q))为

$$X_t = \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \cdots + \varphi_p X_{t-p} - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q}$$

式中，p 和 q 是模型的自回归阶数和移动平均结束；θ, φ是不为零的待定系数；ε_t是独立的误差项；X_t是平稳，正态，零均值的时间序列。

2. 在上述定义中，设$\{X_t, t \in Z\}$是零均值平稳序列，对任意 t，满足线性差分方程

$$X_t - \varphi_1 X_{t-1} - \cdots - \varphi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q}$$

其中，$\{Z_t\} \sim WN(0, \sigma^2)$，p 阶自回归多项式:

$$\varphi(z) = 1 - \varphi_1 z - \cdots - \varphi_p z^p$$

与 q 阶滑动平均多项式:

$$\theta(z) = 1 - \theta_1 z - \cdots - \theta_p z^p$$

无公共因子，$\varphi_p \neq 0, \theta_q \neq 0$,则称$\{X_t, t \in Z\}$是 p 阶自回归和 q 阶滑动平均序列，简称为$\mathrm{ARMA(p, q)}$序列，称$\{X_t, t \in Z\}$满足$\mathrm{ARMA(p, q)}$模型。则$\mathrm{p, q}$分别称为自回归阶数和滑动平均阶数，实参数$\varphi_1, \varphi_2, \cdots, \varphi_p$为自回归参数，实参数$\theta_1, \cdots, \theta_q$为滑动平均参数。

若 B 为延迟算子，则可简介为:

$$\varphi(B) X_t = \theta(B) Z_t, t \in Z$$

于是$\mathrm{ARMA(p, 0)}$模型即为$\mathrm{AR(p)}$即 p 阶自回归模型，而$\mathrm{ARMA(0, q)}$模型就是$\mathrm{MA(q)}$即 q 阶移动平均模型，有$\mathrm{p + q}$个参数要估计，可以用计算机编程实现。

其中，如果$\theta(z) \equiv 1$，则称满足线性差分方程

$$\varphi(B)X_t = Z_t$$

的零均值平稳序列$\{X_t\}$为 q 阶自回归序列，简记为$AR(p)$序列，称$\{X_t, t \in Z\}$满足$AR(p)$模型。

若$\varphi(z) \equiv 1$，则称满足线性差分方程

$$X_t = \theta(B)Z_t$$

的零均值平稳序列$\{X_t\}$为 q 阶滑动平均序列，简记为$MA(q)$序列，称为$MA(q)$模型。

3. 快速傅里叶变换(Fast Fourier Transform)，即利用计算机计算离散傅里叶变换(DFT)的高效快速计算方法的统称，简称 FFT。该算法使得运算的时间大为减少，具有较好的时间效率。

   DFT 的操作公式如下：

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn}, k = 0, \cdots N-1$$

   将一个数组(一组数据)变为其频谱数据，而 FFT 是使用计算机的二进制的计算特点降低其运算次数。

   使用 DFT 与 FFT 可以对数据进行消噪。对数据平滑化处理，以得到较为平稳的时间序列，再使用时间序列的方法。(事实上此处也可以使用小波变换处理，但如何消噪不是关键，操作所需时间效率也对收益率影响不大，所以此处使用快速傅里叶变换即可。)

4. 在本问题中需要对比较复杂的纯粹时间序列进行详细的分析，即使经过消噪，数据的平滑性可能仍然无法满足要求。而要想对有独立变量的时间序列进行

预测，指数平滑更是无能为力，所以需要强有力的模型。ARIMA 模型（整合自回归移动平均模型）更加平滑和精细，且再 Python 中有相应的库函数。ARIMA 模型对时间序列进行差分来消除序列的不平稳成分，使其变成平稳的时间序列，并估计 ARMA 模型，估计之后再转变该模型，使之适应于差分之前的序列模型。

按照我们地模型，所得地操作方案简要分为四个部分：

(1) 选股：在这一个部分中，我们需要考虑各方面的因素，综合考虑选股。首先是风险与收益的平衡，主要在于选择不同类型股票，以及不同走势的股票，以达到风险与收益的一个平衡。其次充分考察背景，考察各支股票背后公司的收支等情况，以及国家环境的大背景。由上述等方面综合选股。

(2) 预测：在这一部分中，我们通过搜集得到的各组数据，假定前 80%的数据是已知的，后 20%的数据是未知的，用这 80%的数据预测后 20%的数据。

(3) 决策：在我们已经预测到未来数据的情况下，我们只需要对照所得的数据对投资比例，买卖时间与周期等因素综合考察，以设计出一套最终的实际操作方案。

(4) 评估：事实上后 20%的数据我们已经得到，所以我们需要对决策的合理性及收益率作一个计算，若结果不理想，则作出调整。最后得到收益率。

具体的操作方案以及算例如下，以中国股市为例，取 10 年内地数据作为已知，作出决策并计算收益率测试，评价我们所建模型与操作方案的优点与不

足。

(1) 选股:

若要得到较好的年化收益率，选股策略十分关键，我们决定使用 Piotroski 选股方法，对所有的股票（稳定型或者创业型）统一操作，并挑选 6 个具有代表性的财务指标，然后根据各自的评分标准打分，得出综合评分。再由综合评分可以选取具有投资价值的高分组股票。将这些股票作为研究对象，作出买卖以及资金分配的决策。

就如上面所述的那样，Piotroski 选股方法是基于价值投资的思想。首先选出“市净率”值(PB 值)最低的前 20%的股票，再对 6 个财务指标进行打分，对于得分越高的股票，越有可能获得超额收益。“市净率”，即为账面价值，是指每股的“股价”与每股的“净资产”的比率，Piotroski 将市净值作为一个衡量企业是否被低估的指标，体现出企业的投资潜力。事实上，若一个企业的市净值很低，那么说明企业的真正价值被低估，所以我们选取市净值最低前 20%的股票作为可能的投资对象。

作为已经选出的待考察股票，考察了上市公司在盈利水平、偿还债务能力、运作经营情况和发展潜力等四大方面的因素，从中挑选出 6 个财务指标，同时自定义评分标准，以综合评分为 7 分以上的股票具有很高的投资价值。

我们的程序实现是在 http://www.RiceQuant.com 这一网站上跑的，主要是因为在这个网站上有着设计好的界面和模板，对于投资方案的收益率和风险的计算较便捷，而且可以改变所用数据的时间范围。

在实际的操作中，目标选出财务状况良好，而且整体运营进步的公司，分为两步进行操作:

1) 首先选择目标股票池并且进行一定的排序，这个选择和排序其实是决定这个策略更接近于（Value investment）稳定保值型还是(Growth investment)增长型，本文主要以利润增长作为主要的排序方式。

2) 在目标股票池当中用 6 个因素打分机制进一步进行筛选，这 7 个因素主要来自于股票的三个方面：

1.盈利性：(ROA 资产收益率; OCF 货运费用)

ROA(current)>0

ROA(current)>ROA(last year)

OCF(current)>0

OCF(current)>OCF(last year)

2.借债情况：(current_ratio 流动资产除流动负债的比率; debat to equal ratio 等比负债)

Current ratio(current)>current ratio(last year)

Debat to equity ratio(current)>dedat to equity ratio(last year)

3.运营情况：(gross margin 毛利率; asset turnover 资产周转率)

Gross margin(current) > gross margin(last year)

Asset turnover(current)>asset turnover(last year)

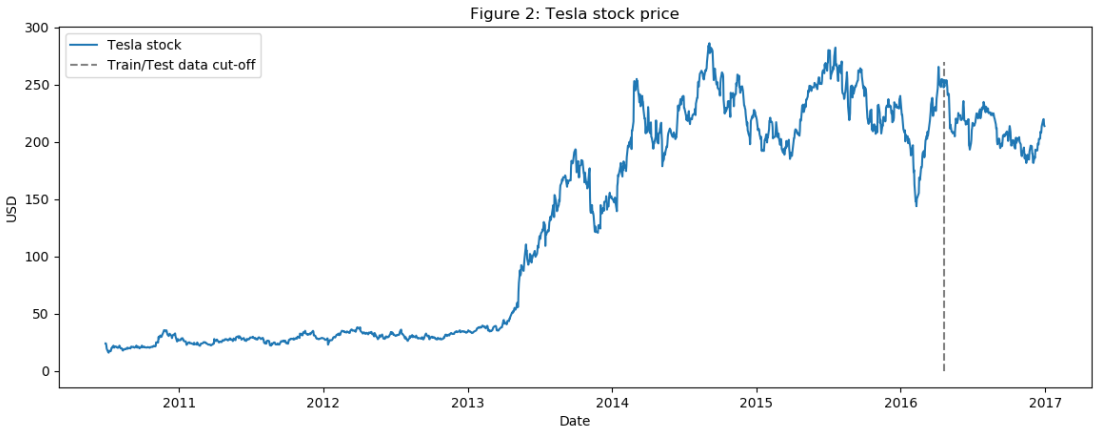
(2) 股价的预测：

对于单支股票，股价的波动近似于一个时间序列，按照我们前面介绍的

ARIMA 模型可以作较好的预测：

我们代入真实的股市行情对我们的操作方案进行测试。此处用的是 Python 编程语言，比如我们取特斯拉从 2010 年 6 月上市到 2017 年初的 1640 个数据进行测试。在这 1640 个数据中，我们采用前 80%数据预测后 20%的数据，结果如下所示：

原始数据如图：

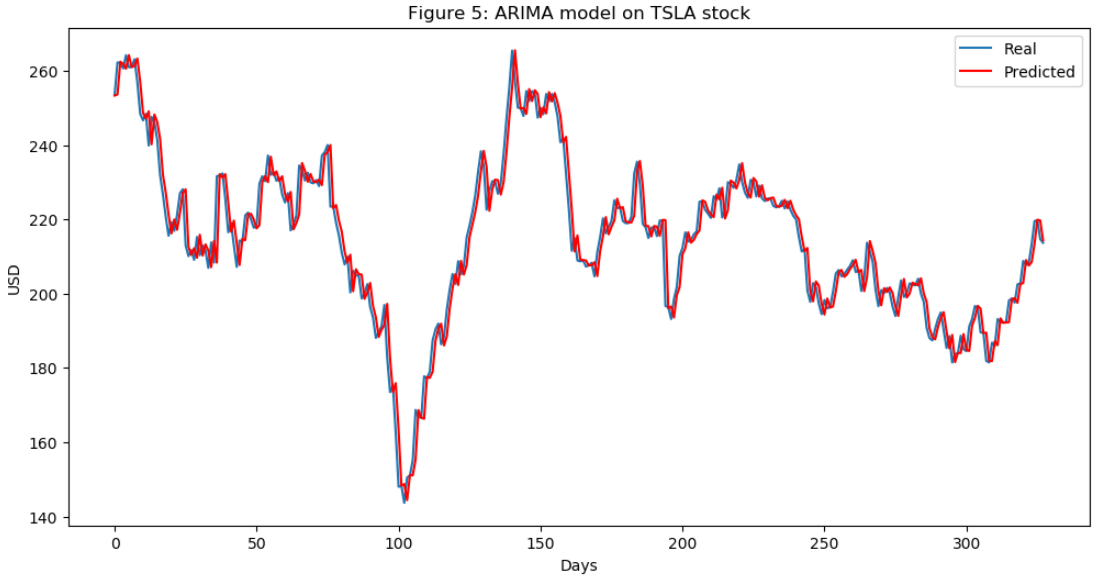


此时红线代表预测，蓝线代表初始数据。

核心代码如下：

```
X = series.values

print("All:",len(X))

size = int(len(X) * 0.8)

print("Train",size)

print("Test",len(X)-size)

train, test = X[0:size], X[size:len(X)]

history = [x for x in train]

predictions = list()

for t in range(len(test)):

    model = ARIMA(history, order=(5,1,0))

    model_fit = model.fit(disp=0)

    output = model_fit.forecast()
```

```
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs),
```



由上图，发现预测的曲线与真实的股价走势几乎重合，预测结果非常好，在没有大的突变的情况下，fft 以及 ARMA 模型对于数据的走势预测非常准确。

(3) 决策：

我们的决策方法如下：

我们的决策策略主要是组合策略，在得到收盘价的预测之后，我们考察未来各天的涨幅，若预测所得未来一天的收盘价的涨幅能够超过设定阈值就持有，否则就卖出，以这个策略我们可以得出买卖时机。在本文中，我们采用的是：

假设股价为 y，则我们如下计算涨幅：

$$r_{inc} = \frac{y_{new} - y_{old}}{y_{old}}$$

我们取涨幅的阈值为：$r = 0.006$,

若$r_{inc} \geq r$，则持有该股；

而若$r_{inc} < r$，则抛出该股，此处不考虑买卖时间。

此时我们已经得到了 16 年 4 月到 17 年 2 月的数据预测，但我们不可以直接采取高抛低买的策略，事实上，因为我们得到的预测数据实际上与前一天的数据有关，所以我们只有前一天才可以预测出后一天的数据，此为自回归的模型，我们无法决定买进的股票什么时候卖，或者卖出后什么时候买。

(4) 评估：

现我们计算收益率，以及风险评估。这一部分包括我们的各部分操作都使用 Python 编程语言实现。

具体的操作步骤如下：

假定买卖周期为 1 个月，在 1 个月内单支股票买入或卖出一次。而同时假设操作者实际由 1000000 现金进行操作。

1) 用 piotroski 算法，先确定 800 支待选股票，先使用上面(1)中的选股策略选择 20 支合适的股票买入。(指标的选取以及取值的详细过程在下面的代码和算法中有具体说明)。

2) 此时在在第一个月中考察这 20 支股票，使用前 10 年的股价数据以及 ARIMA 模型自回归预测每一天的走势，我们在一个月开始的一天买入 (即操作周期的第一天)，注意，在上面我们已经讨论了为了准确预测，实际上我们只用 ARIMA 拟合接下来 1 天的数据。在此处正好可以用阈值判断

卖出时机，若对于某支股票在本月中某一天由预测所得的后一天数据显示预测涨幅大于 r，则我们继续持有该支股票，否则抛出。

3） 由此我们进入下一个操作，再进入操作步骤(1)相同的步骤，使用 piotroski 算法选出 20 支认为趋势最好的股票，在重复(2)中的买卖操作。

由此，比如我们有了 08 年到 18 年的各支股票数据，我们可以计算 19 年的一年内的操作情况，同时计算年华收益率以及风险评估。具体的 Python 代码实现如下(预处理部分不再赘述):

```
def calc_piotroski_scores(stock,fundamentals,fundamental_old):


    return profit_score(stock,fundamentals,fundamental_old) + leverage_score(s
tock,fundamentals,fundamental_old) + operating_score(stock,fundamentals,funda
mental_old)



def profit_score(stock,fundamentals,fundamental_old):


    positive_roa = fundamentals[stock]['return_on_asset']>0

    positive_cash = fundamentals[stock]['cash_from_operating_activities']>0

    increase_roa = fundamentals[stock]['return_on_asset']>fundamental_old[sto
ck]['return_on_asset']

    increase_cash = fundamentals[stock]['return_on_asset']>fundamentals[stoc
k]['cash_from_operating_activities']


    return int(positive_roa)+int(positive_cash)+int(increase_roa)+int(increas
e_cash)


def leverage_score(stock,fundamentals,fundamental_old):


    debt_less = fundamentals[stock]['debt_to_equity_ratio']<fundamental_old[s
tock]['debt_to_equity_ratio']
```

```
    current_more = fundamentals[stock]['current_ratio']>fundamental_old[stoc
k]['current_ratio']


    return int(debt_less)+int(current_more)



def operating_score(stock,fundamentals,fundamental_old):


    gross_margin = fundamentals[stock]['gross_profit_margin']>fundamental_old
[stock]['gross_profit_margin']


    total_asset_turnover = fundamentals[stock]['total_asset_turnover']>fundam
ental_old[stock]['total_asset_turnover']


    return int(gross_margin)+int(total_asset_turnover)


def update_weights(context,stocks):
    if len(stocks) == 0:
        return 0
    else:


        weight = .95/len(stocks)
        return weight
```

我们的测试结果如下:

(最大回撤率：在选定周期内任一历史时点往后推，产品净值走到最低点时的收益率回撤幅度的最大值。最大回撤用于描述买入产品后可能出现的最糟糕的情况。最大回撤是一个重要的风险指标，对于对冲基金和数量化策略交易，该指标比波动率还重要。)

在基准年化收益只有 13.433%时，即股市行情不算特别好但比较稳定的情况下，按照我们的操作方案以及决策结果年化收益率大约 37.5%，相对于巴菲特的 20%左右的收益率，结果已经较好，远远强于基准年化收益。而且最大回撤 50.34%不算特别高，按照各种资料，这个值要尽可能小，所以我们的算法较为合理。由数据的结果可知，我们的策略可操作性较强，且年化收益率较好，同时风险较低。

模型的评价:

假定我们用 1000000 元投入，买卖的年化收益率大约是 37.5%。可以发现年化收益率非常高。但是本问题是理想化的，不考虑买卖差价与时间差。得到最终的预测以及收益率都非常好，模型建立较为合理，但是由于我们的模型

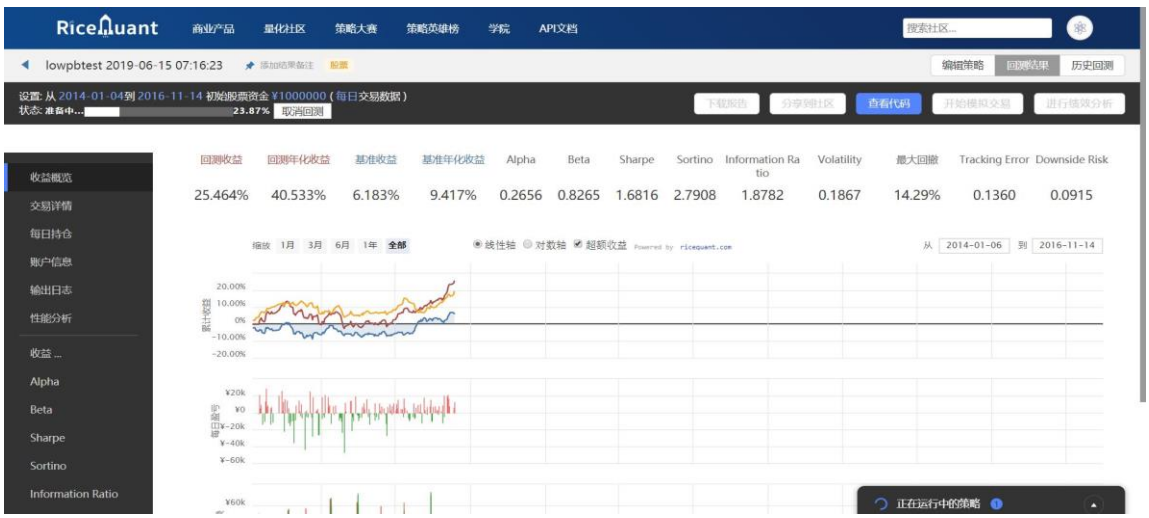
是理想化的，在实际的操作方案中可能还有各种影响因素，预测结果会差一点，操作中也会有各种阻碍，这些因素在模型中并未考虑进去。

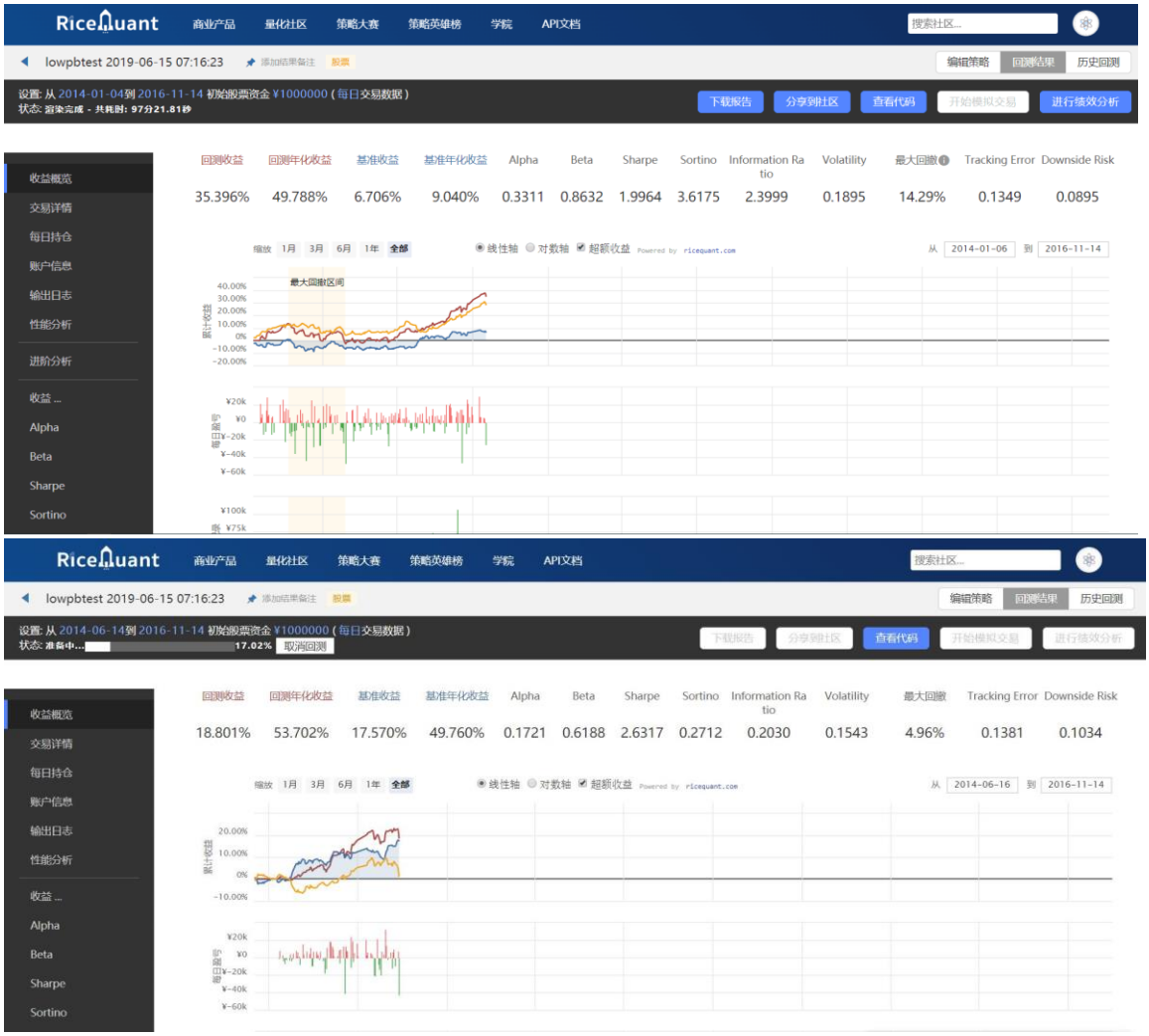
但是，事实上，对于阈值来决定买入和卖出的算法，由于未达到阈值的天数实际不唯一，而我们取的抛出的点是是首次未达阈值的点，但在后续抛出的过程中我们事实上可以选取不同的满足条件的抛售点，可能会使得收益率更高。

故我们对此操作方法改进如下：

对于预测的过程，我们仍然选取 10 年为测试数据，1 年时间作为决策测试，再选取一个月作为操作周期。

对于某个月，我们用自回归整合移动平均 ARIMA 的模型预测出 30 天的数据，测试我们在一个月的月初可以得到整个月股价波动的预测，按照此预测，我们只需要仍然按照月初买进，同时按预测所得的股价最高点抛出。若股价一直在跌，即股价最高点位于月初达到，就不选这支股票。则此时不用定义和求算阈值可以得到买入卖出时间。





下面为我们使用十年数据在网站上的结果，由上图第 2 幅图中我们可以看出在图中终止的时间点处，在基准年化收益率只有 9.040%，即股市行情不算特别好的情况下，我们的操作方案回测年化收益率为 49.788%，而最大回撤只有 14.29%，说明风险非常小。在第三幅图中，在基准年化收益率为 49.760%（牛市）时，我们得到的年化收益率为 53.702%，超过一半，且最大回撤只有 4.96%，说明该决策方案的风险非常小。所以改进后的算法相对于之前的算法有了极大的该进。但该操作方法在计算机的 Python 算法实现时所用的时间较长，所以我们在此处并未跑出在最后一天的平均年化收益率，但实际上我们观察行情可以看出股价上涨的高潮在后半部分 14 年以后，故我们跑完以

后我们可以得出的年化收益率是大于 50%的。

Python 被改动源码如下：

```
for stock in context.stocks:
       if stock in context.lastyearstocks:
          score = calc_piotroski_scores(stock,context.fundamental_df,context.
fundamental_old_df)
          if score >=7 and len(stocklist)<20:
              stocklist.append(stock)
              #只选 20 只 piotroski_scores>=7 的股票



   logger.info("candidate list")
   logger.info(context.stocks)
   logger.info("the selecting list")
   logger.info(stocklist)


   for stock in context.portfolio.positions:
       if stock not in stocklist:
          order_target_percent(stock, 0)


   max = {} #记录抛售股票的时间
   for stock in stocklist:
          logger.info("history")
          logger.info(history(300,'1d','close')[stock].values);
          his = history(300, '1d', 'close').fillna(0)[stock].values.tolist()
          flag = True
          for i in range(10):
              if his[-i] != his[-i-1]:
                  flag = False #奇异
                  break;
```

```
            if not(flag):
                max[stock] = 0; #如果 max=0 则不买入，如果 max>0 则在每天检查 max 卖出股
票
                stockmax = 0;
                for i in range(21):
                    model = ARIMA(his, order=(5,1,0))
                    model_fit = model.fit(disp=0)
                    output = model_fit.forecast()
                    his.append(output[0])
                    if output[0] > stockmax:
                        stockmax = output[0]
                        max[stock] = i
                if max[stock] ==0:
                    stocklist.remove(stock)#如果股票一直在跌，不保留股票
                    order_target_percent(stock,0)
            else: #不保留股票
                stocklist.remove(stock)
                order_target_percent(stock,0)



    weight = update_weights(context, stocklist)
    for stock in stocklist:
        if weight != 0:
            order_target_percent(stock,weight)
```

四 参考文献

[1]《数学模型》(第五版) 姜启源 谢金星 叶俊 编 高等教育出版社

[2]《算法导论》原书第三版 Thomas H.Cormen 等著，殷建平等译 机械工业出版社 2018 年 8 月第 1 版

[3]《数学模型》(第二版) 谭永基 蔡志杰 编著 上海 复旦大学出版社 2011 年 1 月出版

[4]优化建模与 LINDO/LINGO 软件 谢金星 北京 清华大学出版社 2005 年 7 月

[5]应用时间序列分析 何书元编著 北京 北京大学出版社 2003 年 9 月第 1 版

[6] http://cdmd.cnki.com.cn/Article/CDMD-10674-1014356483.htm 小波与神经网络

[7]基于 Piotroski 方法和 ARIMA-SVR 模型的股票投资策略研究 余天伦 2016 年 5 月

五 附录

完整的 Python 算法在附件的包中，直接在 http://www.RiceQuant.com 这一网站上运行上面的代码，可以直接得出年化收益结果。