基于资金流向的支持向量机策略

杨昀昶[1]

（西安邮电大学经济与管理学院,陕西省西安市 710121）

通信作者联系方式：（Email:yangyunchang001@gmail.com 电话：029-68801085 手机：18292577417）
投稿日期：2018-7-27,
作者简介：
杨昀昶（1997-），男，本科，从事量化投资与市场微观结构理论的研究。
Email:yangyunchang@gmail.com

摘 要 随着量化投资的快速发展，各种新兴的学科也开始在金融领域大放异彩。而量化投资研究中，一个核心的问题是，对股价的预判。如果能够对股价有效的预判，则投资者即可在投资中获得超额的收益。而股价又是一种极度不稳定，又难以预测的时间序列。因此，找到合理的影响股价波动的因素和适当的算法，就是量化投资急需解决的问题。

本文认为，资金的流入流出是影响股价短期波动的重要因素，因此，本文尝试对个股资金流向的建模，以提高股价预测的精度。而传统的时间序列分析方法中，又大多偏向于线性的分析方法，而忽略了股价波动的非线性因素。因此，本文利用模式识别中常用的一种感知器——支持向量机对资金流向进行建模，以预判下一交易日个股股价是上升还是下跌，并以此为依据进行股票交易。

本文首先介绍了策略概况；其次，以文献综述形式分析了国内外相关的研究进展；再次，介绍了模型基本背景和算法；最后，通过回测分析和性能分析来评价策略的各方面表现。

策略回测结果较好，但回撤幅度较大，本文认为与当时市场环境存在很大关系。策略 2010 年 1 月 4 日至 2018 年 7 月 30 日的回测结果为：总收益率 523.04%(对比沪深 300 指数-1.26%)，年化收益率 24.54%，夏普率 0.85，最大回撤 43.03%。
关键词 资金流 支持向量机 感知器 股价预测 量化投资

Abstract With the rapid development of quantitative investment, various emerging disciplines have begun to shine in the financial field. In the quantitative investment research, a core issue is the pre-judgment of stock prices. If the stock price can be effectively pre-judged, the investor can get excess income in the investment. The stock price is an extremely unstable and unpredictable time series. Therefore, finding a reasonable factor affecting stock price volatility and appropriate algorithms is a problem that quantitative investment needs to be solved urgently.

This paper believes that the inflow and outflow of funds is an important factor affecting the short-term fluctuation of stock prices. Therefore, this paper attempts to model the flow of individual stocks to improve the accuracy of stock price forecasting. In the traditional time series analysis method, most of them are biased towards linear analysis methods, and the

nonlinear factors of stock price fluctuation are ignored. Therefore, this paper uses a perceptron commonly used in pattern recognition, support vector machine to model the flow of funds, to predict whether the stock price of the next trading day will rise or fall, and use this as a basis for stock trading.

This paper first introduces the strategy overview. Secondly, it analyzes the relevant research progress at home and abroad in the form of literature review. Thirdly, it introduces the basic background and algorithm of the model. Finally, through back analysis and performance analysis to evaluate the performance of various aspects of the strategy.

Strategy backtesting better results, but the retracement by a big margin, we believe that there is a great relationship with the current market environment. The results of the strategy from January 4, 2010 to July 30, 2018 are: Total Return Rate is 523.04% (compared to CSI 300 index -1.26%), Annualized Rate of Return of 24.54%, Sharpe Ratio of 0.85, Maximum Retracement is 43.03%.

Key Word Cash flow; Support Vector Machine; Perceptron; Stock Price Forecast; Quantitative Trading

1 策略概述

1.1 研究目的

随着量化投资的发展，传统的量化择时和量化选股模型已经不能满足当前量化投资者对市场预测的精度需要。尤其是传统模型中，线性模型的广泛使用并没有起到明显优于技术分析和基本分析的预测效果。因此，非线性模型开始在量化投资领域逐步成为主流。近年来，随着人工智能、机器学习的发展，大量的非线性方法开始成为量化投资的主流。由于股票价格存在极强的非线性性、不稳定性、混沌性，因此机器学习方法可以较传统时间序列分析方法，更加准确的预测股票价格走势。

传统的机器学习理论主要是 BP 神经网络。但该算法是基于经验风险最小的原则，最终解依赖于初值，因此存在过度学习，过度拟合和局部最优解的问题。而且收敛速度慢，还存在隐网络节点个数难以确定的问题。[1,2]

而近年来，支持向量机(Support Vector Machine) 在理论研究和算法实现方面都取得了突破性进展,并成为克服 “维数灾难”和“过学习”等传统困难的有力手段。关于支持向量机在经济预测中特别是股票价格预测却是刚刚起步, 很多问题有待研究和探索。[3]

另一方面，自 1997 年以来，以莱昂为代表的市场微观结构理论迅速发展，为量化择时，尤其是高频交易提出了新的思路。莱昂认为市场中大型机构的交易行为，会反映出他们所掌握的私有信息，这些私有信息有一部分可能是内幕信息。[4]因此，通过对大型机构的交易行为的挖掘，有可能获得其中的内幕信息，并在半强式有效市场或弱有效市场中获取超额利润。现今，大量的研究认为中国股票市场处于半强式有效状态，也亦有分析认为是弱有效市场。因此，可以通过这一手段获取超额收益。大型机构的交易行为，只有通过他们在市场中发出的指令流的净额才能反应出来。由于广发量化平台的技术水平限制，暂时没有提供指令流数据，本文采用资金流向指标作为指令流指标的替代指标。

1.2 理论基础

由于股票是一种资产。资产是一种权利，一单位的某种资产被视为在一定时期内获得一定数量的物品或货币的权利。而权利是多维的“束”。资产的权利束最终是由其收益向量表征的。

由于资产是在不确定性环境下的，因此其收益向量亦是不确定的。因此金融学中认为，资产的价格是与状态相关的，即在 t 时期买入单位资产，在 t+1 期，发生某种状态 s 时，资产最终收益才被确定。因此经济学和金融学中，对资产有如下定义：

定义 1.1 资产是状态相依的或有要求权。

具体的说：

定义 1.2 一单位某种资产，是一种在 t=1 时期、状态 s(s∈S, S 表示自然状态)发生时，获得物品、服务或货币 1 的一个数量r_s的权利。其权利束由收益向量 r 来表征。

由此可以看出，资产的最终收益是由其状态决定的，而金融学中状态的描述，是由信息集来完成的。即当有新的信息进入市场时，资产价格就会随之改变。

因此：

定理 1.1 状态是信息集的函数

$s = f(\aleph)$ $\aleph$*为当前时刻市场中全部信息*

但是，1965-1970 年，尤金·法玛提出了效率市场假说,即认为一个市场的效率性体现在其信息接受方式。法玛简单将其划分为三类：

1° 市场仅反应过去价格的信息

2° 市场反应所有公开信息

3° 市场反应全部信息

由于是不确定性的市场，因此：

定理 1.2 市场可以被描述为一个概率测度空间$(\Omega, \mathcal{F}, P)$。其中，$\Omega = \{\omega\}$是基本事件空间，$\mathcal{F}$是Ω的子集的 σ -代数，P 是概率

测度空间$(\Omega,\mathcal{F})$上的概率测度。

而市场有效假说被数学化的描述为：

定理 1.3 对市场上的任一金融资产 S，都有如下性质：对于一个无风险银行账户 B 和局部等价于 P 的概率测度 p，使得，该资产的价格运动过程是鞅过程：

$$\frac{S}{B}=\left(\frac{S_n}{B_n}\right) \text{ 是 p}-\text{鞅，且F}_x-\text{可测}$$

并且：

$$\hat{E}\left(\frac{S_{n+1}}{B_{n+1}}\middle|\mathcal{F}\right)=\frac{S_n}{B_n}$$

而 S/B 的鞅性对信息集F_1有效即为弱势有效市场，F_2为半强式有效，F_3为强式有效市场。

而由于中国市场是弱有效的，因此可以认为，市场价格对公开信息和内幕信息的反应并非是随机游走的，因此可以通过挖掘内幕信息来获取超额回报。

而本策略的主要思路就是通过对资金流向数据的分析，挖掘出私有信息，包括公开信息和内幕信息，以此为基础获利。因此，策略的基本思路是合理的。

1.3 策略主要思路和创新点

本策略的主要思路是对资金流向数据进行挖掘，以期通过支持向量机分析出其中存在的私有信息F_3，得出涨跌预测。

具体来说，本文首先考虑止损策略，当大盘(沪深 300 指数)的 20 日线死叉 30 日线时，清仓并保持空仓状态。而当大盘指数没有发出止损信号时，策略开始进行执行交易。

其次，在没有止损信号时，策略会先选择股票池。股票池是用来选择其中下一个交易日可能上升的股票的备选列表。策略会去除掉次新股、ST、退市整理板、创业板的股票。

再次，通过遍历的手段，对每一只股票的资金流向数据进行获取和数据清洗，以及对数据进行归一化处理。同时，为了便于训练，需要给股票每日的数据加标签。标签是为了模型学习历史数据用的。当第 i 个交易日证券 j 的涨幅超过 0.5%时，则标记该证券的 i-1 交易日的标签为 1，即为“上涨”；反之标记 0，即为“下跌”。

再次，对整理好的数据进行 SVM 训练，本文对每只股票过去 120 日的资金流向数据进行训练。并得出对下一交易日的预测。如果预测为涨，即 SVM 预测模型返回值“1”，并且该证券的 5 日均线金叉 10 日均线，则将该股票放入“买入池”中。反之，返回为 0 时，则卖出该证券，如果没有持有该证券，则不进行操作。

最后，买进按照等股票数的方式（即每种股票的持股数相等）买进“买入池”中全部股票。

策略流程图：

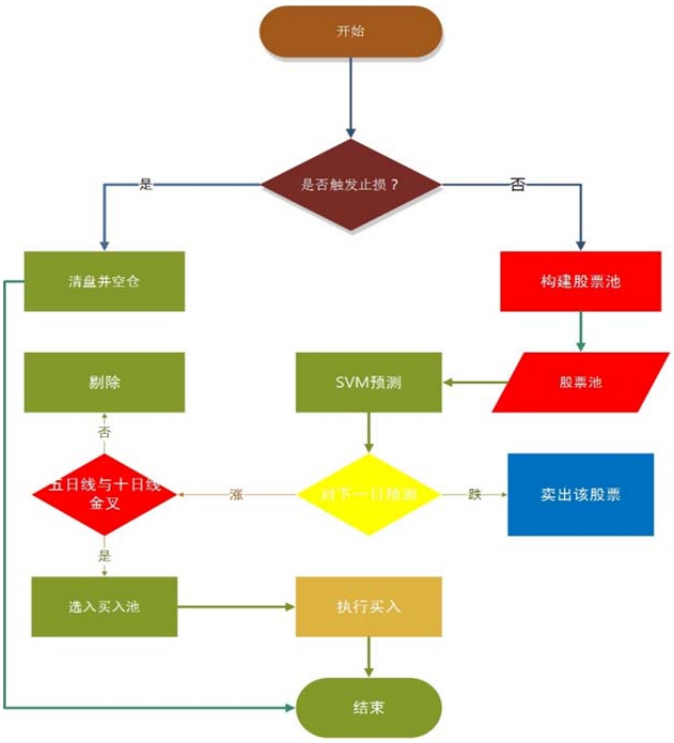

图 1.1 策略流程图

本文的创新之处在于，将资金流向与机器学习模型有机结合，可以获取到私有信息，大幅提高策略的收益率和夏普率。其次，本文通过引入大盘指数作为止损指标，克服了 SVM 模型在市场下跌时容易带来大幅回撤的缺点。最后，本文将均线系统引入预测系统，降低了单一使用 SVM 模型带来的预测失误的风险。

1.4 研究内容

本文主要内容安排如下：

第一章为策略概述。简单介绍资金流向

和 SVM 模型选股与择时的基本原理，并阐述了其策略思路的合理性，最后简单阐述了策略整体思想和整体流程，以及策略的创新点。

第二章为文献综述。通过回顾国内外研究人员利用资金流向和 SVM 模型进行股价预测的相关文献，整理和分析上述两种方法的内容及发展现状、相关概念及特点进行了简要的介绍、梳理现有的相关量化策略。

第三章为模型设定。本章通过对策略流程的详细介绍和模型构建，来呈现本策略的具体建模思路和交易流程。

第四章为回测与实证研究。本章通过对策略在广发量化平台上的回测以及相关性能分析，来分析策略的具体表现状况和程序源码设计的合理性。

第五章为结论。本章阐述了全文的结论以及策略的不足之处和未来完善的方向。

2 文献综述

2.1 SVM 模型在量化投资中的文献综述

Kim(2003)[6]利用 SVM 模型对韩国综合股价指数(KOSPI) 进行研究预测，结果表明 SVM 优于 BPN 和 CBR。HUANG Wei 等(2005)[7]提出结合多元分类的支持向量机模型，对 S&P 500 指数、日本 NIKKEI 225 指数进行研究分析，结果表明 SVM 是金融预测的有效工具，结合多元分类方法会提高预测性能。

彭丽芳等(2006) [8]利用基于时间序列的 SVM 股票预测方法，对 2002 年 3 月 14 日到 8 月 19 日的沙河股份数据进行研究分析，结果表明，与神经网络方法以及时间序列方法相比，SVM 的预测精度更高，在某些非线性时间序列的 预测中有很好的表现，解决了传统时间序列预测 模型无法解决的非线性问题。

林琦等(2010) [9]利用基于相空间重构的 LS-SVM 模型对股票价格进行预测，发现利用相空间重构对数据进行预处理和贝叶斯优化参数后，再用最小二乘支持向量机进行股价预测可以取得更有效的结果。

丁玲娟 (2012) [10]用小波分析对原始时间序列进行去噪，并对去噪后的序列进行小波分解，得到平稳的小波序列和非平稳的尺度序列,然后对平稳的小波序列建立 ARMA 模型进行预测，对非平稳的尺度序列建立 SVM 模型进行预测，最后整合得到对原始数据的预测结果。研究表明，组合模型完全达到了预想的高精度标准。近几年，投资者关注度和投资者情绪对股票市场的影响引起研究者的广泛关注。

Dzielinski (2011) [11]的研究表明，以搜索量指数度量的投资者关注与整个股票市场之间的收益和波动存在显著的相关性。Yu 等(2011)[12]发现在不同的波动率计算方法下，投资者情绪对风险收益关系的影响是一致的，在低情绪期，风险收益关系显著为正，而在高情绪期，风险收益关系会被削弱。

田鑫(2012)[13]利用神经网络模型和情绪指数对上证指数进行预测，研究结果证明应用神经网络对上证指数进行预测具有一定的有效性，加入构造的情绪指数能够显著提高模型的预测精度。

高大良 (2013)[14]的研究结果说明，投资者情绪对市场波动的总体影响是通过影响平均相关性和平均方差同时实现的，影响股票平均方差在总体影响中占据着主导地位，而投资者情绪对平均相关性的影响对总体影响起到一定的反向修正作用。

周胜臣等(2013)[15]基于微博搜索和 SVM 对股价进行预测，结果表明其构建的预测模型比传统的时间序列模型具有更好的预测性能和泛化能力。

Francis 和 Lijuan Cao（2003）[16]同时利用神经网络和支持向量机对单只股票的价格走势进行预测，从他们研究的实证结果可以看出，支持向量机的各项性能也是更为优异。

王彦峰等人（2006）[17]在对股价运用支持向量机进行预测时，结合了滚动时间窗口，预测结果令人满意。

阎纲（2008）[18]在对上证综合指数的走势预测中,将支持向量机和其他智能预测方法对比，发现支持向量机的预测准确率更高，

该研究同时指出基于结构风险最小化原则是支持向量机表现优于其他方法的重要原因。

宋绍峰（2010）[19]对支持向量机的预测性能研究是通过其在不同市场行情下的表现进行的，研究表明，支持向量机在趋势市中的预测准确性较强，在震荡市中的预测能力较弱。

金得宝（2010）[20]为了提高支持向量机预测的准确率，对数据进行了模糊聚类处理，训练时通过各个类的数据进行训练，从而使得预测性能有所提升。

田静（2010）[21]对股票指数与个股的走势进行预测，也是同时利用神经网络和支持向量机，实证结果可以看出，支持向量机的各项性能更为优异。

黄朋朋（2010）[22]对于股市拐点的预测使用了支持向量机，得到了较好的预测结果。徐国祥，

杨振（2011）[23]对沪深 300 指数的相关特征指标运用了主成分分析法，同时结合了遗传算法进行参数寻优，取得了较好的预测结果。

2.2 资金流向的文献综述

中南大学周可峰基于 TopView 数据研究了机构投资者的资金流向及其持股比率与股价的波动性的影响 ，首次以完备交易数据分析了机构资金流对于股价的决定性作用以及相互作用关系。

Chan 和 Lakonishok (1995) [24]认为资金流向与历史收益率呈正相关关系。Kim (2000) [25]对资金流向数据进行了研究，发现订单的不平衡性与股票收益率具有负相性。

Bennett 和 Sia (2001) 利用 1997 年 6 月至 1998 年 6 月期间纽约证券交易所的 1622 家上市公司数据，发现不仅资金流向与同期收益率正相关，而且历史资金流向可以预测未来资金流向和收益率。

Campbell 和 Ramadorai (2009) [26]认为机构日订单流对股票未来收益有明显作用，特别是机构的委卖订单对股票收益的这种负面作用更为显著。

侯丽薇，谢赤 (2010) [27]通过研究 A 股市场分钟级高频交易数据分析资金流向和当期股票收益率以及未来股票收益率之间的关系，发现资金流向对股票的收益率有一定的预测作用。

王义和柳会 (2011)[28] 从目前中国股市市场主力 (机构投资者) 与个人投资者之间的博弈关系出发，研究资金流向对股价短期波动的机理，并总结提分析验证主力资金进出对沪深股指、板块指数以及个股走势的主导作用。

2.3 小结

综上所述，可以看出，资金流数据确实可以反映部分订单流所隐含的信息，且跟踪这些信息可以获得超额收益。而另一方面，SVM 模型在金融价格预测领域的进展则主要在对模型本身的优化、修改以及和其他时间序列分析模型组合，以获取更高的收益。但是，在机器学习中，特征工程才是其核心。因此，选择良好的特征有事半功倍的效果。但过往的文献中很少有利用不同特征作为输入值进行训练的文献，而与市场微观结构理论相关的机器学习类文献，国内更是没有，因此，本文开创性的通过引入不同的特征的方法来进行策略的构建。

3 模型设定

3.1 模型假定

假定 3.1 市场组合为全部 A 股和一个空集的并集，$\mathrm{M} = \{s_i \cup \emptyset,\ i = 1, 2, \ldots\}$

假设 3.2 市场组合的子集的集合的交集、并集、联集、差集运算均是封闭的。即市场集合是一个可测空间。

假设 3.3 市场是有摩擦的。买入时佣金万分之三，卖出时佣金万分之三加千分之一印花税, 每笔交易佣金最低扣 5 块钱。但不考虑冲击成本。

假设 3.4 所有证券的价格均是动态复权的。

假设 3.5 策略于每日 9 时整开始运行，不考虑时间复杂度。因此策略在 9 时 30 分

前便确定了当日的交易策略，并在开盘时执行该策略。

假设 3.6 不考虑分红和相关的税收。

3.2 止损设置

策略运行的第一步会判断当前是否符合止损条件，如果符合，则进行清仓操作并保持空仓状态，直到止损信号消失。

定义 3.1 止损条件

假设沪深300指数的过去若干个交易日的收盘价格为价格序列p_t，那么通过移动平均算法，可以计算出其过去 20 个交易日和 30 个交易日的平均价格，并分别记为 ma20 和 ma30。则止损条件是一个布尔值 stop，其函数形式为：

$$stop = \begin{cases} 0\,, ma20 > ma30 \\ 1\,, \mathrm{ma20} < \mathrm{ma30} \end{cases}$$

当 stop=1 时，则为止损信号，当 stop=0 时，则为正常，策略即执行下文的计算。

3.3 股票池的构建

首先，策略需要对市场组合进行粗选，过滤掉一些导致策略亏损概率较大的股票。

定义 3.2 股票池

股票池$S^1 = \{s_j \cup \emptyset, j = 1,2,\dots n\}$是一个证券的集合，它是市场组合的子集，即$S^1 \subset$ M。∃S中的元素 s 满足如下条件时，该一系列证券集合 S 中计数测度最大的一个集合即为股票池S^1：

1° 证券 s 的名称中不含 ST、*ST 和*的证券

2° 证券 s 不是创业板股票

3° 证券 s 上市时间超过 1 年

4° 证券 s 在策略运行当日开盘时没有出现涨停、跌停或停牌等无法交易的状况

股票池应满足如下性质：

性质 3.1 S^1的计数测度是存在的，且应严格大于 0。

性质 3.2 S^1的运算是完备的。

性质 3.3 S^1是时变的。

再策略确定股票池后，会遍历股票池中每一个证券元素，并对每一个元素进行择时判断，当元素满足择时判断条件时，该元素即为买入池的元素。下文会阐述对单个证券的分类算法。

3.4 对单个证券的择时判断

定义 3.3 择时判断函数 class

对于任意一支证券s_j，抽取该证券 s 对应的四个资金流向数据（下文的小节会阐述这些数据的性质）x1，x2，x3，x4，作为 SVM 模型（函数 SVM）的自变量，SVM 模型会返回一个布尔值 pre，当 pre=1 时，策略还会计算该证券 s 的 5 日均线 ma5 和 10 日均线 ma10，当 ma5 大于 ma10 时，该证券被放入买入池 B；若小于，则不放入。若 pre=0，则卖出该证券 s，若本身没有 s 的持仓，则不做任何动作。

$$class(s_j)\begin{cases} 放入买入池，pre = 1\ 且ma5 > ma10 \\ \emptyset, pre = 1\ 且ma5 < ma10 \\ 卖出，\ \mathrm{pre} = 0 \end{cases}$$

该函数是本策略的核心，它帮助策略构建出买入集，并在每日交易开始时买入该集合内的证券。

下面各小节将会详细阐述策略输入 SVM 模型的数据是什么样的，SVM 模型是如何训练并在新数据进入时如何做出预测的，最后，还将详细阐述策略是如何交易买入池的证券的。

3.4.1 资金流向数据

策略会提取股票池中每一个证券元素的资金流向数据，具体包括四种。它们将作为 SVM 模型的训练数据引入。

1° 前一日的主力净流入额 x1，主力净流入额被定义为大于 10 万股或 20 万元的所有成交单的金额净和

2° 前两日的主力净流入额 x2

3° 证券 s 前一日的涨跌幅 x3

4° 主力净占比，即主力净额/成交额 x4

3.4.2 数据清洗

接下来进入数据清洗的工作，由于机器学习算法的特性，如果不进行数据处理。其后期证券 s 提出的四个特征 x 极有可能超过原先训练集的最大最小值范围，而造成预测

失误。即模型给出的预测值依然是训练集内的，因此，要对数据进行清洗处理。

本文采用最简单的数据处理方法 Max-Min 方法

公理 3.1 Max-Min 方法

对于一个数据序列x_t，计算每一个数据在其最大值和最小值之间的差值的位置，即可得出一个取值介于 0 到 1 之间的序列y_t：

$$y_i = \frac{x_i - x_t^{min}}{x_t^{max} - x_t^{min}}$$

其中：

x_t^{min}——x_t中的最小值

x_t^{max}——x_t中的最大值

对 3.4.1 节中提到的四个数据分别做公理 3.1 的处理，即可得到处理后的数据，本文定义为 y。

3.4.3 标签

由于支持向量机属于监督学习算法，因此需要给予其一个目标，这个目标就是判定当新的数据进入后，模型能否得出正确的涨跌判断。

定义 3.4 标签

假设标签 label 为布尔值。对于证券 s，如果在某个交易日已经收盘后的时刻 t 时，其当日的涨幅 pct 大于 0.5%，则 label 值记为 1，否则即为 0

$$label = \begin{cases} 1, pct > 0.5\% \\ 0, \text{tx} \end{cases}$$

3.4.4 利用 SVM 模型进行分类

支持向量机算法由 Vapnik 等人提出，相关理论至今仍在不断的完善与发展，是一种主要运用于数据挖掘或机器学习的方法[29]。支持向量机算法在模型的复杂性与学习能力之间寻找平衡，弥补了传统神经网络学习算法多项不足，它在解决模式识别和回归问题时，性能优越[30]。

本文用最简单的方式解释 SVM 模型的输入与输出，后文会详细介绍 SVM 模型的原理和算法。

原理 3.1　SVM 模型的计算原理

在做了之前的数据准备后，SVM 模型的训练即可开始。由于 SVM 模型是一个典型的二分类模型，因此其最终预测值应为是或否这样的布尔值。

假设 SVM 的训练模型 SVM train 和预测模型 Model 均为一个黑箱。

对于一个证券 s，其输入集合 IN 为：$IN_t\{[y_1^t, y_2^t, y_3^t, y_4^t], label_t | s_j\}$，输出为一个训练好的分类函数（预测模型）。当一个新的特征向量$y^{new} = [y_1^{new}, y_2^{new}, y_3^{new}, y_4^{new}]$进入预测模型时，该模型会返回一个布尔值 pre。

$$IN_t \xrightarrow{SVM\ train} Model$$

$$y^{new} \xrightarrow{Model} pre$$

3.4.4.1 感知机

感知机是二分类的线性分类模型，输入为实例的特征向量，输出为实例的类别（取 +1 和-1）。感知机对应于输入空间中将实例划分为两类的分离超平面。

感知机旨在求出该超平面，为求得超平面导入了基于误分类的损失函数，利用梯度下降法对损失函数进行最优化。而支持向量机就是这样一种感知机，其算法的核心是损失函数最小化

模型 3.1 感知机模型

假设输入空间（特征向量）是 x 属于 n 维空间，输出空间为 Y 属于{-1，+1}①，输入 X 表示实例的特征向量，对应于输入空间的点，输出 Y 表示实例的类别，则由输入空间到输出空间的表达形式为：

$$\mathrm{f}(\mathbf{x}) = \mathrm{sign}(\boldsymbol{\omega}\mathbf{x} + \mathrm{b})$$

上面该函数称为感知机，其中 w，b 称为模型的参数，w 称为权值，b 称为偏置，w*x 表示为 w，x 的内积。这里的 sign 指符号函数。

如果将 sign 称之为激活函数的话，感知机与 logistic regression 的差别就是感知机激活函数是 sign，logistic regression 的激活函数是 sigmoid。

sign(x)将大于 0 的分为 1，小于 0 的分为-1；sigmoid 将大于 0.5 的分为 1，小于 0.5 的分为 0。因此 sign 又被称为单位阶跃函数，logistic regression 也被看作是一种概率估计。

①当然本策略的所有模型均设标签为{0,1}，这并不影响模型本身。

该感知机线性方程表示为：w*x+b=0,它的几何意义如下图所示：

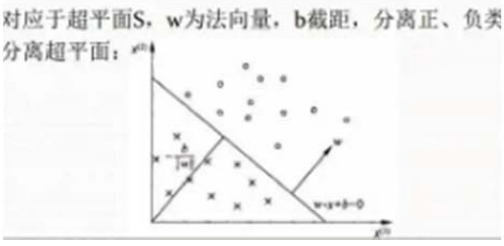

图 3.1：感知器方程的几何意义

模型其实就是在学习参数 w 与 b，确定了 w 与 b，图上的直线（高维空间下为超平面）也就确定了，那么以后来的数据点，策略就会用训练好的模型进行预测判断，如果大于 0 就分类到+1，如果小于 0 就分类到-1。(附录 1 将证明为何 w 是直线的法向量)

而如何保证经过模型训练，一定可以用一个超平面将两个凸集分到两边呢？

定理 3.1 超平面分离定理

设 f 是线性空间 X 上的线性泛函，c 为常数，则称线性流形$H_c^f = \{x|f(x) = c\}$为 X 的一个超平面。H_c^f应具有下文列出的性质。

在 X 上，存在两个不交的凸集 E,F，一定存在一个超平面H_c^f分离集合 E 和 F

$$\begin{cases} f(x) < c, \forall x \epsilon E \\ f(x) > c, \forall x \epsilon F \end{cases}$$

性质 3.4 $H_c^f = x_0 + H_0^f$，其中$f(x_0) = c$

性质 3.5

设$x_0 \in X \backslash H_0^f$，则$X = \text{span}\{x_0\} \oplus H_0^f$

性质 3.6 M 为 X 的真子空间，$x_0 \in X \backslash M$，

s.t. $X = \text{span}\{x_0\} \oplus M$

则存在 X 上的线性泛函 f，满足：

$$f(x_0) \neq 0, f(M) = 0$$

推论 3.1 M 为超平面是 M 为极大子空间的充分必要条件

推论 3.2 若 X 为B^*空间，则 f ∈ X是 M 闭的充分必要条件

例：如下图所示，在大于 0 的时候，我将数据点分类成了 D 类，在小于 0 的时候，我将数据点分类成了 C 类。

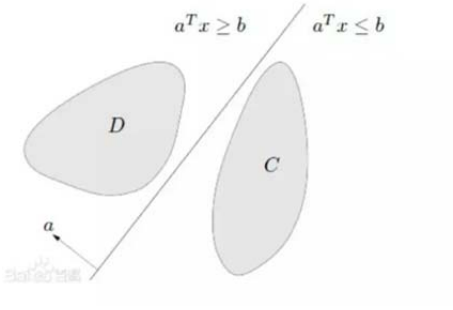

图 3.2 超平面分离定理示意图

下面关键是如何学习出超平面的参数 w，b，这就需要用到我们的学习策略。

众所周知，机器学习模型，需要首先找到损失函数，然后转化为最优化问题，用梯度下降等方法进行更新，最终学习到模型的参数 w，b。很自然的会想到用误分类点的数目来作为损失函数，使得误分类点个数越来越少，感知机本来也是做这种事的，只需要全部分对就好。但是不幸的是，这样的损失函数并不是 w，b 连续可导（你根本就无法用函数形式来表达出误分类点的个数），无法进行优化。于是我们想转为另一种选择，误分类点到超平面的总距离（直观来看，总距离越小越好）：

公式 3.1 距离公式

$$\frac{1}{\|\boldsymbol{\omega}\|}\left|\boldsymbol{\omega}^T \boldsymbol{x_0} + b\right|$$

而对于每一个误分类点都满足 -yi(w*x+b)>0。因为当数据点正确值为+1 的时候，模型误分类了，则模型的判断为-1，则算出来(w*x0+b)<0,所以满足-yi(w*x+b)>0，反之亦然。则可以将绝对值符号去掉，得到误分类点的距离为：

公式 3.2 误分点距离公式

$$\frac{-1}{\|\boldsymbol{\omega}\|} y_i(\boldsymbol{\omega}^T \boldsymbol{x_i} + b)$$

求和得到总距离公式：

公式 3.3 总距离公式

$$\frac{-1}{\|\boldsymbol{\omega}\|} \sum_{x_i \epsilon M} y_i \left(\boldsymbol{\omega}^T \boldsymbol{x_i} + b\right)$$

由于 w 范数分之一为常数不影响最优化，因此可以得到损失函数为：

公式 3.4 损失函数

$$L(\boldsymbol{\omega}, b) = -\sum_{x_i \epsilon M} y_i \left(\boldsymbol{\omega}^T \boldsymbol{x_i} + b\right)$$

只要找到合适的一组{ω, b}，使得 L 最小化，则超平面 wx+b 即为训练好的模型。当新数据进入时，只需要和 wx+b 进行比较，即可将其划分到对应的凸集中去，即返回涨或跌的信号。

由于求解该函数最优化需要将其转化为对偶的感知机，对偶形式就是将参数 w，b 表示为实例 xi 和标记 yi 的线性组合的形式，通过求解其系数而求得 w 和 b,对误分类点进行更新。因此在 Python 等语言中，也是通过赋予{$\boldsymbol{\omega}, b$}大量不同的取值的方法，找出其中使 L 最小化的一组值即为最优的{$\boldsymbol{\omega}, b$}*取值。

3.4.4.2 支持向量机

上一节，简单的说明了感知机的基本算法原理，然而支持向量机的具体算法又与传统的感知机有所不同。它的性质要更优于传统感知机

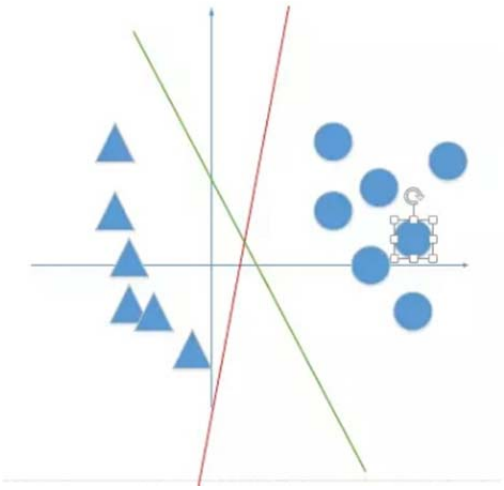

图 3.3 感知机和 SVM 的评价标准异同

在 SVM 的评价标准中绿线是要比红线好的，而传统感知机认为是一样好的。原因是绿线上的支持向量严格少于红线。

SVM 模型的基本原理是，将原有数据点映射到更高维的空间中，再利用支持向量最少的原理进行分类，可以更加简单的求出分隔面。

定理 3.2 高维可分定理

低维空间线性不可分的数据通过非线性映射到高维特征空间则可能线性可分。

假设一个数据集中有 m 个样本，其一定可以在 m−1 维空间中线性可分。

定义 3.5 支持向量

所谓支持向量，即在最优分隔超平面H_c^f一定距离 k 内出现的数据点 y。

$$y\epsilon\{x||f(x)| < c \pm k\}$$

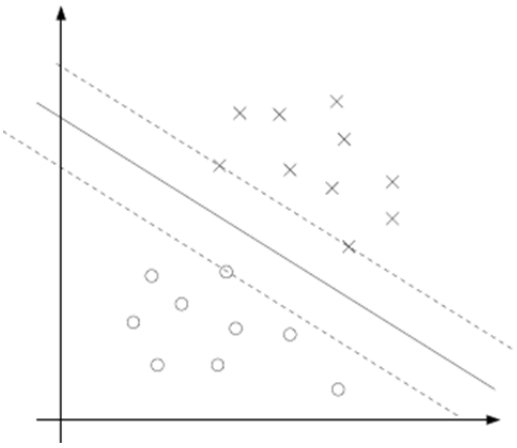

图 3.4 支持向量示意图（与虚线相交的即为支持向量）

支持向量即那些 SVM 分类过程中被认为“坏”的向量。即距离分离面太近的向量，若这些向量个数越少，则分类越可靠，错误分类越少。

首先，分析线性可分情况下的支持向量机。

与 3.4.4.1 中模型设定类似：

存在超平面：

$$\mathrm{f}(\mathbf{x}) = \mathrm{sign}(\boldsymbol{\omega}\mathbf{x} + \mathrm{b})$$

当 wx+b 为正时取 1，反之为-1，wx+b=0 时，函数为 0.

对于每一个点x_0，其到分隔面的距离公式为：

公式 3.1 距离公式

$$\frac{1}{\|\boldsymbol{\omega}\|}\left|\boldsymbol{\omega}^T \boldsymbol{x_0} + b\right|$$

而对于支持向量来说，其距离为 0 则分类间隔ρ：

公式 3.5 分类间隔

$$\rho = \frac{1}{\|\boldsymbol{\omega}\|} = -\frac{1}{\|\boldsymbol{\omega}\|} = \frac{2}{\|\boldsymbol{\omega}\|}$$

由于最佳分类平面的间隔最大，因此问

题转化为寻找合适的$\boldsymbol{\omega}$，使得$\rho = \frac{2}{\|\boldsymbol{\omega}\|}$最大。因此，该问题近一步转化为：

公式 3.6 最大距离问题

$$\text{Min } \frac{\|\boldsymbol{\omega}\|^2}{2} \quad \text{s.t. } y_i(\boldsymbol{\omega}^T\boldsymbol{x_i} + b) \geq 1$$

考虑到有噪声样本，因此引入松驰变量$\xi_i, i = 1,2,\ldots,n$ 予以解决。因此进一步演化为：

公式 3.7 引入松驰变量的最大距离问题

$$\text{Min } \frac{\|\boldsymbol{\omega}\|^2}{2} + C\sum \xi_i$$
$$\text{s.t. } y_i(\boldsymbol{\omega}^T\boldsymbol{x_i} + b) \geq 1 - \xi_i, \xi_i > 0$$

其中，使$\frac{\|\boldsymbol{\omega}\|^2}{2}$最小可以，提高泛化能力。而$C\sum\xi_i$能使误差尽量小。因此，对于 C：

定义 3.6 惩罚系数

C 是用于平衡正则化部分和经验风险部分的平衡系数。此参数被认为是惩罚参数，当 C 越大时，对错误分类的修正越大。

再利用拉格朗日最优化方法，将上述最优化问题转化为对偶问题。由于对偶问题的求解可以降低计算复杂度，因此，对于公式 3.7 中的约束条件将其改写为：

公式 3.8 对偶问题的约束条件

$$g_i(\boldsymbol{\omega}) = -y_i(\boldsymbol{\omega}^T\boldsymbol{x_i} + b) + 1 \leq 0$$

从 KKT 条件得知只有函数间隔是 1（离超平面最近的点）的线性约束式前面的系数$\alpha_i > 0$,对应的$g_i(\boldsymbol{\omega}) > 0$。对于其他的不在线上的点$g_i(\boldsymbol{\omega}) < 0$，有$\alpha_i = 0$，因为极值不会在他们所在的范围内取得。

注意图 3.4，实线是最大间隔超平面，假设×号的是正例，圆圈的是负例。在虚线上的点就是函数间隔是 1 的点，那么他们前面的系数$\alpha_i > 0$，其他点都是$\alpha_i = 0$。这三个点称作支持向量。构造拉格朗日函数如下：

公式 3.9 引入拉格朗日算子的最优化问题

$$\mathrm{L}(\boldsymbol{\omega}, \mathrm{b}, \alpha) = \frac{\|\boldsymbol{\omega}\|^2}{2} - \sum_{i=1}^{m}\alpha_i[y_i(\boldsymbol{\omega}^T\boldsymbol{x_i} + b) - 1]$$

注意到这里只有α_i,是因为原问题没有等式约束。

对于公式 3.9 的问题：

1° 首先求解$\mathrm{L}(\boldsymbol{\omega}, \mathrm{b}, \alpha)$的最小值，固定α，因此：

公式 3.10 最优化求解

$$\nabla_{\boldsymbol{\omega}} L(\boldsymbol{\omega}, b, \alpha) = \boldsymbol{\omega} - \sum_{i=1}^{m}\alpha_i y_i\boldsymbol{x_i} = 0$$

$$\frac{\partial}{\partial b}L(\boldsymbol{\omega}, b, \alpha) = \boldsymbol{\omega} - \sum_{i=1}^{m}\alpha_i y_i = 0$$

$$\therefore \boldsymbol{\omega} = \sum_{i=1}^{m}\alpha_i y_i\boldsymbol{x_i}$$

2° 将上式带回到拉格朗日函数中得到，此时得到的是该函数的最小值（目标函数是凸函数）。代入后，化简，最后得到：

公式 3.11 求解后的新最优化方程

$$\mathrm{L}(\boldsymbol{\omega}, \mathrm{b}, \alpha) = \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j\langle\boldsymbol{x_i}, \boldsymbol{x_j}\rangle$$

3° 进一步求解公式 3.11：

公式 3.12 新的拉格朗日方程

$$\max_{\alpha}\min_{\boldsymbol{\omega}} L(\boldsymbol{\omega}, b, \alpha)$$

$$\text{s.t. } \alpha_i \geq 0, \sum_{i=1}^{m}\alpha_i y_i = 0$$

前面提到过对偶问题和原问题满足的几个条件，首先由于目标函数和线性约束都是凸函数，而且这里不存在等式约束 h。$\exists\boldsymbol{\omega}$使得对于所有的 i，有$g_i(\boldsymbol{\omega}) < 0$。因此，一定存在$\{\boldsymbol{\omega}^*, \alpha^*\}$，使得$\boldsymbol{\omega}^*$是原问题的解，$\alpha^*$为对偶问题的解。随后，利用网格搜索法，找到最优的α，再根据公式 3.10，即可求出最优的 w，最后再根据下式计算出 b

公式 3.13 b 的求解

$$b^* = -\frac{\max\limits_{i:y_i=-1}\boldsymbol{\omega}^{T*}\boldsymbol{x_i} + \min\limits_{i:y_i=1}\boldsymbol{\omega}^{T*}\boldsymbol{x_i}}{2}$$

而对于线性不可分的情况与上文描述类似，只是要将分类超平面方程修改为：

公式 3.14 非线性超平面方程

$$\mathrm{f}(\mathbf{x}) = \mathrm{sign}(\sum\alpha_i^* y_i K(\boldsymbol{x_i}, y_i) + \mathrm{b}^*)$$

——$K(x_i, y_i)$为核函数，应满足 Mercer 条件。用核函数来免去高维变换，直接用低维度的参数带入核函数来等价计算高维度的向量的内积，这样可以避免出现“维数灾难”的问题。同时，核函数的引入，也使得 SVM 可以训练出非线性的分隔面。

即可。其后的运算步骤与上文公式 3.5~3.13 类似，这里不再展开。

由于股价呈现非线性的形态，因此求出最优的 a，b 即可得到分隔面，也即可对证券 s 进行涨跌预判。

下文简单介绍核函数。

概念 3.1 在机器学习中，核方法被形式化为特征空间的向量内积。

即有如下定义：

定义 3.7 核函数

设样本集 X(XϵR^k)，中有两条样本 x 和 z，非线性函数 φ 实现输入空间R^k到特征空间R^n希尔伯特空间的映射，k ≪ n，则：

$$\mathrm{K}(\mathrm{x},\mathrm{z}) = \langle \varphi(x), \varphi(z) \rangle$$

——K(x, z)为核函数

——⟨∗⟩为希尔伯特空间中的内积

那么，什么样的核函数形式是可以替代上述内积运算的呢？自此，引出如下定理：

定理 3.3 核函数有效性判定定理（Mercer 条件）

如果函数 K 是$R^n \times R^n \to R$上希尔伯特空间的映射，并且如果 K 是个有效的核函数，那么当且仅当对于训练集$\{x_i, i = 1,2,\dots\}$来说，其相应的核函数矩阵是半正定的。

简单的表述定理 3.3，即：

定理 3.3* K 是有效的核函数是核函数矩阵 K 半正定的充分必要条件。

而常用的核函数形式如下三种：

公式 3.15 常见核函数形式

1° 多项式核：$\mathrm{K}(\mathrm{x},\mathrm{y}) = \mathrm{x} \cdot \mathrm{y}$

2° RBF 核：$\mathrm{K}(\mathrm{x},\mathrm{y}) = \mathrm{e}^{-\gamma\|x-y\|^2}$

3° 傅立叶核：

$$\mathrm{K}(\mathrm{x},\mathrm{y}) = \frac{1-q^2}{2(1-2q\cos(x-y))+q^2}$$

本文选用 RBF 核，因为 RBF 核具有良好的性态，在实际应用中表现出了良好的性能。

3.4.4.3 支持向量机的参数说明

本小节说明策略使用 SVM 模型时，设定的参数。

参数名称	设定值
惩罚系数 C	1.0
核函数类型	RBF 核
核函数参数	不适用
核函数系数	自动调整
核函数独立项	不适用
概率估计	不启用
启发式收缩方式	是
误差精度 g	0.001
训练使用内存	200MB
多个惩罚系数	否
多线程运算	否
最大迭代次数	不设上限②
伪随机发生器	不启用

表 3.1 参数设定表

3.5 策略的执行

经过 3.4 节的全部运算后，即可得到一个买入池：

定义 3.8 买入池

买入池$\mathrm{B} = \{s_j, j = 1,2,\dots\}$是一个证券的集合，它是股票池的子集，即$\mathrm{B} \subset S^1$。∃S中的元素 s 满足 3.4 节中定义 3.3 的条件时，该一系列证券集合 S 中计数测度最大的一个集合即为买入池 B。

性质 3.7 B 的计数测度是存在的，且应大于等于 0。

性质 3.8 B的运算是完备的。

性质 3.9 B是时变的。

在得到买入池后，策略即执行等股票数的买入操作：

定义 3.9 买入操作

在某一时刻 t，策略经过计算并获取买入池$\mathrm{B} = \{s_j, j = 1,2,\dots n\}$后，立即获取当前剩余现金 Cash，并计算每支股票应买入的股份数 Share：

②已经证明（参见附录 2）SVM 模型是可以在有限次训练中收敛的，因此不必担心死循环状况。

$$\text{Share} = \frac{Cash}{\sum_{i=1}^{n} p(s_i)}$$

——p(s)是证券 s 的市场价格

买入池中每一支证券 s 都有相同的买入股份数。策略立即在开盘时按照每支证券 Share 股买进买入池的所有证券元素 s。

当策略执行完上述小节各个步骤后，一天的交易即完成。在下一个交易日 t+1 时，将从头执行本章全部步骤，并执行交易。

4 回测与实证研究

4.1 数据说明

本文的策略以中国内地所有 A 股为研究对象，选择 2010 年 1 月 4 日至 2018 年 7 月 30 日合共 2100 多个交易日的历史数据进行实证研究。数据来源是广发量化平台的 API 接口。

策略主要的变量选择：

变量名称	说明
输入变量	资金流向数据
输出变量	下一交易日的涨跌

表 4.1 策略主要使用的数据

本文的回测是采用滚动回测的方式。回测是模拟从 2010 年 1 月 4 日开始，每一个交易日都模拟运行一次。对于每一支股票 s，用其回测时间前 120 日的前一日的主力净流入额、前两日的主力净流入额、前一日的涨跌幅和主力净占比的数据作为特征，并按照定义 3.4 中方法计算出标签，作为训练集，以回测当日的四项数据为测试集，进行回测。因此，不存在引用未来数据的情况。

4.2 工具说明

本策略使用 Python 2.X/3.X 语言在广发量化平台上编译运行的（详细代码参见附录 3）。并且需要提前安装如下模块：

模块名称	说明
Numpy	NumPy 系统是 Python 的一种开源的数值计算扩展。
Pandas	pandas 是基于 NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。
Sklearn	基于 python 的机器学习模块
Jqdata	广发量化平台自带模块

表 4.2 Python 模块说明

4.3 策略归因分析

本文对单个指标的分析能力进行了分析，因此，本文首先做了单个指标的回测：

指标名称	准确度
前一日的主力净流入额	48.92%
前两日的主力净流入额	37.01%
前一日的涨跌幅	59.37%
主力净占比	46.91%

表 4.3 单个指标回测结果

从回测结果来看，单个指标回测结果尚可，可以认定，特征选取正确。

4.4 回测结果

本文采用广发量化平台进行回测。

假定交易成本只有买入时佣金万分之三，卖出时佣金万分之三加千分之一印花税，每笔交易佣金最低扣 5 块钱。但不考虑冲击成本。

主要结果：

项目	结果
总收益	523.04%
策略年化收益	24.54%
基准收益	-1.69%
Alpha	0.213
Beta	0.173
Sharpe	0.849
Sortino	1.108
Information Ratio	0.807
Algorithm	0.242

Volatility	
Benchmark Volatility	0.233
胜率	0.408
日胜率	0.487
盈亏比	2.534
盈利次数	102
亏损次数	148
最大回撤	43.025%
最大回撤出现时间	2015-06-02~2015-06-26

表 4.4 策略回测结果概况

从回测结果看，策略大幅跑赢市场，且夏普率为 0.85，也算是较高的水平。另外，策略的风险度较市场略高（策略波动率 0.242，对比市场 0.233），考虑到投资者是风险厌恶的，虽然策略承担了较高的风险，但是考虑到均衡市场下，定价是线性的，策略的收益要远远高于其承担的风险。策略的盈亏比为 2.534，但亏损次数多余盈利次数，说明策略的止损条件设定较好，在盈利时能较好的获利，而损失时，能较好的止损。最大回撤较大，但本文认为出现在特定时期，不代表策略止损条件或风险控制的设定存在严重漏洞。

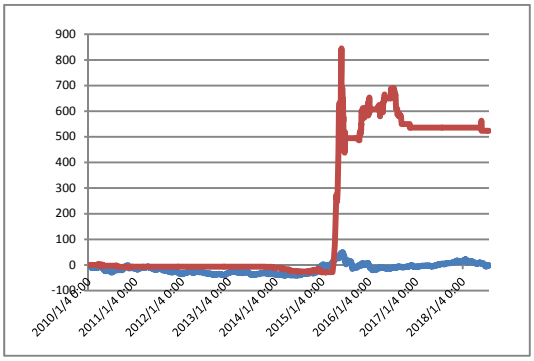

图 4.1 回测表现图（红色为策略，蓝色为市场）

由图中走势看出，策略在牛市表现较好，在熊市中，也能长期跑赢大盘。2015 年 6 月有大幅回撤，但是很快就收回了损失，在其他熊市周期中，策略回撤幅度也小于市场。因此，可以认为策略是可以穿越牛熊周期跑赢市场的。

4.5 性能分析

策略代码的时间复杂度消耗主要在一些获取数据，读取数据的步骤，以及 SVM 训练和预测的步骤。本文认为，可以通过硬件优化进一步提高算法性能。（性能分析具体内容参见附录 4）。但根据时间分析，策略运算时间远远小于 30 分钟，即可以在每日开盘交易前就做出交易决策，减少市场带来的冲击成本。

回测总计运行时间 438.0s，在可接受范围内。

4.6 交易成本分析

本策略的交易成本只有个股交易时的各种税费，详见有关规定。

另一方面，策略可能需要支付数据获取费用，因为使用的因子是需要专门计算或向服务商购买的。

再次，策略采用市价买入，市场摩擦应该不大，不存在冲击成本。

最后，如果出现股票开盘时直接涨停，有可能造成无法买入而导致部分资金闲置的成本。

5 结论

5.1 策略结论

策略明显跑赢大市，可以认为本策略是有效的，且可以获得长期复利增长。策略时间复杂度也在合理范围内，因此可以认为本策略可以在正常情况下完成交易。本文亦证明了支持向量机在股价预测领域有较高的预测准确率和强大的容错能力。

5.2 策略不足与展望

策略存在以下不足：

1.特征选择较为简单，容易被模仿或探测到，导致策略失效。

2.策略在 SVM 计算时，仅仅使用最传统的 SVM 模型。实际上，在 SVM 的大家族中

还有许多更新更好的模型，本文并未采用，或许效果更好。

3.止损设定过于简单，不能避免市场大幅度异常波动带来的损失。

4.采用均线系统辅助判断，精确度可能不够。

5.数据清洗过于简单，可能无法排除异常点，造成精度损失。

针对以上情况，策略将来需要做出如下修改：

1.利用自身对量化投资理解选取大量的可能的特征。再用决策树、GBDT 模型等方法递归消除来选择合适特征。或者，通过信息熵的计算选择可用的特征。再利用集成学习的方法进行精细选择。最后，对特征进行各种数学变换、处理（简单相加等），得到有用的特征。

2.修正 SVM 参数不断回测实验，得出最优参数。或是更换更深的网络进行机器学习，获得更高的准确率，利用 LSTM，GoogleNet，深度 Q 学习等方法。（由于当前广发平台不支持tensorflow模块，因此暂时无法修改）还可以，利用遗传算法将多个模型组合起来使用。亦或是采用并行算法。

3.设定更好的止损。例如利用隐马尔可夫链算法，预判当前市场处于牛市、熊市亦或是震荡市，当市场环境为震荡市或熊市时止损。

4.采用其他量化择时模型配合判断，例如牛熊线模型、Hurst 指数、分形原理等。

5.可以采用聚类模型对数据进行预处理，或是通过 One Class SVM 模型或 Isolation Forest 模型对离群点进行处理。另外，数据处理时，可以考虑采用小波分析过滤掉噪音序列，提高模型精度，获取更真实的私有信息。

参考文献

[1] ZangJia-shu, LiHeng-chao, XiaoXian-ci. ADCT domain quadratic predict or forreal-time prediction of continuous chaoticsignal [J] . Acta Physical Sinica, 2004, 53(3):710-716.

[2] LPMaguire, BRoche, TMM cginnity. Predicting achaotic time series using a fuzzy neural network [J] . Information Sciences, 1998, 112:125-136

[3] 徐维维,高风.灰色算法在股票价格预测中的应用[J] .计算机仿真, 2007, 24 (11):274-276

[4] Richard K. Lyons. Microstructure Approach to Exchange Rate [M]. 2001: 17-19

[5] Fama Eugene F. ,“ Market efficiency , long - term returns, and behavioral finance” , CRSP Working Paper, Graduate School of Business, University of Chicago 1997.

— 1970, “Efficient capital market: a review of theory and empirical work ', Journal of Finance , Vol. 25, No. 2, (May 1970) , pp383- 417.

— 1965, “The behavior of stock market prices,” Journal of Finance, (Jan. 1965) , pp70.

[6] KIM K J. Financial time series forecasting using support vector machines[J]. Neuro computing，2003，55 (1 -2) : 307 -319.

[7] HUANG Wei, NAKAMO RIY, WANG Shouyang. Forecasting stock market movement direction with support vector machine[J].Computers & Operations Research， 2005，32(10) : 2513-2522.

[8] 彭丽芳，孟志青，姜华. 基于时间序列的支持向量机在 股票预测中的应用 [J] .计算技术与自动化，2006(3) : 88 － 91.

[9] 林琦，吴少雄. 基于相空间重构的 LS-SVM 股票价格预测[J]. 福建工程学院学报，2010(3) : 300 － 303.

[10] 丁玲娟.基于小波分析和 ARMA-SVM 模型的股票指数预测分析[D]. 上海: 华东师范大学，2012.

[11] DZIELINSKI M. Measuring economic uncertainty and its impact on the stock market [J]. Finance Research Letters，2011，9(3) : 167 － 175.

[12] YU Jianfeng，YUAN Yu. Investor sentiment and the mean-variance relation [J]. Journal of Financial Economics，2011，100(2) : 367 － 381.

[13] 田鑫. 基于情绪指数和神经网络的上证指数预测研究 [D]. 哈尔滨: 哈尔滨工业大学，2012.

[14] 高大良. 投资者情绪及其对股票市场收益的影响研究 [D]. 长沙: 湖南大学，2013.

[15]周胜臣，施询之，瞿文婷等. 基于微博搜索和

SVM 的股市时间序列预测研究［J］. 计算机与现代化，2013（4）：22-26

[16] Francis E.H. Tay, Lixiang Shen and Lijuan Cao. Ordinary Shares, Exotic Methods Financial Forecasting Using Data Mining Techniques[M].World Scientific Publishing, 2003.

[17] 王彦峰,高风.基于支持向量机的股市预测[J].计算机仿真,2006,11:256-258,321.

[18] 阎纲.支持向量机在股市预测中的应用[J].科学技术与工程,2008,2:507-509,533.

[19] 宋绍峰.基于 SVM 的量化择时方法[EB/OL].量化投资报告,2010
-http://wenku.baidu.com/view/a7abb7d484254b35eefd34ae.html.

[20] 金得宝.基于支持向量机的股市预测研究[D].杭州:浙江大学,2010.

[21] 田静.基于支持向量回归机模型的股市预测研究[D].北京:北京交通大学,2010.

[22] 黄朋朋,韩伟力.基于支持向量机的股价反转点预测[J]. 计算机系统应用,2010,9:214-218.

[23] 徐国祥,杨振建.PCA-GA-SVM 模型的构建及应用研究——沪深 300 指数预测精度实证 分析[J].数量经济技术经济研究,2011,2:135-147.

[24] Chan K．and Lakonishok J．The Behavior of Stock Prices around Institutional Trades［J］. Journal of Finance，1995，1147 － 1174．

[25] Kim，Jung － Wook An Analysis of the Impact of Trades on Stock Returns Using Money Flow Data［J］. Harvard University Paper，2000．

[26] Campbell J． Y，Ramadorai T． and Schwartz A．Caught on Tape: Institutional Trading，Stock Returns and Earnings Announcements， Journal of Financial Economics，2009，66 － 91．

[27] 侯丽薇，谢赤，戴军，葛新元． A 股市场资金流向指标应用分析［J］. 证券市场导报，2010，No．22011: 72 － 77．

[28] 王义，柳会．基于资金流向分析的股票短期投资决策研究 ［J］. 中国证券期货，2011，09: 39 － 40．

[29] 吴亚军.基于非线性方法和 VaR 的均线交易系统研究[D].哈尔滨:哈尔滨工业大学, 2014

[30] 汤凌冰.机器学习在量化投资中的运用研究[M].北京:电子工业出版社,2014.

附录 1 ω是分隔面的法向量证明

证明 附.1

超平面方程为$\boldsymbol{\omega^T x} + b = 0$

设平面上任意两点x_1, x_2满足：

$$\begin{cases} \boldsymbol{\omega^T x_1} + b = 0 \ldots\ldots ① \\ \boldsymbol{\omega^T x_2} + b = 0 \ldots\ldots ② \end{cases}$$

用①-②，得：

$$\boldsymbol{\omega^T (x_1 - x_2)} = 0$$

$$\boldsymbol{x_1 - x_2 = x_1 x_2}$$

则$\boldsymbol{\omega^T}$是与$\boldsymbol{x_1 x_2}$垂直的。其他的点类似，则$\boldsymbol{\omega^T}$垂直于超平面上任意直线，则根据：

公理 附.1 欧式几何公理

一条直线垂直于一个面上的任意一条直线，则这条线垂直于这个面。

∴ $\boldsymbol{\omega^T}$是超平面的法向量。

证毕

附录 2 有限次迭代收敛证明

证明 附.2

将 b 并入权重向量 w，记作：$\hat{\boldsymbol{\omega}} = (\boldsymbol{\omega^T}, b)^T$，$\hat{\boldsymbol{x}} = (\boldsymbol{x^T}, 1)^T$，$\hat{\boldsymbol{x}}, \hat{\boldsymbol{\omega}} \in R^{n+1}, \hat{\boldsymbol{x}}\hat{\boldsymbol{\omega}} = \boldsymbol{\omega^T x} + b$

设训练集T $= \{(x_1, y_1), (x_2, y_2), \ldots\}$是线性可分的，则有如下定理成立：

定理 附.1 存在满足条件$\|\hat{\boldsymbol{\omega}}\| = 1$的超平面$\hat{\boldsymbol{x}}\hat{\boldsymbol{\omega}} = 0$，且存在γ > 0，对所有 i = 1,2,…,N，有：

$$y_i(\hat{\boldsymbol{x}}\hat{\boldsymbol{\omega}}) \geq \gamma$$

定理 附.2 令R $= \max_{1 \leq i \leq N} \|\hat{\boldsymbol{x}}_i\|$感知机算法在训练集的误分类次数（支持向量个数）k 满足不等式：

$$\mathrm{k} \leq \left(\frac{R}{\gamma}\right)^2$$

∴对于一个线性可分的数据集，感知机至少做 k 次迭代运算即可得到一个将数据集完美分隔的超平面，对于非线性可分集，亦有类似结论，证明思路类似。

证毕

证明 附.2.1 对于定理 附.1 的证明

在希尔伯特空间中，存在一个超平面$\hat{\boldsymbol{x}}\hat{\boldsymbol{\omega}} = 0$，使得$\|\hat{\boldsymbol{\omega}}\| = 1$，对于超平面中任意两点均有$y_1(\hat{\boldsymbol{x}}_1\hat{\boldsymbol{\omega}}) \geq 0$和$y_2(\hat{\boldsymbol{x}}_2\hat{\boldsymbol{\omega}}) \geq 0$,。那么∃γ $= \min_i\{y_1(\hat{\boldsymbol{x}}_1\hat{\boldsymbol{\omega}})\}$，则$y_1(\hat{\boldsymbol{x}}_1\hat{\boldsymbol{\omega}}) \geq \gamma$。

证毕

证明 附.2.2 对于定理 附.2 的证明

令$\hat{\boldsymbol{\omega}}_{k-1}$是第 k 个误分类实例前的扩元权值向量，即：

$$\hat{\boldsymbol{\omega}}_{k-1} = (\boldsymbol{\omega^T}_{\boldsymbol{k-1}}, b_{k-1})^T$$

第 k 个误分类实例的条件是：

$$y_i(\hat{\boldsymbol{x}}_i\hat{\boldsymbol{\omega}}_{k-1}) \leq 0$$

则，对 w，b 进行更新：

$$\widehat{\boldsymbol{\omega}}_k = \widehat{\boldsymbol{\omega}}_{k-1} + \eta y_i \widehat{\boldsymbol{x}}_i$$
$$b_k = b_{k-1} + \eta y_i$$

则根据：

(I) $\widehat{\boldsymbol{\omega}}_k \cdot \widehat{\boldsymbol{\omega}} \geq k\eta\gamma$

(II) $\|\widehat{\boldsymbol{\omega}}_k\|^2 \leq k\eta^2 R^2$

有：

$$\mathrm{k}\eta\gamma \leq \widehat{\boldsymbol{\omega}}_k \cdot \widehat{\boldsymbol{\omega}} \leq \|\widehat{\boldsymbol{\omega}}_k\| \cdot \|\widehat{\boldsymbol{\omega}}\| \leq \sqrt{k}\eta R$$

整理得：

$$\mathrm{k} \leq \left(\frac{R}{\gamma}\right)^2$$

证毕

证明 附.2.2.1 (I)的证明

$$\widehat{\boldsymbol{\omega}}_k \cdot \widehat{\boldsymbol{\omega}} = \widehat{\boldsymbol{\omega}}_{k-1} \cdot \widehat{\boldsymbol{\omega}} + \eta y_i \widehat{\boldsymbol{\omega}} \cdot \widehat{\boldsymbol{x}}_i \geq \widehat{\boldsymbol{\omega}}_{k-1} \cdot \widehat{\boldsymbol{\omega}} + \eta\gamma$$

则可以得到：

$$\widehat{\boldsymbol{\omega}}_k \cdot \widehat{\boldsymbol{\omega}} \geq \widehat{\boldsymbol{\omega}}_{k-1} \cdot \widehat{\boldsymbol{\omega}} + \eta\gamma \geq \widehat{\boldsymbol{\omega}}_{k-2} \cdot \widehat{\boldsymbol{\omega}} + 2\eta\gamma \geq \cdots \geq k\eta\gamma$$
$$\therefore \widehat{\boldsymbol{\omega}}_k \cdot \widehat{\boldsymbol{\omega}} \geq k\eta\gamma$$

证毕

证明 附.2.2.2 (II)的证明

已知：$\mathrm{R} = \max_{1 \leq i \leq N} \|\widehat{\boldsymbol{x}}_i\|$。证明：$\|\widehat{\boldsymbol{\omega}}_k\|^2 \leq k\eta^2 R^2$。

$$\begin{aligned}
\|\widehat{\boldsymbol{\omega}}_k\|^2 &= \|\widehat{\boldsymbol{\omega}}_{k-1}\|^2 + 2\eta y_i \widehat{\boldsymbol{\omega}}_{k-1} \widehat{\boldsymbol{x}}_i + \eta^2 \|\widehat{\boldsymbol{x}}_i\|^2 \\
&\leq \|\widehat{\boldsymbol{\omega}}_{k-1}\|^2 + \eta^2 \|\widehat{\boldsymbol{x}}_i\|^2 \\
&\leq \|\widehat{\boldsymbol{\omega}}_{k-1}\|^2 + \eta^2 R^2 \\
&\leq \|\widehat{\boldsymbol{\omega}}_{k-2}\|^2 + 2\eta^2 R^2 \\
&\cdots\cdots \\
&\leq k\eta^2 R^2
\end{aligned}$$

证毕

附录 3 策略代码

```
# 导入函数库
import jqdata
from sklearn import svm

# 初始化函数，设定基准等等
def initialize(context):
    # 设定沪深 300 作为基准
    set_benchmark('000300.XSHG')
    # 开启动态复权模式(真实价格)
    set_option('use_real_price', True)
    # 输出内容到日志 log.info()
    log.info('初始函数开始运行且全局只运行一次')
    # 过滤掉 order 系列 API 产生的比 error 级别低的 log
    # log.set_level('order', 'error')
```

```
    ### 股票相关设定 ###
    # 股票类每笔交易时的手续费是：买入时佣金万分之三，卖出时佣金万分之三加千分之一印花税, 每笔交易佣金最低扣 5 块钱
    set_order_cost(OrderCost(close_tax=0.001, open_commission=0.0003, close_commission=0.0003, min_commission=5), type='stock')

    ## 运行函数（reference_security 为运行时间的参考标的；传入的标的只做种类区分，因此传入'000300.XSHG'或'510300.XSHG'是一样的）
      # 开盘前运行
    run_daily(before_market_open, time='before_open', reference_security='000985.XSHG')
    # 初始化买入股票池
    g.rise = {'a':0}
    # 打开性能分析，以便于分析算法复杂度
    enable_profile()

## 导入资金流数据，并进行模型训练
def training(s,context):
    pred = 0
    # 获取某只股票的资金流数据
    moneyflow = jqdata.get_money_flow(s,end_date = context.current_dt.date(),count=120)

    # 数据检查
    if len(moneyflow)<25:
        return 0

    # 为标准化做准备
    # 引入主力净额和主力净占比，标准化是防止出现拟合失败的问题
    amax = moneyflow['net_amount_main'].max()
    amin = moneyflow['net_amount_main'].min()
    amean = moneyflow['net_amount_main'].mean()
    astd = moneyflow['net_amount_main'].std()

    pmax = moneyflow['net_pct_main'].max()
    pmin = moneyflow['net_pct_main'].min()
    pmean = moneyflow['net_pct_main'].mean()
    pstd = moneyflow['net_pct_main'].std()

    X = []
    Y = []

    i = 4

    # 利用 Max-Min 标准化后的的数据做训练
```

```
    while i < len(moneyflow) - 5:

        if moneyflow['change_pct'][i] > 0.5:
            Y.append(1)
        else:
            Y.append(0)

        x1 = (moneyflow['net_amount_main'][i-1] - amin) / (amax - amin)
        x2 = (moneyflow['net_amount_main'][i-2] - amin) / (amax - amin)
        x3 = moneyflow['change_pct'][i-1] / 10
        x4 = (moneyflow['net_pct_main'][i-1] - pmin) / (pmax - pmin)

        X.append([x1,x2,x3,x4])

        i = i + 1

    # 利用 sklearn 模块进行 SVM 模型训练
    clf = svm.SVC()
    clf.fit(X, Y)

    #预测
    x1 = (moneyflow['net_amount_main'][len(moneyflow)-1] - amin) / (amax - amin)
    x2 = (moneyflow['net_amount_main'][len(moneyflow)-2] - amin) / (amax - amin)
    x3 = moneyflow['change_pct'][len(moneyflow)-1] / 10
    x4 = (moneyflow['net_pct_main'][len(moneyflow)-1] - pmin) / (pmax - pmin)
    X1 = [x1,x2,x3,x4]

    pred = clf.predict(X1)

    return pred

## 构建股票池

# 过滤 ST、*ST 和退市整理板股票
def filter_paused_and_st_stock(stock_list):
    current_data = get_current_data()
    return [stock for stock in stock_list if not current_data[stock].paused
    and not current_data[stock].is_st and 'ST' not in current_data[stock].
    name and '*' not in current_data[stock].name and '退' not in current_data[stock].name]

# 过滤掉创业板股票
def filter_gem_stock(context, stock_list):
    return [stock for stock in stock_list    if stock[0:3] != '300']
```

```
# 过滤掉次新股
def filter_old_stock(context, stock_list):
    tmpList = []
    for stock in stock_list :
        days_public=(context.current_dt.date() - get_security_info(stock).start_date).days
        # 上市超过 1 年
        if days_public > 365:
            tmpList.append(stock)
    return tmpList

# 过滤涨跌停股票
def filter_limit_stock(context, data, stock_list):
    tmpList = []
    curr_data = get_current_data()
    for stock in stock_list:
        # 未涨停，也未跌停
        if curr_data[stock].low_limit < data[stock].close < curr_data[stock].high_limit:
            tmpList.append(stock)
    return tmpList

# 筛选出股票池
def select_stocks(context):
    # 选取流通市值小于 100 亿的 100 只股票
    q = query(valuation.code, valuation.circulating_market_cap).order_by(
            valuation.circulating_market_cap.asc()).filter(
            valuation.circulating_market_cap <=100).limit(50)
    df = get_fundamentals(q)
    stock_list = list(df['code'])

    # 过滤掉停牌的和 ST 的
    stock_list = filter_paused_and_st_stock(stock_list)
    #过滤掉创业板
    stock_list = filter_gem_stock(context, stock_list)
    # 过滤掉上市小于 1 年的
    stock_list = filter_old_stock(context, stock_list)
    # 选取前 N 只股票放入“目标池”
    stock_list = stock_list[:30]
    return stock_list

## 止损函数，当沪深 300 指数的 20 日线下穿 30 日线时，卖出所有股票
def stop(context):
    Price = attribute_history('000300.XSHG',60,'1d', ['close'])
    ma30 = Price['close'][-30:-1].mean()
    ma20 = Price['close'][-20:-1].mean()
```

```
    if ma20<ma30:
        return False;
    else:
        return True

## 交易
def before_market_open(context):

    # 清空字典，初始化买入股票池
    g.rise.clear()
    # 股票池
    g.df = select_stocks(context)
    # 如果止损信号发出
    if stop(context) == False:
        #卖掉所有股票
        for s in context.portfolio.positions:
            order_target(s,0)
    # 如果止损信号未发出
    elif stop(context) == True:
        for s in g.df:
            predict = training(s,context)
            if predict == 1:
                # 预测为涨，则计算某只股票的 5 日、10 日移动平均，金叉则放入买入
池
                Price = attribute_history(s,20,'1d', ['close'],skip_paused=False)
                ma5 = Price['close'][-5:-1].mean()
                ma10 = Price['close'][-10:-1].mean()
                if ma5 > ma10:
                    price = Price['close'][-1]
                    g.rise[s] = price
            # 如果预测为跌，卖出所持的该股票
            elif predict == 0 and context.portfolio.positions[s].closeable_amount > 0:
                # 全部卖出
                order_target(s, 0)
                # 记录这次卖出
                log.info("Selling %s" % (s))

        # 用所有可用 cash 买进买入池的股票
        cash = context.portfolio.available_cash
        total_price = sum(g.rise.values())
        volume = cash/total_price

        # 下单
        for s in g.rise.keys(): #s 是股票 list 的遍历
```

```
            order_target(s, volume)
            # 记录这次买入
            log.info("Buying %s" % (s))

    # 输出运行时间
    log.info('函数运行时间(before_market_open)：'+str(context.current_dt.time()))
```

附录 4 性能分析

Timer unit: 1e-06 s

Total time: 354.224 s
File: /tmp/codejail/71/128871/tmp3u6l6c/user_code.py
Function: training at line 29

```
Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    29                                           def training(s,context):
    30      6351       6558.0      1.0      0.0      pred = 0
    31                                               # 获取某只股票的资金流数据
    32      6351  270387502.0  42574.0     76.3      moneyflow = jqdata.get_money_flow(s,end_date = context.current_dt.date(),count=120)
    33
    34                                               # 数据检查
    35      6351      29545.0      4.7      0.0      if len(moneyflow)<25:
    36       118        103.0      0.9      0.0          return 0
    37
    38                                               # 为标准化做准备
    39                                               #引入主力净额和主力净占比，标准化是防止出现拟合失败的问题
    40      6233    1034086.0    165.9      0.3      amax = moneyflow['net_amount_main'].max()
    41      6233     281387.0     45.1      0.1      amin = moneyflow['net_amount_main'].min()
    42      6233     342477.0     54.9      0.1      amean = moneyflow['net_amount_main'].mean()
    43      6233     358105.0     57.5      0.1      astd = moneyflow['net_amount_main'].std()
    44
    45      6233     807417.0    129.5      0.2      pmax = moneyflow['net_pct_main'].max()
    46      6233     245339.0     39.4      0.1      pmin =
```

```
moneyflow['net_pct_main'].min()
    47          6233          303462          48.7          0.1          pmean =
moneyflow['net_pct_main'].mean()
    48          6233          333680          53.5          0.1          pstd =
moneyflow['net_pct_main'].std()
    49
    50      6233          7475      1.2       0.0      X = []
    51      6233          5338      0.9       0.0      Y = []
    52
    53      6233          5523      0.9       0.0      i = 4
    54
    55                                                 # 利用 Max-Min 标准化后的的
数据做训练
    56    652335        1653515     2.5       0.5      while i < len(moneyflow) - 5:
    57
    58     646102       11243729     17.4      3.2                if
moneyflow['change_pct'][i] > 0.5:
    59    285412        304761      1.1       0.1                Y.append(1)
    60                                                       else:
    61    360690        379169      1.1       0.1                Y.append(0)
    62
    63      646102        10625314        16.4        3.0              x1 =
(moneyflow['net_amount_main'][i-1] - amin) / (amax - amin)
    64      646102        10535313        16.3        3.0              x2 =
(moneyflow['net_amount_main'][i-2] - amin) / (amax - amin)
    65      646102        10617227        16.4        3.0              x3 =
moneyflow['change_pct'][i-1] / 10
    66      646102        10614673        16.4        3.0              x4 =
(moneyflow['net_pct_main'][i-1] - pmin) / (pmax - pmin)
    67
    68    646102        836166      1.3       0.2                X.append([x1,x2,x3,x4])
    69
    70    646102        573895      0.9       0.2                i = i + 1
    71
    72                                                 # 利用 sklearn 模块进行 SVM
模型训练
    73      6233          166261      26.7      0.0      clf = svm.SVC()
    74      6233          6406627     1027.9    1.8      clf.fit(X, Y)
    75
    76                                                 #预测
    77          6233          242317          38.9          0.1          x1 =
(moneyflow['net_amount_main'][len(moneyflow)-1] - amin) / (amax - amin)
    78          6233          140858          22.6          0.0          x2 =
(moneyflow['net_amount_main'][len(moneyflow)-2] - amin) / (amax - amin)
```

79 6233 127231 20.4 0.0 x3 = moneyflow['change_pct'][len(moneyflow)-1] / 10
80 6233 119605 19.2 0.0 x4 = (moneyflow['net_pct_main'][len(moneyflow)-1] - pmin) / (pmax - pmin)
81 6233 7844 1.3 0.0 X1 = [x1,x2,x3,x4]
82
83 6233 15472587 2482.4 4.4 pred = clf.predict(X1)
84
85 6233 8755 1.4 0.0 return pred

Total time: 9.06265 s
File: /tmp/codejail/71/128871/tmp3u6l6c/user_code.py
Function: filter_paused_and_st_stock at line 90

Line # Hits Time Per Hit % Time Line Contents
==
90 def filter_paused_and_st_stock(stock_list):
91 2082 133664 64.2 1.5 current_data = get_current_data()
92 106182 4518777 42.6 49.9 return [stock for stock in stock_list if not current_data[stock].paused
93 91035 3871582 42.5 42.7 and not current_data[stock].is_st and 'ST' not in current_data[stock].
94 90136 538631 6.0 5.9 name and '*' not in current_data[stock].name and '退' not in current_data[stock].name]

Total time: 0.06978 s
File: /tmp/codejail/71/128871/tmp3u6l6c/user_code.py
Function: filter_gem_stock at line 97

Line # Hits Time Per Hit % Time Line Contents
==
97 def filter_gem_stock(context, stock_list):
98 92098 69780 0.8 100.0 return [stock for stock in stock_list if stock[0:3] != '300']

Total time: 1.58651 s
File: /tmp/codejail/71/128871/tmp3u6l6c/user_code.py
Function: filter_old_stock at line 101

Line # Hits Time Per Hit % Time Line Contents
==

```
    101                                                 def  filter_old_stock(context,
stock_list):
    102         2082          1746         0.8         0.1         tmpList = []
    103        37316         25596         0.7         1.6         for stock in stock_list :
    104                 35234                 1520386                 43.2                 95.8
days_public=(context.current_dt.date() - get_security_info(stock).start_date).days
    105                                                     # 上市超过 1 年
    106        35234         25841         0.7         1.6             if days_public > 365:
    107        11864         11739         1.0         0.7                 tmpList.append(stock)
    108         2082          1207         0.6         0.1         return tmpList

Total time: 0 s
File: /tmp/codejail/71/128871/tmp3u6l6c/user_code.py
Function: filter_limit_stock at line 111

Line #          Hits              Time    Per Hit      % Time    Line Contents
==============================================================
    111                                             def  filter_limit_stock(context,  data,
stock_list):
    112                                                 tmpList = []
    113                                                 curr_data = get_current_data()
    114                                                 for stock in stock_list:
    115                                                     # 未涨停，也未跌停
    116                                                                                 if
curr_data[stock].low_limit < data[stock].close < curr_data[stock].high_limit:
    117
tmpList.append(stock)
    118                                                 return tmpList

Total time: 6.65145 s
File: /tmp/codejail/71/128871/tmp3u6l6c/user_code.py
Function: stop at line 122

Line #          Hits              Time    Per Hit      % Time    Line Contents
==============================================================
    122                                             def stop(context):
    123             3224           4415206       1369.5          66.4             Price   =
attribute_history('000300.XSHG',60,'1d', ['close'])
    124             3224           1371173        425.3          20.6             ma30   =
Price['close'][-30:-1].mean()
    125             3224            858000        266.1          12.9             ma20   =
Price['close'][-20:-1].mean()
    126         3224             4590         1.4         0.1         if ma20 ma10:
    174          399            23061        57.8         0.0                         price  =
```

Price['close'][-1]
175 399 843 2.1 0.0 g.rise[s] = price
176 # 如果预测为跌，卖出所持的该股票
177 5685 13342902 2347.0 3.0 elif predict == 0 and context.portfolio.positions[s].closeable_amount > 0:
178 # 全部卖出
179 72 624263 8670.3 0.1 order_target(s, 0)
180 # 记录这次卖出
181 72 233187 3238.7 0.1 log.info("Selling %s" % (s))
182
183 # 用所有可用 cash 买进买入池的股票
184 1142 50442 44.2 0.0 cash = context.portfolio.available_cash
185 1142 32422 28.4 0.0 total_price = sum(g.rise.values())
186 1142 2829 2.5 0.0 volume = cash/total_price
187
188 # 下单
189 1541 2764 1.8 0.0 for s in g.rise.keys(): #s 是股票 list 的遍历
190 399 3992473 10006.2 0.9 order_target(s, volume)
191 # 记录这次买入
192 399 1386193 3474.2 0.3 log.info("Buying %s" % (s))
193
194 # 输出运行时间
195 2082 5410223 2598.6 1.2 log.info('函数运行时间(before_market_open)：'+str(context.current_dt.time()))