2022 年第三届“大湾区杯”粤港澳
金融数学建模竞赛

一种高频交易的策略设计和异常交易识别模型

摘　要

高频交易（HFT）是利用计算机计算量大、反应迅速的特点，基于极短的市场变化而获利的一种电子化交易。本文针对中国股票市场，对不同股票采用不同的高频交易策略，针对性地提高收益，利用本模型解决特定股票组合的高频交易问题，同时加入异常事件检测系统，使模型可以适应不同的应用场景。

针对问题一，本文提取出频率、峰度[1]、收益差三个主要因子，以及波动股价绝对值、偏度[1]、方差三个次要因子，对其进行**熵权法-TOPSIS**[5]-三因子综合评价[2]，利用综合得分对 30 只股票进行排序，选出最适合进行高频交易的前十只股票为：300014、600872、000049、002352、002475、000513、002138、600325、600048、000028。

针对问题二，本文构建 CNN **神经网络分类模型**，对大盘蓝筹股票和中小盘震荡股票进行分类，进而分别对大小盘股票设计 CNN **预测模型**。根据预测股价结合当前股价作出买入卖出操作，交易量与预测股价和当前股价差相关，最大程度避免被其他高频交易者发现潜在买单和卖单，同时较经典网格交易策略的收益更高。

针对问题三，本文加入所要求的条件，根据问题一排序的股票，选择前两只股票：300014、600872，计算四只股票 601318、000333、300014、600872 的 21 天累计收益分别为 $2.1528e^6$ 、$8.4356e^6$ 、$4.2670e^7$ 、$3.8526e^6$，绘出**收益率曲线**，计算**最大回撤率**分别为：1.37%、2.16%、2.71%、2.82%。

针对问题四，本文设置涨停预警，确保异常涨停[6]后能快速处理持仓股票；针对大股东减持等已知事件，提前进行减持，减少股价大幅下跌带来的损失，改进后的平均收益率增加了 0.3683%。

本文使用的模型在经典模型和论文的基础上创新性地提出一些针对性的算法和模型[9]，适合一般的高频交易场景，鲁棒性高、结果合理，可结合高效拆单算法使用以满足不同用户，如不留隔夜仓等需求。

关键词：熵权法-TOPSIS　CNN 分类模型　最大回撤率　收益率曲线　CNN 预测模型

一、 问题重述

由于高频交易需要极快的反应速度，因此高频交易只能由计算机程序运行复杂的评估算法实现。

问题一要求仅针对中国市场，对 30 只股票进行分类选择，根据某些指标选出 10 只最适合高频交易的股票。

问题二要求针对不同股票的特点，建模并设计一种较经典高频交易策略更优秀的策略。

问题三要求利用问题二设计的高频交易策略，对持 400 万流动资金和 4 只各 150 万股票市值的某公司进行高频交易，并计算股票的收益率和绘出收益率曲线。其中，要求每日的持仓股票数量不变，且每天单只股票的高频交易量不得超过全天交易量的 10%。最后，按照 0.1%交易金额收取印花税和 0.015%交易金额收取佣金（不足 5 元按 5 元计算）。

问题四要求针对光大证券乌龙指事件和巴菲特减持比亚迪股票事件进行高频交易的改进，并分析改进效果。

二、 问题分析

2.1 问题一的分析

高频交易（HFT）是利用计算机计算量大、反应迅速的特点，基于极短的市场变化而获利的一种电子化交易。由于中国股票市场与国外市场存在交易方式不同、投机性大、涨跌率限制、交易单位不同等等区别，因此问题一要求针对中国市场的特点进行选股。由于高频交易对时间和价格变动较为敏感，因此应当选择与时间和涨幅相关的因子进行股票评估，再从中选择 10 只股票。

2.2 问题二的分析

由于不同股票的波动性有所不同，因此问题二要求针对股价走势稳健的蓝筹股票和波动明显的中小盘，设计出可适应不同波动性的股票的高频交易策略，面对波动剧烈的股票，高频交易策略应适当提高检测频率、减小预测时间；反之则反。此外，还要根据

2.3 问题三的分析

四只股票分别为中国平安、比亚迪、美的、光大证券，各市值为150万，由10月31日的的收盘价计算出各股票的数量，从而在收盘时保证股票总数不变。

由于高频交易股票数量不能超过全体成交量的10%，因此为了将高频交易的盈利最大化，需要利用好一个周期内的最大值和最小值，尽可能在这两点卖出买入，且交易的利润不及交易佣金和印花税时应当不进行交易，以达到盈利最大化的目的。最后按照一定比例缴纳佣金和印花税。

2.4 问题四的分析

针对光大证券乌龙指事件和大股东巴菲特减持比亚迪股票事件，要求高频交易策略能够判断出异常情况并立即停止交易或执行购入、抛售的程序以规避风险。本文利用BP神经网络对这两大类异常事件所导致股价快速上涨或快速下跌的程度进行分析，得到判断矩阵，从而对高频交易策略进行改进，比较改进前后的收益变化。

三、 模型假设

1. 除题目要求外不考虑黑天鹅事件。
2. 股票买卖立即完成，资金到账没有时延。
3. 计算高频交易策略的收益率时使用10月10日至10月11日的数据。
4. 由于中国股市实行T+1制度，但由于可以通过其他手段实现T+0制度，因此假设所有的股票都以T+0的方式交易。
5. 市场股价不会受本文使用高频交易的影响。
6. 假设高频交易有部分单无法完成交易，故在本文中的高频交易成交量计算中已经撤去无法完成的订单。
7. 假设粤港澳基金公司购买了交易所提供的L2行情信息，能够完全确保信息的完整和实时性。
8. 由于持股量较大，暂时不考虑其余高频交易散户竞争者，因此所得结果不考虑被其他高频交易竞争者抢先的情况。
9. 假设最后收盘时使用高效的拆单策略[4]、抛售时间相对总交易过程可以忽略不计。
10. 假设高频交易过程中未出现负收益大于流动资金的情况。

四、 符号说明

符号	说明	单位

W_j	第 j 只股票的收益差	¥
I_t	单周期股价增长量（Increase）	¥
D_t	单周期股价减小量（Decrease）	¥
P_t	t 时刻时的股价	¥
t_0	开盘时刻	s
t_1	收盘时刻	s
N	一日内发生股价增减的次数（波峰个数）	次
V_t	累积到 t 时刻的成交量（Volume）	股
P_t^{sum}	累积到 t 时刻的成交额	￥
Q_t	t 时刻的持仓股票	股
$RVar_j$	第 j 只股票的方差	
$RKurt_j$	第 j 只股票的峰度	
$freq_j$	第 j 只股票的频率	Hz
$RSkew_j$	第 j 只股票的偏度	
$AVSF_j$	第 j 只股票的股价波动绝对值	￥
ΔQ	持仓股票变化量	股
ΔP	股价变化量	￥

五、 模型的建立与求解

5.1 问题一模型的建立与求解

高频交易是利用极短的市场变化而获利的一种电子化交易。问题一要求针对中国市场的特点进行选股，由于高频交易关注一个极小的时间窗口内的股价变化，因此可见股价变化越剧烈、越频繁，高频交易相对于中低频交易的优势越大。为评估股票价格的波动性，本文选取了几类高频因子，并对其进行相关性筛选。

首先根据附件中的数据‘成交额’和‘成交量’，可以得到一小段时间内的平均股价

$$P_t = \frac{P_t^{sum} - P_{t-\Delta t}^{sum}}{V_t - V_{t-\Delta t}} \tag{1}$$

其中：P_t^{sum} 为累计到时刻 t 的成交额，V_t 为累计到时刻 t 的成交量

因为两数据的时间间隔短，因此可以将 t 时刻的股价以 Δt 内的平均股价代替。

5.1.1 数据预处理

由于 t 时刻的股价有超过最高价和最低价的异常值出现，因此本文在去除异常值

后使用平均插值法，保证数据样本大小一致。

5.1.2 收益差因子W_j

现定义单周期股价增量为：一次股价增减所消耗的间内的股价的增加量；单周期股价减量为：一次股价增减所消耗时间内的股价的减小量。定义如下图：

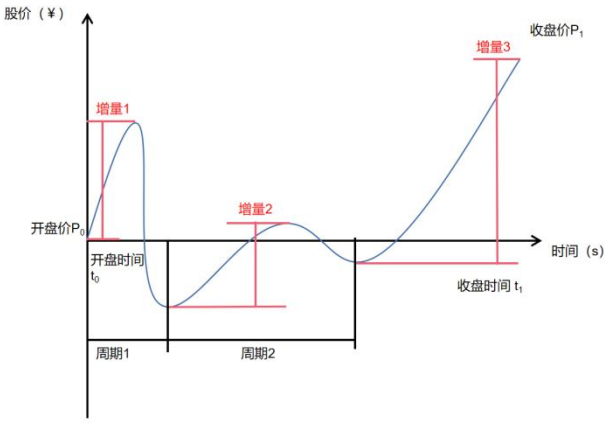

图 5.1 单周期内股价增量定义示意图

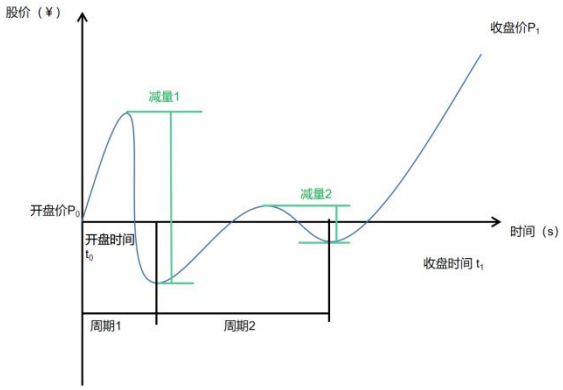

图 5.2 单周期内股价减量定义示意图

由于理想状态下的高频交易收益为所有小周期内的增量之和，中低频交易，特别是 T+1 制度下的交易理想收益往往是一天内收盘价与开盘价之差。因此定义利益差 W 为两者理想收益之差：

$$W_j = \left(\sum_{t=t_0}^{t_1} I_t\right) - \left(P_{t_1} - P_{t_0}\right) \tag{2}$$

其中：

I_t 当日内该股价在每一个小周期内的增长量之和

P_{t_0} 为开盘价

P_{t_1} 为收盘价

t_0 为开盘时刻 9：30

t_1 为收盘时刻 15：00

由公式可知当股票的波动性较小，或出现单边市场时，高频交易的优势无法体现，

甚至可能不及中低频交易（考虑到手续费），W 值变小；而当股市处于震荡期时，收盘价与开盘价相差不大而每个震荡内的股价增量大时，高频交易竞争力提高，W 值增大。

5.1.3 方差因子 $RVar_j$ [1]

方差反映了数据的偏移量

$$RVar_j = \sum_{t=t_0}^{t_1} \left(P_t - \overline{P}\right)^2 \tag{3}$$

其中：P_t 为 t 时刻的股价，$\overline{P}$ 为股价平均值

5.1.4 峰度因子 $RKurt_j$ [1]

峰度是用来衡量数据中极端值多少的重要指标

$$RKurt_j = \frac{(t_1 - t_0)\sum_{t=t_0}^{t_1} \left(P_t - \overline{P}\right)^4}{RVar} \tag{4}$$

其中，$RVar_j$ 是方差因子

5.1.5 频率因子 $freq_j$

频率反映了股价变动的快慢，变化越快则高频交易优势越明显

$$freq_j = \frac{N}{t_1 - t_0} \tag{5}$$

其中，N 为波峰个数，$t_1 - t_0$ 为一日内股市开放的时间。

5.1.6 偏度因子 $RSkew_j$ [1]

$$RSkew_j = \frac{\sqrt{(t_1 - t_0)}\sum_{t=t_0}^{t_1} \left(P_t - \overline{P}\right)^2}{RVar^{3/2}} \tag{6}$$

由于偏度是指数据分布的偏斜方向和程度，在本文中的含义是高股价和低股价的分布情况，左偏则低价股出现频率高，右偏同理，对高频交易无明显影响，作为无效因子剔除[1]。

5.1.7 波动股价绝对值因子 $AVSF_j$

$$AVSF_j = abs\left(\sum_{t=t_0}^{t_1} \left(I_t - D_t\right)\right) \tag{7}$$

其中：

$AVSF$ 为股价波动绝对值（Absolute value of stock fluctuation）

I_t 当日内该股价在每一个小周期内的增长量之和

D_t 当日内该股价在每一个小周期内的减小量之和

5.1.8 熵权法-TOPSIS 评估选股方法

首先获取 30 只股票的波动股价绝对值因子 AVSF，频率 freq，峰度 RKurt，方差 RVar，利用熵权法计算各项因子权重

构造评价矩阵 R 如图 5.3：

股票	X1(AVSF)	X2(freq)	X3(RKurt)	X4(RVar)	Xi(…)	Xm
Y1						
Y2						
Y3						
Y4						
Yj(…)						
Yn						

图 5.3 评价矩阵 R 示意图

再对 R 进行标准化处理，并更新矩阵 R：

上述四因子均为正指标，因此

$$r_{ij} = \frac{r_{ij} - Min_i\left(r_{ij}\right)}{Max_i\left(r_{ij}\right) - Min_i\left(r_{ij}\right)} + 0.0001 \tag{8}$$

其中：i 为因子序号，j 为股票序号，Min_i 函数取第 i 个因子中最小值，Max_i 函数取第 i 个因子中最大值。加 0.0001 是为了避免最小项为 0。

计算各因子熵值 H_i

$$\begin{cases} H_i = -\dfrac{1}{\ln(n)} \sum_{j=1}^{n} f_{ij} \ln\left(f_{ij}\right) \\ f_{ij} = \dfrac{r_{ij}}{\sum_{j=1}^{n} r_{ij}} \end{cases} \tag{9}$$

从而计算各因子熵权 w_i

$$w_i = \frac{1 - H_i}{\sum_{i=1}^{m}(1 - H_i)} \tag{10}$$

至此得到四个因子的权重分别为：W_{AVSF}=0.527367341014268、W_{freq}=0.140366987093279、W_{RKurt}=0.273805198652305、W_{RVar}=0.389888520428729

TOPSIS 对 R 进行标准化

$$r_{ij} = \frac{r_{ij}}{\sqrt{\sum_{j=1}^{n} r_{ij}^{\ 2}}} \tag{11}$$

由熵权法求出的权重构造加权规范阵

$$r_{ij} = w_j \cdot r_{ij} \tag{12}$$

计算解间距离 d_j^{-} 和 d_j^{+}，计算出第 j 只股票的得分 S_j

$$\begin{cases} r_i^{\min} = Min_i(r_{ij}) \\ r_i^{\max} = Max_i(r_{ij}) \\ d_j^{-} = \sqrt{\sum_{i=1}^{n}\left(r_{ij} - r_i^{\min}\right)^2} \\ d_j^{+} = \sqrt{\sum_{i=1}^{n}\left(r_{ij} - r_i^{\max}\right)^2} \\ S_j = \dfrac{d_j^{-}}{d_j^{-} + d_j^{+}} \end{cases} \tag{13}$$

利用每只股票的得分进行排序，结果如下：

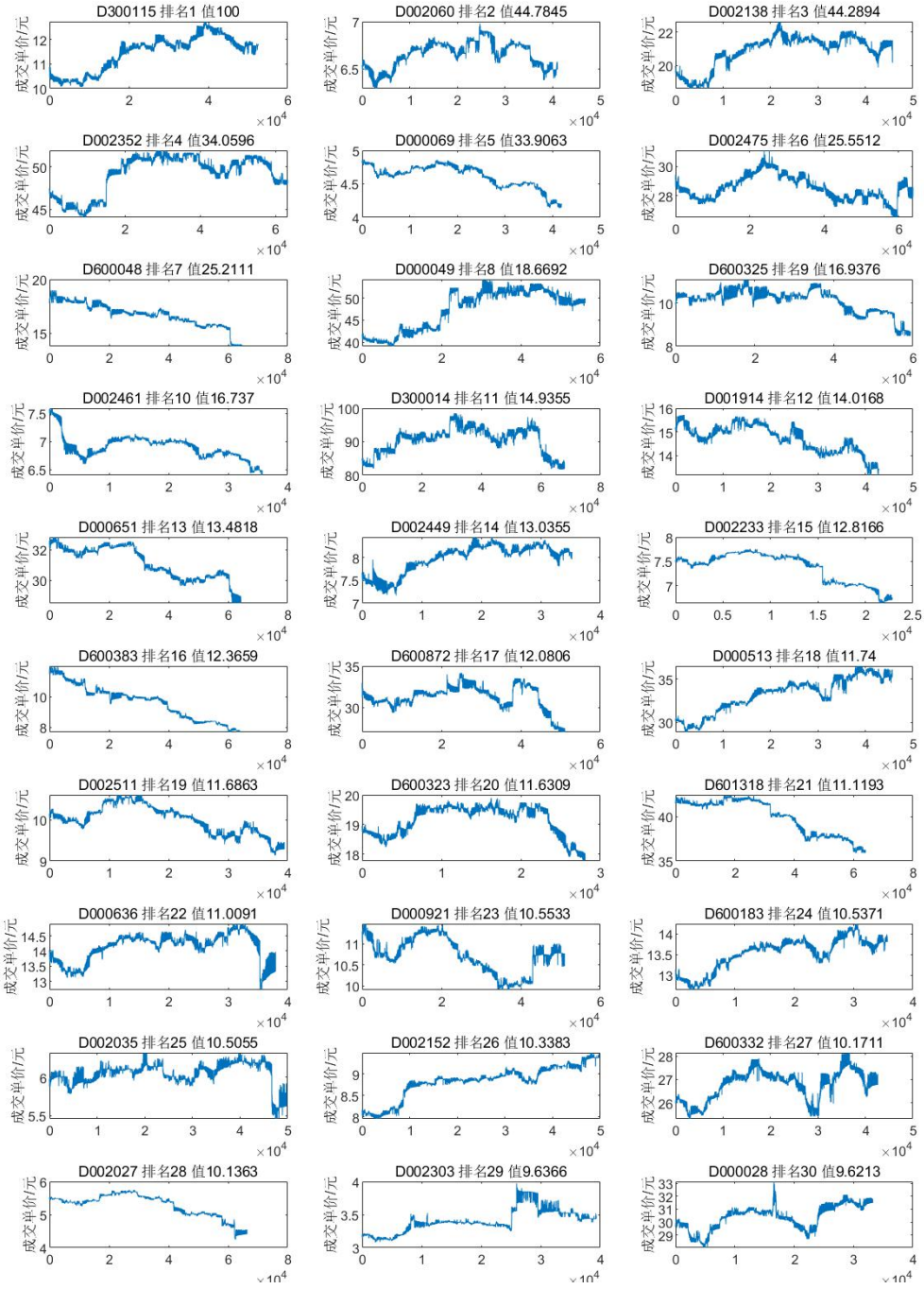

图 5.4 基于熵权法-TOPSIS 评估选股方法结果

5.1.9 收益差-频率-峰度三因子选股方法

定义第 j 只股票的高频适应度V_j为

$$V_j = W_j \cdot freq_j \cdot RKurt_j \tag{14}$$

利用各股票的高频适应度进行排序：

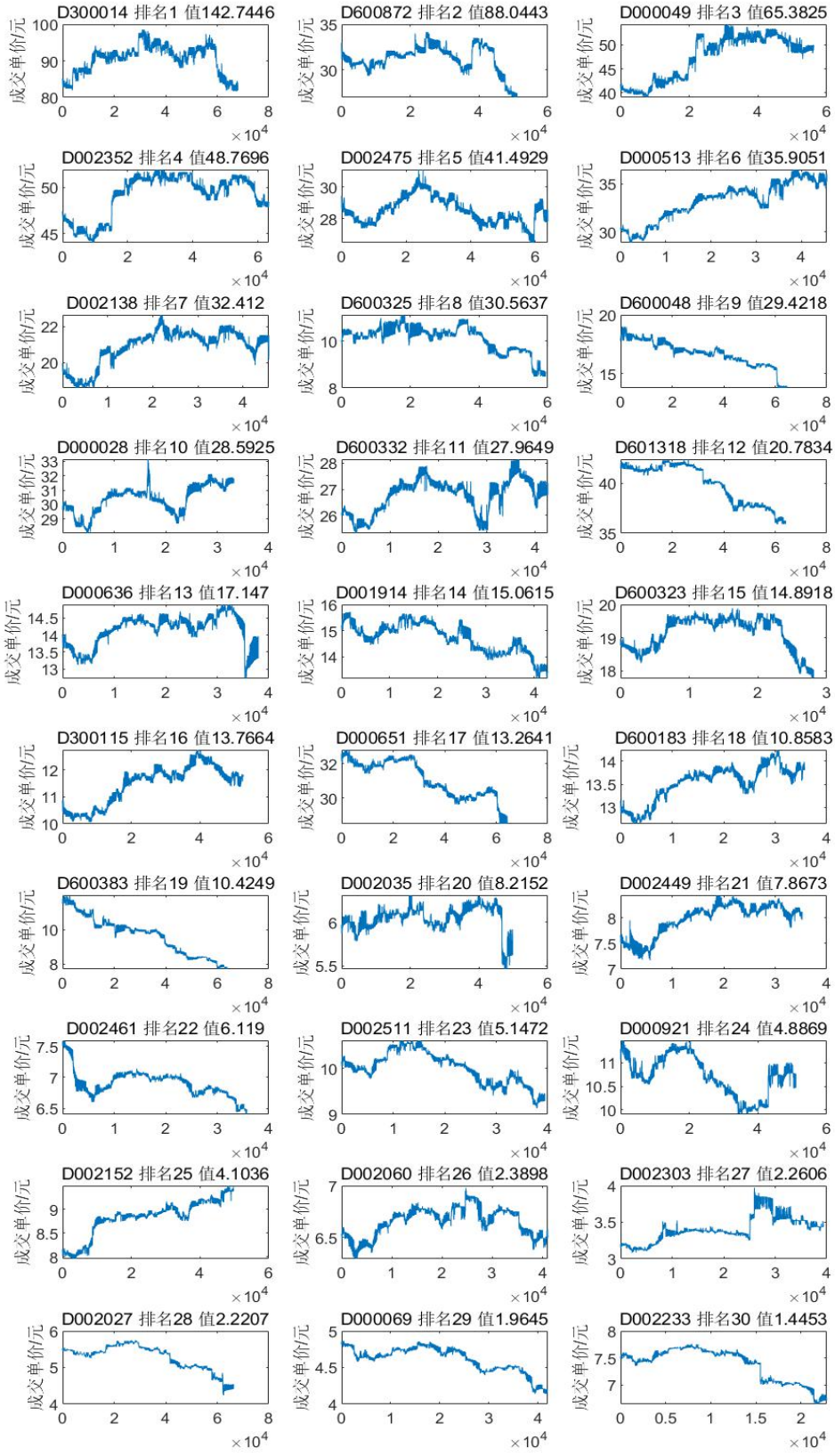

图 5.5 基于收益差-频率-峰度三因子选股方法结果

从以上两图可见，低频波动性小的股票 002060 利用熵权法-TOPSIS 评价法排到第 2

位，而在三因子模型中却排到了第 26 位；排在第一的波动剧烈的股票 300014 却在熵权法-TOPSIS 评价法中排在了第 11 位，可见熵权法只能利用信息的丰富度进行加权，对于真正有效的高频因子不能很好的析出，因此本文综合考虑主客观因素，采用

$$Score_j = 0.9 \cdot S_j + 0.1 \cdot V_j \tag{15}$$

即 0.9 倍收益差-频率-峰度三因子选股 +0.1 倍熵权-TOPSIS 进行排序选股，最终选出十只最适合进行高频交易的股票：300014、600872、000049、002352、002475、000513、002138、600325、600048、000028

5.2 问题二模型的建立与求解

由于问题二要求针对不同特点的股票，尤其是针对涨跌平缓的大盘蓝筹股票和波动显著的中小盘股票设计高频交易策略。因此本文根据市场公认大盘（中国平安 601318、中国石化 600028、万科 000002、工商银行 600028、宝钢股份 600019、招商银行 600036、中国铝业 601600、中远海控 601919）和中小盘（国电星光 002449、顺络电子 002138、广电运通 002152、华帝股份 002035、珠江啤酒 002461、美盈森 002303、顺丰控股 002352、粤水电 002060、塔牌集团 002233 、分众传媒 002027）作特征提取分类。其中大盘的数据来自同花顺软件 11.5-11.7 的全天真实数据。

5.2.1 高低频分类机

本文根据对大盘和中小盘股票进行分类，构建了一个 2 个隐藏层、每层 10 个神经元的 CNN 神经网络，输入层为特征因子：频率 freq 和收益差 W，输出 0 或 1，分别对应分为小盘和大盘的倾向。取大小盘股票 70%的数据作为训练集，15%作为测试集，15%作为验证集，得到 CNN 分类模型如图：

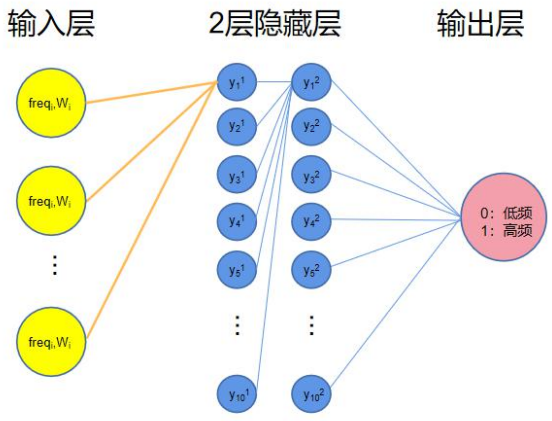

图 5.6 高低频分类机各层示意图

其误差如图：

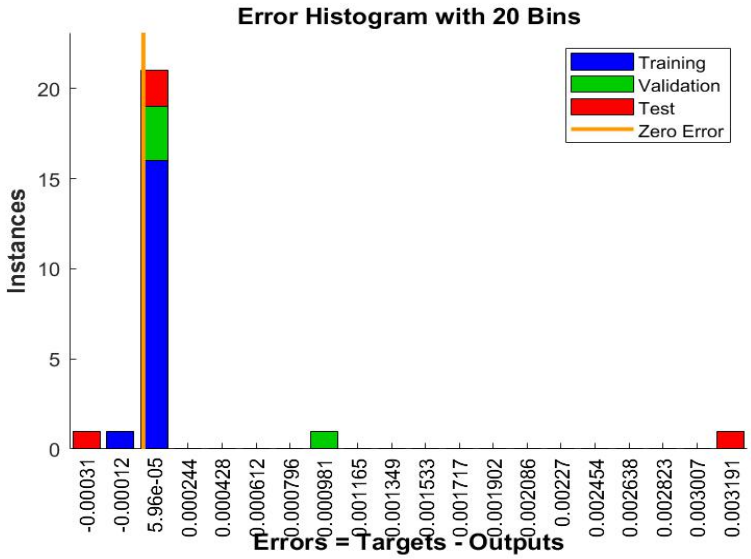

图 5.7 分类误差分布柱状图

5.2.2 股价预测系统

本文使用分类机分别对震荡股市和平缓股市建立两个股价预测模型。

本文构建了一个 2 个隐藏层、每层 10 个神经元的 BP 神经网络，输入层为当前时刻及此前 9 个时刻的股价共 10 个数据，输出层为下一时刻的股价，激活函数使用 sigmoid 函数，利用公式（16）更新权重矩阵：

$$\begin{cases} h_i^k = \sum_j^n w_{ij}^k \cdot x_j^{k-1} + b_{ij}^k \\ sigmoid(x) = \dfrac{1}{1+e^{-x}} \\ y_i^k = sigmoid\left(h_i^k\right) \end{cases} \tag{16}$$

其中 n 为第 k 层的神经元数量，则该 CNN 网络结构示意图如下：

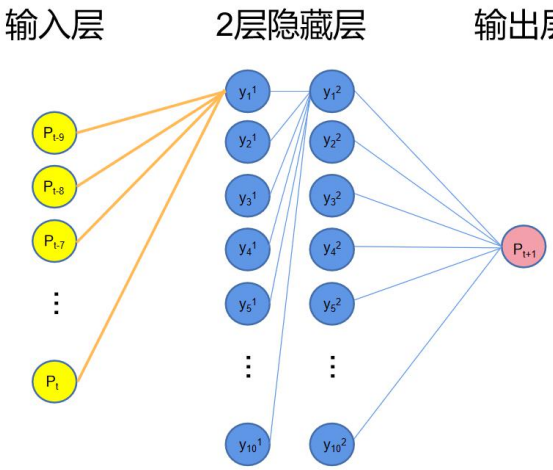

图 5.8 预测模型各层示意图

将 30 只股票 70%的数据用以训练，15%用作验证集，15%用作测试集。经过 44 次迭代，最终预测准确度为 0.99996，回归结果和误差结果如图：

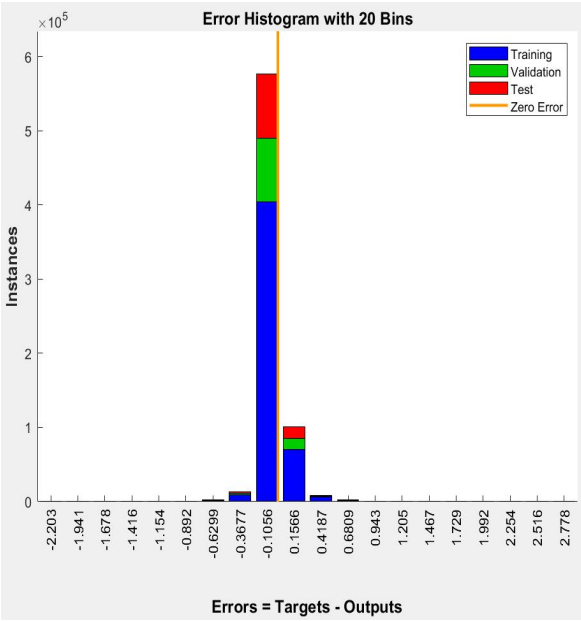

图 5.9 预测结果误差分布柱状图

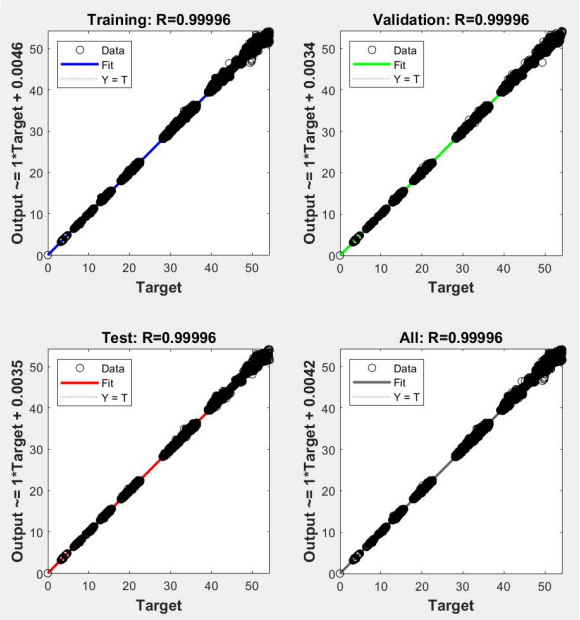

图 5.10 回归直线-散点图

5.2.3 詹姆斯-西蒙斯网格交易策略

基于詹姆斯提出的经典网格交易策略[9]，股价每上涨一个单位股价 ΔP 买入固定单数 ΔQ；每下降一个单位股价 ΔP 卖出固定单数 ΔQ，设经过一完整涨跌周期后，股价变回初始股价 P_0，网格交易策略操作流程如图，在红线标记处卖出，绿线标记处买入；橙线代表买入的股票在何时卖出，反映了其收益大小恰为一单位股价， ΔQ 设为定值

5 股。由于当今股票中高频交易的交易量已占市场成交量的 30%--70%，因此本文设置交易限额为$0.6\cdot(V_{t+1}-V_t)$，当高频交易挂单量超过该限额，则只有当前市场成交量数值大小的高频交易股票数量能够成交，其余挂单均被撤销。

$$\Delta Q_t = Min\left(5\ ,\ 0.6\cdot(V_{t+1}-V_t)\right) \tag{17}$$

经典网格交易策略的股票进出量与时间的关系如下图：

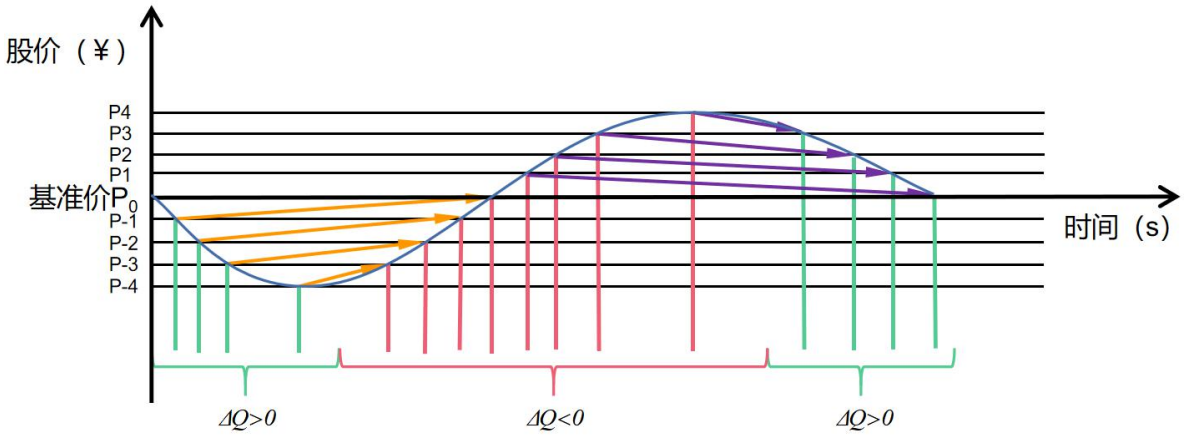

图 5.11 网格交易示意图

5.2.4 经典网格策略的收益率和平均成本计算

当预测系统判断下一时刻相较本时刻的股价有所下跌时，进行买入操作，并按照以下公式更新平均成本：

$$\overline{C_{t+1}} = \frac{\overline{C_t}\cdot Q_t + P_{t+1}\cdot \Delta Q}{Q_t + \Delta Q} \tag{18}$$

每波动一次，网格交易策略可赚取一单位股价的差价并从中获利。由收益率=收益/成本，可得每次卖出股票的收益率$Yield^{cla}{}_t$为

$$Yield^{cla}{}_t = \frac{\Delta P\cdot \Delta Q}{\overline{C_t}} \tag{19}$$

其中：$\overline{C_t}$为第 t 时刻持仓股票的平均成本

5.2.5 基于市场当前成交量计算交易单数

虽然网格策略简单、快速、无需预测、依靠周期性的涨跌便可从中获利，但受股票整体走势的影响，且单位股价极小，导致总收益较小，且容易被市场上其他高频交易竞争者发现规律，从而影响自己的，无法充分利用高频交易的优势。因此本文在网

格交易的基础上，加入神经网络预测系统，得到 t+1 时刻的股价 P_{t+1}，再与本时刻（t）的股价进行对比，得到股价差 ΔP。由于高频交易数量不宜过大，设置上下限为 1000 股，通过 ΔP 公式实时改变 ΔQ 表示如下：

$$\Delta Q = \frac{1000 \arctan(\Delta P)}{\pi/2} \tag{20}$$

其中：ΔQ 为持仓股票变化量，买进为正、卖出为负，ΔP 为下一时刻的股价与本时刻的股价之差。

上述公式说明：若股价有增长，且此时涨幅不明显，则购入少量股票，但若此时股价涨幅明显，则购入大量股票；若股票有减小的趋势，且此时跌幅不明显，则卖出少量股票，但若此时股价跌幅明显，则卖出大量股票。涨跌幅和股票进出量间的关系如图：

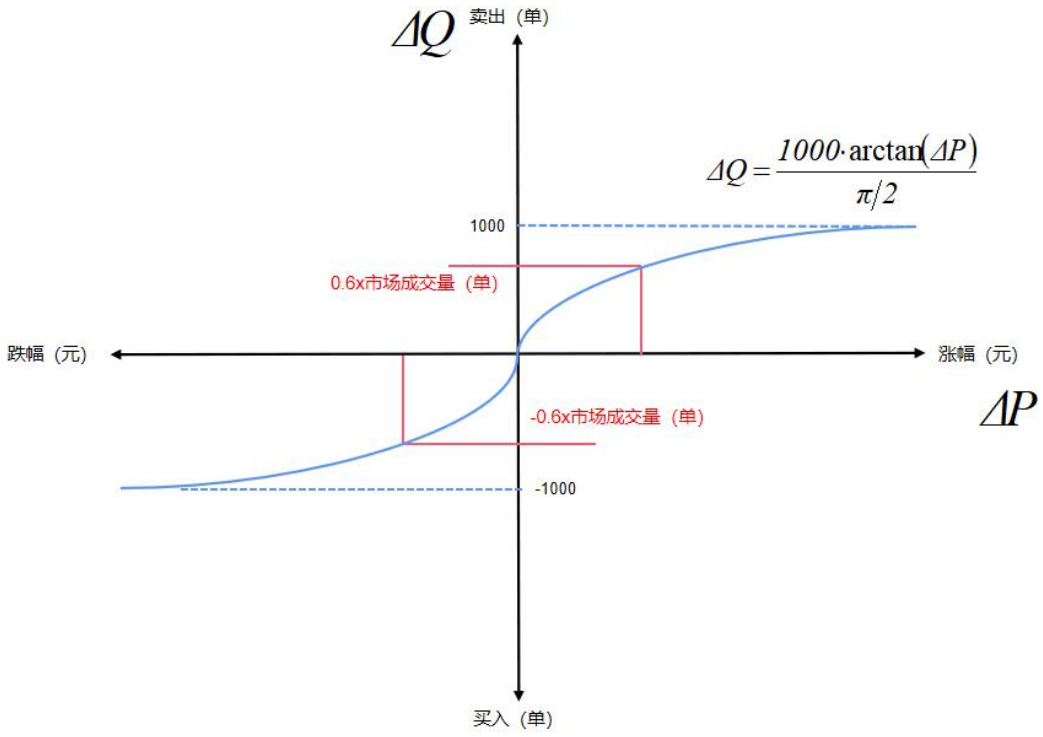

图 5.12 改进后的网格交易单数示意图

由于市场吞吐量有限，不可能成交所有挂单，由于当今股市的高频交易量达到了 30%-60%，因此设置吞吐量上限为 0.6 倍的当前市场成交量。

5.2.6 改进网格的收益率和平均成本计算

当预测系统判断下一时刻相较本时刻的股价有所下跌时，进行买入操作，并按照公式（16）更新平均成本。

当预测系统判断下一时刻相较本时刻的股价有所上涨时，进行卖出操作，可得 t 时刻的收益率 $Yield^{imp}{}_t$ 为：

$$Yield^{imp}{}_t = \frac{P_t - \overline{C_t}}{\overline{C_t}} \tag{21}$$

其中：P_{t+1} 为第 t+1 时刻购入股票的股价，ΔQ 为购入数量

特别地，当 $t=t_0$ 时，平均成本 $\overline{C_t}$ 为当前持仓股票的开盘价

5.2.7 最大回撤计算

回撤率是评估股票风险的重要指标，反映了未来可能的最大亏损。其计算流程为：

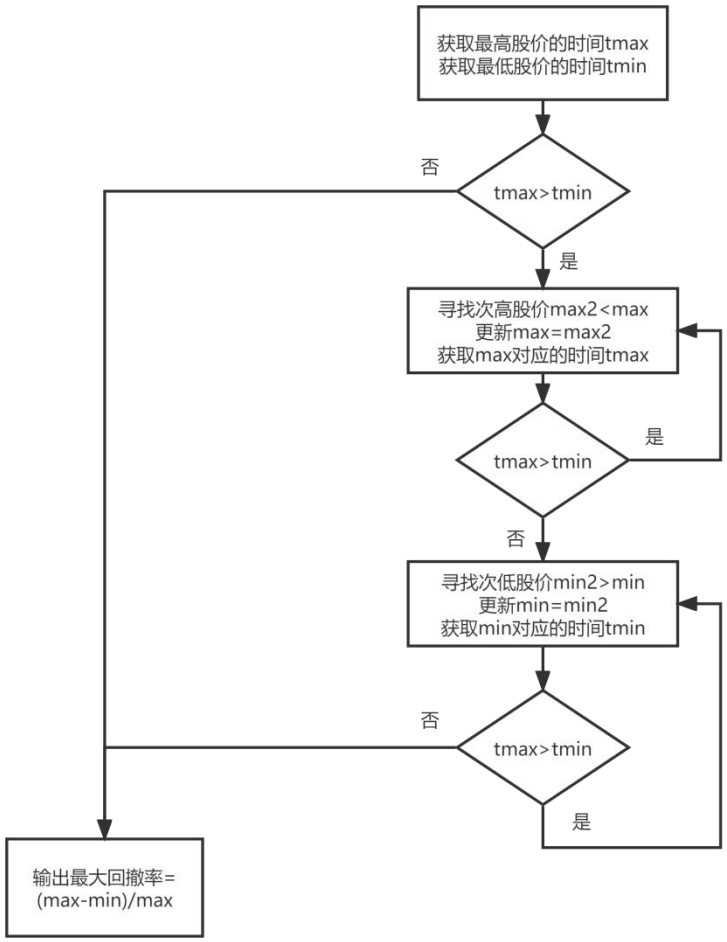

图 5.13 最大回撤率计算法流程图

5.2.8 超参数更新

由于持仓股票数量始终不能为负数，且流动资金有限，定义 t 时刻持仓股票数量为Q_t，t 时刻的流动资金为M_t，有以下约束条件：

$$\begin{cases} Q_t \geq 0 \\ M_t \geq 0 \end{cases} \tag{22}$$

5.2.9 交易流程

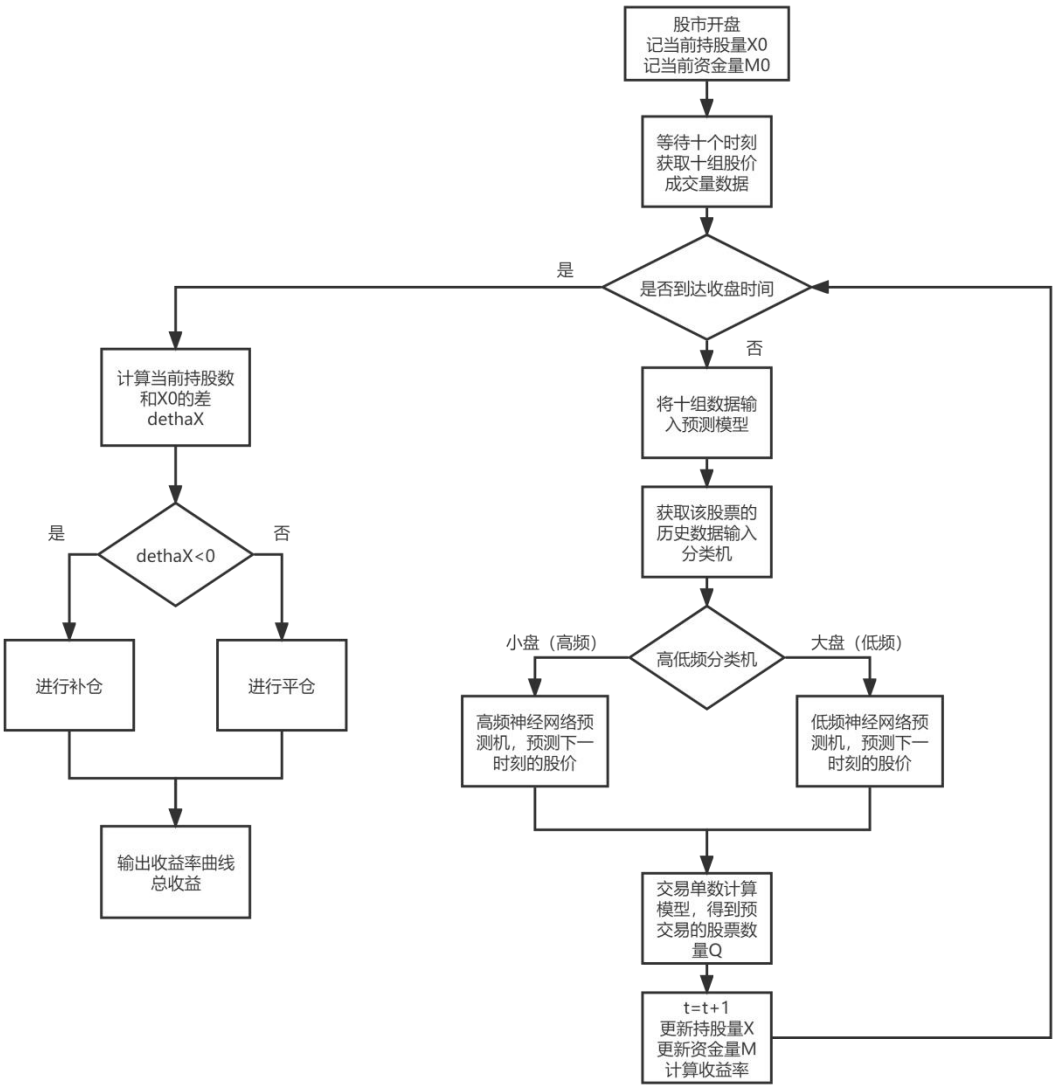

图 5.14 高频交易流程图

5.2.10 收益率对比

对比经典网格交易策略和改进网格交易策略，可得两者的收益率曲线：

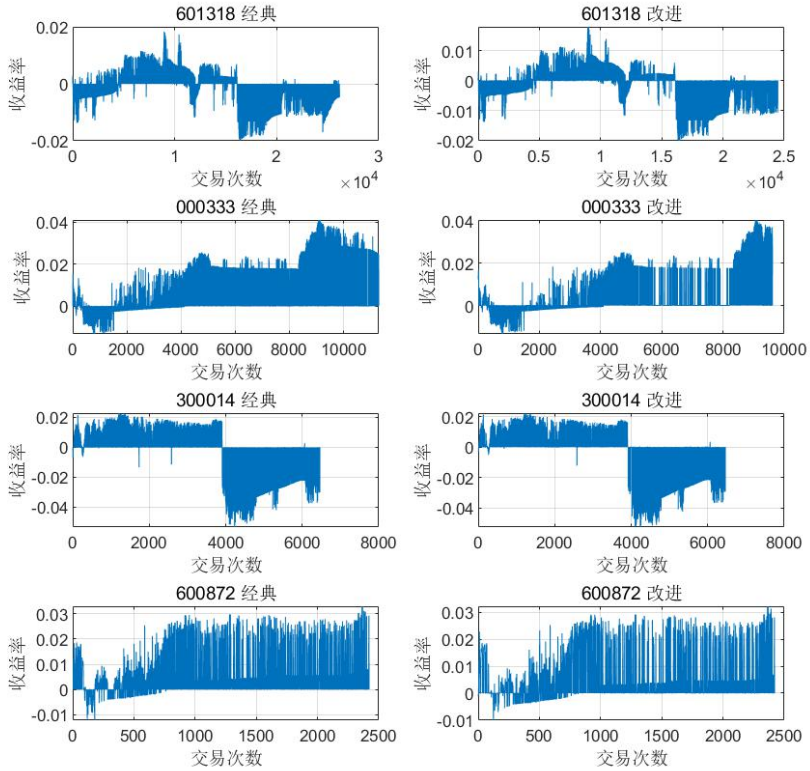

图 5.15 经典高频交易策略和改进高频交易策略收益率曲线对比

两者的四只股票平均收益率分别为：601318：-0.0011，-0.0007；000333：0.0064，0.0044；300014：-0.0025，-0.0022；600872：0.005606，0.005641

可见，对问题三即将使用的四只股票，使用改进后的网格交易策略能够获得比经典网格交易更大的收益率，故选用改进后的高频交易策略作为问题三的高频交易策略。

5.3 问题三模型的建立与求解

问题三要求从任务一中选取两只股票与中国平安和美的集团股票一同进行高频交易，故本文选择了任务一种综合评分最高的两只股票：300014、600872，与 601318、000333 共四只股票进行高频交易。交易策略选用收益率较高的改进网格策略。

5.3.1 超参数更新

由于问题三要求每日持股数不变，因此修改公式（20）得

$$\begin{cases} Q_t \geq 0 \\ Q_{t_0} = Q_{t_1} \\ M_t \geq 0 \\ M_{t_0} = 4000000 \end{cases} \tag{23}$$

其中：Q_{t_0} 为开盘时的持股数，Q_{t_0} 为收盘时的持股数。

5.3.2 佣金和印花税扣除

每次卖出股票，进行佣金和印花税扣除操作：

$$\begin{cases} M_t = M_{t-1} - P_t \cdot \Delta Q - tax - charge & charge > 5 \\ M_t = M_{t-1} - P_t \cdot \Delta Q - tax - 5 & charge \le 5 \\ charge = 0.015\% \cdot P_t \cdot \Delta Q \\ tax = 0.1\% \cdot P_t \cdot \Delta Q \end{cases} \tag{24}$$

每次买入股票，进行佣金扣除操作：

$$\begin{cases} M_t = M_{t-1} - P_t \cdot \Delta Q - charge & charge > 5 \\ M_t = M_{t-1} - P_t \cdot \Delta Q - 5 & charge \le 5 \\ charge = 0.015\% \cdot P_t \cdot \Delta Q \end{cases} \tag{25}$$

根据以上公式更新流动资金量 M_t

5.3.3 累计交易量

由于高频交易成交量不得超过总成交量 V_{t_1} 的 10%，设当前股票累计进行高频交易数量为 $Q_t^{sum} = \sum_{i=t_0}^{t} \Delta Q_i$，更新公式（18）为：

$$\begin{cases} \Delta Q = \dfrac{1000 \arctan(\Delta P)}{\pi/2} & , Q_t^{sum} \le 10\% \cdot V_{t_1} \\ \Delta Q = 0 & , Q_t^{sum} > 10\% \cdot V_{t_1} \end{cases} \tag{26}$$

5.3.4 交易流程

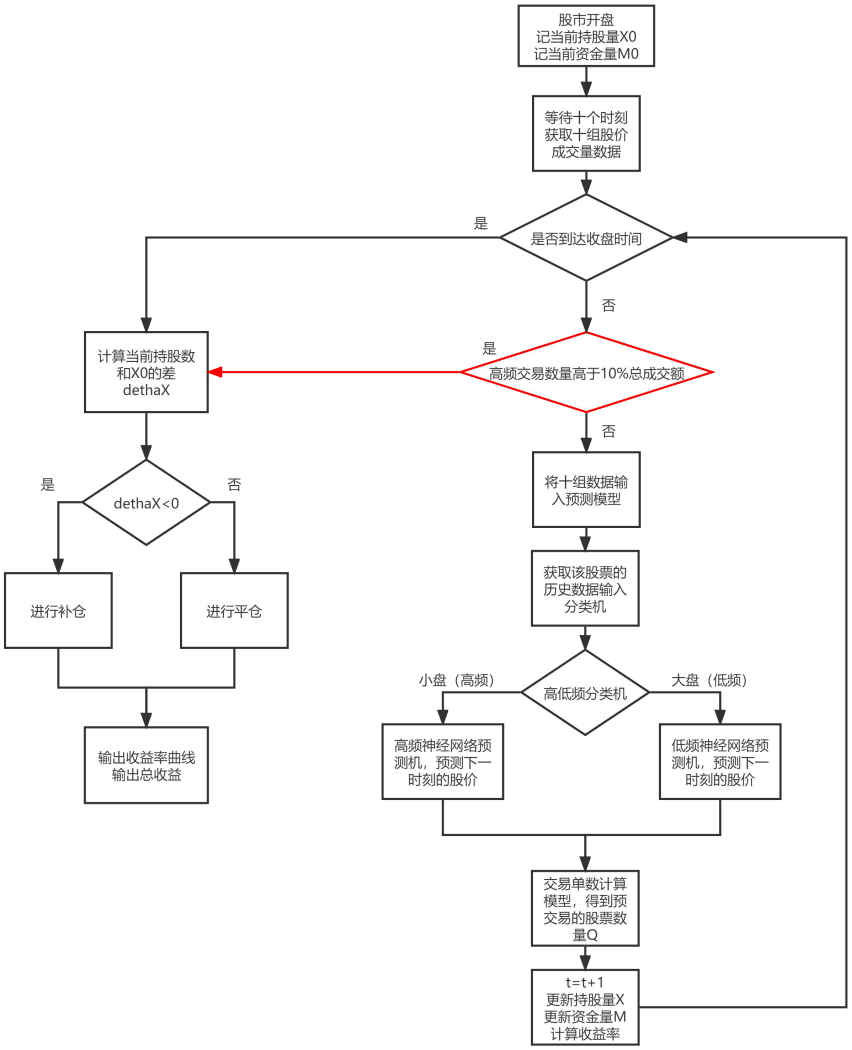

图 5.16 针对任务三改进高频交易流程

5.3.5 收益与收益率、最大回撤率

利用回撤率计算公式：最大回撤率 $=\frac{Max-Min}{Min}$，可得四只股票的最大回撤率分别为：

601318：1.37%、000333：2.16%、300014：2.71%、600872：2.82%

求得四只股票的收益分别为

601318：$2.1528e^{6}$ 、000333：$8.4356e^{6}$ 、300014：$4.2670e^{7}$ 、600872：$3.8526e^{6}$

四只股票的收益率曲线如下图：

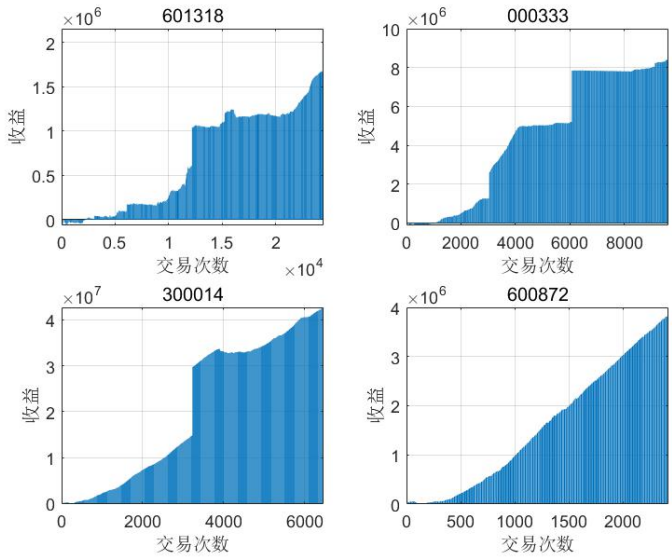

图 5.17 四只持有股票的收益率曲线

由图可见，最大负收益未超过流动资金 $4e^5$，因此假设（10）成立。

5.4 问题四模型的建立与求解

针对大量异常买单导致的涨停事件和大股东巴菲特抛售股票事件，本文分别提出了两种针对性高频交易策略。

5.4.1 大股东减持应对策略

针对巴菲特减持比亚迪股票，由于证券法规定，大股东减持股票前需要先进行公示，因此，当得知大股东减持的公告后，应针对该股票单独改变高频交易策略。

由于大股东减持是对该股票不看好或是急需套现的行为表现，可以预见的，未来一段时间内股票的价格会持续性下降，由于单边市场不适合高频交易者，因此应当抛出适量股票保值，等待股价大跌后演变为震荡股市再入场。

因此，修改 ΔQ 的更新方法为：

$$\begin{cases} \Delta Q = -\dfrac{1}{t} & ,\ \Delta P < 0 \\ \Delta Q = -\dfrac{\partial \Delta P}{\partial \Delta t} & ,\ \Delta P \geq 0 \end{cases} \tag{27}$$

其中，Δt 表示时间步长，t 表示大股东开始减持后的第 t 时刻

上述公式表明，在股价持续下跌的过程中，抛售的股票数量与时间成反比关系，离大股东开始减持的时间越短，紧跟着抛售的股票数量也越大，有利于股票保值。当

发生股价回升时，加快抛售速度，抛售速度与股价回升速度成正相关。

作高频交易策略改进前后的收益率曲线如下图：

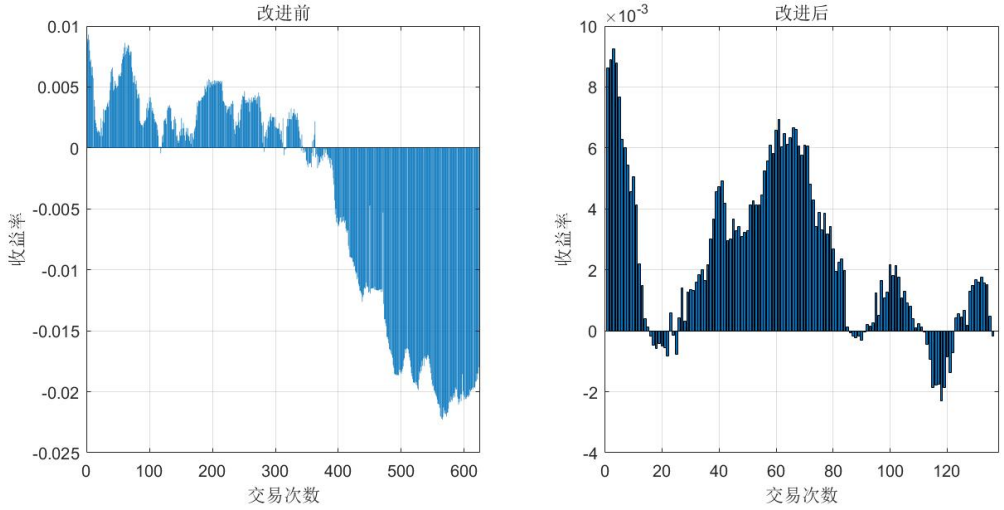

图 5.18 高频交易策略改进前后收益率曲线对比图

可见，在改进高频交易后，平均收益率提高了 0.3683%，针对大股东减持时间收益率有了较大提升，大大减小了大股东减持所带来的附加亏损。

5.4.2 异常买单涨停[8]策略

针对大量异常买单涌入市场导致的股票涨停[6]，本文提出利用问题二中提出的预测机模型，增加输出层数量为 3 个时刻的股价预测，若任一预测股价达到涨停线，则开始大量抛售股票增加涨停前的收益，提早抛售增加股票被卖出的机率。若无涨停预警则按照公式（20）更新 ΔQ 。其中改进后的预测模型结果如下：

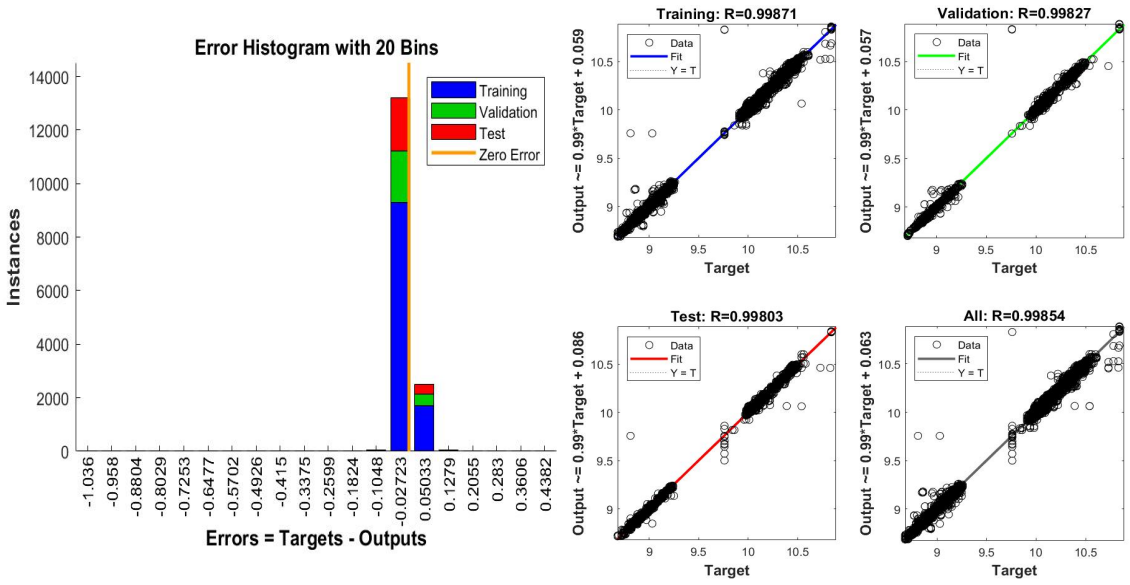

图 5.19 涨停预警模型的误差分布和预测准确率示意图

六、 模型的分析与检验

针对题目条件：$4e^5$ 的流动资金和 $1.5e^5$ 的每股股票市值对收益所受影响作灵敏度分析，将股票市值从 $0.5e^5$ 按步长 $0.1e^5$ 增加到 $2.5e^5$，将流动资金从 $3.5e^5$ 按步长 $0.1e^5$ 增加到 $4.5e^5$，将收益变化量作为分析目标，结果如下：

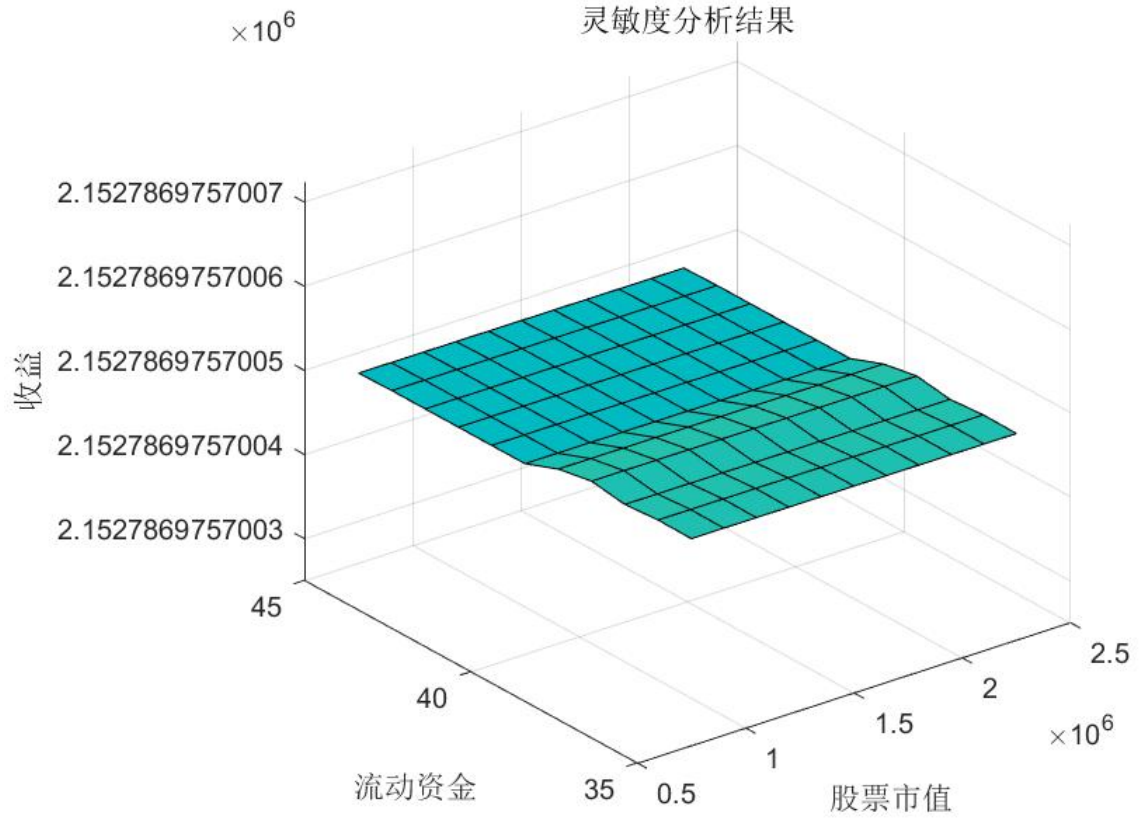

图 5.20 收益与初始流动资金、股票市值的关系

可见，在流动资金高于 $4e^5$ 时，收益不变；当流动资金高于 $4e^5$ 时，收益变化了 0.000001 元，变化量极小，因此认为收益对流动资金的灵敏度低。

初始股票市值变化间接影响持股数，由于交易数始终小于持股数，因此影响微弱

因此可认为该模型鲁棒性高，当初始条件变化时，输出结果较稳定，对初始条件的敏感度较小。

七、 模型的评价、改进与推广

7.1 模型的优点

创新性地使用了高频因子作为条件对适合高频交易的股票进行筛选。

高频因子选取合理有效，相关性较强。

回测和收益率结果有较强的说服力，反映该交易策略的有效性。

在经典模型和论文的基础上创新性地提出一些针对性的算法和模型，适合一般的高频交易场景，鲁棒性高、结果合理。

7.2 模型的缺点

将 0.6 倍的市场成交量作为模型的成交量限制，暂时忽略了其他高频交易竞争者对股市的影响。

忽略最后收盘时为保持持仓股票数量一致而忽略了拆单卖出所需的时间。

7.3 模型的改进

利用动态规划算法优化卖出单数，使拆单单量减小。

进一步延长预测时间，增加隐藏层深度和神经元数量，以进一步减小风险，使决策更合理有效。

7.4 模型的推广

加入高效的拆单策略[4]，以满足不同用户如不留隔夜仓等需求。

模块化设计，方便修改买卖系统的单次交易量函数获得更有市场针对性的交易策略。

八、 参考文献

[1]王怡宁. 基于 XGBoost 的高频交易选股研究[D]. 东华大学,2021.DOI:10.27012/d.cnki.gdhuu.2021.001289.
[2]喻术奇. 基于时变加权 LightGBM 的多因子选股交易策略设计[D].上海师范大学,2020.DOI:10.27312/d.cnki.gshsu.2020.000484.
[3]姜启源, 谢金星, 叶俊.数学模型（第三版）[M]. 北京: 高等教育出版社, 2003: 274-324
[4]姚海博,茹少峰,张文明.基于动态交易量预测的 VWAP 算法交易卖出策略[J].运筹与管理,2015,24(02):215-220.
[5]陈艳,王宣承.基于变量选择和遗传网络规划的期货高频交易策略研究[J].中国管理科学,2015,23(10):47-56.DOI:10.16381/j.cnki.issn1003-207x.2015.10.006.
[6]彭志.量化投资和高频交易:风险、挑战及监管[J].南方金融,2016,No.482(10):84-89.
[7]施镇海. 股票涨停后个人投资者决策行为的实证研究[D].浙江财经大学,2014.
[8]赵晓光. 涨停股票的投资策略研究[D].西安工业大学,2012.
[9]胡天福. 高频交易在中国证券市场的应用研究[D].上海交通大学,2012.

附录

附录 1：代码名字

介绍：支撑材料的文件列表

Analyze1.m
EntropyMethod.m
Fix001.m
TopsisMethod.m
Analyze2.m
Analyze3.m
CNNJudge20221106.m
CoreInput10tOutputNum.m
CoreUpdate11tHoldStocks.m
IkunBuySellBasis.m
IkunBuySellClassic.m
IkunBuySellReverseRatio.m
IkunLineFitting.m
IkunLineFittingRand.m
IkunStaticPredict001.m
IkunVibrantPredict001.m
neuralPredictionTest001.m
PackUpJudgeHistory.m
PackUpOutputFunstion.m
AnswerProfiRate001.mlx
AnswerCodeBlock1.mlx
Answer1Dat1.mat
Answer2Dat1.mat
NeuralTeachingsDat1.mat
OriginalData2.mat
大盘训练数据：从同花顺软件上下载的 11.5、11.6、11.7 三日的实盘全天数据

附录 2：PackUpOutputFunstion.m

作用：最外层函数，计算持仓数目、持有金额、平均成本、收益率、回撤等；进行交易的主体部分
使用 MATLAB 语言编写

```
function [HoldCountRouting, CashTests, AveCost, SellNumber, AveProfitRate, MaximemRetfinal, TradeNumbers, HoldCountTest, PureProfit] = PackUpOutputFunstion (Input11ArrayOrigin, Input11DealOrigin, BeforeHoldCount, BeforeCash, EndPriceYesterday)
%%
% 2022.11.6
% 大打包函数
% 输入：
```

```
%
% m 是同一只股票的不同的时间点!
%
% Input11ArrayOrigin (m, 1) = 每时刻股价原始数据，m 行 1 列
% Input11DealOrigin (m, 1) = 每时刻交易量原始数据，m 行 1 列
% BeforeHoldCount (1,1) = 从前的持仓数目
% BeforeCash(1,1) = 从前的持有金额
% EndPriceYesterday(1,1) = 从前的收盘价
%
% 输出:
% HoldCountRouting (m-10, 1) = 持仓数目（过程）
% CashTests (m-10,1) = 最终的持有金额
% AveCost (m-10, 1) = 平均成本
% SellNumber (m-10, 1) = 每次买入的数量（买正卖负）
% AveProfitRate (m-10, 1) = 平均收益率
% MaximumRetracement (1, 1) = 最大回撤
% HoldCountTest (1,1) = 最终的持仓数目 （终点）
% PureProfit (1,1) = 最终收益

% 变量设置部分
InputMatrixTest = zeros(1, 10);
DealMatrixTest = zeros(1, 10);
HoldCountTest = BeforeHoldCount(1, 1);
CashTests = zeros([], 1);
CashTest = 0;
sellnumber = 0;
SellNumber = zeros([], 1);
AveCost = zeros([], 1);
AveProfitRate = zeros([], 1);
HoldCountRouting = zeros([], 1);
avecostcell = EndPriceYesterday;
MaximumRetracement = 0;
maximumrec = 0;
minimumrec = 0;
TradeNumber = 0;
TradeNumbers = zeros([], 1);
%aveprofitrate = 0;
m = 0;
%beforecash = BeforeCash;
beforeholdcount = zeros(1,1);
beforeholdcount = BeforeHoldCount(1, 1);        %分量，用于存储从前的
持仓数目

k = 90;                          %一种止损线的设置策略，股价低于基准价的 k%时
```

```
    就卖股止损
basisprice = 0;          %基准价
IsVibrant = PackUpJudgeHistory (-1, Input11ArrayOrigin, Input11DealOrigin);


% 主循环
for i = 1 : size(Input11ArrayOrigin, 1)

    %计算回撤部分
    if i > 2
        if Input11ArrayOrigin(i, 1) - Input11ArrayOrigin(i-1, 1) < 0 && Input11ArrayOrigin(i-1, 1) - Input11ArrayOrigin(i-2, 1) >= 0    %极大值

            if Input11ArrayOrigin(i-1, 1) >= maximumrec
                MaximumRetracement = max(MaximumRetracement, maximumrec - minimumrec);
                MaximemRetfinal = MaximumRetracement/ maximumrec;
                maximumrec = Input11ArrayOrigin(i-1, 1);
                minimumrec = Input11ArrayOrigin(i-1, 1);
            else  %Input11ArrayOrigin(i-1, 1) < maximumrec
            end
            maximumrec = Input11ArrayOrigin(i-1, 1);
        end
        if Input11ArrayOrigin(i, 1) - Input11ArrayOrigin(i-1, 1) > 0 && Input11ArrayOrigin(i-1, 1) - Input11ArrayOrigin(i-2, 1) <= 0    %极小值
            minimumrec = min(minimumrec, Input11ArrayOrigin(i-1, 1));
        end
    end
    % 交易部分
    if i <= 10           %前 10 个时刻不进行任何交易
        InputMatrixTest(1, i) = Input11ArrayOrigin(i, 1);
        DealMatrixTest(1, i) = Input11DealOrigin(i, 1);
        basisprice = basisprice + 0.1 * Input11ArrayOrigin(i, 1); %基准价

    elseif ( i == size(Input11ArrayOrigin, 1) && (HoldCountTest > beforeholdcount)) || mod(i, floor(size(Input11ArrayOrigin, 1)/21)) == 0       %最后一刻，剩下的全部卖出，最终得到高频交易所需股票
```

```
        holdcounttest = HoldCountTest;  %前一时刻的持仓

        [~,~,InputMatrixTest, DealMatrixTest, ~] = CoreUpdate11tHoldStocks (InputMatrixTest, DealMatrixTest, Input11ArrayOrigin(i, 1), Input11DealOrigin(i, 1), HoldCountTest, CashTest, IsVibrant, basisprice, k, beforeholdcount);
        sellnumber = HoldCountTest - beforeholdcount;
        HoldCountTest = HoldCountTest - sellnumber;
        CashTest = CashTest + sellnumber * Input11ArrayOrigin(i, 1);

        if  mod(i, floor(size(Input11ArrayOrigin, 1)/21)) == 0
            basisprice = Input11ArrayOrigin(i, 1);          %基准价每天调整，这里是 21 天的情况
        end

        m = m + 1;

        if sellnumber < 0       % 买入
            avecostcell = ((holdcounttest * avecostcell) - (Input11ArrayOrigin(i, 1)) * sellnumber) / (HoldCountTest);     %计算平均成本
            aveprofitrate = ((Input11ArrayOrigin(i, 1) - avecostcell) / avecostcell);       %计算收益率
        else                    % 卖出
            aveprofitrate = 0;
        end

          AveCost(m, 1) = avecostcell;
          SellNumber(m, 1) = sellnumber;
          HoldCountRouting(m, 1) = HoldCountTest;
          AveProfitRate(m, 1) = aveprofitrate;
      else                                  %进行交易
          holdcounttest = HoldCountTest;  %前一时刻的持仓
          [HoldCountTest, CashTest, InputMatrixTest, DealMatrixTest, sellnumber] = CoreUpdate11tHoldStocks (InputMatrixTest, DealMatrixTest, Input11ArrayOrigin(i, 1), Input11DealOrigin(i, 1), HoldCountTest, CashTest, IsVibrant, basisprice, k, beforeholdcount);

          m = m + 1;

          if sellnumber < 0       % 卖出
              avecostcell = ((holdcounttest * avecostcell) - (In
```

```
    put11ArrayOrigin(i, 1)) * sellnumber) / (HoldCountTest);     %计
    算平均成本
113.             aveprofitrate = ((Input11ArrayOrigin(i, 1) - aveco
    stcell) / avecostcell);       %计算收益率
114.         else                    % 买入
115.             aveprofitrate = 0;
116.         end
117.
118.         AveCost(m, 1) = avecostcell;
119.         SellNumber(m, 1) = sellnumber;
120.         HoldCountRouting(m, 1) = HoldCountTest;
121.         AveProfitRate(m, 1) = aveprofitrate;
122.         PureProfit = HoldCountTest - BeforeCash;
123.         CashTests(m, 1) = CashTest;
124.     end
125.     TradeNumber = TradeNumber + abs(sellnumber);
126.     TradeNumbers(i, 1) = TradeNumber;
127.
128.     if TradeNumber > 0.1 * sum(Input11DealOrigin)   %持仓情况
129.
130.                 holdcounttest = HoldCountTest;  %前一时刻的持仓
131.
132.         [~,~,InputMatrixTest, DealMatrixTest, ~] = CoreUpdate1
    1tHoldStocks (InputMatrixTest, DealMatrixTest, Input11ArrayOrigi
    n(i, 1), Input11DealOrigin(i, 1), HoldCountTest, CashTest, IsVib
    rant, basisprice, k, beforeholdcount);
133.         sellnumber = HoldCountTest - beforeholdcount;
134.         HoldCountTest = HoldCountTest - sellnumber;
135.         CashTest = CashTest + sellnumber * Input11ArrayOrigin(
    i, 1);
136.
137.         m = m + 1;
138.
139.         if sellnumber < 0       % 买入
140.             avecostcell = ((holdcounttest * avecostcell) - (In
    put11ArrayOrigin(i, 1)) * sellnumber) / (HoldCountTest);     %计
    算平均成本
141.             aveprofitrate = ((Input11ArrayOrigin(i, 1) - aveco
    stcell) / avecostcell);       %计算收益率
142.         else                    % 卖出
143.             aveprofitrate = 0;
144.         end
145.
146.         AveCost(m, 1) = avecostcell;
```

```
            SellNumber(m, 1) = sellnumber;
            HoldCountRouting(m, 1) = HoldCountTest;
            AveProfitRate(m, 1) = aveprofitrate;

           break;
        end
        if HoldCountTest < 0
           break;
        end

    end

    end

    % 函数引用关系
    % PackUpOutPutFunstion()
    %     -> PackUpJudgeHistory()
    %         -> CNNJudge20221106()
    %     -> CoreUpdate11tHoldStocks()
    %         -> CoreInput10OutputNum()
    %             -> IkunBuySellBasis()
    %             -> IkunBuySellClassic()
    %             -> IkunVibrantPredict001()
    %             -> IkunStaticPredict001()
```

PackUpJudgeHistory.m
股票类型基于历史数据的判断
使用 MATLAB 语言编写

```
clc
%%
% 2022.11.2
% 金融数模股票数据提取程序
%%
% 导入最初的数据到这里，生成 origin 矩阵：
% 注意：导入时要用数值矩阵的方式！不要用默认的表的方式！
origin = D002594;
%%
origin = transpose(origin);    %原始数据，矩阵转置
%origindata = size(origin);
newtradematrix = zeros([], 6, []);                    %得出最终数据的矩阵
outdatamatrix = zeros(100,2);                         %输出信息的矩阵
matrixindex = zeros(1, 2);                            %上面的矩阵的索引
day = 1;                    %每一天对应的情况分析
```

```
tempnum = 0.0;
tempval = 0.0;
temptime = 0.0;
tempsingle = 0.0;


k = 0;
k1 = 0;
k2 = 0;
vibration = zeros([],[]);           %两个相邻数据间的振幅（单价元/时间秒）
frequency = zeros([],[]);           %频率记录（秒）
motion = zeros(1, []);               %股价差有效数据
motion2 = zeros(1, []);              %成交量差有效数据
    %%
        % newtradematrix 矩阵说明：
        % 第 1 维度：每一天的情况
        % 第 2 维度：
        %   第 1 行：成交量差（手）
        %   第 2 行：成交额差（元）
        %   第 3 行：平均成交单价（元）
        %   第 4 行：时间差（秒）
        %   第 5 行：时间差从 1 到 i（index）的累积和（秒）

        %   第 6 行：两个相邻数据间的振幅（单价元/时间秒）

        % outdatamatrix 矩阵说明：
        % 行数：第 n 天
        % 列数：
        %   第 1 列：总升
        %   第 2 列：总减
    %%
    % 数据提取部分
for i = 1 : size(origin,2)-1

    if (origin(2,i+1) - origin(2,i) >= 500000.0)     %判断是否到了第二天
        day = day + 1;
        matrixindex(day) = 1;
        k = 0;      %循环变量
        k1 = 0;     %循环变量
    end
    if (origin(2,i+1) - origin(2,i) <= 150.0)
        if (origin(7,i+1) - origin(7,i) > 0)        %滤掉成交额和
```

```
    成交量为0的数据
58.             tempnum = origin(8, i+1) - origin(8, i);     %成交量差
59.             tempval = origin(7, i+1) - origin(7, i);     %成交额差
60.             temptime = ((origin(2, i+1) - origin(2, i))) * 0.6;
     %时间差
61.             tempsingle = (tempval / tempnum) / 100;
    %平均单价
62.
63.             if (tempsingle > min(origin(6,i+1), origin(6, i)) &&
     tempsingle < max(origin(5,i+1), origin(5, i)))   %滤掉低于最低
    价、高于最高价的值
64.
65.                 if (abs(tempsingle - (tempval / tempnum) / 100)
    < 10^100 || k2 <= 1)
66.                     motion(k2 + 1) = (tempval / tempnum) / 100;
67.                     motion2(k2 + 1) = tempnum;
68.                 else
69.                     motion(k2 + 1) = motion(k2);
70.                     motion2(k2 + 1) = tempnum;
71.                 end
72.                 k2 = k2 + 1;
73.
74.                 matrixindex(day) = matrixindex(day) + 1;
75.                 newtradematrix(day,1,matrixindex(day)) = tempnum
    ;  %导入数据到矩阵里面
76.                 newtradematrix(day,2,matrixindex(day)) = tempval
    ;
77.                 newtradematrix(day,3,matrixindex(day)) = tempsin
    gle;
78.                 newtradematrix(day,4,matrixindex(day)) = temptim
    e;
79.
80.                 %加入数据到总升、总减中
81.                 if matrixindex(day) > 1
82.                     change = newtradematrix(day,3,matrixindex(da
    y)) - newtradematrix(day,3,matrixindex(day)-1);
83.                     if change >= 0  %加入总升
84.
85.                         if k < 0
86.                             k1 = k1 + 1;
87.                             frequency(day, k1) = abs(k);
88.                             k = 0;
89.                         end
90.                         k = k + abs(temptime);
```

```
91.
92.                         outdatamatrix(day,1) = outdatamatrix(day
   ,1) + change;
93.                     else            %加入总减
94.
95.                         if k > 0
96.                             k1 = k1 + 1;
97.                             frequency(day, k1) = abs(k);
98.                             k = 0;
99.                         end
100.                           k = k - abs(temptime);
101.
102.
103.                           outdatamatrix(day,2) = outdatamatrix(d
   ay,2) + change;
104.                        end
105.                        newtradematrix(day,6,matrixindex(day)) = c
   hange / temptime;
106.                        vibration(day, matrixindex(day)) = change
   / temptime;     %振幅(改为总升-总降)
107.                    end
108.
109.                    if matrixindex(day) == 1    %时间差累积和
110.                        newtradematrix(day, 5, matrixindex(day)) =
    0;
111.                    else
112.                        newtradematrix(day, 5, matrixindex(day)) =
    newtradematrix(day, 5, matrixindex(day)-1) + temptime;
113.                    end
114.                end
115.            end    %判断是否为空数据
116.        end
117.    end    %for 循环
118.
119.    %%
120.    % 绘图部分
121.    %subplotn = 4;   %画图子图每行的个数
122.    %subplotm = ceil(day / 4);
123.    %for j = 1 : day
124.        % grid on;
125.        %plotx = zeros(1, 2);
126.        %ploty = zeros(1, 2);
127.        %k = 0;
128.        %for i = 1 : matrixindex(1, j)
```

```
129.          %if newtradematrix(j, 3, i) ~= 0
130.              %k = k + 1;
131.              %plotx(k) = newtradematrix(j, 5, i);
132.              %ploty(k) = newtradematrix(j, 3, i);
133.          %end
134.      %end
135.      %subplot(subplotm, subplotn, j);
136.              %xlabel('累计时间;');
137.              %ylabel('成交单价');
138.              %title('002594');
139.      %hline1 = plot(plotx, ploty, 'r');
140.  %end
141.  %hline1 = plot(plotx, ploty, 'r');
142.
143.  %%
144.  % 数据整合部分
145.  vibrationbind2 = [];
146.  vibrationbind = abs(reshape(vibration, 1, []));
147.  stabilitybind = reshape(frequency, 1, []);
148.  stabilitybind2 = [];
149.  k = 0;
150.  for i = 1 : size(vibrationbind, 2)
151.      if vibrationbind(1, i) ~= 0 && ~isinf(vibrationbind(1, i))
    && ~isnan(vibrationbind(1, i))
152.          k = k + 1;
153.          vibrationbind2(k) = vibrationbind(1, i);
154.      end
155.  end
156.  vibrationbind1 = sum(vibrationbind2);    %总升-总降
157.
158.
159.
160.  %总:
161.  vibrationbind = reshape(vibration, 1, []);
162.  k = 0;
163.  for i = 1 : size(vibrationbind, 2)
164.      if vibrationbind(1, i) ~= 0 && ~isinf(vibrationbind(1, i))
    && ~isnan(vibrationbind(1, i))
165.          k = k + 1;
166.          vibrationbind2(k) = vibrationbind(1, i);
167.      end
168.  end
169.  vibrationbind3 = sum(vibrationbind2);   %总
170.
```

```
%总:
vibrationbind = reshape(vibration, 1, []);
k = 0;
for i = 1 : size(vibrationbind, 2)
    if vibrationbind(1, i) > 0 && ~isinf(vibrationbind(1, i)) && ~isnan(vibrationbind(1, i))
        k = k + 1;
        vibrationbind2(k) = vibrationbind(1, i);
    end
end
vibrationbind4 = sum(vibrationbind2);   %仅总升


%vibrationbind = mean(vibrationbind2);
% 总升-总降

k = 0;
for i = 1 : size(stabilitybind, 2)
    if stabilitybind(1, i) ~= 0
        k = k + 1;
        stabilitybind2(k) = stabilitybind(1, i);
    end
end

stabilitybind1 = [];
k = 0;
for i = 1 : floor(size(stabilitybind2, 2) / 2)
    k = k + 1;
    stabilitybind1(k) = stabilitybind2(1, i) + stabilitybind2(1, i+1);
end
stabilitybind = mean(1.0 ./ stabilitybind1);
% 频率（峰值个数 / 周期），输出为 stabilitybind。

Var = var(motion);
% 所有有效数据的方差

Kurtosis = kurtosis(motion);
% 所有有效数据的峰度

%motiony = 1 : size(motion, 2);
%motionp = polyfit(motiony, motion, 2);
%motiony = motiony .^ 2 * motionp(1) + motiony .* motionp(2) + motionp(3);
```

```
212.
213.
214.
215.  %plot(1 : size(motion,2), motion);
216.  %yticks(0:0.5:500)
217.  %grid on
218.
219.
220.  %%
221.  % 每只股票最终输出的东西，仅 1 行
222.  termindex = 0;                        %输出的数据
223.
224.  vibrationsize = size(reshape(vibration, 1, []), 2);
225.  termindex = termindex + 1;
226.  % 确定下面的 4 个东西
227.  Termindex (termindex, 1) = vibrationbind1 / vibrationsize;   %
     01 总升 - 总降 / 总有效数据量
228.  Termindex (termindex, 2) = stabilitybind;   % 02 频率
229.  Termindex (termindex, 3) = Var;             % 03 方差
230.  Termindex (termindex, 4) = Kurtosis;        % 04 峰度
231.
232.  %总升-总降 / 总有效数据量 * 频率
233.  Termindex (termindex, 5) = vibrationbind1 / vibrationsize * st
   abilitybind * 10^4;
234.
235.  %总升-总 / 总有效数据量 * 频率
236.  Termindex (termindex, 6) = ((vibrationbind4 - vibrationbind3)
   / vibrationsize) * stabilitybind * 10^4;
237.
238.  %总升 - 总 / 总有效数据量 * 峰度
239.  Termindex (termindex, 7) = Termindex (termindex, 6) * Kurtosis
   ;
240.  %%
241.  % 修正代码
242.  unun = 0.0;
243.  for i = 1 : size(vibrationbind2, 2)
244.      if ~isinf(vibrationbind2(1, i)) && ~isnan(vibrationbind2(1
   , i))
245.          unun = unun + vibrationbind2(1, i);
246.      end
247.  end
248.  Termindex(termindex, 1) = unun;
249.
250.  %%
```

```
% 神经网络训练专区
%k3 = 0;
%k3 = k3 + 1
k4 = 0;     %终极大样本处理
%TrainSourceStatic4 = zeros([], 10);
%TrainTeachStatic4 = zeros([], 1);
InputMatrixOrigin = zeros([], 10);
DealMatrixOrigin = zeros([], 10);
Input11ArrayOrigin = zeros([], 1);
Input11DealOrigin = zeros([], 1);
for i = 1 : size(motion, 2) - 11 - 1
    k4 = k4 + 1;
    for j = 1 : 10
    InputMatrixOrigin(k4, j) = motion(i + j - 1);
    DealMatrixOrigin(k4, j) = motion2(i + j - 1);
    end
    Input11ArrayOrigin(k4, 1) = motion(i + 10);
    Input11DealOrigin(k4, 1) = motion2(i + 10);
end
 k4
```

CNNJudge20221106.m
用于判断股票的属于稳定类型或非稳定类型
使用 MATLAB 语言编写

```
function [Y,Xf,Af] = CNNJudge20221106(X,~,~)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 06-Nov-2022 19:57:59.
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 1xTS cell, 1 inputs over TS timesteps
%   Each X{1,ts} = Qx2 matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 1xTS cell of 1 outputs over TS timesteps.
%   Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number
  of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

```

```
% Input 1
x1_step1.xoffset = [11.2703357563171;30.1399999999994];
x1_step1.gain = [0.175058159013876;0.000211194014885296];
x1_step1.ymin = -1;

% Layer 1
b1 = [-4.4989540062893205175;3.2832183566039447875;-2.5147993588133870091;-1.4872656035406071062;-0.74058119999315652482;-0.5759642604831787249 3;-1.4954910588339924704;-2.1223387976968499657;2.9522107129746766319;-4.1046922017187243981];
IW1_1 = [3.6157013176300285551 2.4024632585367746707;-3.3566578321388291783 3.0552204319823501599;2.8912966679054972197 3.3282145762346235784;0.27525816302736028085 -4.4108985238687541397;4.36726095088711741 -0.65336165634369292032;-3.9141491984842184948 -2.1078112523543750534;-0.84987988400778557985 4.3316580516818223856;-4.5706331840530900479 -1.2265440227527566464;3.7445371541904015622 3.3101143212572501362;-3.8744248589769987312 -2.6632004946744340934];

% Layer 2
b2 = 0.010816380447024107406;
LW2_1 = [0.025300873274851701927 -0.0045365828189919375768 -0.01603453290786073515 1 -0.010644140743585314701 0.91532598062228498481 0.066352111232765201287 -0.0091759585716479179218 0.20038087962186892921 0.3534749974598593858 0.0062482476440339921112];

% Output 1
y1_step1.ymin = -1;
y1_step1.gain = 2;
y1_step1.xoffset = 0;

% ===== SIMULATION ========

% Format Input Arguments
isCellX = iscell(X);
if ~isCellX
    X = {X};
end

% Dimensions
TS = size(X,2); % timesteps
if ~isempty(X)
    Q = size(X{1},1); % samples/series
else
```

```
    Q = 0;
end

% Allocate Outputs
Y = cell(1,TS);

% Time loop
for ts=1:TS

    % Input 1
    X{1,ts} = X{1,ts}';
    Xp1 = mapminmax_apply(X{1,ts},x1_step1);

    % Layer 1
    a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*Xp1);

    % Layer 2
    a2 = repmat(b2,1,Q) + LW2_1*a1;

    % Output 1
    Y{1,ts} = mapminmax_reverse(a2,y1_step1);
    Y{1,ts} = Y{1,ts}';
end

% Final Delay States
Xf = cell(1,0);
Af = cell(2,0);

% Format Output Arguments
if ~isCellX
    Y = cell2mat(Y);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
```

```
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

 % Map Minimum and Maximum Output Reverse-Processing Function
 function x = mapminmax_reverse(y,settings)
 x = bsxfun(@minus,y,settings.ymin);
 x = bsxfun(@rdivide,x,settings.gain);
 x = bsxfun(@plus,x,settings.xoffset);
  end
```

CoreUpdate11tHoldStocks.m
用于判断跌停全卖止损、印花税、双向佣金，防止使卖出的股票数量多于持有的股票数量
使用 MATLAB 语言编写

```
function [AfterHoldCount, AfterCash, OutputMatrix, OutputDeal, sellnumber] = CoreUpdate11tHoldStocks (InputMatrix, DealMatrix, Input11Array, Input11Deal, BeforeHoldCount, BeforeCash, IsVibrant, BasisPrice, k, initprice)
%%
% 2022.11.5
% 当前是第 11 时刻
% 输入第 10 时刻的持仓、资金数，以及第 11 时刻的股价，输出第 11 时刻的持仓和股数
%
% 输入矩阵
% InputMatrix (m, n=10)
% m 行：m 个分别的股票
% n 列：必须得是 10 个，每一行前 10 时刻的数据
%
% DealMatrix (m, n=10)
% m 行：m 个分别的股票
% n 列：前 10 时刻各自的成交量（非累计）
%
% Input11Array (m, n=1)
% m 行：m 个分别的股票
% n 列：必须得是 1 个，第 11 时刻的股票票价
%
% Input11Deal (m, n=1)
% m 行：m 个分别的股票
% n 列：必须得是 1 个，第 11 时刻的成交量
%
% BeforeHoldCount(m, 1)
% m 行：m 个分别的股票
% n 列：必须得是 1 个，第 10 时刻持有的股票
```

```
%
% BeforeCash(m, 1)
% m 行：m 个分别的股票
% 1 列：第 10 时刻持有的现金
%
% 中间变量
% sellnumber (m, n=1)
% m 行：m 个分别的股票
% n 列：1 列，第 11 时刻的买卖数量
%
% predicteleven (m, n=1)
% m 行：m 个分别的股票
% n 列：1 列，第 11 时刻预测的股价
%
% BasisPrice (1,1)
% 股票基准价
%
% 输出矩阵
%
% AfterHoldCount(m, 1)
% m 行：m 个分别的股票
% n 列：必须得是 1 个，第 10 时刻持有的股票
%
% AfterCash(m, 1)
% m 行：m 个分别的股票
% 1 列：第 10 时刻持有的现金
%
% OutputMatrix (m, n=10)
% m 行：m 个分别的股票
% n 列：必须得是 10 个，每一行第 2 时刻到第 11 时刻的单价数据
%
% OutputDeal (m, n=10)
% m 行：m 个分别的股票
% n 列：必须得是 10 个，每一行第 2 时刻到第 11 时刻的成交量
%%
sellnumber = CoreInput10tOutputNum(InputMatrix, DealMatrix, IsVibrant, k, BasisPrice, initprice, BeforeHoldCount);

%%

if (-1 * max(sellnumber)) > BeforeHoldCount(1,1) - 1   %不可能使卖出的股票数量多于持有的股票数量
    sellnumber = -1;
end
```

```
if max(Input11Array(:,1)) < (100-k(1,1)) * 0.01 * BasisPrice (1,1) && sellnumber < 0
    sellnumber = -1 * abs(BeforeHoldcount(1,1) - initprice(1,1));    %跌停全卖止损措施
end


AfterHoldCount = BeforeHoldCount(1,1) - sellnumber;

stamptax = zeros(size(Input11Array, 1), 1);
bidirectcharge = zeros(size(Input11Array, 1), 1);

for i = 1 : size(Input11Array, 1)

    if sellnumber(i, 1) > 0
        stamptax(i, 1) = 10^(-3) * sellnumber(i, 1) * Input11Array(i,1);       %印花税
    end
    if abs((sellnumber(i, 1) * Input11Array(i,1))) > 10^5 / 3
        bidirectcharge(i, 1) = abs((sellnumber(i, 1) * Input11Array(i,1))) * 0.15 * 10^(-3);   %双向 0.15‰佣金
    else
        bidirectcharge(i, 1) = 5;
    end

end
AfterCash = BeforeCash - (sellnumber .* Input11Array) - bidirectcharge - stamptax;

% OutputMatrix = zeros([], max([10, size(InputMatrix, 2)]));
for i = 1 : size(InputMatrix, 2) - 1
   OutputMatrix(:, i) = InputMatrix(:, i + 1);
   OutputDeal(:, i) = DealMatrix(:, i + 1);
end
   OutputMatrix(:, 10) = Input11Array(:, 1);
   OutputDeal(:, 10) = Input11Deal(:, 1);

    end
```

CoreInput10tOutputNum.m
用于预测下一时刻的股票价格，及买入/卖出量
使用 MATLAB 语言编写

```
function [sellnumber, predicteleven] = CoreInput10tOutputNum (InputMatrix, DealMatrix, IsVibrant, k, Basisprice, initprice, BeforeHoldCount)
```

```
%%
% 2022.11.5
% 输入前 10 时刻的数据，决定第 11 时刻的买卖量
%
% 输入矩阵
% InputMatrix (m, n=10)
% m 行：m 个分别的股票
% n 列：必须得是 10 个，每一行前 10 时刻的数据
%
% DealMatrix (m, n=10)
% m 行：m 个分别的股票
% n 列：前 10 时刻各自的成交量（非累计）
%
% 输出矩阵
% sellnumber (m, n=1)
% m 行：m 个分别的股票
% n 列：1 列，第 11 时刻的买卖数量
%
% predicteleven (m, n=1)
% m 行：m 个分别的股票
% n 列：1 列，第 11 时刻预测的股价
%
%%
% 注意使用情形：
% 建议 m = 1;

SELECT_PREDICTION_MACHINE = IsVibrant(1,1);
SELECT_BUYSELL_MACHINE = 1;
% 预测机：0 = 高频，1 = 低频，2 = 拟合
% 买卖机：0 = Classic，1 = Basis, 99 = 巴菲特股价事件，100 = 光大事件

% 中间变量 tempt11 (m, n = 1)
% 第 11 的股价输出值


%%
if SELECT_PREDICTION_MACHINE <= 0
    tempt11 = IkunStaticPredict001 (InputMatrix);
elseif SELECT_PREDICTION_MACHINE >= 1
    tempt11 = IkunVibrantPredict001 (InputMatrix);
else
    tempt11 = SELECT_PREDICTION_MACHINE * IkunVibrantPredict001 (InputMatrix) + (1-SELECT_PREDICTION_MACHINE) * IkunStaticPredic
```

```
   t001 (InputMatrix);
end

%tempt11 = IkunLineFitting (InputMatrix);


% 中间变量 tempbuysellorigin (m, n = 1)
% 原始的第 11 时刻股价的预测价格
tempbuysellorigin = zeros(1,1);
if SELECT_BUYSELL_MACHINE == 0
    tempbuysellorigin = IkunBuySellClassic (InputMatrix, tempt11, 0, DealMatrix);
elseif SELECT_BUYSELL_MACHINE == 1
    tempbuysellorigin = IkunBuySellBasis (InputMatrix, tempt11, 0, DealMatrix);
elseif SELECT_BUYSELL_MACHINE == 99
    if InputMatrix(1, size(InputMatrix, 2)) > InputMatrix(1, size(InputMatrix, 2)-1)
    tempbuysellorigin = -50 * 1000 * (InputMatrix(1, size(InputMatrix, 2)) - InputMatrix(1, size(InputMatrix, 2)-1));
    elseif InputMatrix(1, size(InputMatrix, 2)) < InputMatrix(1, size(InputMatrix, 2)-1)
    tempbuysellorigin = 10 * 1000 * (InputMatrix(1, size(InputMatrix, 2)) - InputMatrix(1, size(InputMatrix, 2)-1));
    else

    end
end
if SELECT_BUYSELL_MACHINE == 100
    if tempt11 > (100 + k(1,1)) * 0.01 * Basisprice(1,1)
    %涨停
        tempbuysellorigin(:, 1) = -1 * abs(BeforeHoldcount(1,1) - initprice(1,1));
    else
        tempbuysellorigin = IkunBuySellBasis (InputMatrix, tempt11, 0, DealMatrix);
    end
end

% 中间变量 tempbuysellfinal (m, n = 1)
% 原始的第 11 时刻股价的决定价格
if SELECT_BUYSELL_MACHINE ~= 99 && SELECT_BUYSELL_MACHINE ~= 100
    tempbuysellfinal = zeros([], 1);
    for m = 1 : min([size(DealMatrix, 1), size(InputMatrix, 1)])
```

```
        if tempbuysellorigin (m, 1) <= mean(DealMatrix(m, :)) *
    2
            tempbuysellfinal(m, 1) = tempbuysellorigin(m, 1);
        else
            tempbuysellfinal(m, 1) = floor(mean(DealMatrix(m, :)
    ));
        end
    end

    sellnumber = -abs(floor(tempbuysellorigin));

else
    sellnumber = floor(tempbuysellorigin(1,1));
end
predicteleven = tempt11;
%tempbuysellfinal;


end
```

IkunBuySellBasis.m
网格买入卖出函数（改进法）
使用 MATLAB 语言编写

```
function [BuySellArray, ClosedValue] = IkunBuySellBasis (I
nputMatrix, PredictArray, InitValue, DealMatrix)
%%
% 2022.11.4
% 网格买入卖出函数（基本的）
% 每 1 分钱（0.01 元）一格
% 每个买入/卖出 100 个

% InputMatrix:
% 输入矩阵：列数：前 n 个点          size: 行 * 列
%           行数：样本个数
%
% PredictArray:
% 输入数组：1 列                     size: 行 * 列
%           行数：样本个数
%
% InitValue:
% 初始持仓数
%
% BuySellArray: 买卖数量决定
% 输出数组：2 列                     size: 行 * 列
%           行数：样本个数
```

```
%            列数：第 1 列：买/卖的数量
% ClosedValue:
% 最终持仓数
%%
basegrid = 10 ^ (-4);                          % 基准格数大小
topgrid = 10 ^ (-1);
diffrence = 0;
ClosedValue = 0;
basecountpergrid = min(150, 2 * max(DealMatrix(:, 10)));
        % 每个买入或卖出的股数
BuySellArray = zeros([], 1);
for i = 1 : size(InputMatrix, 1)    % 逐行操作
    difference = InputMatrix(i, size(InputMatrix, 2)) - PredictA
  rray(i, 1);
    sellnumber = 0;
    sellnumber = sellnumber + IkunConvertFunction002(difference,
     basegrid, topgrid);    %美的模型
    BuySellArray (i, 1) = sellnumber;
end
ClosedValue = InitValue + sum(BuySellArray(:, 1));


end

function y = IkunConvertFunction002 (x, basegrid, topgrid)
%     if x == 0
%         y = 0;
%     elseif x > 0 && x <= basegrid
%         y = floor(1 / basegrid);
%     elseif x > basegrid && x <= topgrid
%         y = floor(1 / x);
%     elseif x > topgrid
%         y = 0
%     elseif x >= - basegrid && x < 0
%         y = ceil(1 / -basegrid);
%     elseif x >= -topgrid && x < - basegrid
%         y = ceil(1 / x);
%     elseif x < -topgrid
%         y = 0
%     end
%
%     if y > 0
%     y = floor(y * 0.1);
```

```
%     else
%     y = ceil(y * 0.1);
%     end

y = atan(x) / (pi/2) * 1000;

  end
```

IkunBuySellClassic.m
网格买入卖出函数（经典法）
使用 MATLAB 语言编写

```
function [BuySellArray, ClosedValue] = IkunBuySellClassic(InputMatrix, PredictArray, InitValue, DealMatrix)
%%
% 2022.11.4
% 网格买入卖出函数（最经典）
% 每 1 分钱（0.01 元）一格
% 每个买入/卖出 100 个

% InputMatrix:
% 输入矩阵：列数：前 n 个点          size: 行 * 列
%           行数：样本个数
%
% PredictArray:
% 输入数组：1 列                     size: 行 * 列
%           行数：样本个数
%
% InitValue:
% 初始持仓数
%
%
% DealMatrix (m, n=10)
% m 行：m 个分别的股票
% n 列：前 10 时刻各自的成交量（非累计）
%
%
% BuySellArray: 买卖数量决定
% 输出数组：2 列                     size: 行 * 列
%           行数：样本个数
%           列数：第 1 列：买/卖的数量
% ClosedValue:
% 最终持仓数
%%
gridsize = 0.01;                    % 每格大小（元）
```

```
33.diffrence = 0;
34.sellnumber = 0;
35.ClosedValue = 0;
36.basecountpergrid = min(5, 0.015 * max(DealMatrix(:, 10)));
          % 每个买入或卖出的股数
37.BuySellArray = zeros([], 1);
38.for i = 1 : size(InputMatrix, 1)    % 逐行操作
39.    difference = InputMatrix(i, size(InputMatrix, 2)) - PredictA
   rray(i, 1);
40.    sellnumber = ceil(ceil(difference / gridsize) * basecountper
   grid);
41.    BuySellArray (i, 1) = sellnumber;
42.end
43.ClosedValue = InitValue + sum(BuySellArray(:, 1));
    44.    end
```

IkunVibrantPredict001.m
适用于动态型股票的股票单价预测函数
使用 MATLAB 语言编写

```
    1. function [YmyIkunBabe,IkunXksFinalls,IkunAaffFinalls] = Ik
unVibrantPredict001(IloveIkunEkkks,~,~)
2. %IkunVibrantPredict001 neural network simulation function.
3. % 2022.11.4
4. % 动态型股票预测工具
5. %
6. % [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
7. %
8. %   X = 前 10 个时刻的单价 [ [], 10 ]
9. %   Each X{1,ts} = Qx10 matrix, input #1 at timestep ts.
10.%
11.% and returns:
12.%   Y = 下一个时刻（时刻 11）的单价 [ [], 1 ]
13.%   Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
14.%
15.% where Q is number of samples (or series) and TS is the number
   of timesteps.
16.
17.%#ok<*RPMT0>
18.
19.% ===== NEURAL NETWORK CONSTANTS =====
20.
21.% Input 1
22.Ikun_Eks1Stepps.xoffset = [4.24256610824746;4.24256610824746;4.2
   4256610824746;4.24256610824746;4.24256610824746;4.24256610824746
```

```
;4.24256610824746;4.24256610824746;4.24256610824746;4.24256610824746];
Ikun_Eks1Stepps.gain = [0.0212080390745486;0.0212080390745486;0.0212080390745486;0.0212080390745486;0.0212080390745486;0.0212080390745486;0.0212080390745486;0.0212080390745486;0.0212080390745486];
Ikun_Eks1Stepps.ymin = -1;

% Layer 1
BloodyIkun1 = [-1.960780873534755608 4;-1.5512494425426599509;0.5196481383631084138 8;-0.32839525230466609651;0.16764248042493773427;0.16209003379596614858;-0.29852571141016986944;0.76269885255049230111;1.2536669590492042214;-1.4009266708133554236];
IterationWayIkun1_1 = [1.066163798171222421 -0.00907774006146117 20303 0.40749884105355521635 0.61027406982139109637 0.5430527710 0764770548 -0.1762842734203632078 0.29398811817218106768 0.02127 0298613613725786 0.4358680247411998443 0.008428155479710345773 8;0.283762178850379021 -0.38413987895757140123 0.18064971928648343 447 0.034924573922473403642 -0.24174474177361421345 0.4580658767 3842802388 -0.12217061801523548814 0.31367794821257743987 -0.178 85200986800983691 0.79614281613384185743;-0.70587722817030573719 0.38845872177272267045 0.90028649599149757066 -0.37486097604385 004711 -0.16925970101872134244 -0.11617706547148318186 -1.263349 5336688504995 0.56924271745894883257 -0.30496477677755318147 0.0 3429081999630208960 1;0.86170222837122034676 0.180435542247991659 3 0.28929802111338498438 -0.62640704197646013274 0.2098981339847 312555 -0.85124623735201798702 -0.23319188296694368101 0.1858602 359384422531 0.82557492274149624212 0.6566416803662999957;0.4745 1071954423612587 -0.31538580486357481458 0.17619689754092665579 0.83306390650428729927 -0.8730042540841294052 0.1989787186112767 5031 -0.10767768511247433061 0.22821467728463223912 0.0285489634 24765073926 0.18052843649312810825;0.39782381381228892891 -0.144 15914061942325386 -0.45616137895821146087 0.54117647153286807526 -0.29973135778527237871 -0.54460028450579078818 0.4799330478714 4380752 -0.36353279122670240087 0.30426815478204038978 -0.988779 07772590878377;-0.51229789036903861099 -0.092545175978048427323 -0.03745603742451596857 -0.72067091589226850434 0.93212772721612 946469 -0.24982996466414181258 0.25625286670752794871 0.24226664 071340620055 0.77475562822398080787 -0.17875740528616326785;0.63 089844870098676921 0.33618849378797616145 0.14623538752848849742 -0.4313769072529046178 0.32276561433612382945 -0.55062977909286 159317 0.085667623156283942421 0.19034727329900569259 0.05689373 3005752511467 0.5621630542577540357;0.082247213536436200587 -0.0 0084792494052055586776 0.0085001120179894300299 0.04424187539382
```

```
    832407 0.083737483381090216383 0.17869418501342745897 0.13160419
    509659679327 0.025799543717397410009 0.018011551811082150004 0.2
    1469793902535119767;-0.30468736088315490074 1.475451773553189349
    6 0.44855108835894186292 0.52295125133084063851 0.17232363392567
    623395 -0.60743310358158597584 -0.17863018473695763122 -0.100713
    44445734134454 0.1640344668700584041 -1.3292792387606910953];
29.
30.% Layer 2
31.BloodyIkun2 = 0.27640291685644718944;
32.LongWayIkun2_1 = [0.001489056331631473471 1 0.486110242700109052
    2 -0.18381181071863417209 -0.012124688473971806033 0.23679920736
    104345669 -0.19554093389799889757 0.20388370703411404228 0.06444
    4848737714041653 0.99069049972105349688 0.65313543205382440515];
33.
34.% Output 1
35.Ikun_WaaiStepps1.ymin = -1;
36.Ikun_WaaiStepps1.gain = 0.0212080390745486;
37.Ikun_WaaiStepps1.xoffset = 4.24256610824746;
38.
39.% ===== SIMULATION ========
40.
41.% Format Input Arguments
42.isCellX = iscell(IloveIkunEkkks);
43.if ~isCellX
44.    IloveIkunEkkks = {IloveIkunEkkks};
45.end
46.
47.% Dimensions
48.TS = size(IloveIkunEkkks,2); % timesteps
49.if ~isempty(IloveIkunEkkks)
50.    Q = size(IloveIkunEkkks{1},1); % samples/series
51.else
52.    Q = 0;
53.end
54.
55.% Allocate Outputs
56.YmyIkunBabe = cell(1,TS);
57.
58.% Time loop
59.for ts=1:TS
60.
61.    % Input 1
62.    IloveIkunEkkks{1,ts} = IloveIkunEkkks{1,ts}';
63.    Xp1 = mapminmax_apply(IloveIkunEkkks{1,ts},Ikun_Eks1Stepps);
```

```
64.
65.     % Layer 1
66.     a1 = tansig_apply(repmat(BloodyIkun1,1,Q) + IterationWayIkun
   1_1*Xp1);
67.
68.     % Layer 2
69.     a2 = repmat(BloodyIkun2,1,Q) + LongWayIkun2_1*a1;
70.
71.     % Output 1
72.     YmyIkunBabe{1,ts} = mapminmax_reverse(a2,Ikun_WaaiStepps1);
73.     YmyIkunBabe{1,ts} = YmyIkunBabe{1,ts}';
74.end
75.
76.% Final Delay States
77.IkunXksFinalls = cell(1,0);
78.IkunAaffFinalls = cell(2,0);
79.
80.% Format Output Arguments
81.if ~isCellX
82.     YmyIkunBabe = cell2mat(YmyIkunBabe);
83.end
84.end
85.
86.% ===== MODULE FUNCTIONS ========
87.
88.% Map Minimum and Maximum Input Processing Function
89.function y = mapminmax_apply(x,settings)
90.y = bsxfun(@minus,x,settings.xoffset);
91.y = bsxfun(@times,y,settings.gain);
92.y = bsxfun(@plus,y,settings.ymin);
93.end
94.
95.% Sigmoid Symmetric Transfer Function
96.function a = tansig_apply(n,~)
97.a = 2 ./ (1 + exp(-2*n)) - 1;
98.end
99.
100.  % Map Minimum and Maximum Output Reverse-Processing Function
101.  function x = mapminmax_reverse(y,settings)
102.  x = bsxfun(@minus,y,settings.ymin);
103.  x = bsxfun(@rdivide,x,settings.gain);
104.  x = bsxfun(@plus,x,settings.xoffset);
    105.  end
```

IkunStaticPredict001.m
适用于动态型股票的股票单价预测函数
使用 MATLAB 语言编写

```
function [YmyIkunBabe,IkunXksFinalls,IkunAaffFinalls] = IkunStaticPredict001(IloveIkunEkkks,~,~)
%IkunStaticPredict001 neural network simulation function.
% 2022.11.4
% 静态型股票预测工具
%
% [Y] = myNeuralNetworkFunction(X,~,~) takes these arguments:
%
%   X = 前 10 个时刻的单价 [ [], 10 ]
%   Each X{1,ts} = Qx10 matrix, input #1 at timestep ts.
%
% and returns:
%   Y = 下一个时刻（时刻 11）的单价 [ [], 1 ]
%   Each Y{1,ts} = Qx1 matrix, output #1 at timestep ts.
%
% where Q is number of samples (or series) and TS is the number of timesteps.

%#ok<*RPMT0>

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
Ikun_Eks1Stepps.xoffset = [0;0;0;0;0;0;0;0;0;0];
Ikun_Eks1Stepps.gain = [0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473;0.0368330423961473];
Ikun_Eks1Stepps.ymin = -1;

% Layer 1
BloodyIkun1 = [1.5741523261778780363;-0.92054562680776608197;-0.81307114897169485701;0.1145514640261342254;-0.5014785491645796788989;-0.091357223546991273011;-0.74561288396174951743;1.1394570559354848527;1.2777376174664178965;-1.6714543228740783576];
IterationWayIkun1_1 = [-0.78235435292572830246 -0.2504091350610265776 0.39346185520231552468 -0.72045156701183432624 -0.34035571882064763471 -0.59311198956301358631 -0.76682181819543882018 0.03708831636947604315 5 0.63886873658158382927 -0.65364450604112456222;-0.32368946928660913276 -0.9904865713428023799 -0.40768114365701690893 0.35863323792395590539 0.37825436141052548589 0.37490
```

```
346740650593027 0.35450794216309156059 0.31627393306264539596 0.
23724966320733806291 0.22471930959988700716;0.222136567641474380
61 0.45851917280604548299 0.81877360098452522941 0.4487694650941
7081632 -0.68149439998728933432 -0.23649032511234932552 -0.85316
473319526819896 0.26834935724326725026 -0.40651428812583034755 0
.026130025938361396698;-0.70084716868629992614 -0.60259298899046
020992 0.39065938154307205954 0.27244273964763343621 0.188070793
45729138531 -0.26041853910348561341 -0.53775963028509787822 1.05
20106576244367957 -0.40498611215167756860 0.23275370791811764160;0
.52940002601364288815 0.042977844401281294073 0.2958765042832882
1084 0.21043192028792997994 -0.12664073563742189732 -0.477136731
00520992421 -0.29082310114633791454 -0.59294397220592243158 0.33
002691638658038231 -0.18013168932655218568;-0.537963974577105541
46 -0.69099771683579314097 0.40939519423226999884 0.309584930557
82349002 0.69960299310564544317 -0.5028793765724610676 -0.317211
54143419261207 0.86328550007333970662 0.042497618836963013678 0.
34825855371801206495;-0.66847902269787096152 0.54865742275173079
445 0.30917117330669841024 0.27065896724007304508 0.712358824577
94930112 0.4375111759653239063 0.16959011546591198916 0.29035544
029047311021 0.72484667185252179135 0.44370143067027212336;0.390
12211677442559798 0.047360367469080191793 -0.3416314727731539768
7 0.55476989363298834235 -0.86993660405517236889 0.5666534692480
6991868 -0.32838999075670682881 -0.88187797604307083255 -0.33707
967118083781433 0.039475109029116446646;0.24082771342592065866 -
0.51859121881627656681 0.25083354269991037233 0.2599482048208726
6367 0.36393627010973522706 -0.39420510393517965708 -0.046780936
929910005295 0.0042477738715207781148 0.63454453799631582722 0.0
325169974869047237 46;-0.76519480482749768768 0.19302198267844028
479 -0.3341701403869153375 -0.89890906933835301729 0.25889998820
241483246 -0.1125155488809339438 -0.9533095202515750044 -0.56470
769225270400771 -0.29206227388948924339 -0.2044278202240026876];

% Layer 2
BloodyIkun2 = 0.29474327878742812015;
LongWayIkun2_1 = [0.005264171828304067477 0.35256522496677850098
 0.82112752831978930157 -0.79591115548892477083 -0.6938395618175
6434312 0.48051843051024323294 -0.002535485929759656066 6 -0.1311
9680566494537621 0.49693264210030735351 -0.000740830367598077350
71];

% Output 1
Ikun_WaaiStepps1.ymin = -1;
Ikun_WaaiStepps1.gain = 0.0368330423961473;
Ikun_WaaiStepps1.xoffset = 0;
```

```
38.
39.% ===== SIMULATION ========
40.
41.% Format Input Arguments
42.isCellX = iscell(IloveIkunEkkks);
43.if ~isCellX
44.    IloveIkunEkkks = {IloveIkunEkkks};
45.end
46.
47.% Dimensions
48.TS = size(IloveIkunEkkks,2); % timesteps
49.if ~isempty(IloveIkunEkkks)
50.    Q = size(IloveIkunEkkks{1},1); % samples/series
51.else
52.    Q = 0;
53.end
54.
55.% Allocate Outputs
56.YmyIkunBabe = cell(1,TS);
57.
58.% Time loop
59.for ts=1:TS
60.
61.    % Input 1
62.    IloveIkunEkkks{1,ts} = IloveIkunEkkks{1,ts}';
63.    Xp1 = mapminmax_apply(IloveIkunEkkks{1,ts},Ikun_Eks1Stepps);
64.
65.    % Layer 1
66.    a1 = tansig_apply(repmat(BloodyIkun1,1,Q) + IterationWayIkun
   1_1*Xp1);
67.
68.    % Layer 2
69.    a2 = repmat(BloodyIkun2,1,Q) + LongWayIkun2_1*a1;
70.
71.    % Output 1
72.    YmyIkunBabe{1,ts} = mapminmax_reverse(a2,Ikun_WaaiStepps1);
73.    YmyIkunBabe{1,ts} = YmyIkunBabe{1,ts}';
74.end
75.
76.% Final Delay States
77.IkunXksFinalls = cell(1,0);
78.IkunAaffFinalls = cell(2,0);
79.
80.% Format Output Arguments
```

```
if ~isCellX
    YmyIkunBabe = cell2mat(YmyIkunBabe);
end
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
y = bsxfun(@minus,x,settings.xoffset);
y = bsxfun(@times,y,settings.gain);
y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n,~)
a = 2 ./ (1 + exp(-2*n)) - 1;
end

 % Map Minimum and Maximum Output Reverse-Processing Function
 function x = mapminmax_reverse(y,settings)
 x = bsxfun(@minus,y,settings.ymin);
 x = bsxfun(@rdivide,x,settings.gain);
 x = bsxfun(@plus,x,settings.xoffset);
  end
```

AnswerProfitRate00.mlx
多功能数据输出、绘图脚本
使用 MATLAB 语言编写

```
clc
%iplot001 = 1;
%iplot001 = iplot001 + 1


[HoldCountTest, CashTest, AveCost, SellNumber, AveProfitRate, MaximumRetracement, TradeNumber] = PackUpOutputFunstion (Input11ArrayOrigin, Input11DealOrigin, 150000, 0, origin(3,1));
%%如果要跑一天数据，记得改 PackUpJudgeHistory 的 21 天哦

subplot(1, 2, 1)
bar(AveProfitRate)
grid on;
xlabel('交易次数');
ylabel('收益率');
```

```
title(['改进前']);
origin(3,1) %前一天的收盘价
tempProfitRate = mean(AveProfitRate)         %平均收益率
holdcounttest = HoldCountTest(size(HoldCountTest, 1), 1)     %最终持仓数量
cashtest = CashTest(size(CashTest, 1), 1) %最终持有金额
MaximumRetracement  %最大回撤
HoldCountTest(size(HoldCountTest, 1)-1, 1)
TradeNumber(size(TradeNumber, 1)-1, 1)      %交易量
    sum(Input11DealOrigin) / 10       %市场总交易量的 10%
```

EntropyMethod.m
熵权法评价函数
使用 MATLAB 语言编写

```
%%
% 熵权法评价函数
function [scores, evaluation] = EntropyMethod (givendata)
%%
% 输入矩阵: givendata[m, n]
% m 行：对象(样本)
% n 列：各个对象的相关指标
givendata = transpose(givendata);
tempdata = mapminmax(givendata', 0.002, 1); %标准化到 0.002-1 区间
tempdata = tempdata';

%%
% 得到信息熵
[m, n] = size(tempdata);
p = zeros(m,n);
for j = 1 : n
    p(:, j) = tempdata(:, j) / sum(tempdata(:, j));
end
for j = 1 : n
    % EE 是一个行向量，一共有 n 个元素，对应得到的是每一个指标的信息熵。
    % 这里所针对的是每一个指标，而不是每一个样本!
    EE(j) = -1 / log(m) * sum(p(:, j) .* log(p(:, j)));
end

%%
% 计算权重
%
evaluation = (1 - EE) / sum(1 - EE);

%%
```

```
% 计算得分
scal = tempdata * evaluation';
scores = 100 * scal / max(scal);
%scores = mapminmax(log(log(log(log(log(100 * scal / max(scal) +
    1)+1)+1)+1)+1), 0, 1);
%scores = mapminmax((log(100 * scal / max(scal) + 1)), 0, 1);


%%
% 输出：行向量，scores
end
```

TopsisMethod.m
TOPSIS 法评价函数
使用 MATLAB 语言编写

```
%%
% TOPSIS 评价函数
function [score] = TopsisMethod(X, W)
%%
%X 输入的数据，W 各指标的权重
W = transpose(W);
[n,~]=size(X);
%Z=zscore(X);
Z = X ./ repmat(sum(X.*X) .^ 0.5, n, 1); %矩阵归一化
V_D = sum(((Z - repmat(max(Z),n,1)) .^ 2 ) .* repmat(W,n,1) ,2)
.^ 0.5;
V_X = sum(((Z - repmat(min(Z),n,1)) .^ 2 ) .* repmat(W,n,1) ,2)
.^ 0.5;
S = V_X ./ (V_D+V_X); %未归一化得分
Score_S = S / sum(S); %归一化得分,即为每个企业的投资风险评分，值越大，
投资风险也越大
% score=Score_S;
 score=100*Score_S/max(Score_S);
  end
```