法律声明

本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

第六课

进入专业量化赛道的必修课

量化交易实战：

策略编写与系统搭建

内容介绍

- 最根本的数据处理：Level1 Ticks
- 深入讨论两个实战量化策略
- 量化策略的完整研发和系统搭建
- 如何优化你的量化策略：再谈人工智能
- 预告：你未来必踩的坑－怎么上实盘？

最根本的数据处理

Level1的tick数据处理

为什么强调行情数据的处理

量化的基础

获取价格走势

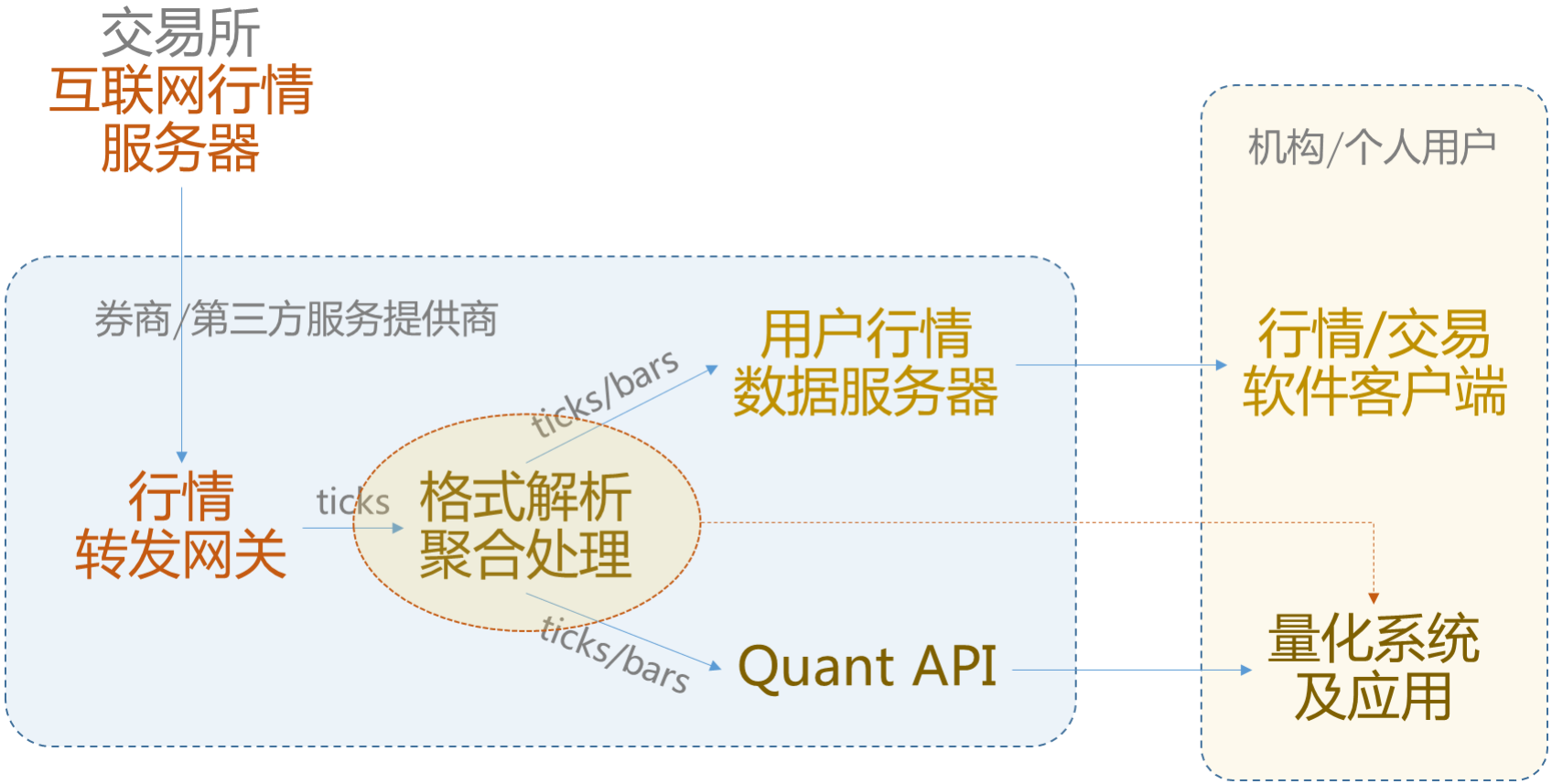
计算技术指标，
产生交易信号

统计分析建模，
制定量化策略

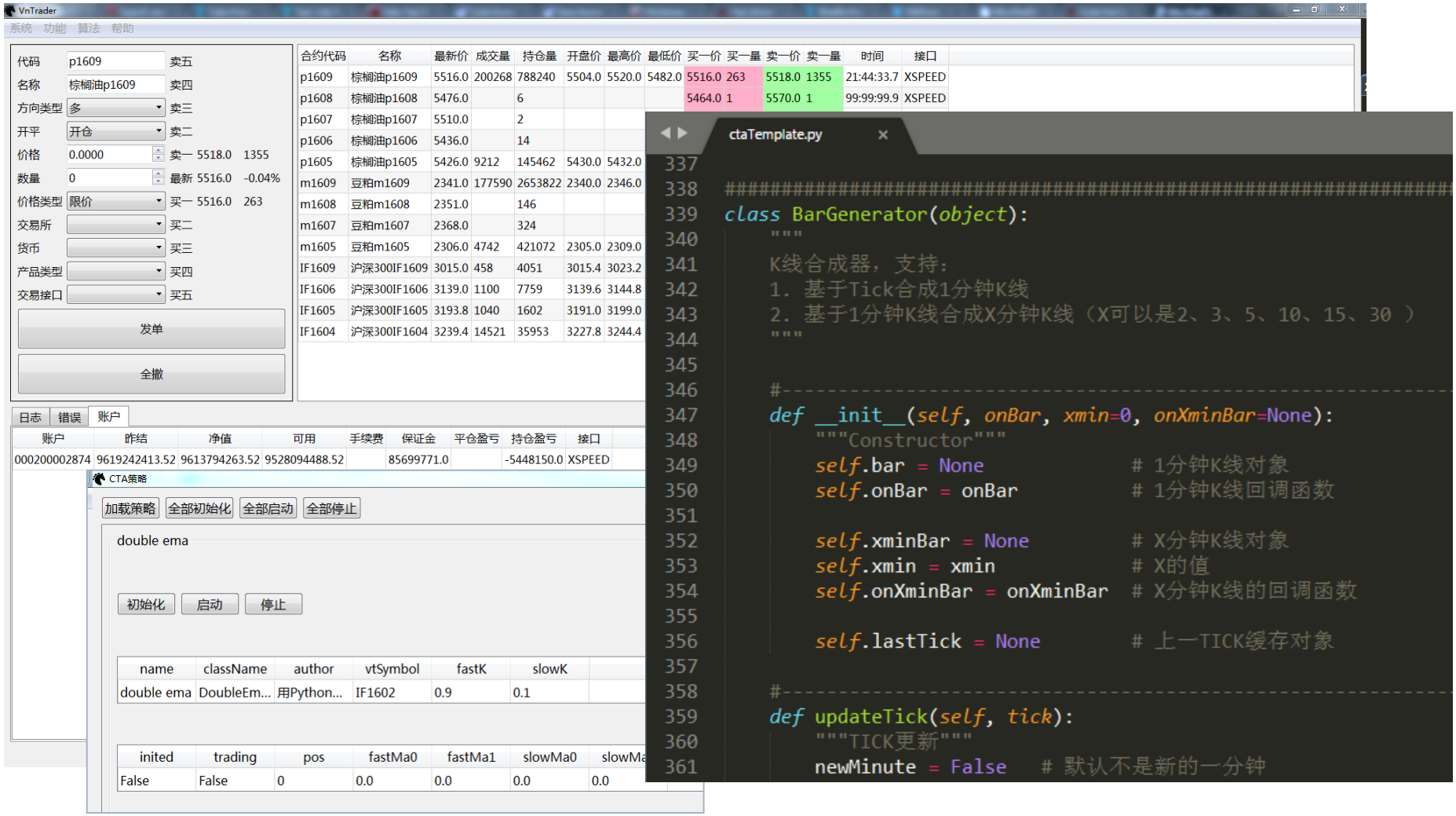
执行程序化交易

排查错误异常

搭建量化平台的需要


交易所
互联网行情
服务器
券商/第三方服务提供商
行情
转发网关
ticks
格式解析
聚合处理
ticks/bars
用户行情
数据服务器
ticks/bars
Quant API
机构/个人用户
行情/交易
软件客户端
量化系统
及应用

程序化交易的需要


VnTrader
系统 功能 算法 帮助
代码 p1609
名称 棕榈油p1609
方向类型 多
开平 开仓
价格 0.0000
数量 0
价格类型 限价
交易所
货币
产品类型
交易接口
卖五
卖四
卖三
卖二
卖一 5518.0 1355
最新 5516.0 -0.04%
买一 5516.0 263
买二
买三
买四
买五
发单
全撤
合约代码 名称 最新价 成交量 持仓量 开盘价 最高价 最低价 买一价 买一量 卖一价 卖一量 时间 接口
p1609 棕榈油p1609 5516.0 200268 788240 5504.0 5520.0 5482.0 5516.0 263 5518.0 1355 21:44:33.7 XSPEED
p1608 棕榈油p1608 5476.0 6 5464.0 1 5570.0 1 99:99:99.9 XSPEED
p1607 棕榈油p1607 5510.0 2
p1606 棕榈油p1606 5436.0 14
p1605 棕榈油p1605 5426.0 9212 145462 5430.0 5432.0
m1609 豆粕m1609 2341.0 177590 2653822 2340.0 2346.0
m1608 豆粕m1608 2351.0 146
m1607 豆粕m1607 2368.0 324
m1605 豆粕m1605 2306.0 4742 421072 2305.0 2309.0
IF1609 沪深300IF1609 3015.0 458 4051 3015.4 3023.2
IF1606 沪深300IF1606 3139.0 1100 7759 3139.6 3144.8
IF1605 沪深300IF1605 3193.8 1040 1602 3191.0 3199.0
IF1604 沪深300IF1604 3239.4 14521 35953 3227.8 3244.4
日志 错误 账户
账户 昨结 净值 可用 手续费 保证金 平仓盈亏 持仓盈亏 接口
000200002874 9619242413.52 9613794263.52 9528094488.52 85699771.0 -5448150.0 XSPEED
CTA策略
加载策略 全部初始化 全部启动 全部停止
double ema
初始化 启动 停止
name className author vtSymbol fastK slowK
double ema DoubleEm... 用Python... IF1602 0.9 0.1
inited trading pos fastMa0 fastMa1 slowMa0 slowMa
False False 0 0.0 0.0 0.0 0.0
ctaTemplate.py
337
338 ##
339 class BarGenerator(object):
340 """
341 K线合成器，支持：
342 1. 基于Tick合成1分钟K线
343 2. 基于1分钟K线合成X分钟K线（X可以是2、3、5、10、15、30 ）
344 """
345
346 #--
347 def __init__(self, onBar, xmin=0, onXminBar=None):
348 """Constructor"""
349 self.bar = None # 1分钟K线对象
350 self.onBar = onBar # 1分钟K线回调函数
351
352 self.xminBar = None # X分钟K线对象
353 self.xmin = xmin # X的值
354 self.onXminBar = onXminBar # X分钟K线的回调函数
355
356 self.lastTick = None # 上一TICK缓存对象
357
358 #--
359 def updateTick(self, tick):
360 """TICK更新"""
361 newMinute = False # 默认不是新的一分钟

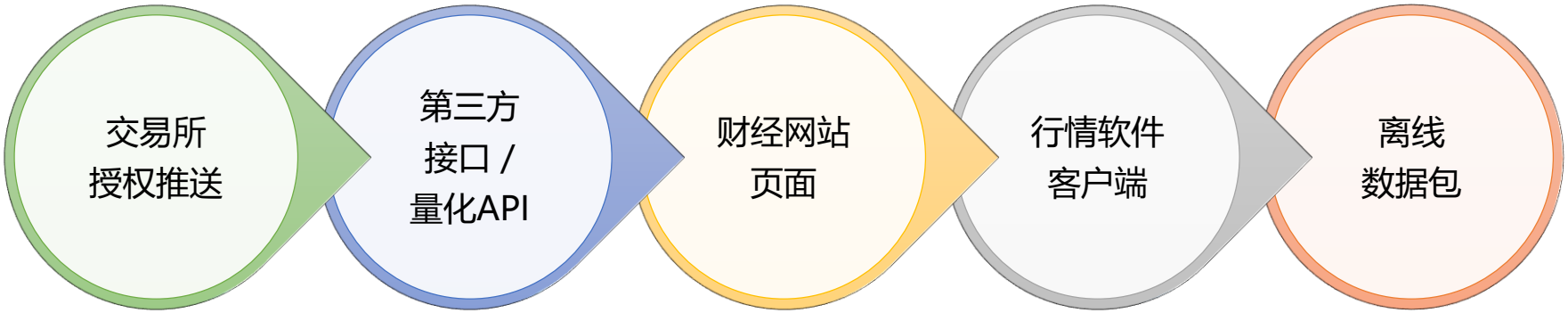
排查错误的需要

- 不同交易软件处理后的分钟数据不完全一致
  - 切分点不同（OHLC值的差异）
  - 左右边界（时间戳）靠拢方式不同
- 附加信息的处理
  - 交易状态
  - 涨跌停价的处理
  - 资金流向（超大中小单）统计
  - 换手率、量比、成交量/额等的累积

Level 1 vs. Level 2

Tick级别	Level 1	Level 2
行情数据类别	快照行情	逐笔行情
推送频度	固定间隔（~3秒）	委托队列和逐笔成交明细
盘口报价显示	五档行情	十档行情（深交所千档）
使用方式	免费（客户端）	付费使用

行情数据来源（例）



上海
证券交易所
深圳
证券交易所

Wind
东财Choice
掘金量化
Tushare
通联数据
JQData

新浪财经
东方财富
巨潮资讯

通达信
东方财富
同花顺
飞狐交易师
图表交易软件
券商软件

交易所
Wind
淘宝
预测者

Ticks 分时 Bars（K线/蜡烛图）

时间	2018-03-15 14:59:04		
证券代码	600000	证券名称	浦发银行
交易所	沪A	交易状态	交易
昨收盘价	12.39	成交量	17287774.0
今开盘价	12.35	成交额	213972089.0
今最高价	12.41	委比	34.18
今最低价	12.34	换手率	0.0006
最新价	12.38	量比	0.7274
涨停价	13.63	涨跌额	-0.01
跌停价	11.15	涨跌幅%	-0.0008
卖五价	12.43	卖五量	61400.0
卖四价	12.42	卖四量	473398.0
卖三价	12.41	卖三量	177146.0
卖二价	12.40	卖二量	228798.0
卖一价	12.39	卖一量	178500.0
买一价	12.38	买一量	570273.0
买二价	12.37	买二量	168500.0
买三价	12.36	买三量	258100.0
买四价	12.35	买四量	800658.0
买五价	12.34	买五量	484200.0

证券代码	600000
证券名称	浦发银行
日期	2013-01-04
交易状态	交易
开盘价	10.1
最高价	10.26
最低价	9.93
收盘价	10.02
昨收盘价	9.92
涨停价	10.91
跌停价	8.93
成交量	261436810
成交额	2633796691.0
换手率	1.7519
量比	1.6842
涨跌额	0.10
涨跌幅%	0.010080
调整因子	6.90526

任务定义

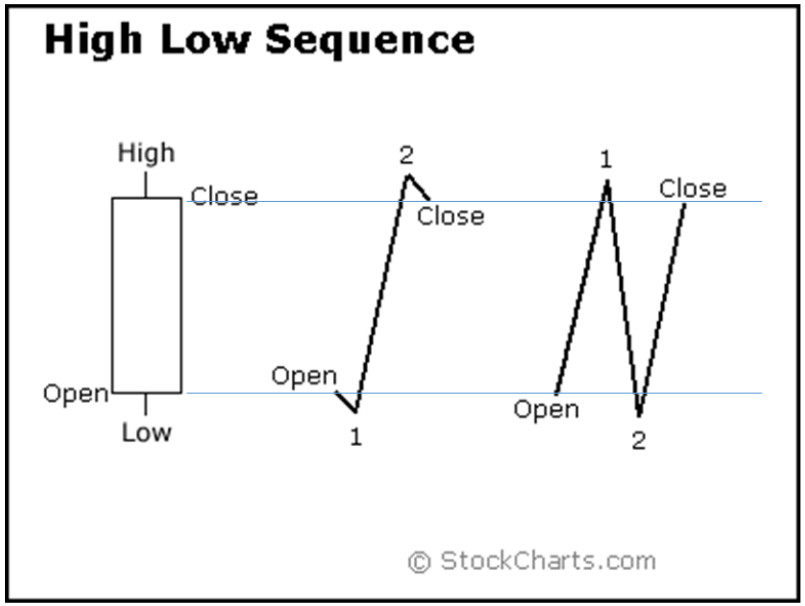
行情数据累积：

- Ticks -> 1min/日
- 1min -> 5m/15m/30m/60m(1h)/日
- 日K线（不复权 vs. 复权）
- 日 -> 周/月

特别注意事项：

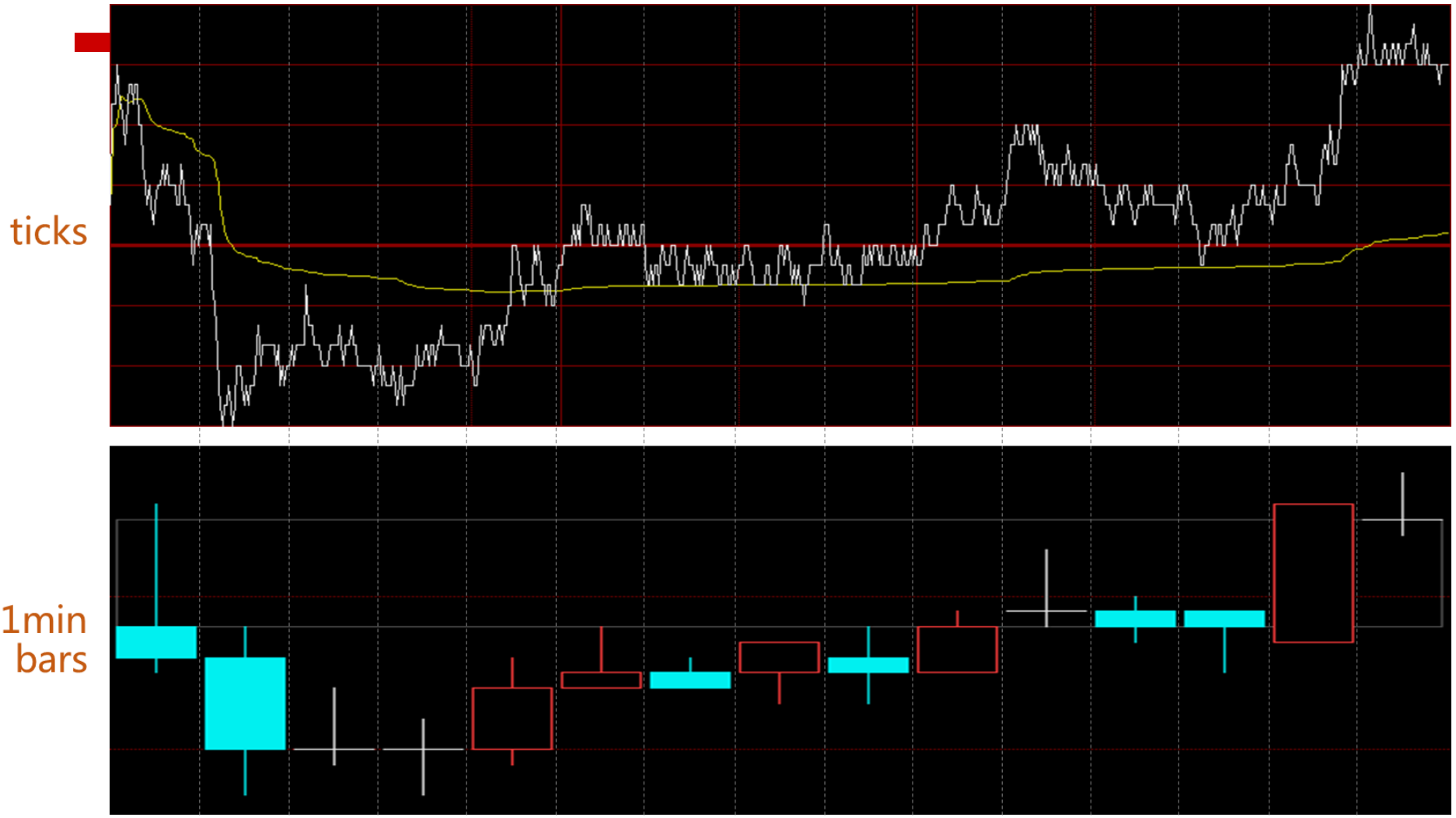
- 集合竞价
- 除权/除息（复权因子）
- 涨跌停价
- 交易状态（停复牌）

从ticks到1分钟K线



Level-1 ticks	
code	证券代码
time	快照时间
open	当日开盘价
latest	最新价
high	当日最高价
low	当日最低价

1-min bars	
code	证券代码
time	K线时间
open	开盘价
high	最高价
low	最低价
close	收盘价


ticks
1min
bars

```python
""" Ticks到1min线的转换，该脚本每天运行一次 """

# 所有累积完成的1min线放到一个dict中，以股票代码作为key
bars_by_codes = dict()

def ticks_to_1m_bars(one_tick):
    code = one_tick["code"]          # 该tick所属的股票代码
    ptr = bars_by_codes.get(code)  # 该代码对应的1min线计算用的数据结构
    if ptr is None:
        ptr = bars_by_codes[code] = \
            {
                "bars_queue":     list(),  # 已完成累积的1min线的队列
                "last_bar_time":  None,    # 上次tick对应的K线时间戳
                "day_high":       None,    # 截至当前时刻，当日的最高价
                "day_low":        None,    # 截至当前时刻，当日的最低价
                "current_bar":    dict()   # 当前累积中的1min线数据结构
            }

    # 临时取出当前工作的数据结构，便于后面的读写
    bars_queue = ptr["bars_queue"]
    last_bar_time = ptr["last_bar_time"]
    day_high = ptr["day_high"]
    day_low = ptr["day_low"]
    current_bar = ptr["current_bar"]

    # 接收到的每个tick中的时间戳格式举例："2018-03-12 09:41:27"
    current_bar_time = one_tick["time"][:-3]
```

```
28
29      if current_bar_time != last_bar_time:  # 进入下一分钟的第一个tick
30          if last_bar_time is not None:  # 不是当天的第一个tick
31              # 把前面刚累计好的1min线输出
32              current_bar["time"] += ":00"  # 把时间戳中的"秒"字段补成零
33              bars_queue.append(current_bar)  # 添加到1min线队列的末尾
34
35          # 把新的一根1min线的工作数据结构初始化
36          current_bar["time"] = current_bar_time
37          if last_bar_time is None:  # 当日第一个tick（也是第一个1min线）
38              current_bar["open"] = one_tick["open"]  # 当日开盘价就是第一根1min线的开盘价
39              day_high = one_tick["high"]  # 以截至到第一个tick时的当日最高价初始化
40              day_low = one_tick["low"]    # 以截至到第一个tick时的当日最低价初始化
41          else:  # 非当日第一个tick
42              current_bar["open"] = one_tick["latest"]  # 以tick最新价初始化该1min线的开盘价
43              if one_tick["high"] > day_high:  # 本分钟内见到了当日的新的最高价
44                  current_bar["high"] = one_tick["high"]  # 更新本根1min线的最高价
45                  day_high = one_tick["high"]  # 记录到目前为止当日的最高价
46              elif one_tick["latest"] > current_bar["high"]:  # tick最新价创该分钟内新高
47                  current_bar["high"] = one_tick["latest"]  # 更新本根1min线的最高价
48
49              if one_tick["low"] < day_low:  # 本分钟内见到了当日的新的最低价
50                  current_bar["low"] = one_tick["low"]  # 更新本根1min线的最低价
51                  day_low = one_tick["low"]  # 记录到目前为止当日的最低价
52              elif one_tick["latest"] < current_bar["low"]:  # tick最新价创该分钟内新低
53                  current_bar["low"] = one_tick["latest"]  # 更新本根1min线的最低价
```

```
54
55          current_bar["close"] = one_tick["latest"]  # 以tick最新价更新该1min线的收盘价
56
57          ptr["last_bar_time"] = current_bar_time  # 记录当前1min线的时间戳
58
59      else:  # 还在同一个分钟内的tick
60          if one_tick["high"] > day_high:  # 本分钟内见到了当日的新的最高价
61              current_bar["high"] = one_tick["high"]  # 更新本根1min线的最高价
62              day_high = one_tick["high"]  # 记录到目前为止当日的最高价
63          elif one_tick["latest"] > current_bar["high"]:  # tick最新价创该分钟内新高
64              current_bar["high"] = one_tick["latest"]  # 更新本根1min线的最高价
65
66          if one_tick["low"] < day_low:  # 本分钟内见到了当日的新的最低价
67              current_bar["low"] = one_tick["low"]  # 更新本根1min线的最低价
68              day_low = one_tick["low"]  # 记录到目前为止当日的最低价
69          elif one_tick["latest"] < current_bar["low"]:  # tick最新价创该分钟内新低
70              current_bar["low"] = one_tick["latest"]  # 更新本根1min线的最低价
71
72          current_bar["close"] = one_tick["latest"]  # 以tick最新价更新该1min线的收盘价
73
74      # 更新保存当日的最高最低价，其它数据结构在上面也已更新
75      ptr["day_high"] = day_high
76      ptr["day_low"] = day_low
77
```


1min
bars
N-min
bars

更大周期的行情数据累积

- Ticks/1min线 -> 日K线（不复权）
- 日K线的复权处理（预存或实时计算）
- 日K线 -> 周K线 / 月K线
- 复权的日K线 -> 复权的周K线 / 月K线

【思考题】

可以从“不复权”的周线直接计算得出“前复权”的周线吗？

A. 可以　　B. 不可以

特别注意事项

- 集合竞价
- 除权/除息 vs. 复权

涨跌停价的处理

- 挂单成交的概率
  - 封涨停时通常买不进
  - 封跌停时通常卖不出

- 涨跌停价格的获取
  - 第三方的量化API（通常已有）
  - 交易所行情接收（盘前信息库、集合竞价消息）
  - 自己计算（正常、ST/*ST、N）

交易状态的处理

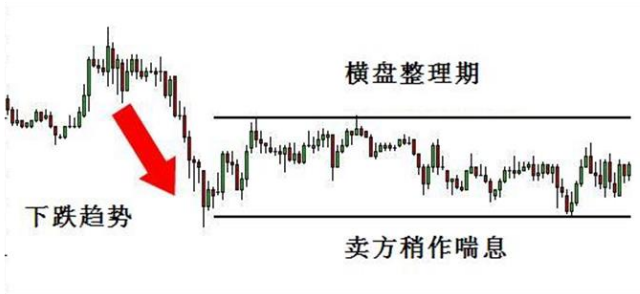
- 记录交易状态（正常交易或停/复牌）的重要性
  - 回测错误和无效交易指令
  - 技术指标计算误差

- 交易状态的获取
  - 第三方的量化API（通常已有）
  - 交易所行情接收
  - 盘前/盘中信息库
  - Tick消息字段
  - 紧急公告消息（临时停牌）

在期货交易中常见的有效策略

Hans123策略及增强

从横盘突破型策略说起...

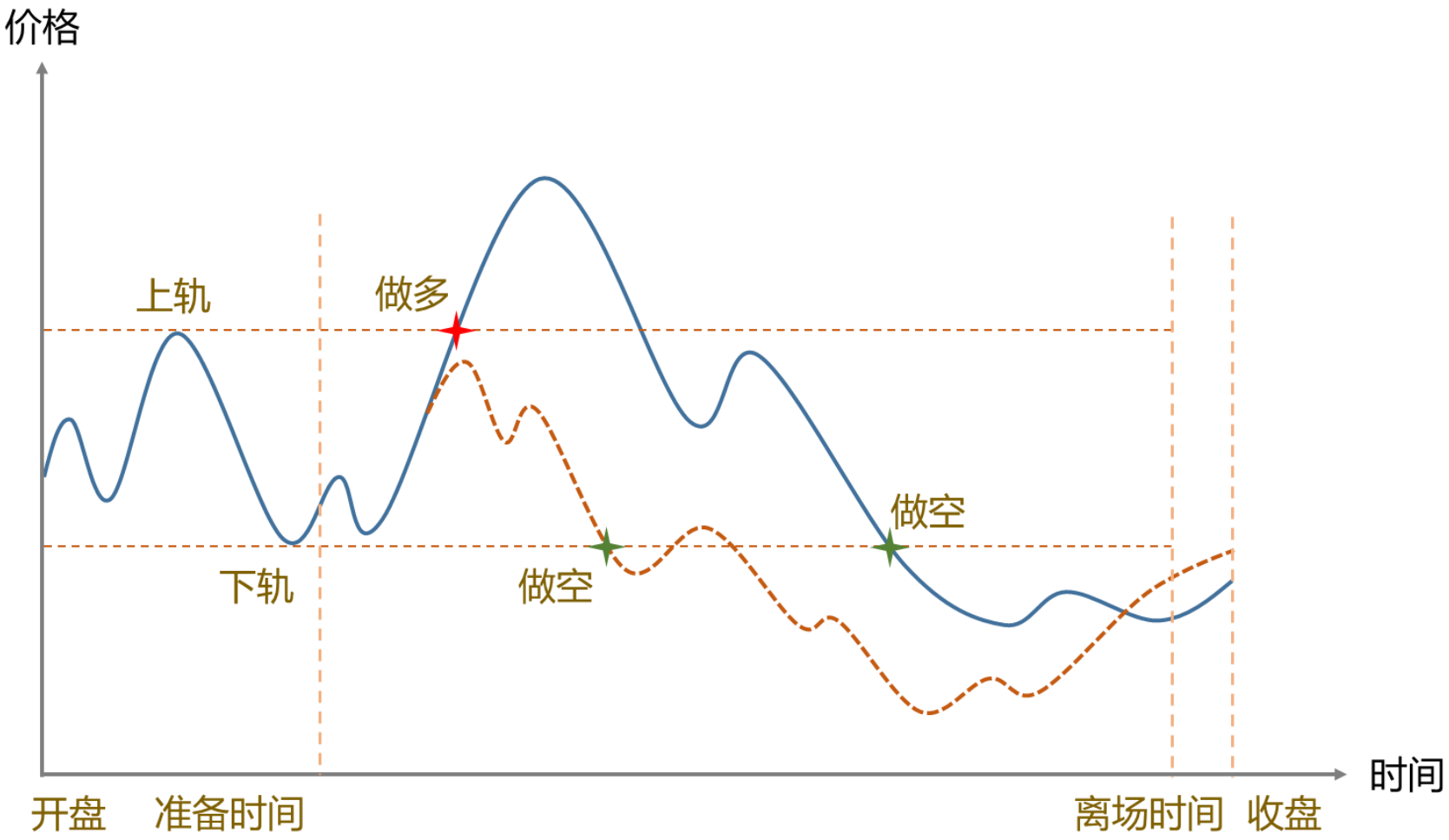


在经历过一轮强劲下跌趋势后，市场稍作喘息，进入横盘整理态势

在反转突破情况下，交易者认为之前趋势不可持续，并推动汇价朝之前反方向运动

Hans123策略原理

- 日内交易策略，趋势突破系统
- 在开盘30（或N）分钟后准备入场
- 上轨＝开盘后30分钟内的高点
- 下轨＝开盘后30分钟内的低点
- 当价格突破上轨，买入开仓
- 当价格跌穿下轨，卖出开仓
- 突破时如已有持仓，则先止损再反手
- 如有持仓，在收盘前无条件平仓


价格
上轨
做多
做空
下轨
做空
时间
开盘
准备时间
离场时间
收盘

```
//准备中间变量
INPUT:SS(1,1,10000,1),NMIN1(30,1,1000,1),NMIN2(10,1,100,1),N1(0,0,100,1);
N:=BARSLAST(DATE<>REF(DATE,1))+1;

开盘30分钟最高价:=VALUEWHEN(TIME<=090000+NMIN1*100,HHV(H,N));
开盘30分钟最低价:=VALUEWHEN(TIME<=090000+NMIN1*100,LLV(L,N));
手数:=SS;
上轨:开盘30分钟最高价+N1*MINDIFF;
下轨:开盘30分钟最低价-N1*MINDIFF;
//条件
开多条件:=C>上轨;
开空条件:=C<下轨;
//交易系统
IF TIME>090000+NMIN1*100 AND TIME<150000-10*100 THEN BEGIN
    开多:BUY(开多条件 AND HOLDING=0,手数,MARKET);
    开空:BUYSHORT(开空条件 AND HOLDING=0,手数,MARKET);
END
//平仓
IF TIME>=150000-NMIN2*100 THEN BEGIN
    收盘平多:SELL(1,手数,MARKET);
    收盘平空:SELLSHORT(1,手数,MARKET);
END

当前持仓:HOLDING,COLORGRAY,LINETHICK0;
当前资产:ASSET,NOAXIS,COLORGRAY; //输出当前资产，但不影响坐标最高最低值
```

思考：该策略流程的实现是否完整？

编程语言：PEL
运行环境：金字塔
（系统交易示例）

编程语言： Python
运行环境：掘金量化
来源：http://bbs.pinggu.org/thread-5884854-1-1.html

策略文件：Hans123.py
(分5页显示)

```
# encoding: utf-8
from gmsdk.api import StrategyBase
from gmsdk import md
from gmsdk.enums import *
import arrow
import time


OPEN_VOL = 5  # 每次开仓量
MAX_TRADING_TIMES = 50  # 最大开仓次数

class Hans123(StrategyBase):
    def __init__(self, *args, **kwargs):
        super(Hans123, self).__init__(*args, **kwargs)
        self.time_flag = False  # 是否已获取当天时间标识
        self.data_flag = False  # 是否已获取当天上、下轨数据
        self.long_holding = 0;  # 持仓量
        self.short_holding = 0;
        self.trading_times = 0;  # 当天交易次数
        self.__get_param()

    def __get_param(self):  # 获取配置参数
        self.trade_symbol = self.config.get('para', 'trade_symbol')  # 交易证券代码
        pos = self.trade_symbol.find('.')
        self.exchange = self.trade_symbol[:pos]
        self.sec_id = self.trade_symbol[pos + 1:]
        self.open_time = self.config.get('para', 'open_time')  # 开盘时间
        self.hans_time = self.config.get('para', 'hans_time')  # hans时间
        self.ex_time = self.config.get('para', 'ex_time')  # 强制平仓时间
```

```
def __get_time(self, cur_utc):  # 获取当天的重要时间参数
    utc = arrow.get(cur_utc).replace(tzinfo='local')
    cur_date = utc.format('YYYY-MM-DD')

    FMT = '%s %s'
    self.today_open_time = FMT % (cur_date, self.open_time)
    self.today_hans_time = FMT % (cur_date, self.hans_time)
    today_ex_time = FMT % (cur_date, self.ex_time)
    self.ex_time_utc = arrow.get(today_ex_time).replace(tzinfo='local').timestamp
    self.hans_time_utc = arrow.get(self.today_hans_time).replace(tzinfo='local').timestamp

def __init_band_data(self, bar_type):  # 获取上、下轨数据
    bars = self.get_bars(self.trade_symbol, bar_type, self.today_open_time,
                         self.today_hans_time)
    close_list = [bar.close for bar in bars]
    self.upr_band = max(close_list)  # 上轨
    self.dwn_band = min(close_list)  # 下轨

def on_tick(self, tick):  # tick行情事件
    self.last_price = tick.last_price  # 获取最新价

def on_bar(self, bar):  # bar周期数据事件
    # 获取当天的时间参数
    if self.time_flag is False:
        self.__get_time(bar.utc_time)
        self.time_flag = True
```

思考：该策略的实现细节是否有偏差?

```
# 计算上、下轨
if bar.utc_time < self.ex_time_utc and bar.utc_time > self.hans_time_utc:
    if self.time_flag is True and self.data_flag is False:
        self.__init_band_data(bar.bar_type)
        self.data_flag = True
```

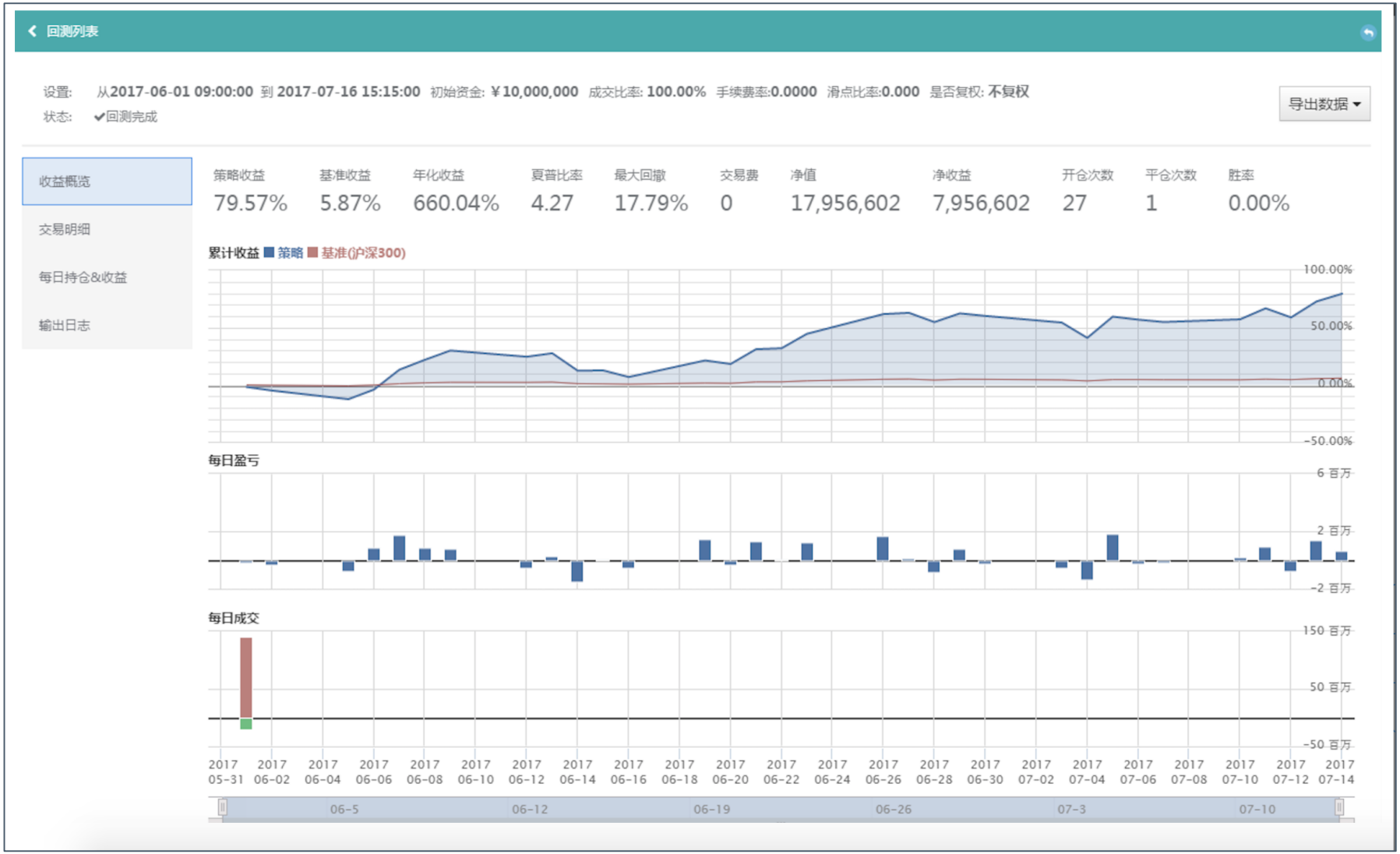
```
# 休市前强平当天仓位
if bar.utc_time > self.ex_time_utc:
    if self.long_holding > 0:
        self.close_long(self.exchange, self.sec_id, 0, self.long_holding)
        print('exit time close long: %s, vol: %s' % (self.trade_symbol, self.long_holding))
        self.long_holding = 0

    elif self.short_holding > 0:
        self.close_short(self.exchange, self.sec_id, 0, self.short_holding)
        print('exit time close long: %s, vol: %s' % (self.trade_symbol, self.short_holding))
        self.short_holding = 0
    return

if self.trading_times > MAX_TRADING_TIMES:
    print('trading times more than max trading times, stop trading')
    return

# 交易时间段
if bar.utc_time > self.hans_time_utc and bar.utc_time < self.ex_time_utc:
    if bar.close > self.upr_band:
        if self.short_holding > 0:
```

```
        # 有空仓，先平空仓
        self.close_short(self.exchange, self.sec_id, 0, self.short_holding)
        print('close short: %s, vol:%s' % (self.trade_symbol, self.short_holding))

        self.short_holding = 0

    # 开多仓
    self.open_long(self.exchange, self.sec_id, 0, OPEN_VOL)
    print('open long: %s, vol:%s' % (self.trade_symbol, OPEN_VOL))
    self.long_holding += OPEN_VOL

    # 开仓次数+1
    self.trading_times += 1
elif bar.close < self.dwn_band:
    if self.long_holding > 0:
        # 有多仓，先平多仓
        self.close_long(self.exchange, self.sec_id, 0, self.long_holding)
        print('close long: %s, vol:%s' % (self.trade_symbol, self.long_holding))
        self.long_holding = 0

    # 开空仓
    self.open_short(self.exchange, self.sec_id, 0, OPEN_VOL)
    print('open short: %s, vol:%s' % (self.trade_symbol, OPEN_VOL))
    self.short_holding += OPEN_VOL

    # 开仓次数+1
    self.trading_times += 1
```

```
if __name__ == '__main__':
    hans123 = Hans123(config_file='Hans123.ini')
    ret = hans123.run()
print(hans123.get_strerror(ret))
```


配置文件：Hans123.ini

```
[strategy]
td_addr=localhost:8001
username=
password=
strategy_id=
mode=4
subscribe_symbols=CFFEX.IF1707.tick,CFFEX.IF1707.bar.60

[backtest]
start_time=2017-6-01 09:00:00
end_time=2017-7-16 15:15:00
initial_cash=10000000
transaction_ratio=1
commission_ratio=0
slippage_ratio=0

[para]
trade_symbol=CFFEX.IF1707
open_time=09:15:00
hans_time=09:45:00
ex_time=15:10:00
limit_times=3
```


回测列表
设置: 从2017-06-01 09:00:00 到 2017-07-16 15:15:00 初始资金: ￥10,000,000 成交比率: 100.00% 手续费率:0.0000 滑点比率:0.000 是否复权: 不复权
状态: ✔回测完成
导出数据
收益概览
交易明细
每日持仓&收益
输出日志
策略收益 基准收益 年化收益 夏普比率 最大回撤 交易费 净值 净收益 开仓次数 平仓次数 胜率
79.57% 5.87% 660.04% 4.27 17.79% 0 17,956,602 7,956,602 27 1 0.00%
累计收益 策略 基准(沪深300)
100.00%
50.00%
0.00%
-50.00%
每日盈亏
6 百万
2 百万
-2 百万
每日成交
150 百万
50 百万
-50 百万
2017 05-31
2017 06-02
2017 06-04
2017 06-06
2017 06-08
2017 06-10
2017 06-12
2017 06-14
2017 06-16
2017 06-18
2017 06-20
2017 06-22
2017 06-24
2017 06-26
2017 06-28
2017 06-30
2017 07-02
2017 07-04
2017 07-06
2017 07-08
2017 07-10
2017 07-12
2017 07-14
06-5
06-12
06-19
06-26
07-3
07-10

遇到极端行情怎么办？

- Hans窗口内的振幅限制
- 日内止盈/止损（小周期K线）
- 大周期涨跌空间及后市判断
- 每交易日内的开平仓次数限制

如何适用于股票自动交易？

- T+1制度的限制
- 指数/板块到个股
- 从日内到日间
- 止盈/止损措施

休息一下

5分钟后回来

别拿豆包不当干粮

越简单的越有效

经典买入三法

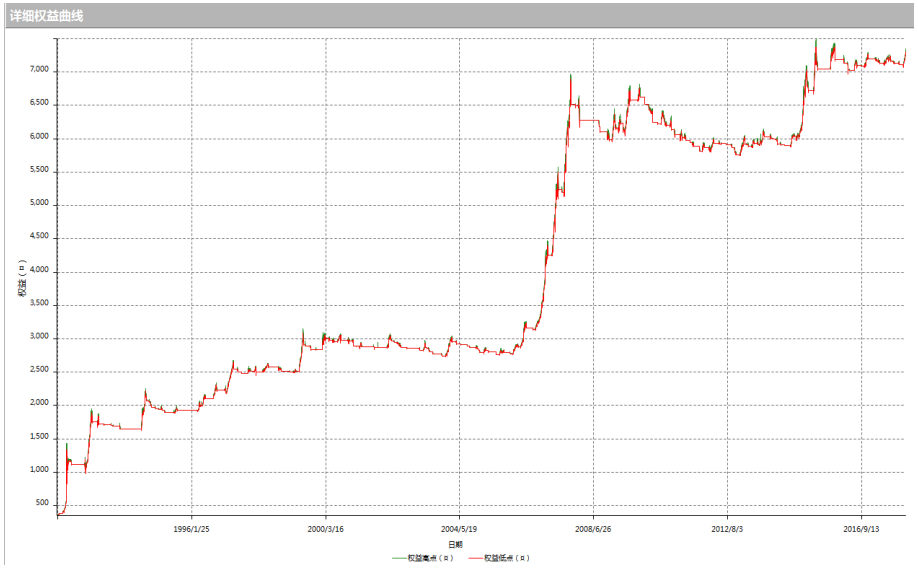
```
Variables: var0(0), var1(0), kline(0), dline(0);
Variables: avgPriceBuy(0), avgVol(0), avgPriceExit(0);

// 计算慢速KD值，经典KDJ指标，参数为经典参数14，3，3
value1 = Stochastic(High, Low, Close, 14, 3, 3, 1, var0, var1, kline, dline);

// 计算10日价格均线，20日价格均线，10日成交量均值
avgPriceBuy = Average(Close, 10);
avgVol = Average(Volume, 10);
avgPriceExit = Average(Close, 20);

// 在K上穿D，同时当日成交大于10日均值，当日收盘大于10日收盘价时买入
if (kline > dline and kline[1] <= dline[1] and Volume > avgVol and Close > avgPriceBuy) then
    buy 1 contract at this bar close;

// 当日收盘价格跌破20日均线时卖出
if (Close < AvgPriceExit) then sell at this bar close;
```

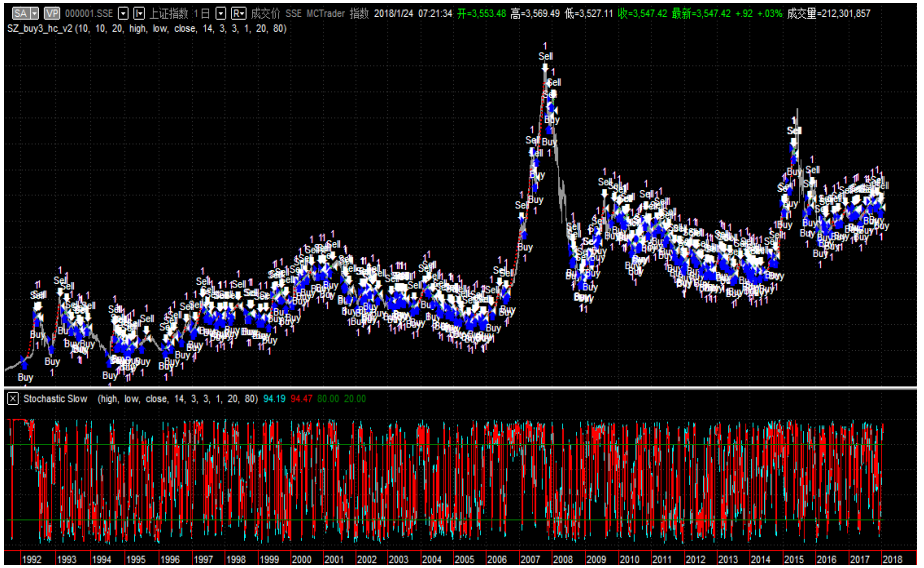
经典买入三法

策略标的：上证指数；策略时间：1992.2-2018.1；手续费：买卖0.2%

策略账户由350点上升到6970点，同期上证指数由350点上升到3560点

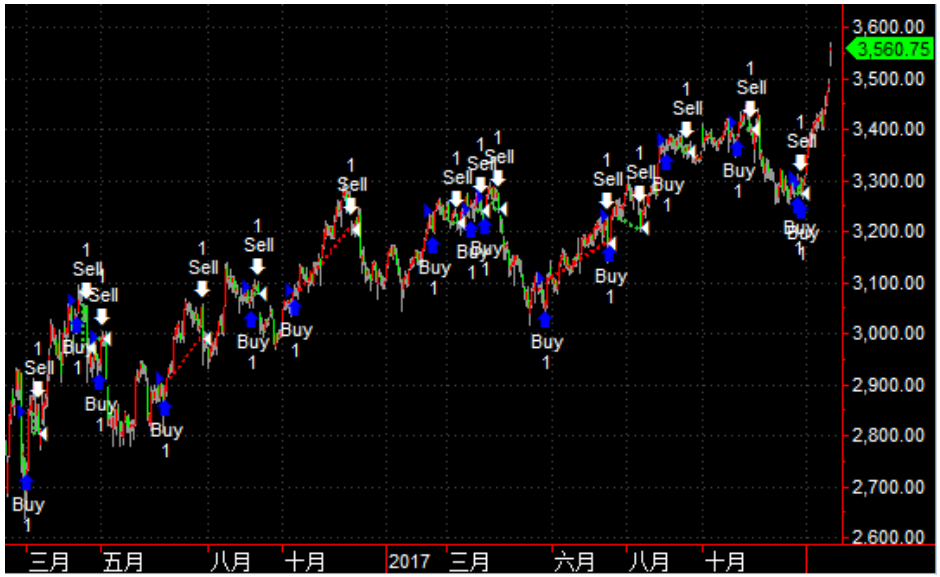
策略账户最大回撤1190点，上证指数最大回撤4460点





不同行情下量化信号与我们的习惯相反

缓步上涨行情中-避免错失机会



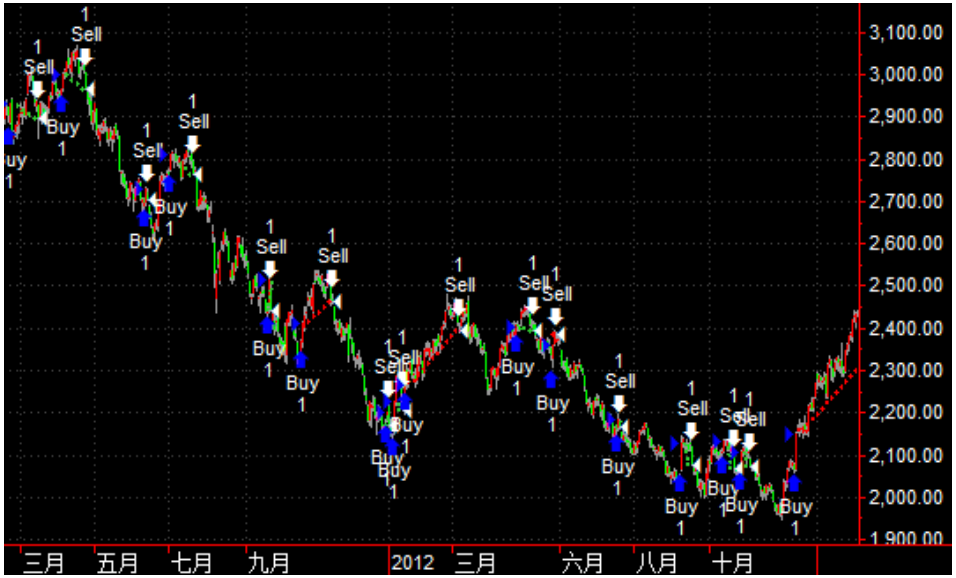
缓步上涨，抓住机会不错过

量化系统告诉我们：该开始交易啦，交易不要过少

避免常见的：经过了慢慢熊途，连账号密码都忘了。涨上去了才发觉要进场。

心理偏差纠正：保守偏差

漫漫熊途-避免深套



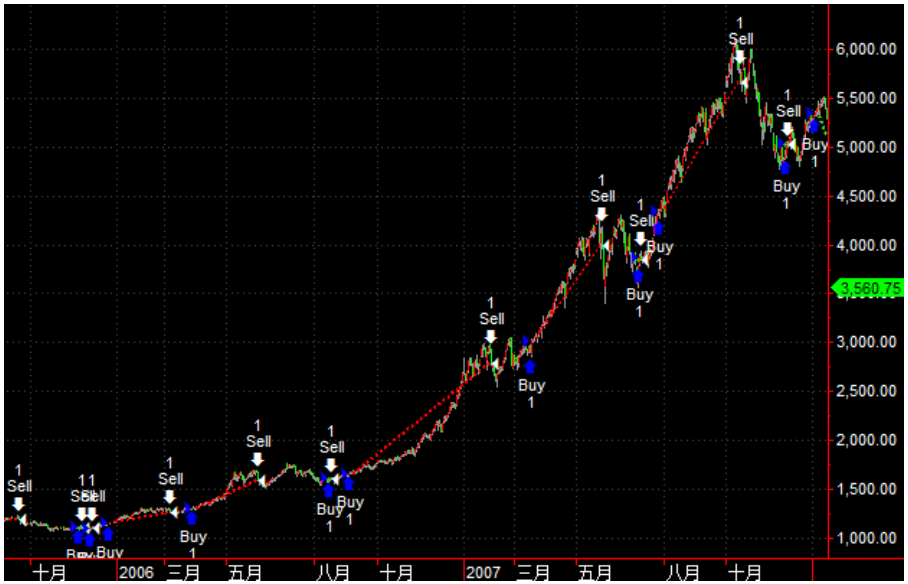
漫长的熊市中及时出场很重要，反而要交易频繁一些。量化交易系统告诉我们，到了位置要出场。先出了再说。保存子弹，等待机会。

避免常见的：一旦套牢就不卖，就是不认输离场。过了一段再一看，没钱回本了。

心理偏差纠正：自我确认偏差

不同行情下量化信号与我们的习惯相反

急速上涨行情中-避免持股不牢



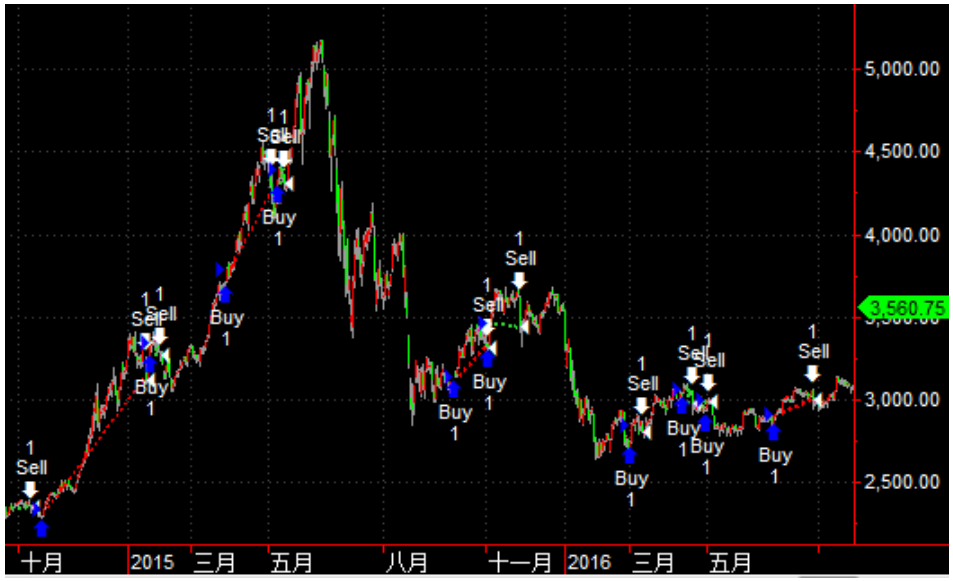
急速上涨，长期持股很重要

量化交易系统告诉我们，不到位置不要卖。牛市2006.1-2007.10中只有7笔交易。平均持仓大于3个月。

避免常见的：今天买，明天卖，看了盈利就逃跑，结果错过大行情，或者只吃一段，错过大部分利润

心里偏差纠正：损失厌恶偏差；避免后悔偏差

急速下跌行情-避免抢反弹



急速下跌中，不抢反弹很重要

在2015.6.12到2015.8.26一个入场信号都没有。

避免常做的事：看了大跌抢反弹，碰到跌停就套牢，套牢了也不卖，还要补仓降低成本

心里偏差纠正：心锚，固定和调整偏差；代表性偏差

实战中怎么操作

- 指数怎么交易？
  - 指数跟踪？
  - 指数ETF？
- 个股怎么做？
  - 分别操作？
  - 自己编制指数（股票池）？

实战中怎么操作

```
// 在K上穿D，同时当日成交大于10日均值，当日收盘大于10日收盘价时买入
if (kline > dline and kline[1] <= dline[1] and Volume > avgVol and Close > avgPriceBuy) then
    buy 1 contract at this bar close;

// 当日收盘价格跌破20日均线时卖出
if (Close < AvgPriceExit) then sell at this bar close;
```

有什么问题？

成交价

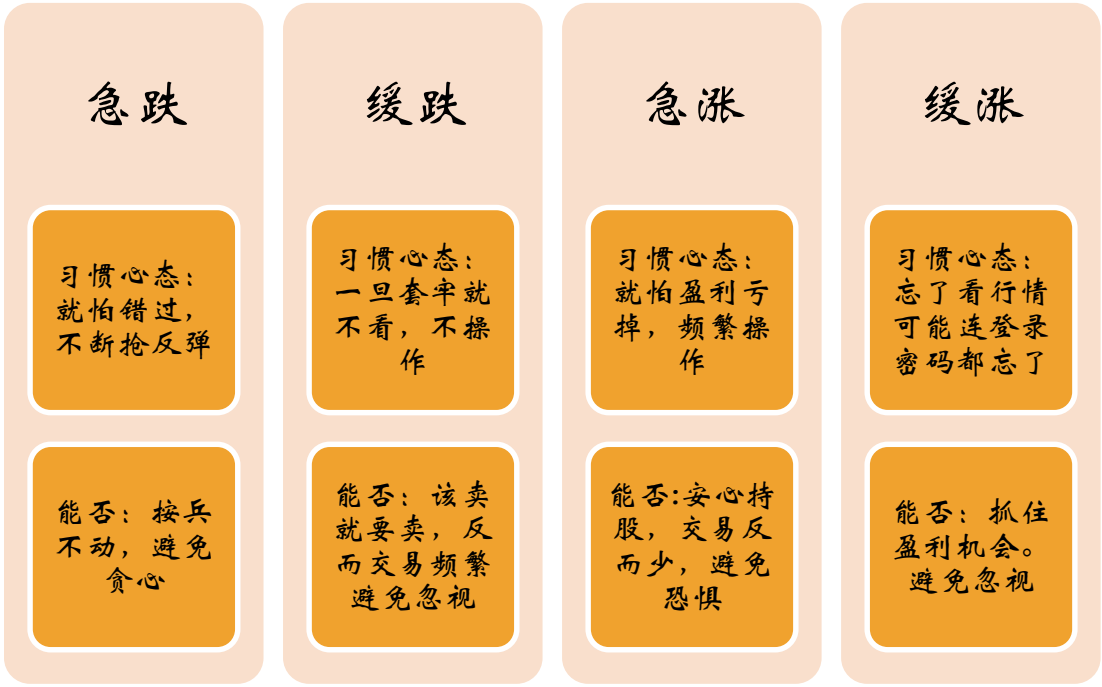
头寸

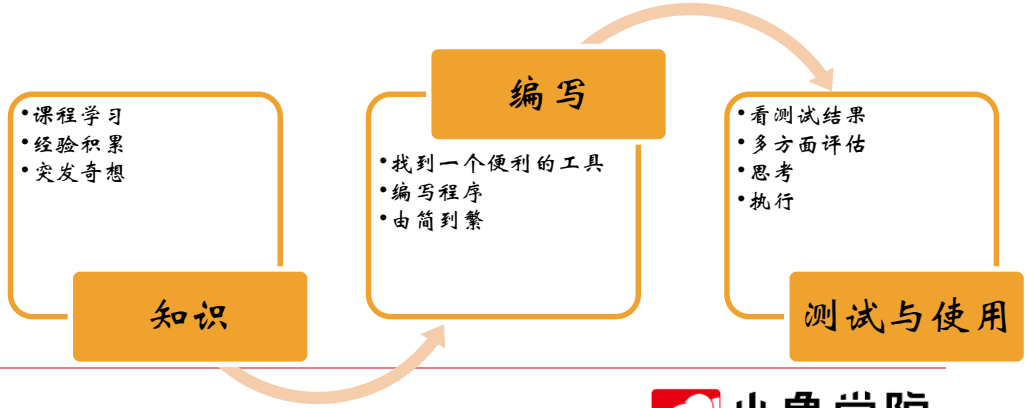
你理解的实战是什么

量化策略的完整研发和系统搭建

使用量化系统避免和利用心理偏差

- 投资中的困难一：心态难以把握，自己是否贪婪与恐惧
  - 量化投资固定了入场出场点
- 投资中的困难二：每天看真辛苦
  - 一旦程序写成，每日自动生成
  - 日间选股，日内信号
- 投资中困难三：方法真的有效吗
  - 程序可进行历史回测，查看历史收益





如何构造一个好的量化系统

行情

- 行情类型要全面
- 不可以只测试牛市
- 熊市能不能生存更重要

- 大跌，缓跌，缓涨，大涨都要数据
- 数据要足够长，日线数据一两年不够，多少个数据点就够了？

数据

策略

- 什么叫好策略？
- 策略敏感性？

- 条件越复杂越好吗？
- 参数越多越好，越少越好？

条件

量化系统工作过程


策略制定

行情判断


股票池筛选


信号产生


头寸管理


交易执行


人工干预


数据


评价和优化

流程细则与选择

数据收集
数据校验
数据比对
数据清洗
数据对齐
管理配置

高速通信通道
托管机房
网络架构

- 基础模型
- 回测平台选择
- 统计检验
- 数据样本检验
- 协整分析
- 样本内外测试
- 过拟合检验
- 鲁棒性分析

- 回测平台选择与特性
- 计算层面维度选择：TICK/MIN
- 策略层面基本逻辑
- 系统层面操作系统选择：如LINUX操作系统，无多余的可视化构件，适合专业高速的交易团队与策略。
- 编程语言层面：C++，python
- 再次进行样本内外测试
- 再次进行鲁棒性检验

举例：期货接口比较	高速飞马	一般CTP
内部逻辑事件	5个步骤	13个步骤
柜台延迟	小于300us	大于3ms
高速行情	支持	不支持

系统搭建实例演示

- 看到不等于做到
- 做到不等于做对
- 做对不等于真对
- 真对的人从不得瑟

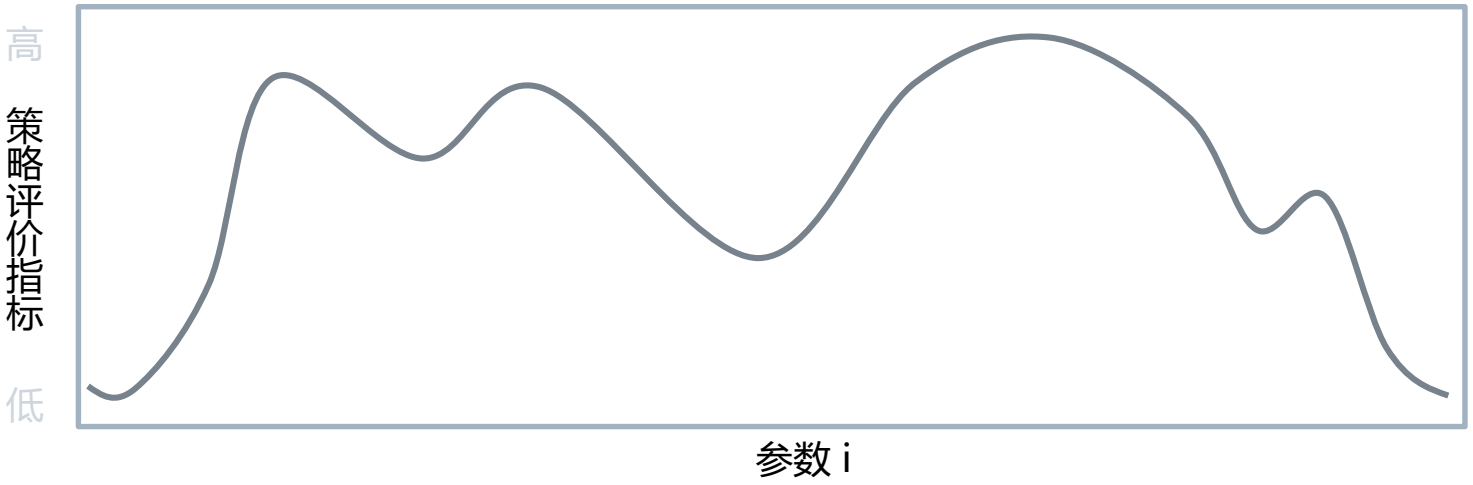
从简单的原理出发

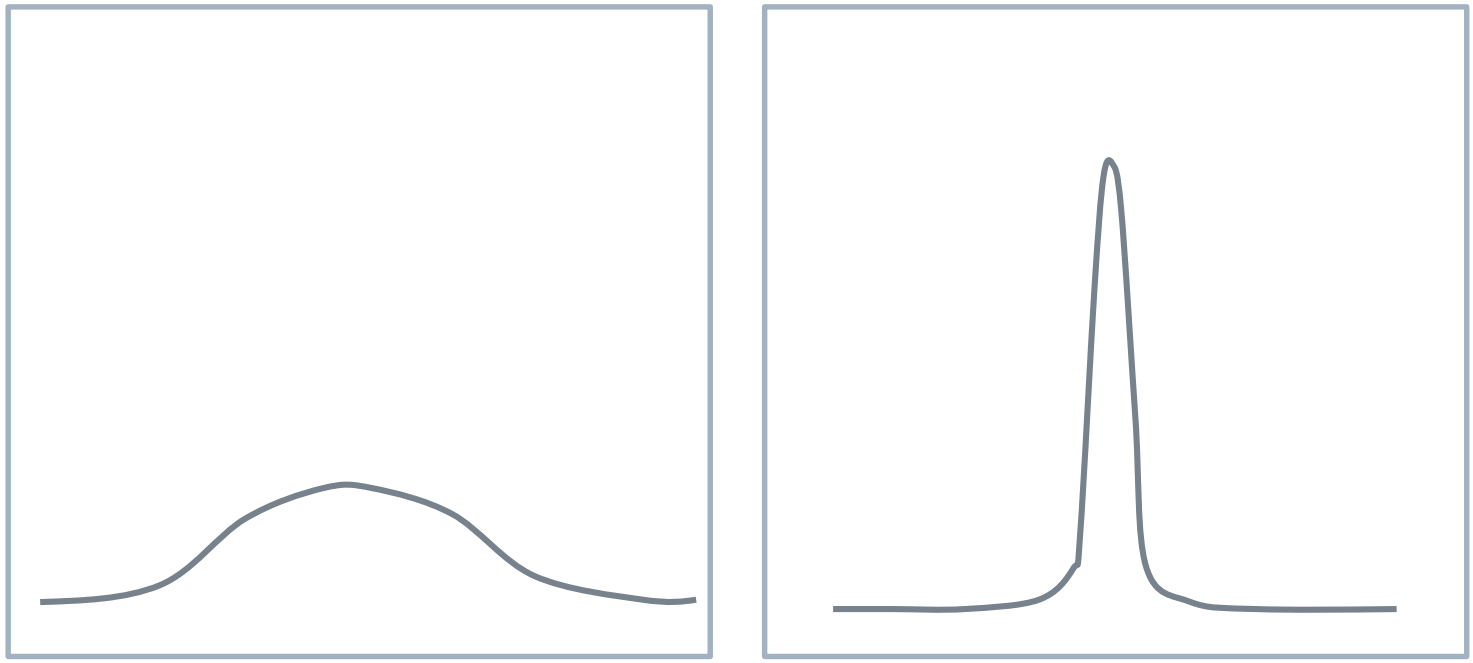
如何优化你的量化策略：再谈人工智能

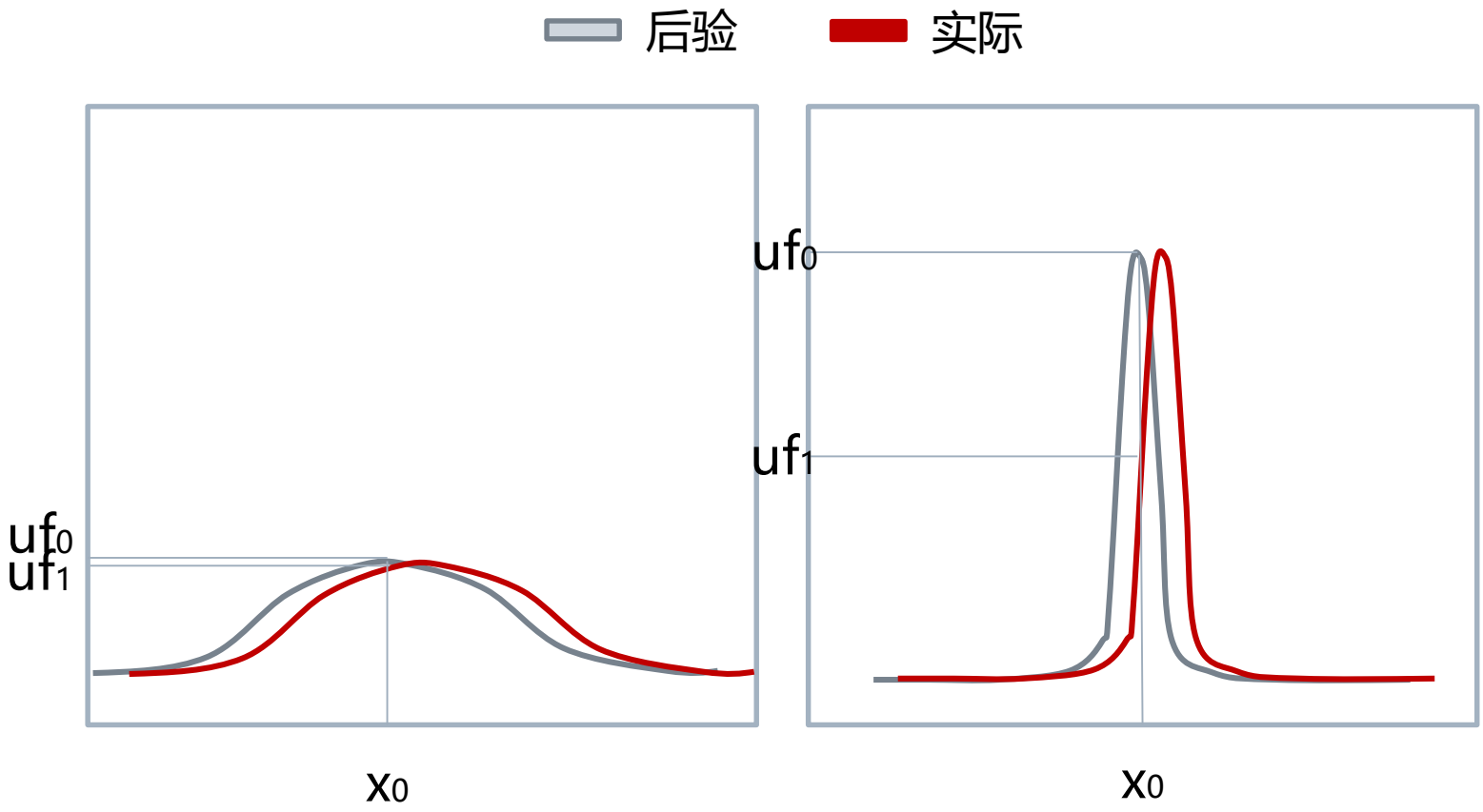
简单粗暴

- 假设有足够的算力，尽量获得参数值域 vs. 策略评价指标的完整函数曲线
- 策略评价指标是什么？

$$UF(\cdot) = \sum w_k \cdot f_k$$

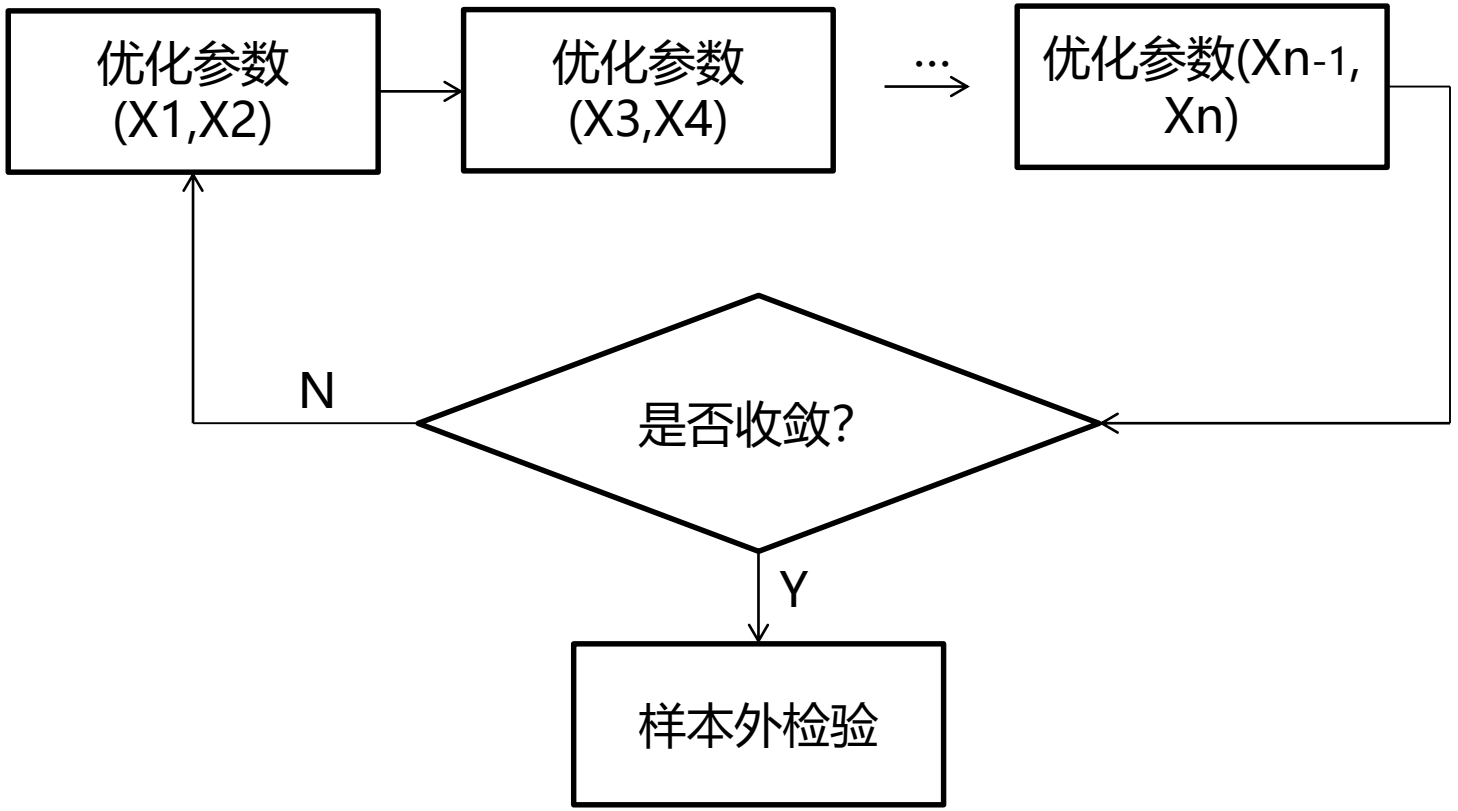






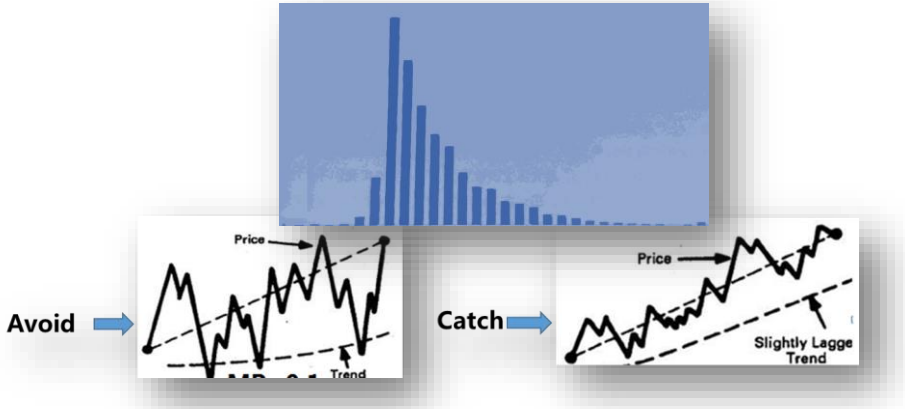
与寻求极值不同 我们的目标是策略对未来行情的描述能力

对于参数向量 X=（X1, X2, ... Xn）



量化投资更像哪一个？

那么现在你的选择是什么



抛掉你的自恋和自负

预告：你未来必踩的坑 – 怎么上实盘？

你有几种死法

- 策略没有严格验证，一上来满仓杠杆不止损，直接out
- 认真做策略了，实盘一上来很谨慎，逐渐开始有浮盈，仓位开始加重，突发一个意外，系统来不及反应（根本就没有相应预案），蒙受巨大回撤后，长期萎靡不振
- 两三年都能稳定盈利，开始膨胀，风控跟不上资产规模的增长，忽略了持续的策略评估，当出现温水煮青蛙也不自知，任何一次市场的大波动都是压垮骆驼的最后稻草
- 离开牛市，你什么都不是！

说说普量学员

- 中国二级市场三十年不败的超级大家
- 2017年成功狙击野村证券的私募大佬
- 掌管80亿资金的机构掌门
- 各类专业机构投资者...

- 越是高手越谦虚
- 越是不懂越自恋

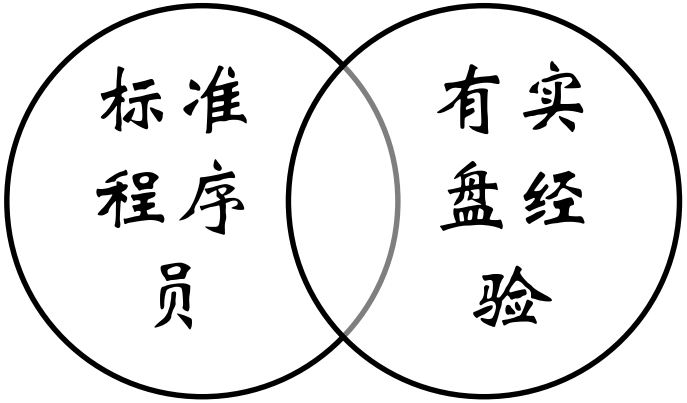
你要走的路 – 量化交易也是交易

去金融机构写代码

- 重点学习量化系统架构和模块的搭建方法
- 理解业务本质

自己做宽客

- 恶补实战课
- 认真学逻辑
- 忘掉代码 – 会coding对于现在的你其实什么也不是
- 1000元实操训练



策略逻辑

- 用量化手段来检验你之前的逻辑靠不靠谱

策略实现

- Coding不是重点
- 重点是找一个称手工具

策略实战

- 全是细节
- 不停诊断优化测试

普量风控 – 股票型策略预警机制

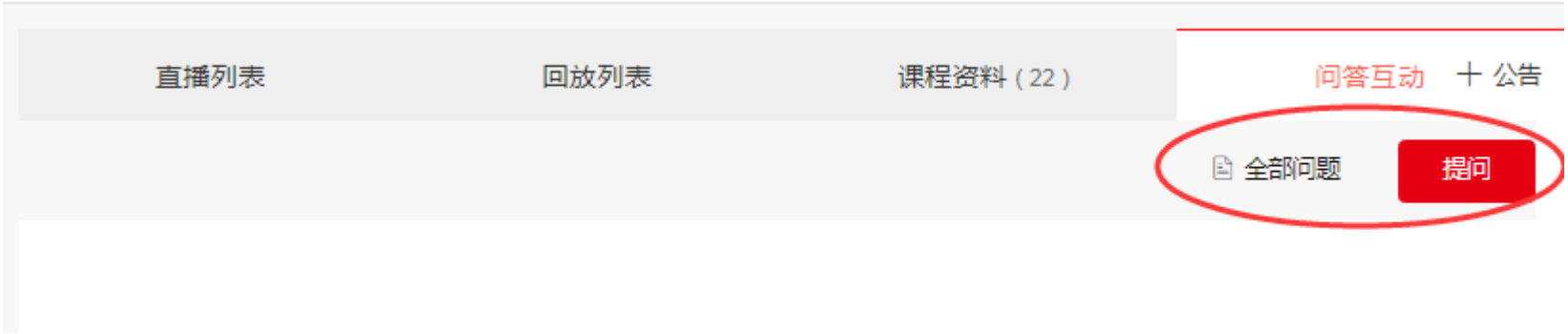
- 如单日产品净值回撤超过1%，发出一级预警，需要进行详细复盘分析，以确认该回撤的原因，并确定是否可以继续执行既定策略；
- 如产品净值累计最大回撤超过4%，发出二级预警，需要暂停策略自动交易，同时发起风控委员会临时会议决定是否进行策略调整；
- 如产品净值累计最大回撤超过8%，发出三级预警，需要主动降低仓位至25%以下，同时发起风控委员会临时会议决定是否进行策略调整或者是否停止策略的执行；
- 如产品净值下降到0.85，由风控委员会决定是否启用清盘程序；
- 盘中总仓位不得超过90%，隔夜总仓位不超过80%；
- 单只股票开仓总仓位<2%单方向总仓位；
- 单只股票最大亏损不超过15%；
- 单只股票每日最大交易额<min(流通市值*1%,该股票当日成交额*10%)；
- 单只股票最大持仓<10%流通市值

后续课程

- 系统化构建量化交易体系－案例系列课程

问答互动

在所报课的课程页面，
1、点击“全部问题”显示本课程所有学员提问的问题。
2、点击“提问”即可向该课程的老师和助教提问问题。



联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**

THANKS