基于超额十字星概念的量化交易策略

——注解和分析

王晨宇 **41432088**
2014 级经济与管理国际化人才创新班
周三晚 **10——12** 节

2017.06

基于超额十字星概念的量化交易策略：

出处：https://www.ricequant.com/community/topic/2086/

1. 代码理解和注释：

```
'''股票选择 '''
def pick_stocks(context):
    #数据转置剔除市值前 1%和 3%以后的股票
    df=context.df.T
    stock_list = list(df.index)
    #记入日志信息（满足条件的股票数）
    logger.info(len(stock_list))
    #过滤停牌股票
    stock_list = filter_paused_stock(stock_list)
    logger.info(len(stock_list))
    #过滤 ST 及其他具有退市标签的股票
    stock_list = filter_st_stock(stock_list)
    logger.info(len(stock_list))
    # 筛选符合超额十字星的股票
    stock_list = filter_crossstar_stock(stock_list)
    logger.info(len(stock_list))
    # 过滤前 20 日涨幅为负的股票：
    stock_list = filter_minus_rate(stock_list)
    #记入日志信息（满足条件的股票数）
    logger.info(len(stock_list))
    #记入日志信息（满足条件的股票）
    logger.info(stock_list)
    # 选取指定可买数目的股票
    if len(stock_list) > context.buy_list_count:
        stock_list = stock_list[:context.buy_list_count]
    return stock_list

'''功能控件'''
# 过滤停牌股票
def filter_paused_stock(stock_list):
    return [stock for stock in stock_list if not is_suspended(stock)]

# 过滤 ST 及其他具有退市标签的股票
def filter_st_stock(stock_list):
```

```
    return [stock for stock in stock_list if not is_st_stock(stock)]

# 筛选符合超额十字星的股票
def filter_crossstar_stock(stock_list):
    #创建名为 stock_list_1 的列表
    stock_list_1 = []
    #建立指数 index_pr 为所选市场指数过去 240 分钟内每分钟截止价格的历史数据（以中证 500 指数为基础比较）
    index_pr = history(240, '1m', 'close')['399905.XSHE']
    #建立指数 index_t 为所选市场指数过去 2 天每天收盘价格的历史数据
    index_t = history(2,'1d','close')['399905.XSHE']
    #以 T-1 日的收盘价为基准计算指数第 T 日中截止第 t 分钟的累积收益率
    Ri = index_pr/index_t-1
    #对于初步筛选得到的股票进行如下操作
    for stk in stock_list:
        #取该股票过去 240 分钟每分钟截止价格的历史数据
        stk_pr = history(240, '1m', 'close')[stk]
        #取该股票过去 2 天每天收盘价的历史数据
        stk_t = history(2,'1d','close')[stk]
        #以 T-1 日收盘价为基准计算股票在第 T 日中截止第 t 分钟的累积收益率
        Rs = stk_pr/stk_t-1
        #计算股票的累积超额收益率
        Re = Rs – Ri

        ‘”画超额收益烛线图‘”

        #股票超额收益高值
        High = Re.max()
        #股票超额收益低值
        Low = Re.min()
        #股票超额收益开盘值
        Open = Re.ix[0]
        #股票超额收益收盘值
        Close = Re.ix[-1]
        #计算开收盘差的绝对值作为烛体长度
        bar = abs(Re.ix[0]-Re.ix[-1])
        #如果开盘价>收盘价：
        if Open>Close:
            #上影线=最高价-开盘价
            upline = High – Open
            #下影线=收盘价-最低价
            downline = Close – Low
        #否则：
        else:
```

```
            #上影线=最高价-收盘价
            upline = High – Close
            #下影线=开盘价-最低价
            downline = Open – Low
        #判断烛体长度是否小于 0.001 且上、下影线长度超过其 3 倍
        if bar < 0.001 and upline > bar*3 and downline > bar*3:
            #满足则在列表 stock_list_1 中增加此股票并返回列表
            stock_list_1.append(stk)
    return stock_list_1

    # 过滤前 20 日涨幅为负的股票：
    def filter_minus_rate(stock_list):
        #建立名为 stock_list_2 的列表
        stock_list_2 = []
        #对于初步所得的股票进行如下筛选：
        for stk in stock_list:
            #判断过去 20 天每天的收盘价情况
            close = history(20,'1d','close')[stk]
            #判断前一天收盘价是否高于后一天：
            if close[0] > close[-1]:
                #满足则在列表 stock_list_2 中增加该股票并返回列表
                stock_list_2.append(stk)
        return stock_list_2

    def init(context):
        #调仓频率
        context.period = 20
        #定时器
        context.day_count = 0
        #是否调仓
        context.adjust = False
        #买入股票的数目
        context.buy_list_count = 20
        #设置调仓频率的小时数

        context.adjust_position_hour = 9
        #设置调仓频率的分钟数
        context.adjust_position_minute = 51

    #编写初始化逻辑，context 对象算法策略方法之间做传递
    def before_trading(context):

#建立 dataframe 更新和维护股息信息，使用 get_fundamentals 存取迄今为止股息率情况
context.df=get_fundamentals(query(fundamentals.eod_derivative_indicator.market_cap).order_by
```

```
(fundamentals.eod_derivative_indicator.market_cap.asc() ))

        #定时器调节天数增加一天
        context.day_count += 1

        #这里不做任何处理，直接跳过
        pass

    #调仓动作：
    def handle_bar(context,bar_dict):
        #设置 hour 为现时整时数
        hour = context.now.hour
        #设置 minute 为现时分钟数
        minute = context.now.minute

        #判断时间是否满足调仓所规定的小时和分钟数：
        if hour == context.adjust_position_hour and minute == context.adjust_position_minute:
            #满足则执行调仓
            do_handle_data(context, bar_dict)

    #调仓记录
    def do_handle_data(context, bar_dict):
        #记录调仓日志信息
        logger.info("调仓日计数  [%d]" %(context.day_count))
        #判断是否此时点进行过调仓
        if context.day_count % context.period == 0:
            #满足则记录已调仓的日志信息
            logger.info("==>  满足条件进行调仓")
            #股票买入记录
            buy_stocks = pick_stocks(context)
            #记录选股日志
            logger.info("选股后可买股票: %s" %(buy_stocks))
            #判断投资组合中该股持仓数是否大于 0：
            if len(context.portfolio.positions)>0:
                #以列表形式返回投资组合中该股票的持仓键值：
                for stk in context.portfolio.positions.keys():

                    #判断股票是否在买入股中:
                    if stk not in buy_stocks:
                        #不满足则调整相应持仓比例为 0
                        order_target_percent(stk,0)
            #对满足条件的股票执行如下操作：
            for stk in buy_stocks:
                #调整相应持仓比例为 0.99/20
```

order_target_percent(stk,0.99/20)

2. 策略归纳分析

此策略充分将证券投资数据的金融分析和 python 的量化操作相结合。大多数十字星形态在证券交易中属于明确调仓信号。十字星是指实体柱很短、上下影线较长的蜡烛图形态。是多空双方争夺激烈前期趋势将尽，趋势反转即将来临。以底部十字星为例，在股票价格经历较长下跌之后，如果出现十字星形态，表明市场空方力量已开始衰竭，多方在逐步蓄积反攻的力量，多空博弈从空方占优演变为势均力敌，股价走势转向的可能性也迅速升高。

而编者则是以十字星为基础构建了超额十字星的新指数来进行量化交易操作。以下为超额十字星的计算方法：

为了定义超额十字星，首先需计算每只股票的超额蜡烛图（或称超额 K 线）。对于特定股票第 T 日的超额蜡烛图，其计算步骤如下：

（1）以股票第 T-1 日的收盘价 P(T-1,Close)为基准，计算股票第 T 日中截止至第 t 分钟的累积收益率：Rs(T,t) = P(T,t)/P(T-1,Close)-1；

（2）以指数第 T-1 日的收盘价 Q(T-1,Close)为基准，计算指数第 T 日中截止至第 t 分钟的累积收益率：Ri(T,t) = Q(T,t)/Q(T-1,Close)-1；

（3）基于上述两个变量，计算股票第 T 日中第 t 分钟的累积超额收益：Re(T,t)= Rs(T,t)-Ri_(T,t)；

（4）取第 T 日累积超额收益的开盘值、最高值、最低值、收盘值，构成第 T 日的超额蜡烛图。

3. 运行效果

超额收益率十字星概念将十字星的反馈价值进一步深化改进，强化了调仓信号并使得收益更加稳健。从编写者的实际操作验证上看，也可以得到有效的证实（如下图所示）：

回测收益	回测年化收益	基准收益	Alpha	Beta	Sharpe	最大回撤
1.0758	-28.995%	135.773%	24.627%	38.8%	0.1862	0.5082

如果以 SHIBOR 作为无风险利率并运用策略中的中证 500 作为市场指数来验证此策略是否优于市场，我们可以采用时间序列分析和预测得出相应时段的市场 risk premiun 为-0.012956，进一步可得市场夏普比率为-0.15503，该数值远小于该量化交易策略下投资组合的夏普比率 0.1862，因此该策略是有效的且在相应时段优于市场指数。

4. 思考和感悟

此代码的运行过程中我认为最主要分为两个重点，第一是如何去过滤剔除掉常态下表现不佳的股票，第二是怎样设置和记录调仓动作。在设置调仓动作的时候，还可以进一步通过研究分析找到更合适的时间差，从统计分析的角度上讲，检测频率越高越有利于把握准确的调仓时点，从金融投资分析的角度上讲，如何根据投资组合中各股的历史表现来自动改善检测和调仓频率是可以进一步研究和改善的部分。

当今大数据时代，金融研究分析已经离不开程序语言的运用，将二者紧密结合起来不仅对于数据分析的便捷性和准确性有很大提升，更重要的是可以推进相关行业乃至整个市场的不断量化发展，解放出更多人力物力。作为一个大学生虽然还不能去判断人工操盘是否在某些方面能够有程序语言不具备的绝对优势，但量化交易的发展一定会是一场潮流，而且会不断地发展壮大下去。