启林量化平台说明文档

王鸿勇

2017年3月

目录

1 量化ALPHA简介 **3**

1.1 量化投资 3

1.2 股票ALPHA多因子模型 3

1.3 我们的ALPHA因子系统框架 3

1.4 多空回测体系简介 3

1.5 因子有效性评价 4

2 启林量化平台简介 **4**

2.1 平台安装 4

2.2 平台简介 4

2.3 config文件配置 5

2.4 策略运行和结果展示 5

3 可用的数据和函数 **5**

3.1 数据 5

3.2 函数 5

3.3 全局函数 7

3.4 全局变量 8

3.5 日期函数 8

4 operation算符说明 **9**

5 ALPHA策略的编写 **9**

5.1 数据声明和获取 10

5.2 策略的编写 10

5.3 命名规范 11

6 策略的评估 **11**

7 ALPHA策略的优化 **14**

7.1 优化方法 14

7.2 一些idea来源 14

1 量化ALPHA简介

1.1 量化投资

量化投资就是借助现代统计学、数学的方法，从海量历史数据中寻找能够带来超额收益的多种大概率策略，并纪律严明地按照这些策略所构建的数量化模型来指导投资，力求取得稳定的、可持续的、高于平均的超额回报。量化投资属主动投资范畴，本质是定性投资的数量化实践，理论基础均为市场的非有效性或弱有效性。

1.2 股票ALPHA多因子模型

我们模型利用市场的无效性进行统计套利。总的来说，预测一个股票的走势非常困难，但如果是一篮子股票，并且从统计上预测期望为正，则组合长期能获得稳定的超额收益， 我们把这个超额收益叫做股票的alpha。 多因子模型就是寻找到某些和股票收益率最相关的一系列因子。并根据相应的因子来构建股票组合。具体来说，每一个因子是一个对股票走势进行预测的数学表达式或数学模型，其由各种金融数据、运算符和函数构成。有效地因子构建的股票组合能够在长期稳定的获取超额收益。

1.3 我们的ALPHA因子系统框架

图一显示了我们系统框架下一个ALPHA因子的分析流程，首先我们会构建一个基础的ALPHA模型，该模型会对全市场的所有股票进行评分。 比如我们定义ALPHA=1/close, 这里close代表股票前一天收盘价，则股票的分数为前一天收盘价的倒数。这个分数构成了模型对所有股票的一个原始阿尔法值(raw alpha values)。 进一步我们对这个原始ALPHA值通过各种operation算符进行优化，从而得到每个股票最终的position(每日仓位)。回测系统根据每个股票每日的仓位对因子进行回测分析。

图 1: 我们的ALPHA因子系统框架

1.4 多空回测体系简介

我们采用多空的方法对因子进行回测。具体来说，我们将经过operation处理以后的ALPHA绝对值的大小认为是股票的仓位，正负号代表股票多空的方向。 图二给出了股票组合中股票的仓位和方向与其相应收益的关系。通过计算每个股票的每日收益我们

得到 因子股票组合的每日收益，这个基于时间的收益序列将用于对模型ALPHA因子有效性进行评估。

Position	Name	PnL if stock price raised 1%	PnL if stock price loses 1%
$100	Long	$1	-$1
-$100	Short	-$1	$1

图 2: 股票仓位、方向和收益的关系

1.5 因子有效性评价

下面给出了我们常用的几个评价因子有效性的指标:

1.IR(信息比率) IR等于收益的平均值除以波动率。

2.sharpe(夏普) $sharpe = 15.8 * IR$。

3.return 策略的年化收益

4.fitness 一个评价策略表现的函数，由策略因子的收益, 夏普和换手率结合构成，用来评估因子的有效性。 好的策略应该有尽可能高的收益, 夏普和低的换手率。

2 启林量化平台简介

2.1 平台安装

1.找平台系统管理员创建VNC账户

2.在账户下执行如下命令：/data/alphaSystem/release/run/version/1.0.0/setup yourSimFolder

2.2 平台简介

平台下各个文件具体说明如下

文件夹:

1.mysim：策略代码放置文件夹，里面有策略示例CPP，可通过修改相应CPP进行策略开发

2.libs：策略编译后的文件，用来给xml链接

3.pnl：策略运行生成的pnl文件

4.tools：策略结果处理的python工具箱

文件:

1.sim文件：策略运行的执行文件

2.config.xml文件：配置文件，用于配置策略运行的各项参数

2.3 config文件配置

1.Macros 定义宏，策略用到的引用的绝对路径

2.Universe 配置策略运行起始(startDate)和结束时间(endDate)

3.Modules 模块类，用于配置策略运行的数据和ALPHA文件等。

4.Portfolio 回测类，用于配制策略回测的框架和用的operation算符。

2.4 策略运行和结果展示

1.策略编译：./make

1.策略运行：./run config.xml(策略配置xml文件)

2.结果展示： 1) 策略表现展示：python tools/alpha_stat.py pnl/策略id(策略pnl文件路径)

2) 相关性：/data/alphaSystem/release/tools/cor pnl/策略id(策略pnl文件路径)

3 可用的数据和函数

3.1 数据

平台目前可用的数据包括量价数据、基本面数据、日内数据和其他数据。 每种数据具体的类型，结构，名称和说明见数据.xls文档。

3.2 函数

通过QL_Oputils::的方法调用，比如QL_Oputils::std(x)是求序列x的方差，以下是平台目前已有的一些常用函数:

1. const bool QL_less (const T &x, const T &y)

说明：比较x, y 若x<y, 则为true, 否则为false

2. int sign(T A)

说明：返回 A的符号

3. float decay_linear(vector<float> &x)

说明：返回序列x做线性衰减处理的值. $Decay\_linear(x) = (x[0] * n + x[1] * (n-1) + \ldots + x[n-1])/(n + (n-1) + \ldots + 1)$.

4. float decay_exp(vector<float> &x)

说明：返回序列x做线性衰减处理的值. $Decay\_exp(x, f) = (x[0] + x[1] * f + \ldots + x[n-1] * f^{n-1})/(1 + f + \ldots + f^{n-1})$.

5. int rank(vector<float> &x)

说明：对序列x做rank处理，返回序列的有效值个数 比如, 有6个股票的收盘价为[20.2, 15.6, 10.0, 5.7, 50.2, 18.4], rank以后序列为[0.8, 0.4, 0.2, 0.0, 1.0, 0.6].

6. void power(vector<float> &x, float e, bool dorank = true)

说明：对序列x做power处理, 默认会先做dorank处理，即先对序列rank以后再power. 比如, 有6个股票的收盘价为[20.2, 15.6, 10.0, 5.7, 50.2, 18.4], rank以后序列为[0.8, 0.4, 0.2, 0.0, 1.0, 0.6]. 假设设置e为2，则power以后为[0.64, 0.16, 0.04, 0, 1, 0.36]. 如果不希望做rank处理，则设置dorank = false.

7. float sum (vector<float> &x)

说明：求序列x的和.

8. float product (vector<float> &x)

说明：求序列x的乘积.

9. float mean (vector<float> &x)

说明：求序列x的平均值.

10. float median (vector<float> &x)

说明：求序列x的中位数.

11. float std (vector<float> &x)

说明：求序列x的方差.

12. float skew (vector<float> &x)

说明：求序列x的峰度.

13. float kurtosis (vector<float> &x)

说明：求序列x的偏度.

14. float corr(vector<float> &x, vector<float> &y)

说明：计算序列x和y的相关度.

15. llv (T first, T last)

说明：求序列的最小值. 示例：比如有vector x, 则llv(x.begin(),x.end())则返回序列x的最小值.

16. hhv (T first, T last)

说明：求序列的最大值. 示例：比如有vector x, 则hhv(x.begin(),x.end())则返回序列x的最大值.

17. float beta_range (T x_first, T x_last, T y_first, T y_last)

说明：计算序列x相对于序列y的beta. 示例：beta_range(x.begin(),x.end(),y.begin(),y.end()).

18. float slop (T first, T last)

说明：求序列的斜率.

3.3 全局函数

通过GLOBALFUNC::的方法调用，比如GLOBALFUNC::isnan(x)是判断x是否为nan，以下是平台目前已有的一些常用的全局函数：

1. bool isnan(T a)

说明：判断a是否为nan

2. bool isinf(T a)

说明：判断a是否为inf

3. bool iserr(T a)

说明：判断a是否为err(是否为nan或者为inf)

4.bool inline equal(float x, float y)

说明：判断x,y是否相等

5. int inline findInstrumentIdx(const string& sInstrument)

说明：查找某个股票在股票池中的index

3.4 全局变量

通过GLOBAL::的方法调用，比如GLOBAL::Dates(di)是di这个对应的日期：

1. Dates

说明：日期数组，GLOBAL::Dates(di)是di这个对应的日期

2. Instruments

说明：日期数组，GLOBAL::Instruments(ii)是ii这个对应的股票代码

3.5 日期函数

通过QLCalendar::的方法调用，比如QLCalendar::getDi(date)是返还date日期对应的di：

1.int inline getDi(int date)

说明：得到date日期对应的di

2.int inline distance_day(int date_a, int date_b)

说明：得到两个日期之间的天数(不管是不是交易日)，得到的是绝对值，不会有负数产生

3.int inline getDateAfterDays(int date, int days)

说明：向后多少天以后的日期。如果为负数，则为向前

4.int inline distance_tradDay(int date_a, int date_b)

说明：两个日期之间的交易日天数，得到的是绝对值

5.int inline distance_tradDay(int date_a, int date_b)

说明：向后多少交易日以后的交易日。如果为负数，则为向前

6.int inline getDate(int di)

说明：由di查得date

7.int inline returnWeekDay(int date)

说明：得到date是星期几

4 operation算符说明

平台常用operation算符如下

1. Decay 对ALPHA值做衰减处理，可选参数days，代表用来做衰减的天数

Decay(x, n) = (x[date] * n + x[date - 1] * (n - 1) + ... + x[date – n - 1]) / (n + (n - 1) + ... + 1)

其中n代表就是days

2. Power 对ALPHA值做指数处理, 可选参数exp，代表指数的幂次，默认在做指数处理前会做RANK处理

3. Truncate 对ALPHA值做截断处理, 可选参数maxPercent，代表截断的百分比，如果ALPHA值超过所有ALPHA值和的maxPercent,则其等于所有ALPHA值和的maxPercent。

4. IndNeut 对ALPHA值做行业中性处理, 可选参数group，代表做行业中性的具体行业分类。

5 ALPHA策略的编写

一个典型的策略编写流程如图4所示。首先一个粗略的idea通过数学建模生成原始的ALPHA，然后通过operation算符处理得到每个股票模拟时候的仓位，通过回测系统计算 策略每日收益和生成策略各项指标。然后根据各项指标再对策略进行优化改进。

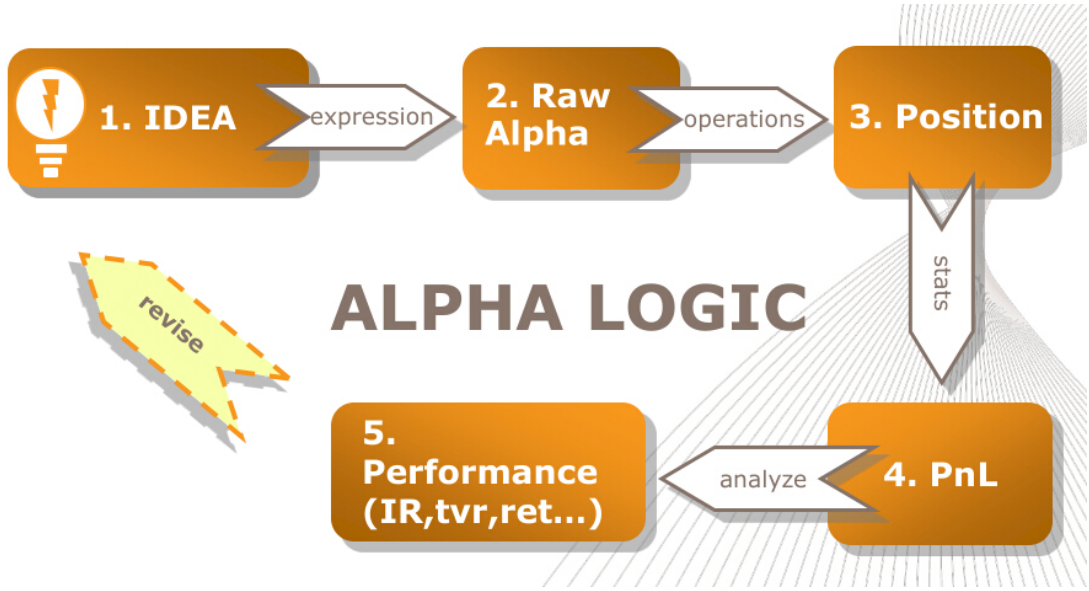

图 3: 典型的策略编写流程

5.1 数据声明和获取

1. 数据声明 数据声明在private下，比如需要股票的日线开盘数据，则可以如下声明。

const QL_MATRIX<QL_FLOAT> &open

2. 数据获取 数据获取在public下，比如需要股票的日线开盘数据，则可以如下获取。

open(dr.getData<QL_MATRIX<QL_FLOAT>>("adj_open"))

5.2 策略的编写

策略的编写通过复写函数$void\ generate(int\ di)$ 实现，其中di代表具体的某一天。具体的逻辑为给股票池每个股票进行打分(分数即为股票的ALPHA值), 即建立一个股票分数和数据之间的函数逻辑关系.

编写过程注意事项为:

1. 要尽可能给每个股票都打分,如果不能给出分数则给出打分为0

2. 目前系统采用delay 1模式， 不能用未来数据给股票打分，即di日的ALPHA只能用di-1日或者更前的数据。delay 1模式的说明见图4。

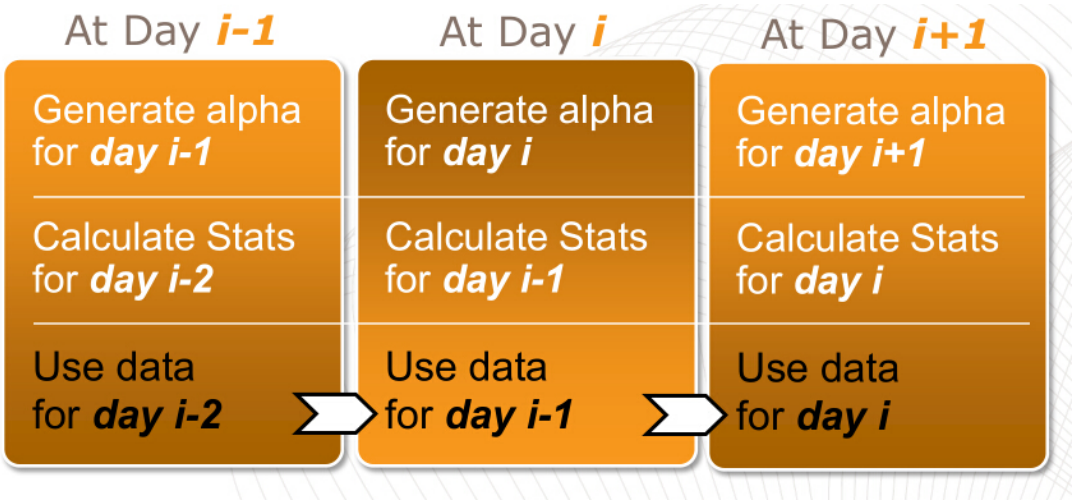

图 4: delay 1流程

一个简单5day-return的策略代码如图5所示:

```
explicit alpha5Dr(XMLCONFIG::Element *cfg):
    AlphaBase(cfg),
    close(dr.getData<QL_MATRIX<QL_FLOAT>>("adj_close")),   /*get close data*/
    Num1(cfg->getAttributeIntDefault("para1",5))  /*get para1*/
{
}
void generate(int di) override
{
    for(int ii = 0; ii < GLOBAL::Instruments.size(); ++ ii)
    {
        if((valid[ di ][ ii ]))
        {
            if(fabs(close[di - delay - Num1][ ii ]) > 1e-4)
            {
                alpha[ ii ] = (-1) * ((close[di - delay][ ii ] / close[di - delay - Num1][ ii ]) - 1) ;
            }
        }
    }
    return;
}
void checkPointSave(boost::archive::binary_oarchive &ar)
{
    ar & *this;
}
void checkPointLoad(boost::archive::binary_iarchive &ar)
{
    ar & *this;
}
std::string version() const
{
    return GLOBALCONST::VERSION;
}
private:
    friend class boost::serialization::access;
    template<typename Archive>
    void serialize(Archive & ar, const unsigned int/*file_version*/)
    {
    }
    const QL_MATRIX<QL_FLOAT> &close;   /*declare the data*/
    int Num1;   /*declare the data*/
```

图 5: 一个简单5day-return的策略代码

5.3 命名规范

1. cpp文件命名格式: username_alphaname.cpp
2. xml文件命名格式: config.username_alphaname.xml
3. xml内部alpha_id命名格式为username_alphaname
4. 团队合作策略命名usernameA_usernameB_alphaname

6 策略的评估

策略的表现通过调用python工具箱中的alpha_stat.py文件实现，上例中5day-return结果如图6所示:

其中dates代表回测周期，目前策略样本内回测周期为200601到201501。long,short代表多空的仓位， ret代表收益，turnover代表换手率，ir代表信息比率，drawdown代表最大回撤，win代表胜率， long_num,short_num代表平均做多股票数目和平均做空股票数目。fitness代表策略评分。up_days和down_days代表最长连续上涨和最长连续下跌日。

dates	long	short	ret	turnover	ir	drawdown	win	long_num	short_num	fitness	up_days	down_days
20060104-20061229	10.00	-10.00	31.63	32.03	0.19	8.43(20060414-20060509)	0.60	509	456	2.98	7	4
20070104-20071228	10.00	-10.00	43.93	31.61	0.20	5.12(20070606-20070612)	0.62	581	517	3.67	13	5
20080102-20081231	10.00	-10.00	61.20	31.14	0.26	5.31(20080801-20080819)	0.62	639	605	5.77	8	6
20090105-20091231	10.00	-10.00	54.85	32.42	0.34	2.92(20090409-20090414)	0.66	707	607	6.98	13	6
20100104-20101231	10.00	-10.00	31.92	31.49	0.18	4.81(20100409-20100507)	0.60	792	696	2.83	10	5
20110104-20111230	10.00	-10.00	23.69	30.52	0.17	4.34(20110805-20110811)	0.55	930	865	2.30	6	5
20120104-20121231	10.00	-10.00	30.19	30.31	0.21	5.48(20120112-20120118)	0.63	1001	957	3.31	9	5
20130104-20131231	10.00	-10.00	30.92	29.45	0.21	3.70(20130621-20130626)	0.61	1057	937	3.41	8	4
20140102-20141231	10.00	-10.00	17.34	29.53	0.12	7.29(20141216-20141229)	0.59	1038	952	1.47	8	7
20150105-20150130	10.00	-10.00	71.13	29.53	0.34	3.08(20150116-20150120)	0.65	1044	946	8.12	6	3
ALL	10.00	-10.00	36.54	30.93	0.21	8.43(20060414-20060509)	0.61	808	734	3.58	13	7

图 6: 5day-return的策略表现

策略的相关性评估通过命令/data/alphaSystem/release/tools/cor pnl/策略id(策略pnl文件路径)实现，5day-return相关性评估结果如图7所示 其中cor, ISSharp, OSSharp, Fitness分别给出了与策略相关性最高和最低的5个策略的相关性，策略样本内夏普，策略样本外夏普和Fitness分数。 count和count_better分别给出了策略与策略库中相关性各个等级的策略个数，以及其中比策略好的策略的个数。

ID	cor	ISSharp	OSSharp	Fitness
0	0.921594	5.037860	3.221901	4.578228
1	0.896607	5.080112	2.354225	3.763726
2	0.863586	5.426197	2.925806	5.223720
3	0.821349	5.840256	4.259636	4.985559
4	0.804823	6.526966	4.474137	7.215757
5	-0.481177	6.156419	7.276820	6.328896
6	-0.519210	2.699387	2.609117	1.697595
7	-0.646017	2.305622	1.506746	1.223826
8	-0.714948	3.680078	4.800074	3.674991
9	-0.780548	2.624980	1.431988	1.376957

lowerb	count	count_better
0.9	1	1
0.8	4	4
0.7	2	2
0.6	10	5
0.5	45	29
0.4	77	36
0.3	137	46
0.2	195	68
0.1	215	47
0.0	175	51
-0.1	116	31
-0.2	41	8
-0.3	27	6
-0.4	15	5
-0.5	8	4
-0.6	1	0
-0.7	1	0
-0.8	2	1
-0.9	0	0
-1.0	0	0

图 7: 5day-return的相关性表现

7 ALPHA策略的优化

7.1 优化方法

1. idea的优化，即模型算法的优化，尽可能从比较新颖的角度去提出想法
2. 模型参数的优化，尝试使用不同的函数
3. operation优化，尝试使用不同的operation算符和参数
4. 尽可能不要对参数做过度拟合
5. 尽量不要用过多的条件逻辑判断

7.2 一些idea来源

1. 价格趋势和反转
2. 波动率分析
3. 价格和成交量结合
4. 长期趋势和短期趋势结合
5. 从各种网站和文献中寻求灵感