基于协整和距离的配对交易

【摘 要】：自在 20 世纪 20 年代，著名交易员 Jesse Livermore 试图用配对策略来交易股票，配对交易正成为一种新的有效的投资方法。基于协整配对和距离配对，构建了一个新的两阶段配对交易策略。在股票配对选择方面，首先通过协整分析选取具有类似股价趋势的候选股票配对;其次，利用欧式距离计算每个候选配对股票的距离，并根据最小距离选择最佳配对股票，以避免同时存在同一股票。买卖卖空的风险很小。在资金配置方面，考虑保证金交易制度的当前背景，在有限资本约束下解决最优资金配置方案，确保模型设计更接近实际交易情况。上证 50 指数的成分股作为实证对象。实证研究结果表明，构建的新两阶段方法可获得超额收益。

【关键字】：协整；欧式距离；配对交易

一、 引言

股票配对的选择是配对交易策略制定中的重要步骤，主要涉及三种方法，分别是由 Gatev 等提出 的基于距离的股票配对方法，由 Vidyamurthy 提出的基于协整的方法和由 Eillott 等提出的基于随机价差的方法。各方法具有不 同的优势和劣势。其中，基于距离的股票配对方法 简单直接，在实证中得到广泛的应用。例如，Gatev 等通过对 S&P500 指数成分股的配对交易研究， 指出基于距离法的配对策略能获得超额收益。 Nath 针对风险引入了止损条件，并证明改进后的 距离配对策略相比基准模型能提供更高的收益。麦永冠和王苏生将距离配对法应用于沪深 A 股市场的投资组合研究中。然而，距离配对法存在着对 价差收敛时间和期望持有时间预测能力不足的缺陷。

协整技术能有效估计两两股票的历史价格走势及其价差偏离程度，被广泛应用于配对策略构造中。例如，Lin Yanxia 等将协整配对法应用于澳大利亚交易所的投资决策研究中，结果表明 该方法能获得持续收益；Hong 和 Raul 应用该方 法对美国市场 6 4 只股票进行配对交易，同样获得显著收益；Vidyamurthy 从理论上对协整配对法进 行了具体阐述。金恺采用协整配对方法对沪深 300 指数成分股进行了配对交易研究。然而，基于协整的配对法仍具有不可避免的缺陷,如模型指定偏 误、参数估计偏误,以及格兰杰检验变量选择等问题。

基于随机误差法的股票配对法认为股票价差由某一隐变量驱动,且其隐变量遵循 Vasicek 过程。然而,该模型限制配对股票的长期收益必须一致，且模型复杂性较高，相关实证研究较少。鉴于各单方法均有优劣，有机结合多种方法的混合配对法能有效克服各单方法的不足，并充分发挥其优势。因此，一些混合多阶段配对交易策略被 相应提出。例如，Miao 结合相关性分析与协整分析提出了一个新的两阶段配对策略，并以美国股票市场为研究对象验证了该策略能产生高于 S&P500 指数的收益。徐沐霖将该方法推广到沪深 300 指数成分股的配对交易中，其结果表明该模型能够 在中国股票市场的投资中带来超额收益。

综上，拟有机组合距离配对法与协整配对法，构建一种新的两阶段配对交易策略。具体的，在股票配对选择上，首先采用协整分析选出股价走势具有协整关系的候选股票对；在此基础上，采用距离法计算各候选配对股票间的欧式距离，以距离最小为依据构建最佳配对组合，从而避免了同一股票同时被买入和卖空的风险。实证研究以上证 50 指数成分股为研究对象，以验证模型的有效性与稳定性。

二、 量化交易描述

1. 量化交易整体框架

在配对阶段：首先将上证 50 指数的成分股的时序价格数据进行两两协整分析，计算出对应的 p 值，将协整性强的股票对留下，即 p 值小于 0.05，但是在计算协整的过程需要注意一些停牌的股票，即 p 值等于 0，而后将第一轮筛选的股票对的时序价格数据进行数据标准化，去除量纲的影响，而后计算它们之间的欧式距离，筛选出欧式距离最小的股票对。

在交易阶段：跟踪股票市场上的动态股票价格，计算股票对之间的价格差判断定价是否出现误差，一旦检测到误差，则买入低估的股票，同时卖出高估的股票

2. 理论依据与方法

下面将介绍两阶段配对交易策略中所采用的相 关技术方法。其中，股票配对阶段结合基于协整的 股票配对法与基于距离的股票配对法。

2.1 基于协整的股票配对法

Alexander 等证明两只股票之间的潜在套利需满足两个条件：一是两者存在长期的均衡利差；二是短期利差偏离长期均衡。相应的，一个有效的配对交易方法必须要满足两个条件：一是要能够有效地模拟股票价格走势，并探测股票价格间的相关关系；二是能够度量短期价差偏离长期均衡价差的程度。协整分析因具有这两方面优势从而被广泛应用于配对交易研究中。

Granger 因果检验是我们再计量经济学中学习过判断协整的方法。协整模型假设，受经济 系统内在机制作用，各经济变量间往往存在着长期的均衡关系，即使某些变量短期内偏离了长期均衡，该内在机制也会促使变量逐渐调整使其回到均衡状态。

如果一个时间序列是平稳的，我们称之为 I（0）过程；如果一个时间序列不平稳，但是经过 d（d > 0）次差分后变成平稳，我们称之为 I（d）过程。协整的定义为，对于多个序列，如果它们多是非平稳的，但是经过一阶差分过后，变为平稳序列，并且存在非零系数是的它们的线性组合是平稳的，那么它们就具有协整关系。

两两股票之间进行协整分析可以通过 python 中第三方库 statsmodels 计算它们之间的 p 值，当 p 值小于 0.05 时，说明两个股票对存在协整关系。

2.2 基于距离的股票配对法

距离方法是一种非参数方法，在配对交易中得到广泛的应用。基于距离的股票配对方法由 Gatev 等提出。首先，计算各股票对的欧式距离：

$$D_{i,i'} = \sqrt{\sum_{t=1}^{T} (P_{i,t} - P_{i',t})^2}$$

P 为股票价格标准化后的结果，D 值越小，说明两个股票的价格走势越接近，越适合作为股票对。

3．配对过程的代码实现

通过量化交易平台---聚宽，可以轻松得到股票的历史交易价格，以及能够模拟进行量化，因此，本次实践的全部代码都是在该量化交易平台上实现的。配对数据来源于 2016 年上证 50 成分股的历史收盘价。

配对代码如下：

```
# 导入函数库
import jqdata
import numpy as np
```

```
import pandas as pd
import statsmodels.api as sm
import seaborn as sns

# 初始化函数，设定基准等等
def initialize(context):
    # 设定基准
    set_benchmark('000016.XSHG')
    # 开启动态复权模式(真实价格)
    set_option('use_real_price', True)
    # 输出内容到日志 log.info()
    log.info('初始函数开始运行且全局只运行一次')
    # 过滤掉 order 系列 API 产生的比 error 级别低的 log
    # log.set_level('order', 'error')

    #获取上证 50 成分股
    g.stocks = get_index_stocks('000016.XSHG', date='2016-12-01')
    #log.info(g.stocks)
    #获取成分股的历史价格
    g.hist_price = get_price(g.stocks, start_date='2016-01-01', end_date='2016-12-31',
frequency='daily', fields=None, skip_paused=False, fq='pre', count=None)['close']
    g.dataframe = pd.DataFrame(g.hist_price)

    #配对结算---两两股票之间进行协整
    # 得到 DataFrame 长度
    n = g.dataframe.shape[1]
    # 初始化 p 值矩阵
    pvalue_matrix = np.ones((n, n))
    # 抽取列的名称
    keys = g.dataframe.keys()
    # 初始化最强协整组即 p 值最小
    pairs = []
    #为了选取最小 p 值，设置两个变量
    new = 0
    old = 1
    print(g.dataframe['600485.XSHG'])
    # 对于每一个 i
    for i in range(n):
        # 对于大于 i 的 j
        for j in range(i+1, n):
            # 获取相应的两只股票的价格 Series
            #if g.dataframe[keys[i]] and g.dataframe[keys[j]]:
            stock1 = g.dataframe[keys[i]]
            stock2 = g.dataframe[keys[j]]
```

```
                # 分析它们的协整关系
                result = sm.tsa.stattools.coint(stock1, stock2)
                # 取出并记录 p 值
                pvalue = result[1]
                pvalue_matrix[i, j] = pvalue
                # 选取最小 p 值
                new = pvalue
                #if new < old:
                    #old = new
                    #pairs = (keys[i], keys[j], pvalue)
                if pvalue < 0.05 and pvalue != 0:
                    # 记录股票对和相应的 p 值
                    #print(keys[i], keys[j], pvalue)
                    #pairs.append((keys[i], keys[j], pvalue))

                    #计算欧式距离
                    st_stock1 = (g.dataframe[keys[i]] - np.mean(g.dataframe[keys[i]]))/np.std(g.dataframe[keys[i]])
                    st_stock2 = (g.dataframe[keys[j]] - np.mean(g.dataframe[keys[j]]))/np.std(g.dataframe[keys[j]])
                    Distance = np.sqrt(np.sum(np.square(st_stock1-st_stock2)))

                    #print(keys[i], keys[j], pvalue, Distance)
                    pairs.append((keys[i], keys[j], pvalue, Distance))

    #以欧式距离排序
    # 获取列表的第四个元素
    def takeSecond(elem):
        return elem[3]

    # 指定第四个元素排序
    pairs.sort(key=takeSecond)
    print(pairs[0])
    best_stock = pairs[0]

    #对配对股进行拟合
    x = g.dataframe[best_stock[0]]
    y = g.dataframe[best_stock[1]]
    k = (np.sum(x*y) - n*np.mean(x)*np.mean(y))/(np.sum(x*x) - n*np.mean(x)*np.mean(x))
    b = np.mean(y) - k*np.mean(x)
    st_x_y = np.std(y-k*x-b)
    print(k, b, st_x_y)

    ## 运行函数（reference_security 为运行时间的参考标的；传入的标的只做种类区分，
```

因此传入'000300.XSHG'或'510300.XSHG'是一样的）

```
    # 开盘前运行
    run_daily(before_market_open, time='before_open', reference_security='000300.XSHG')
    # 开盘时运行
    run_daily(market_open, time='open', reference_security='000300.XSHG')
    # 收盘后运行
    run_daily(after_market_close, time='after_close', reference_security='000300.XSHG')

## 开盘前运行函数
def before_market_open(context):
    pass

def after_market_close(context):
    pass
```

运行结果：

运行结果的第一、二列为股票对的股票代码，后缀（.XSHG）表示的是在上海证券交易所上市的股票，第三列表示协整 p 值，第四列表示欧式距离。下图为部分截图：

```
2016-06-01 00:00:00 - INFO  - ('600016.XSHG', '601857.XSHG', 0.034759365056470218,
9.9607415748370087)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600030.XSHG', 0.021504054796465298, 1
0.76161842134583)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600036.XSHG', 0.0022669569349849424,
8.0050461673368432)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600104.XSHG', 0.022460396444388513,
9.350615439918375)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600518.XSHG', 0.048377377628678025, 1
1.05496702409159)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600519.XSHG', 0.011882776824288847,
9.9392250300307428)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600887.XSHG', 0.02615991903209194, 1
0.729563670679861)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '600919.XSHG', 0.014972123786212543, 2
8.975503456927324)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '601006.XSHG', 0.018771387203518987,
8.9157775125659118)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '601088.XSHG', 0.019152379101554751,
9.3781400419192824)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '601288.XSHG', 0.03966264343128257, 9.
2299865178672995)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '601318.XSHG', 0.0035692436141284436,
8.6203083579590611)
2016-06-01 00:00:00 - INFO  - ('600028.XSHG', '601336.XSHG', 0.00019400106767765064,
 7.5769925031175784)
```

通过 python 中的 sort 函数排序可以得到，以两个股票之间欧氏距离的前十小如下图所

示，特别地，工商银行（601398.XSHG）与 农业银行（601288.XSHG）排名第一,建设银行（601939.XSHG）与中国银行（601988.XSHG）排名第二，并且工商银行（601398.XSHG）与中国银行(601988.XSHG)排名第八,农业银行(601288.XSHG)与建设银行(601939.XSHG)排名第十，说明四家银行之间股票走势之间与其他股票相比具有较强的相关性，因此配对结果为工农中建四家银行股票。

```
2016-06-01 00:00:00 - INFO  - ('601398.XSHG', '601288.XSHG', 0.02922844357548398, 3.
546018741309688)
2016-06-01 00:00:00 - INFO  - ('601390.XSHG', '601800.XSHG', 0.0058969463273736734,
3.448026622363669)
2016-06-01 00:00:00 - INFO  - ('601939.XSHG', '601988.XSHG', 0.0078800204852085189,
3.992224434226707)
2016-06-01 00:00:00 - INFO  - ('601390.XSHG', '601818.XSHG', 0.017369165039894229,
5.493266333220753)
2016-06-01 00:00:00 - INFO  - ('601939.XSHG', '601857.XSHG', 0.024505951892083638,
6.495768264578778)
2016-06-01 00:00:00 - INFO  - ('601390.XSHG', '601901.XSHG', 0.023231271943781582,
6.570527501272991)
2016-06-01 00:00:00 - INFO  - ('601818.XSHG', '601985.XSHG', 0.015244279742223657,
6.998230615385584)
2016-06-01 00:00:00 - INFO  - ('601398.XSHG', '601988.XSHG', 0.011684620889349967,
7.816826102892986)
2016-06-01 00:00:00 - INFO  - ('601985.XSHG', '601989.XSHG', 0.00092099720868446078,
 7.983500824453149)
2016-06-01 00:00:00 - INFO  - ('601668.XSHG', '601985.XSHG', 0.021301767270980324,
9.2247968638126512)
2016-06-01 00:00:00 - INFO  - ('601288.XSHG', '601939.XSHG', 0.014318804578903575, 1
0.0347060149454714)
```

交易代码为（交易时间为：2018/03/15-2018/05/14）：

```
# 通过各个银行股票的实时价格与前一日收盘价格对比，取得最大最小的比率差，
# 超过 0.05 那么就买入比率最小的那个(认为被低估)，同时卖出最大的那个(认为被低估)
# 作者：何家雄

import math
import pandas as pd
import numpy as np
import statsmodels.api as sm
import scipy.stats as scs
import scipy.optimize as sco
import talib as tl
from datetime import timedelta

#enable_profile()
bank_stocks=['601398.XSHG', '601288.XSHG','601939.XSHG','601988.XSHG']  # 设置银行股票 工行，建行
```

```
# 初始化参数
def initialize(context):
    # 初始化此策略
    # 设置要操作的股票池为空，每天需要不停变化股票池
    set_universe([])
    g.riskbench = '000300.XSHG'
    set_commission(PerTrade(buy_cost=0.0002, sell_cost=0.00122, min_cost=5))
    #设置滑点
    #这个价差可以是一个固定的值(比如 0.02 元，交易时加减 0.01 元)，设定方式为：
FixedSlippage(0.02)
    set_slippage(FixedSlippage(0))
    set_option('use_real_price', True)
    # 设置基准对比为沪深 300 指数

    g.inter = 0.005

# 每天交易前调用
def before_trading_start(context):
    #history 读取的是前一天的数据
    g.df_last = history(1, unit='1d', field='close', security_list=bank_stocks, df=False,
skip_paused=True, fq='pre')

# 每个单位时间(如果按天回测,则每天调用一次,如果按分钟,则每分钟调用一次)调用一次
def handle_data(context, data):

    raito = []

    for code in bank_stocks:
        #data[code].close 存放前一个单位时间(按天回测，是前一天，按分钟回测，则是前
一分钟) 的数据
        #print(code,data[code].close,g.df_last[code][-1])
        raito.append( data[code].close / g.df_last[code][-1] )

    if not context.portfolio.positions.keys():
        #context.portfolio.positions.keys()当前持仓的股票代码
        if max(raito) - min(raito) > g.inter:
            min_index = raito.index(min(raito))
            order_value(bank_stocks[min_index], context.portfolio.total_value)
            g.is_stop = True
    else:
        code = context.portfolio.positions.keys()[0]

        index = bank_stocks.index(code)
```

```
        if raito[index] - min(raito) > g.inter:
            order_target(code, 0)
            min_index = raito.index(min(raito))
            order_value(bank_stocks[min_index], context.portfolio.total_value)
            g.is_stop = True

# 每天交易后调用
def after_trading_end(context):
    if bank_stocks[0] in context.portfolio.positions and context.portfolio.positions[bank_stocks[0]].total_amount > 0:
        record(code0=context.portfolio.positions[bank_stocks[0]].total_amount)
    if bank_stocks[1] in context.portfolio.positions and context.portfolio.positions[bank_stocks[1]].total_amount > 0:
        record(code1=context.portfolio.positions[bank_stocks[1]].total_amount)
    if bank_stocks[2] in context.portfolio.positions and context.portfolio.positions[bank_stocks[2]].total_amount > 0:
        record(code2=context.portfolio.positions[bank_stocks[2]].total_amount)
    if bank_stocks[3] in context.portfolio.positions and context.portfolio.positions[bank_stocks[3]].total_amount > 0:
        record(code3=context.portfolio.positions[bank_stocks[3]].total_amount)
```

持仓情况（部分截图）：

日期	品种	标的	多空	数量	可用数量	收盘价/结	市值/价值	盈亏/逐笔	开仓均价	持仓均价	保证金	盯市浮盈	今手数	仓位占比
2018/3/15	股票	建设银行(601939.XSH	多仓	3700股	0股	8.06	29822	148	8.02	-	0	148	3700股	98.90%
2018/3/15		Cash					320.07							
2018/3/16	股票	农业银行(601288.XSH	多仓	7300股	0股	4.07	29711	-146	4.09	-	0	-146	7300股	99.60%
2018/3/16		Cash					131.85							
2018/3/19	股票	农业银行(601288.XSH	多仓	7300股	7300股	4.08	29784	-73	4.09	-	0	73	0股	99.60%
2018/3/19		Cash					131.85							
2018/3/20	股票	农业银行(601288.XSH	多仓	7300股	7300股	4.1	29930	73	4.09	-	0	146	0股	99.60%
2018/3/20		Cash					131.85							
2018/3/21	股票	工商银行(601398.XSH	多仓	4600股	0股	6.49	29854	-230	6.54	-	0	-230	4600股	99.50%
2018/3/21		Cash					154.05							
2018/3/22	股票	建设银行(601939.XSH	多仓	3700股	0股	8.04	29748	148	8	-	0	148	3700股	99.50%
2018/3/22		Cash					135.99							
2018/3/23	股票	工商银行(601398.XSH	多仓	4600股	0股	6.3	28980	184	6.26	-	0	184	4600股	99.20%
2018/3/23		Cash					232.93							
2018/3/26	股票	中国银行(601988.XSH	多仓	7100股	0股	3.99	28329	-568	4.07	-	0	-568	7100股	99.70%
2018/3/26		Cash					91.02							
2018/3/27	股票	建设银行(601939.XSH	多仓	3800股	0股	7.45	28310	266	7.38	-	0	266	3800股	99.60%
2018/3/27		Cash					123.11							
2018/3/28	股票	中国银行(601988.XSH	多仓	7100股	0股	3.96	28116	142	3.94	-	0	142	7100股	98.70%
2018/3/28		Cash					381.03							
2018/3/29	股票	建设银行(601939.XSH	多仓	3800股	0股	7.67	29146	1140	7.37	-	0	1140	3800股	98.70%
2018/3/29		Cash					380.22							
2018/3/30	股票	中国银行(601988.XSH	多仓	7400股	0股	3.93	29082	-296	3.97	-	0	-296	7400股	99.60%
2018/3/30		Cash					106.78							
2018/4/2	股票	工商银行(601398.XSH	多仓	4800股	0股	5.96	28608	-384	6.04	-	0	-384	4800股	99.70%
2018/4/2		Cash					81.59							
2018/4/3	股票	建设银行(601939.XSH	多仓	3800股	0股	7.46	28348	152	7.42	-	0	152	3800股	99.10%
2018/4/3		Cash					261.28							
2018/4/4	股票	中国银行(601988.XSH	多仓	7300股	0股	3.85	28105	-146	3.87	-	0	-146	7300股	98.90%
2018/4/4		Cash					318.05							
2018/4/9	股票	建设银行(601939.XSH	多仓	3800股	0股	7.36	27968	152	7.32	-	0	152	3800股	98.30%
2018/4/9	股票	中国银行(601988.XSH	多仓	100股	0股	3.85	385	2	3.83	-	0	2	100股	1.40%
2018/4/9		Cash					106.29							
2018/4/10	股票	农业银行(601288.XSH	多仓	7400股	0股	3.94	29156	960	3.81	-	0	960	7400股	99.30%

收益情况：

代码模拟交易时间是从 2018/03/15-2018/05/14，为期两个月，收益率为 0.95%，年化收益率为 6.28%，最大回测为 7.631%，模拟交易期间策略的收益率普遍高于基准收益率（沪深 300 指数收益率），α值为 0.28，β值为 0.946.

三、 总结

本次量化交易的结果虽然超过了沪深 300 指数的收益率，但是收益率长期处于负数阶段，说明其收益性会受到整体股市的表现影响，并不能很好的规避市场风险，实现稳定的正收益。

再者，由于本次量化交易在交易过程中没有考虑滑点问题（即程序捕捉到一次的合适价格后，下单买入或者卖出不一定能够以该合适价格成交）。虽然本次选取的股票都是上证 50 成分中的银行股票，交易量较大，价格变动幅度不大，但是在实际的模拟交易过程中依旧可能存在无法以目标价格成交的情况，因此量化交易程序可优化的方向是：增加考虑滑点问题，以及收益的稳定性。

以及回测最大值达到了 7.631%，说明本策略在稳定性方面还有待改善。