Quantitative Trading strategies report

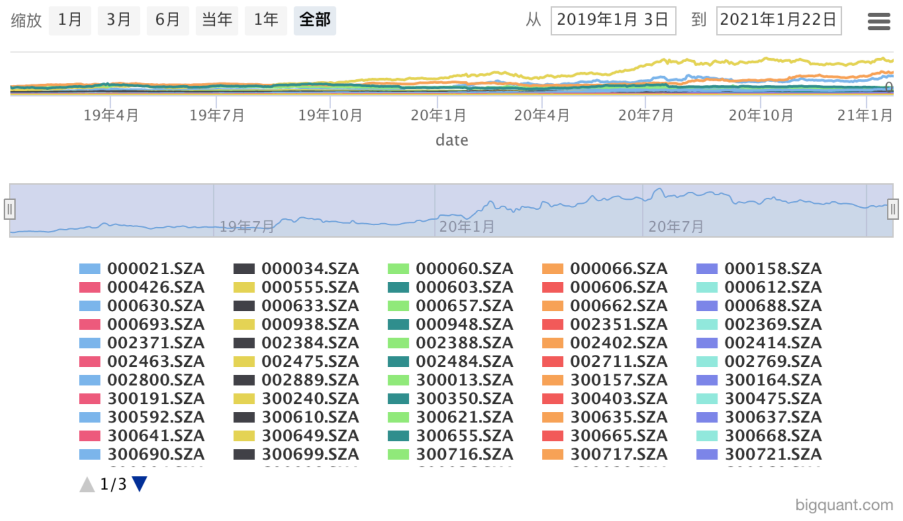
Yan Chen

June 3, 2021

Here I used BigQuant platform `https://bigquant.com/` for the simulation and development of trading strategies listed below.

1 Introduction

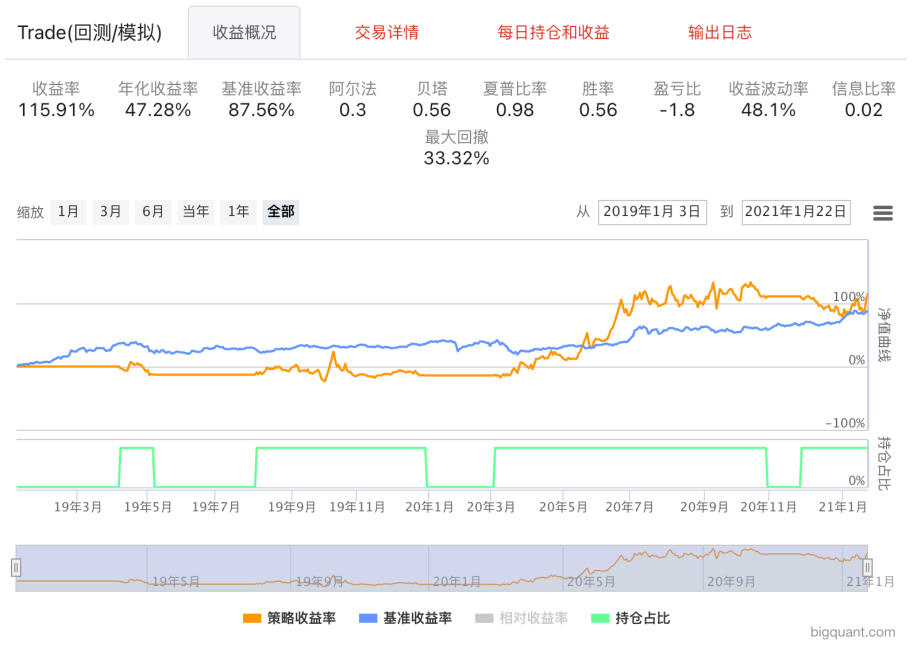
In this strategy, I pick from a prefixed set of stocks to form the portfolio here in my strategies, which includes more than 100 stocks in the file. And here I use the period of 01/03/2019 - 1/22/2021 as the backtesting period. The transaction fee is set as 0.3%, and if the fee is less than 5 RMB, then it's set as 5 RMB. I use "CSI300 Index" as the benchmark. And in the simulation graph, we can find metrics including asset return, sharpe ratio and drawdown, etc. Before diving into any of the strategies below, we may first get a quick glimpse at the general trend of the stock prices.



2 Strategy based on technical indicators

2.1 Strategy 1: MACD + MA

Empirically Traders find that the analysis of 12- and 26-day EMA(exponential moving average) very useful for determining buy-and-sell points. The standard MACD is the 12-day EMA subtracted by the 26-day EMA, which is also called the DIF. The MACD histogram measures the signed distance between the MACD and its signal line calculated using the 9-day EMA of the MACD, which is the DEA. When DIF crosses above DEA, there's a buy signal in the strategy. Another indicator we take into consideration is the simple moving average. The strategy when the 5-day short-term moving average line is above the 20-day long-term moving average line, we'll buy these stocks. So the strategy presented here is a combination of the above two indicators. The detailed strategy is as follows: The rebalance period is every 20 workdays, the maximum number of stocks in portfolio is 10. Now suppose we have 10, 000 RMB, and today is thee first day of trading. We select some stocks according to the indicators listed above, and we distribute the capital evenly on these 10 stocks (when the stock number is less than 10, we'll just evenly distribute cash on them as well). And 20 workdays later, we'll first sell these stocks at their open price, then we'll use the cash obtained to buy the first 10 stocks selected again according to the above indicators, and evenly distribute our cash on them as well, etc. and etc. And the result of backtesting is as follows

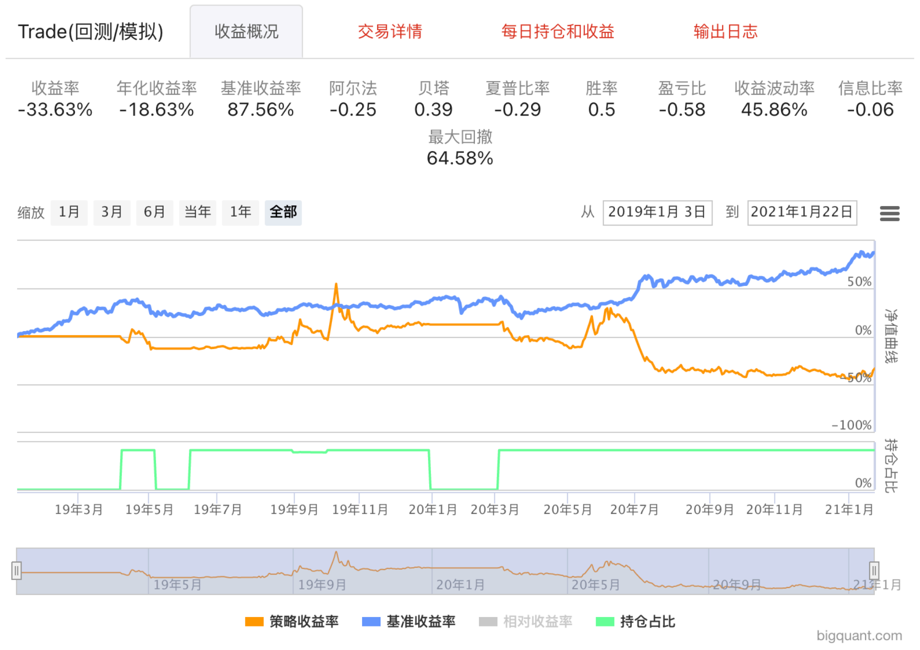

Trade(回测/模拟)
收益概况
交易详情
每日持仓和收益
输出日志
缩放 1月 3月 6月 当年 1年 全部
从 2019年1月 3日 到 2021年1月22日
100%
0%
-100%
净值曲线
持仓占比
0%
19年3月 19年5月 19年7月 19年9月 19年11月 20年1月 20年3月 20年5月 20年7月 20年9月 20年11月 21年1月
19年5月 19年9月 20年1月 20年5月 20年9月 21年1月
策略收益率
基准收益率
相对收益率
持仓占比
bigquant.com

收益率	年化收益率	基准收益率	阿尔法	贝塔	夏普比率	胜率	盈亏比	收益波动率	信息比率	最大回撤
115.91%	47.28%	87.56%	0.3	0.56	0.98	0.56	-1.8	48.1%	0.02	33.32%

As we can see, the strategy is close to the benchmark, whilee the sharpe ratio is 0.98, and drawdown is 33.32%.

2.2 Strategy 2: MACD only

This time, I kept everything else the same and only used MACD indicator, the strategy became worse than the benchmark, and has both a negative yearly return and a negative sharpe ratio.



收益率	年化收益率	基准收益率	阿尔法	贝塔	夏普比率	胜率	盈亏比	收益波动率	信息比率	最大回撤
-33.63%	-18.63%	87.56%	-0.25	0.39	-0.29	0.5	-0.58	45.86%	-0.06	64.58%

Trade(回测/模拟)
收益概况
交易详情
每日持仓和收益
输出日志
缩放 1月 3月 6月 当年 1年 全部
从 2019年1月 3日 到 2021年1月22日
50%
0%
-50%
-100%
净值曲线
0%
持仓占比
19年3月 19年5月 19年7月 19年9月 19年11月 20年1月 20年3月 20年5月 20年7月 20年9月 20年11月 21年1月
19年5月 19年9月 20年1月 20年5月 20年9月 21年1月
策略收益率
基准收益率
相对收益率
持仓占比
bigquant.com

3 Quadratic Programming based methods

Here we define a few variables

1. $x = (x_1, \ldots, x_n)$ represents the weight of each stock, where n is the size of the portfolio. Here $\sum_{i=1}^{n} x_i = 1$

2. $\mu = (\mu_1, \ldots, \mu_n)$ is the expected return for each stock based on historical data. The calculation is based on the average of every 5 workdays.

3. Σ is the covariance matrix of stock returns.

In the following QP strategies, we set the rebalance period as 10 days consistently.

3.1 Strategy 3: classical QP

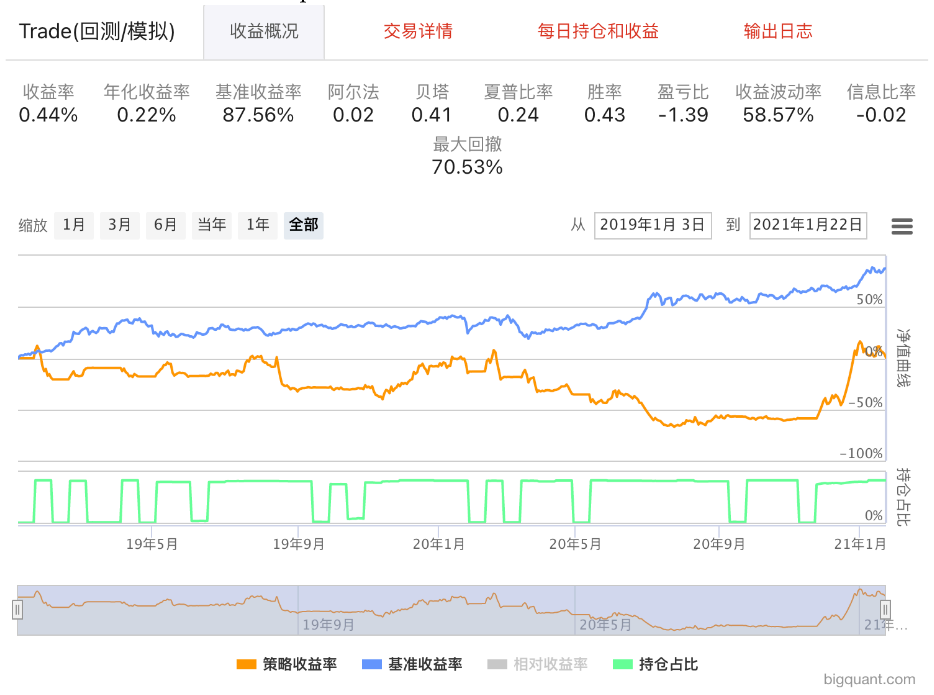
The idea is to obtain a sharpe ratio as high as possible. So here we consider the following Lagrangian strategy with

$$\underset{x}{\textbf{minimize}} \quad -\mu^T x + \lambda \frac{1}{2} x^T \Sigma x$$

$$\textbf{subject to} \quad \sum_{i=1}^{n} x_i = 1$$

$$x_i \geq 0; \ i = 1, 2, \ldots, n$$

For the sake of operating time of the platform, here I selected the first 30 stocks in the stock list and include them in the optimization. The result is as follows.

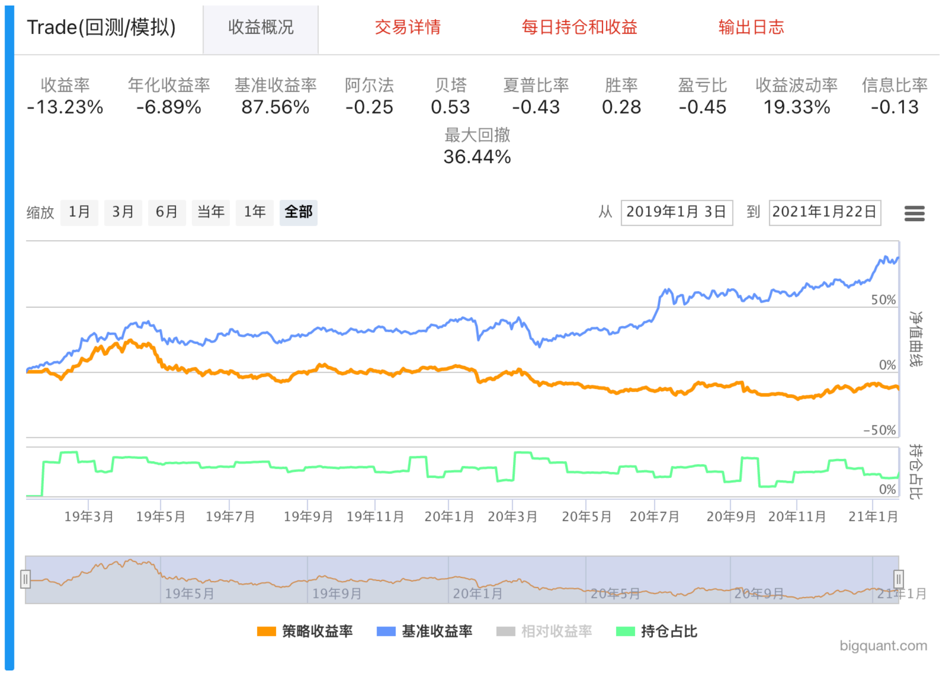


Trade(回测/模拟) 收益概况 交易详情 每日持仓和收益 输出日志

收益率	年化收益率	基准收益率	阿尔法	贝塔	夏普比率	胜率	盈亏比	收益波动率	信息比率	最大回撤
0.44%	0.22%	87.56%	0.02	0.41	0.24	0.43	-1.39	58.57%	-0.02	70.53%

3.2 Strategy 4: diversify the portfolio

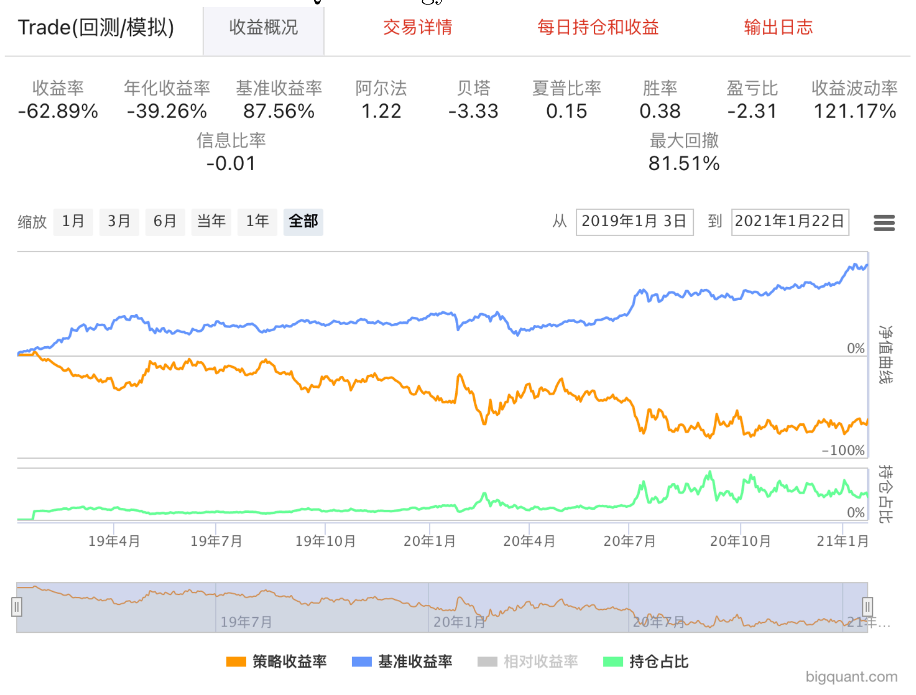
If we use the classical QP method, a potential problem is that some of the weights might be very high while some might be very low, this would be equivalent to selecting only a few stocks among the many stocks, which doesn't diversify away the risk. So another way to tackle this is that we can limit the weight of each stock, and at the same time, also limiting the weights of each sector of the stocks. And in our example of the top 30 stocks, we have 3 sectors which are '710000', '210000', '220000'.



收益率	年化收益率	基准收益率	阿尔法	贝塔	夏普比率	胜率	盈亏比	收益波动率	信息比率	最大回撤
-13.23%	-6.89%	87.56%	-0.25	0.53	-0.43	0.28	-0.45	19.33%	-0.13	36.44%

3.3 Strategy with lower bound on stock weights

The following figure corresponds to the strategy with lower bound on the weights of each stock in addition to the diversified QP strategy above.



收益率	年化收益率	基准收益率	阿尔法	贝塔	夏普比率	胜率	盈亏比	收益波动率	信息比率	最大回撤
-62.89%	-39.26%	87.56%	1.22	-3.33	0.15	0.38	-2.31	121.17%	-0.01	81.51%

Strategies	Return	Sharpe Ratio	Drawdown	Volatility
MACD+MA	115.91%	0.98	33.32%	48.1%
MACD	-33.63%	-0.29	45.86%	64.58%
classical QP	0.44%	0.24	70.53%	58.57%
Diversified QP	-13.23%	-0.43	36.44%	19.33%
lower bound on diversified	-62.89%	0.15	81.51%	121.17%

Table 1: Comparison of different strategies

3.4 Comparison

Seems like so far using technical indicators MACD+MA works best among all the methods, while the diversification did help with lowering volatility. Perhaps we may try to add in more stocks into the original MACD+MA strategy, e.g. trading 20 or 30 stocks instead of just 10, which may help with lowering volatility.

4 Code

4.1 MACD+MA

```
import numpy as np
import pandas as pd
import talib

start_date = '2019-01-03'
end_date = '2021-01-22'
stock_list = pd.read_csv('stock_list.csv', header=None)
instruments = stock_list.values[:,0].tolist()

#initial capitalabs
capital_base = 10000

#benchmark '沪深300'
benchmark = '000300.INDX'

#account rebalance period
rebalance_period = 20

#number of stocks held
```

```
stock_count = 10


#Strategy:

#crossing point of MACD:
#DIF short = 12; DIF long = 26; DEA=9; DIF cross above DEA (long); otherwise short

#MA long:
#MA short = 5; MA long = 20; MA short crosses above MA long (long); otherwise short


def initialize(context):
    #transaction fee
    context.set_commission(PerOrder(buy_cost=0.0003, sell_cost=0.0003, min_cost=5))
    #short moving average (DIF)
    context.short_period = 12 #短期均线
    #long moving average (DIF)
    context.long_period=26 #长期均线
    #DEA
    context.smooth_period = 9

    #observation length for historical data
    context.observation=50

    context.ma_short_period = 5 #MA short
    context.ma_long_period = 20 #MA long


    #context.stock_weights = T.norm([1/stock_count for i in range(0,stock_count)]) #


def handle_data(context,data):
    if context.trading_day_index < context.observation:
        return

    #today is the first transaction day
    today = data.current_dt.strftime('%Y-%m-%d')
    #cash = context.portfolio.cash

    # context.trading_day_index: the index of trading date, the first is 0
    # will only trade every 20 days (hold_day)
    if context.trading_day_index % rebalance_period != 0:
        return

    equities = context.portfolio.positions

    #sell all the stocks in the portfolio
    for equity in equities:
        if data.can_trade(equity):
            context.order_target(equity, 0)

    #buy the top 10 stocks
    stock_candidates = []
    for i, instrument in enumerate(instruments):
        sid = context.symbol(instruments[i]) #stock index
        price = data.current(sid, 'price') # current price
        # historical daily prices
        prices = data.history(sid, 'price', context.observation, '1d')
        if np.isnan(price):
            continue
        # 用Talib计算MACD取值，得到三个时间序列数组，分别为macd, signal 和 hist
        macd, signal, hist = talib.MACD(np.array(prices), context.short_period, cont

        # long term and short term MA
        short_mavg = data.history(sid, 'price',context.ma_short_period, '1d').mean()
        long_mavg = data.history(sid, 'price',context.ma_long_period, '1d').mean() #

        # 计算现在portfolio中股票的仓位
        cur_position = context.portfolio.positions[sid].amount

        # sell all the stocks every hold_day (20) days
        if cur_position > 0 and data.can_trade(sid):
            context.order_target_value(sid, 0)
        # buy (cross above)
        if macd[-1] - signal[-1] > 0 and macd[-2] - signal[-2] < 0:
            # 如果短期均线大于长期均线形成金叉，并且MA 短线在 MA 长线上方时，并且该股
            if short_mavg > long_mavg:
                if data.can_trade(sid): #the stock is tradable
                    stock_candidates.append((sid, price))

        stocks = stock_candidates[: stock_count]
        stocks_len =  len(stocks)
        if stocks_len > 0:
            context.stock_weights = T.norm([1 / stocks_len for i in range(0, stocks_
            stock_weights = context.stock_weights
            sid_prices = list(stocks)
            for i, instrument in enumerate(sid_prices):
                cash = capital_base * stock_weights[i]
                if cash > 0:
                    order_unit =  int(cash/instrument[1]/100)*100
                    context.order(instrument[0], order_unit)

        else:
            pass
```

4.2 Diversified QP

```
def optimal_portfolio(returns):
    n = len(returns)
    returns = np.asmatrix(returns)

    N = 100
    mus = [10**(5.0 * t/N - 1.0) for t in range(N)]

    # objective function
    S = opt.matrix(np.cov(returns))
    pbar = opt.matrix(np.mean(returns, axis=1))


    #constraints
    G_1 = -np.eye(n)
```

```
G_2 = np.eye(n)
G_3 = np.zeros((3,n))
G_3[0, :10] = 1
G_3[1, 10:20] = 1
G_3[2, 20:] = 1
G_np = np.concatenate((G_1, G_2, G_3), axis=0)
#print(G.shape)
G = opt.matrix(G_np)

h_1 = np.zeros((n,1))
h_2 = np.ones((n,1)) * context.max_weight_per_stock # 0.2
h_3 = np.ones((3,1)) * context.max_weight_per_sector # 0.5
h_np = np.concatenate((h_1, h_2, h_3), axis=0)
#print(h_np.shape)
h = opt.matrix(h_np)



A = opt.matrix(1.0, (1, n))
b = opt.matrix(1.0)

# 使用凸优化计算有效前沿
portfolios = [solvers.qp(mu*S, -pbar, G, h, A, b)['x']
              for mu in mus]
## 计算有效前沿的收益率和风险
returns = [blas.dot(pbar, x) for x in portfolios]
risks = [np.sqrt(blas.dot(x, S*x)) for x in portfolios]
m1 = np.polyfit(returns, risks, 2)
x1 = np.sqrt(m1[2] / m1[0])

# 计算最优组合
wt = solvers.qp(opt.matrix(x1 * S), -pbar, G, h, A, b)['x']
return np.asarray(wt), returns, risks
```