Not Over Thinking

Paired Switching

Algorithmic Trading Strategy with Full Code

Haixiang

2023.09 | Vol 30.

hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

This strategy is very flexible. Investors could use stocks, funds, or ETFs as an investment vehicle. We show simple trading rules for a sample strategy from the source research paper.

The investor uses two Vanguard funds as his investment vehicles – one equity fund (VFINX) and one government bond fund (VUSTX). These two funds have a negative correlation as they are proxies for two negatively correlated asset classes. The investor looks at the performance of the two funds over the prior quarter and buys the fund that has a higher return during the ranking period. The position is held for one quarter (the investment period). At the end of the investment period, the cycle is repeated.

BUY	SELL
the fund that has a higher return during the ranking period	The opposite

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Bond, Equity
FINANCIAL INSTRUMENTS	ETFs, funds, stocks
REGION	Global
PERIOD OF REBALANCING	Quarterly
NO. OF TRADED INSTRUMENTS	2
WEIGHTING	Equal weighting
LOOKBACK PERIODS	Quarter
LONG/SHORT	Long only

ALGORITHM

```
from AlgorithmImports import *

class PairedSwitching(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2004, 1, 1)
        self.SetCash(100000)

        self.first_symbol = self.AddEquity("SPY", Resolution.Daily).Symbol
        self.second_symbol = self.AddEquity("AGG", Resolution.Daily).Symbol
        self.recent_month = -1

    def OnData(self, data):
        if self.Time.month == self.recent_month:
            return
        self.recent_month = self.Time.month

        if(self.recent_month % 3 == 0):
            if self.first_symbol in data and self.second_symbol in data:
```

```
                history_call = self.History([self.first_symbol, self.second_symbol],
timedelta(days=90))
                if not history_call.empty:
                    first_bars = history_call.loc[self.first_symbol.Value]
                    last_p1 = first_bars["close"].iloc[0]

                    second_bars = history_call.loc[self.second_symbol.Value]
                    last_p2 = second_bars["close"].iloc[0]

                    # Calculates performance of funds over the prior quarter.
                    first_performance = (float(self.Securities[self.first_symbol].Price) -
float(last_p1)) / (float(self.Securities[self.first_symbol].Price))
                    second_performance = (float(self.Securities[self.second_symbol].Price)
- float(last_p2)) / (float(self.Securities[self.second_symbol].Price))

                    # Buys the fund that has the higher return during the period.
                    if(first_performance > second_performance):
                        if(self.Securities[self.second_symbol].Invested):
                            self.Liquidate(self.second_symbol)
                        self.SetHoldings(self.first_symbol, 1)
                    else:
                        if(self.Securities[self.first_symbol].Invested):
                            self.Liquidate(self.first_symbol)
                        self.SetHoldings(self.second_symbol, 1)
```

BACKTESTING PERFORMANCE

Fig 1. Overall Performance

PSR	2.921%	Sharpe Ratio	0.655
Total Trades	64	Average Win	5.92%
Average Loss	-1.87%	Compounding Annual Return	8.487%
Drawdown	24.100%	Expectancy	2.675
Net Profit	378.448%	Loss Rate	12%
Win Rate	88%	Profit-Loss Ratio	3.17
Alpha	0.039	Beta	0.323
Annual Standard Deviation	0.096	Annual Variance	0.009
Information Ratio	-0.08	Tracking Error	0.134
Treynor Ratio	0.194	Total Fees	$564.41
Estimated Strategy Capacity	$230000000.00	Lowest Capacity Asset	AGG SSC0EI5J2F6T

Fig 2. Performance Metrics

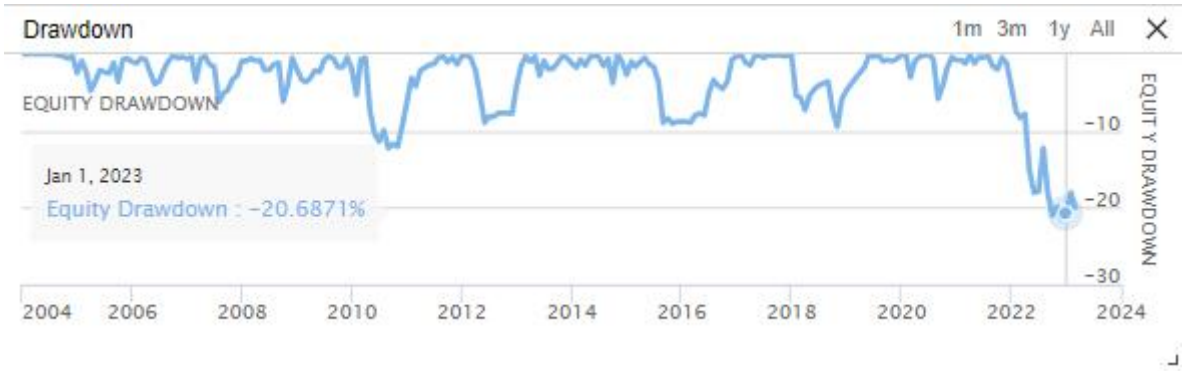

Fig 3. Drawdown