Not Over Thinking

Short Term Reversal with Futures

Algorithmic Trading Strategy with Full Code

Haixiang
2023.09 | Vol 34.
hxyan.2015@gmail.com | github.com/hxyan2020

STRATEGY & ECONOMIC RATIONALE

The investment universe consists of 24 types of US futures contracts (4 currencies, five financials, eight agricultural, seven commodities). A weekly time frame is used – a Wednesday- Wednesday interval. The contract closest to expiration is used, except within the delivery month, in which the second-nearest contract is used. Rolling into the second nearest contract is done at the beginning of the delivery month.

The contract is defined as the high- (low-) volume contract if the contract's volume changes between period from t-1 to t and period from t-2 to t-1 is above (below) the median volume change of all contracts (weekly trading volume is detrended by dividing the trading volume by its sample mean to make the volume measure comparable across markets).

All contracts are also assigned to either high-open interest (top 50% of changes in open interest) or low-open interest groups (bottom 50% of changes in open interest) based on lagged changes in open interest between the period from t-1 to t and period from t-2 to t-1. The investor goes long (short) on futures from the high-volume, low-open interest group with the lowest (greatest) returns in the previous week. The weight of each contract is proportional to the difference between the return of the contract over the past one week and the equal-weighted average of returns on the N (number of contracts in a group) contracts during that period.

BUY	SELL
futures from the high-volume, low-open interest group with the lowest (greatest) returns in the previous week	The opposite

PARAMETER & VARIABLES

PARAMETER	VALUE
MARKETS TRADED	Bond, Commodity, Currency, Equity
FINANCIAL INSTRUMENTS	CFDs, futures
REGION	United States
PERIOD OF REBALANCING	Weekly
NO. OF TRADED INSTRUMENTS	6
WEIGHTING	Equal weighting
LOOKBACK PERIODS	Weekly
LONG/SHORT	Long only

ALGORITHM

```python
from collections import deque
import numpy as np

class ShortTermReversal(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2010, 1, 1)
        self.SetCash(100000)

        symbols:dict = {
            'CME_S1': Futures.Grains.Soybeans,
            'CME_W1': Futures.Grains.Wheat,
            'CME_BO1': Futures.Grains.SoybeanOil,
            'CME_C1': Futures.Grains.Corn,
            'CME_LC1': Futures.Meats.LiveCattle,
            'CME_FC1': Futures.Meats.FeederCattle,
            'CME_KW2': Futures.Grains.Wheat,
            'ICE_CC1': Futures.Softs.Cocoa,
            'ICE_SB1': Futures.Softs.Sugar11CME,

            'CME_GC1': Futures.Metals.Gold,
            'CME_SI1': Futures.Metals.Silver,
            'CME_PL1': Futures.Metals.Platinum,

            'CME_RB1': Futures.Energies.Gasoline,
            'ICE_WT1': Futures.Energies.CrudeOilWTI,
            'ICE_O1': Futures.Energies.HeatingOil,

            'CME_BP1': Futures.Currencies.GBP,
            'CME_EC1': Futures.Currencies.EUR,
            'CME_JY1': Futures.Currencies.JPY,
            'CME_SF1': Futures.Currencies.CHF,

            'CME_ES1': Futures.Indices.SP500EMini,
            'CME_TY1': Futures.Financials.Y10TreasuryNote,
            'CME_FV1': Futures.Financials.Y5TreasuryNote,
        }

        self.period:int = 14
        self.SetWarmUp(self.period, Resolution.Daily)

        self.futures_info:dict = {}
        self.min_expiration_days:int = 2
        self.max_expiration_days:int = 360

        # daily close, volume and open interest data
        self.data:dict = {}

        for qp_symbol, qc_future in symbols.items():
            # QP futures
            data:Security = self.AddData(QuantpediaFutures, qp_symbol, Resolution.Daily)
            data.SetFeeModel(CustomFeeModel())
            data.SetLeverage(5)
            self.data[data.Symbol] = deque(maxlen=self.period)

            # QC futures
            future:Future = self.AddFuture(qc_future, Resolution.Daily, 
dataNormalizationMode=DataNormalizationMode.Raw)
            future.SetFilter(timedelta(days=self.min_expiration_days), 
timedelta(days=self.max_expiration_days))

            self.futures_info[future.Symbol.Value] = FuturesInfo(data.Symbol)

        self.recent_month:int = -1
```

```
    def find_and_update_contracts(self, futures_chain, symbol) -> None:
        near_contract:FuturesContract = None

        if symbol in futures_chain:
            contracts:list = [contract for contract in futures_chain[symbol] if 
contract.Expiry.date() > self.Time.date()]

            if len(contracts) >= 1:
                contracts:list = sorted(contracts, key=lambda x: x.Expiry, reverse=False)
                near_contract = contracts[0]

        self.futures_info[symbol].update_contracts(near_contract)

    def OnData(self, data):
        if data.FutureChains.Count > 0:
            for symbol, futures_info in self.futures_info.items():
                # check if near contract is expired or is not initialized
                if not futures_info.is_initialized() or \
                    (futures_info.is_initialized() and 
futures_info.near_contract.Expiry.date() == self.Time.date()):
                    self.find_and_update_contracts(data.FutureChains, symbol)

        rebalance_flag:bool = False
        ret_volume_oi_data:dict[Symbol, tuple] = {}

        # roll return calculation
        for symbol, futures_info in self.futures_info.items():
            # futures data is present in the algorithm
            if futures_info.quantpedia_future in data and 
data[futures_info.quantpedia_future]:
                if futures_info.is_initialized():
                    near_c:FuturesContract = futures_info.near_contract

                    if self.Securities.ContainsKey(near_c.Symbol):
                        if futures_info.is_initialized():
                            # store daily data
                            price:float = data[futures_info.quantpedia_future].Value
                            vol:int = self.Securities[near_c.Symbol].Volume
                            oi:int = self.Securities[near_c.Symbol].OpenInterest

                            if price != 0 and vol != 0 and oi != 0:
                                self.data[futures_info.quantpedia_future].append((price, 
vol, oi))

                    # new month rebalance
                    if self.Time.month != self.recent_month and not self.IsWarmingUp:
                        self.recent_month = self.Time.month
                        rebalance_flag = True

                    if rebalance_flag:
                        if len(self.data[futures_info.quantpedia_future]) == 
self.data[futures_info.quantpedia_future].maxlen:
```

```
                            # performance
                            prices:list[float] = [x[0] for x in 
self.data[futures_info.quantpedia_future]]
                            half:list[float] = int(len(prices)/2)
                            prices:list[float] = prices[-half:]
                            ret:float = prices[-1] / prices[0] - 1

                            # volume change
                            volumes:list[int] = [x[1] for x in 
self.data[futures_info.quantpedia_future]]
                            volumes_t1:list[int] = volumes[-half:]
                            t1_vol_mean:float = np.mean(volumes_t1)
                            t1_vol_total:float = sum(volumes_t1) / t1_vol_mean
                            volumes_t2:list[int] = volumes[:half]
                            t2_vol_mean:float = np.mean(volumes_t2)
                            t2_vol_total:float = sum(volumes_t2) / t2_vol_mean
                            volume_weekly_diff:float = t1_vol_total - t2_vol_total

                            # open interest change
                            interests:list[int] = [x[2] for x in 
self.data[futures_info.quantpedia_future]]
                            t1_oi:list[int] = interests[-half:]
                            t1_oi_total:float = sum(t1_oi)
                            t2_oi:list[int] = interests[:half]
                            t2_oi_total:float = sum(t2_oi)
                            oi_weekly_diff:float = t1_oi_total - t2_oi_total

                            # store weekly diff data
                            ret_volume_oi_data[futures_info.quantpedia_future] = (ret, 
volume_weekly_diff, oi_weekly_diff)

        if rebalance_flag:
            weight:dict[Symbol, float] = {}

            if len(ret_volume_oi_data) > 4:
                volume_sorted:list = sorted(ret_volume_oi_data.items(), key = lambda x: 
x[1][1], reverse = True)
                half:int = int(len(volume_sorted)/2)
                high_volume:list = [x for x in volume_sorted[:half]]

                open_interest_sorted:list = sorted(ret_volume_oi_data.items(), key = 
lambda x: x[1][2], reverse = True)
                half = int(len(open_interest_sorted)/2)
                low_oi:list = [x for x in open_interest_sorted[-half:]]

                filtered:list = [x for x in high_volume if x in low_oi]
                filtered_by_return:list = sorted(filtered, key = lambda x : x[0], reverse 
= True)
                half = int(len(filtered_by_return) / 2)

                long:list[Symbol] = filtered_by_return[-half:]
                short:list[Symbol] = filtered_by_return[:half]
                if len(long + short) >= 2:
                    # return weighting
                    diff:dict[Symbol, float] = {}
                    avg_ret:float = np.average([x[1][0] for x in long + short])

                    for symbol, ret_volume_oi in long + short:
                        diff[symbol] = ret_volume_oi[0] - avg_ret

                    total_diff:float = sum([abs(x[1]) for x in diff.items()])
                    long_symbols:list[Symbol] = [x[0] for x in long]

                    if total_diff != 0:
                        for symbol, data in long + short:
                            if symbol in long_symbols:
                                weight[symbol] = diff[symbol] / total_diff
                            else:
                                weight[symbol] = - diff[symbol] / total_diff

            # trade execution
            invested:list[Symbol] = [x.Key for x in self.Portfolio if x.Value.Invested]
            for symbol in invested:
                if symbol not in weight:
                    self.Liquidate(symbol)

            for symbol, w in weight.items():
                self.SetHoldings(symbol, w)

class FuturesInfo():
    def __init__(self, quantpedia_future:Symbol) -> None:
        self.quantpedia_future:Symbol = quantpedia_future
        self.near_contract:FuturesContract = None

    def update_contracts(self, near_contract:FuturesContract) -> None:
        self.near_contract = near_contract

    def is_initialized(self) -> bool:
        return self.near_contract is not None

# Custom fee model.
class CustomFeeModel():
    def GetOrderFee(self, parameters):
        fee = parameters.Security.Price * parameters.Order.AbsoluteQuantity * 0.00005
        return OrderFee(CashAmount(fee, "USD"))

# Quantpedia data.
# NOTE: IMPORTANT: Data order must be ascending (datewise)
class QuantpediaFutures(PythonData):
    def GetSource(self, config, date, isLiveMode):
        return
SubscriptionDataSource("data.quantpedia.com/backtesting_data/futures/{0}.csv".format(confi
g.Symbol.Value), SubscriptionTransportMedium.RemoteFile, FileFormat.Csv)

    def Reader(self, config, line, date, isLiveMode):
        data = QuantpediaFutures()
        data.Symbol = config.Symbol

        if not line[0].isdigit(): return None
        split = line.split(';')

        data.Time = datetime.strptime(split[0], "%d.%m.%Y") + timedelta(days=1)
        data['back_adjusted'] = float(split[1])
        data['spliced'] = float(split[2])
        data.Value = float(split[1])

        return data
```

BACKTESTING PERFORMANCE

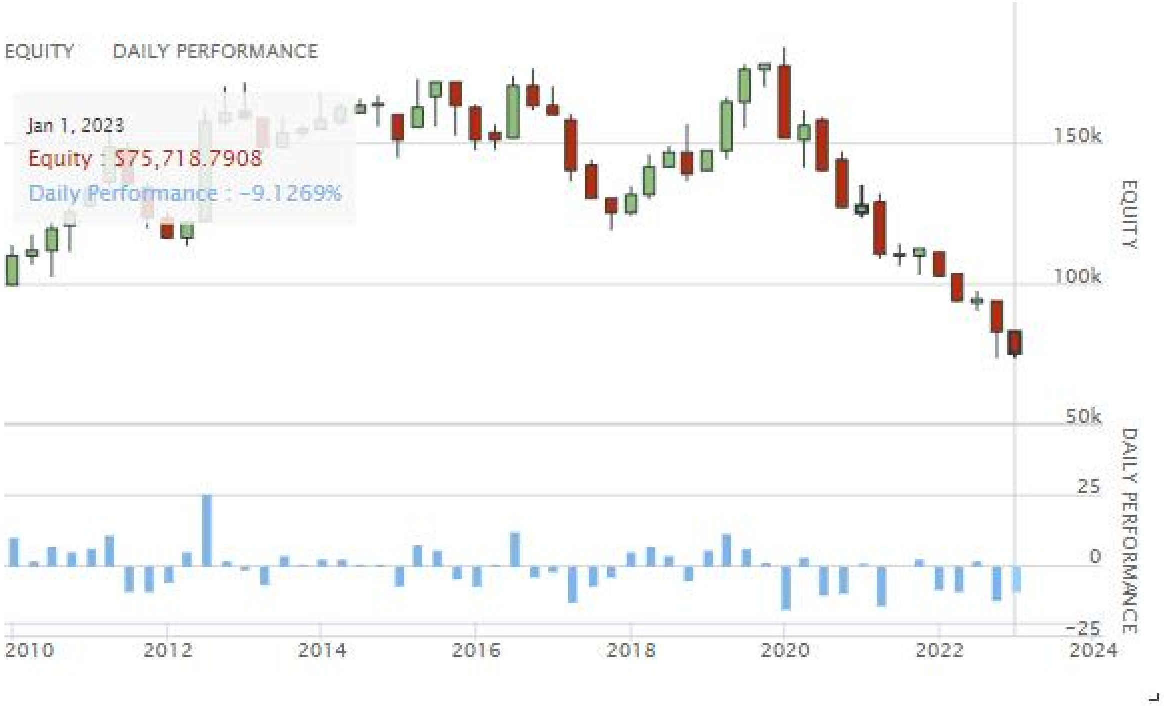
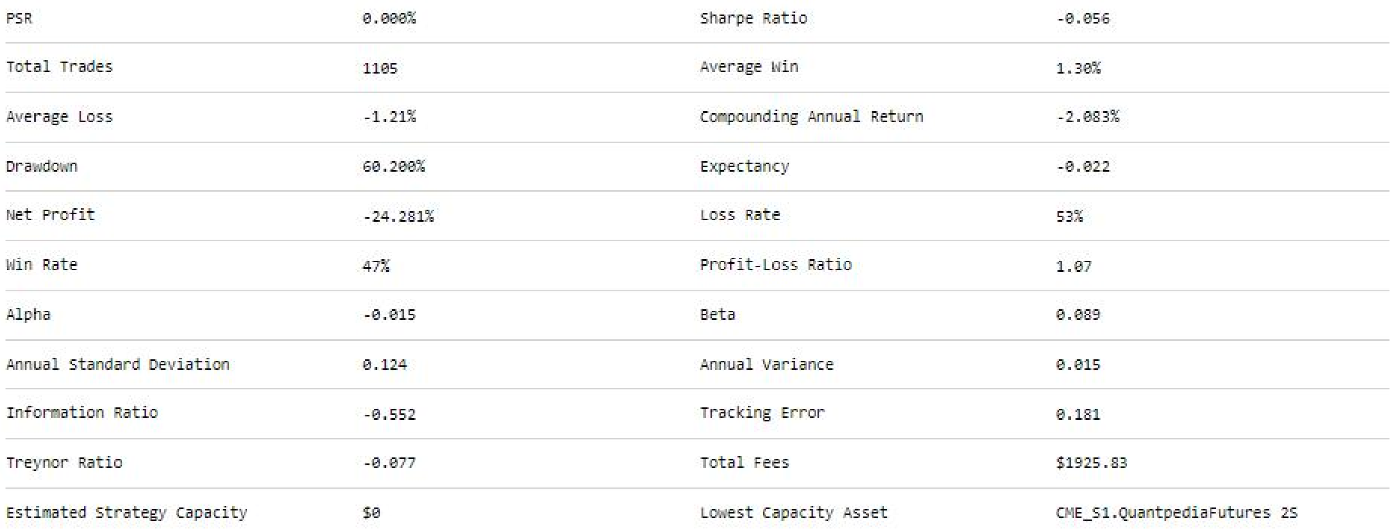


Fig 1. Overall Performance



PSR	0.000%	Sharpe Ratio	-0.056
Total Trades	1105	Average Win	1.30%
Average Loss	-1.21%	Compounding Annual Return	-2.083%
Drawdown	60.200%	Expectancy	-0.022
Net Profit	-24.281%	Loss Rate	53%
Win Rate	47%	Profit-Loss Ratio	1.07
Alpha	-0.015	Beta	0.089
Annual Standard Deviation	0.124	Annual Variance	0.015
Information Ratio	-0.552	Tracking Error	0.181
Treynor Ratio	-0.077	Total Fees	$1925.83
Estimated Strategy Capacity	$0	Lowest Capacity Asset	CME_S1.QuantpediaFutures 2S

Fig 2. Performance Metrics

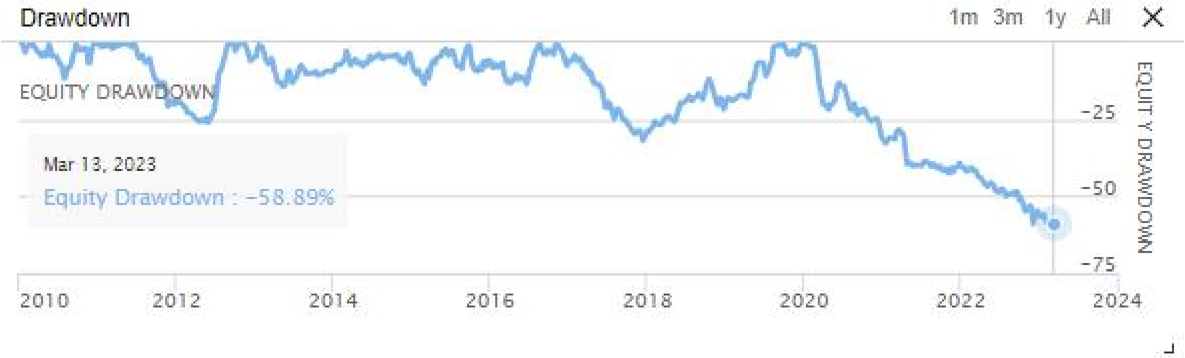


Fig 3. Drawdown

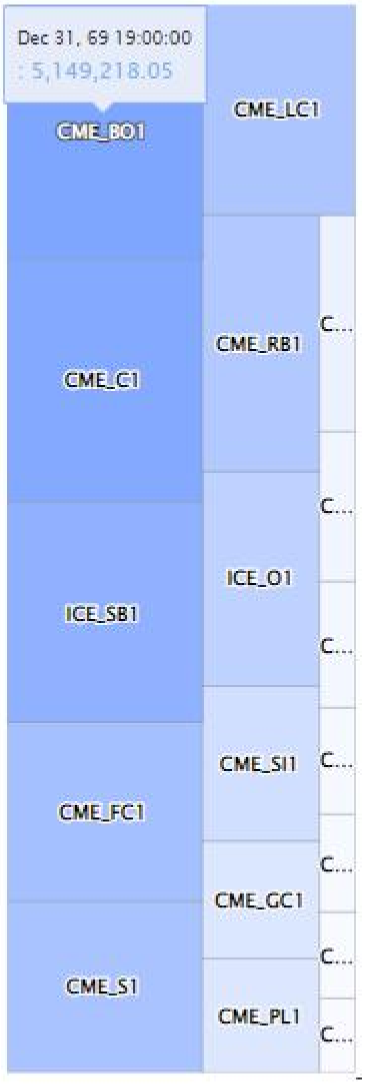


Fig 4. Assets Sales Volume