Group Project Presentation Slides

Modern Portfolio Management With Backtesting

Group Members: Hanfang Chen, Chuanyang Ren,
Sperydon Koumarianos, Guanhua Huang

Portfolio Optimization with Backtesting

Motivation: ETF has been claimed that systematically better than mutual fund. In this project, we would back test multiple strategies on real data to verify this claim. Potentially, it can be scaled up to be applied at positive fund managers

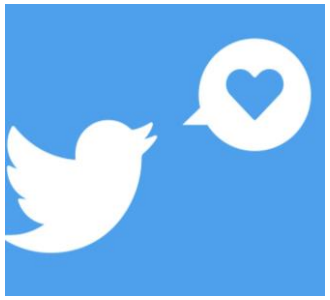
This is one of the best performing ETFs in last year. The strategy is simple, investing in the ten largest tech firms in the U.S.

The goals of this project:

1. Decipher its mechanism by portfolio optimization
2. Try different objective functions and constraints
3. Backtest on real data to generate insights.

Background

FNGU is a 3X micro sector ETF. The most challenging part is to find the 10 most ‘profitable’ stocks. But in this scenario, the assets are pre-defined. FNGU uses options to create 3X leverage. In the backtest part, this might be challenging for us.



Classical Markovitz Efficient Frontier

Minimum Variance Portfolio

$$min\ (w^T \Sigma w)$$

subject to,

$$\sum_{i=1}^{n} w_i = 1$$

$$\mu^T w = r_j$$

- There are n assets in the portfolio. Long and short positions are both available
- $w \in \mathbb{R}^n$, is each assets' allocation in the portfolio
- $\mu \in \mathbb{R}^n$, is each asset's historical return
- $\Sigma \in \mathbb{R}^{n \times n}$, is the covariance matrix
- $r_j \in \mathbb{R}^+$, is the designated portfolio return

Maximize Sharpe Ratio Portfolio

$$max\ (\frac{\mu^T w - r_f}{\sqrt{\mu^T \Sigma w}})$$

subject to,

$$\sum_{i=1}^{n} w_i = 1$$

- There are n assets in the portfolio. Long and short positions are both available
- $w \in \mathbb{R}^n$, is each assets' allocation in the portfolio
- $\mu \in \mathbb{R}^n$, is each asset's historical return
- $\Sigma \in \mathbb{R}^{n \times n}$, is the covariance matrix
- $r_f \in \mathbb{R}^+$, is the risk free rate

Other Alternatives

Minimum Variance Portfolio with Penalty

$$min\ (w^T \Sigma w + \gamma w^T w)$$

subject to,

$$\sum_{i=1}^{n} w_i = 1$$

$$\mu^T w = r_j$$

- There are n assets in the portfolio. Long and short positions are both available
- $w \in \mathbb{R}^n$, is each assets' allocation in the portfolio
- $\mu \in \mathbb{R}^n$, is each asset's historical return
- $\Sigma \in \mathbb{R}^{n \times n}$, is the covariance matrix
- $r_j \in \mathbb{R}^+$, is the designated portfolio return
- $\gamma \in \mathbb{R}^+$, is the penalty coefficient as 'small weights penalty'

Black-Litterman Allocation

$$\hat{\mu} = [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}[(\tau\Sigma)^{-1}\Pi + P^T\Omega^{-1}Q]$$

$$\hat{\Sigma} = \Sigma + [(\tau\Sigma)^{-1} + P^T\Omega^{-1}P]^{-1}$$

$$w = (\delta\Sigma)^{-1}\hat{\mu}$$

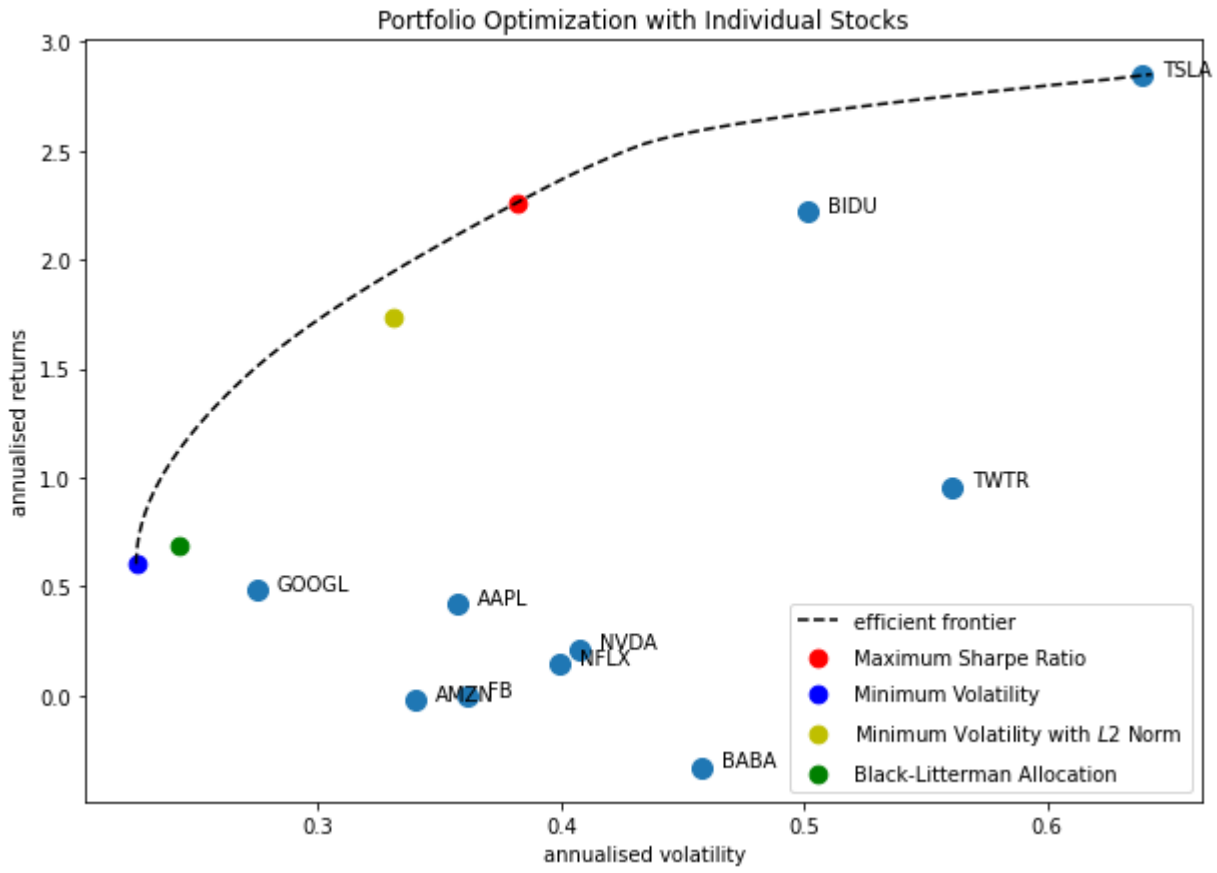
- $\hat{\mu} \in \mathbb{R}^n$ is the adjusted expected return
- Q is vector of views
- P maps views to the universe of assets. Essentially, it tells the model which view corresponds to which asset(s)
- Ω is uncertainty matrix of views
- $\Pi \in \mathbb{R}^n$ is the prior of assets return
- $\Sigma \in \mathbb{R}^{n \times n}$, is the covariance matrix
- τ is a scalar tuning constant

Daily Data From Sep-11-2020 To Jan-08-2021

Date	TWTR	TSLA	AAPL	FB	GOOGL	BABA	NFLX	AMZN	BIDU	NVDA
2020-09-10	-0.015420	0.013815	-0.032646	-0.020568	-0.013689	-0.020502	-0.039025	-0.028605	-0.002010	-0.031715
2020-09-11	-0.000513	0.003716	-0.013129	-0.005521	-0.006743	0.015175	0.002829	-0.018547	0.012252	-0.011960
2020-09-14	-0.004110	0.125832	0.030000	-0.001725	-0.004572	0.009168	-0.011970	-0.004252	0.010446	0.058182
2020-09-15	0.008254	0.071827	0.001560	0.023558	0.017424	0.014082	0.041427	0.017132	0.021415	0.009225
2020-09-16	0.013047	-0.017787	-0.029514	-0.032670	-0.015002	0.000648	-0.024456	-0.024723	0.012370	-0.036679
...	...	...	...	...	...	...	...	...	...	...
2021-01-04	0.007017	0.034152	-0.024719	-0.015449	-0.015126	-0.020968	-0.033048	-0.021585	0.002590	0.004481
2021-01-05	-0.011920	0.007317	0.012364	0.007548	0.008064	0.055080	-0.003940	0.010004	-0.012915	0.022210
2021-01-06	-0.011507	0.028390	-0.033662	-0.028269	-0.009868	-0.053203	-0.038998	-0.024897	-0.046869	-0.058953
2021-01-07	-0.017461	0.079447	0.034123	0.020622	0.029869	-0.003119	0.016784	0.007577	0.019219	0.057830
2021-01-08	-0.016243	0.078403	0.008631	-0.004354	0.013239	0.040943	0.002967	0.006496	0.155659	-0.005040

Test On Real Data

```
--------------------------------------------------------------------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 2.26
Annualised Volatility: 0.38


            TWTR   TSLA  AAPL   FB  GOOGL  BABA  NFLX  AMZN   BIDU  NVDA
allocation   9.8  37.49   0.0  0.0   4.18   0.0   0.0   0.0  48.53   0.0
--------------------------------------------------------------------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.49
Annualised Volatility: 0.24


            TWTR  TSLA  AAPL   FB  GOOGL   BABA   NFLX  AMZN   BIDU  NVDA
allocation  9.07  1.44   0.0  0.0  50.02  11.62  16.88   0.0  10.97   0.0
--------------------------------------------------------------------------------
Minimum Volatility Portfolio With L2 Norm Allocation

Annualised Return: 1.49
Annualised Volatility: 0.3


            TWTR   TSLA  AAPL   FB  GOOGL  BABA  NFLX  AMZN   BIDU  NVDA
allocation  3.93  27.37  6.56  1.8  17.02  2.78  3.18  6.15  24.88  6.32
--------------------------------------------------------------------------------
Black-Litterman Allocation

Annualised Return: 0.49
Annualised Volatility: 0.24


            TWTR  TSLA  AAPL    FB  GOOGL  BABA  NFLX  AMZN  BIDU  NVDA
allocation  12.3  0.15   3.6  6.48  44.71  10.0  5.49   4.4  6.76  6.12
```



Zipline Backtesting

The next question we want to answer is how effective the optimization do in real data. The current pipeline is:

1. Download data from yahoo finance via Python's yahoo finance API. Process data and prepare them for zipline
2. Set up the system of Zipline (virtual environment, dependencies, ingest data, etc.)
3. Code up trading logics, handling portfolio on-the-fly.
4. Test multiple strategies on real data

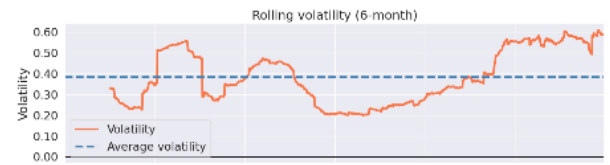
Zipline Backtesting

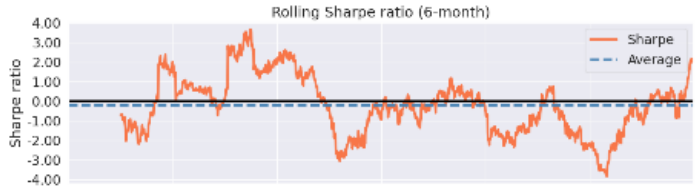
Zipline

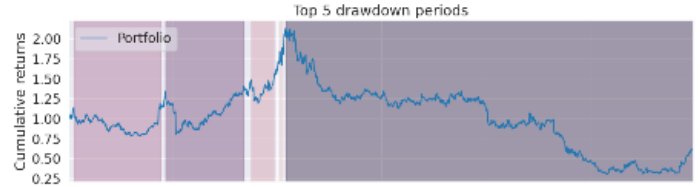
gitter join chat | pypi v1.4.1 | python 2.7 | 3.5 | 3.6 | build passing | build passing | coverage 88%

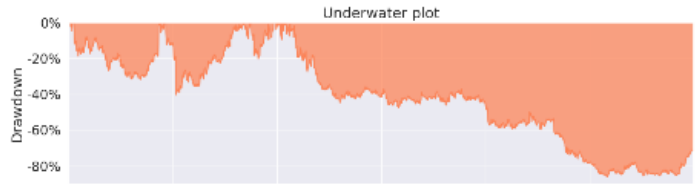
Zipline is a Pythonic algorithmic trading library. It is an event-driven system for backtesting. Zipline is currently used in production as the backtesting and live-trading engine powering Quantopian – a free, community-centered, hosted platform for building and executing trading strategies. Quantopian also offers a fully managed service for professionals that includes Zipline, Alphalens, Pyfolio, FactSet data, and more.

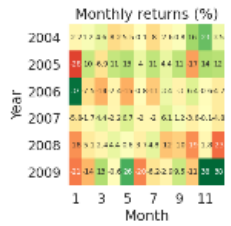
```
zipline run -f main.py --start 2016-1-1 --end 2018-1-1 --capital-base 1000000
–o
 test_zipline.pickle --no-benchmark
```

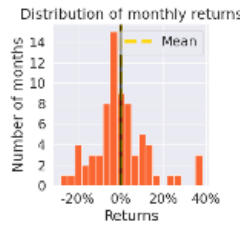


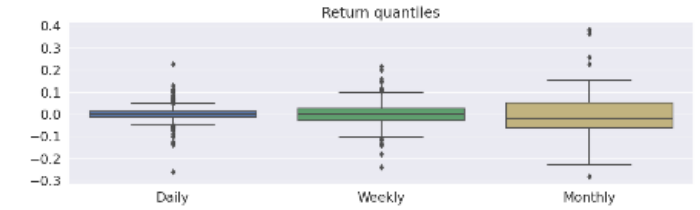












Directions

- We will implement more portfolio optimization strategies under the framework
- Use more advanced techniques, like RNN, LSTM or GRU in time series data for better prediction
- Add more asset classes in to Zipline, including but not limited to equities, fixed income, alternative assets
- Work on an interactive dashboard (Python Dash) to present to the business
- We would conduct frequent code review and agile development to move this project forward