SIAM FM21 Student Programming Challenge[2]

Sponsored by MathWorks

9-10:30am EST, June 3rd
Programming Challenge Committee:
Chair: Matthew Dixon (Illinois Tech)
Stuart Kozola (MathWorks)
Philippe Casgrain (ETH Zurich)

[2] See tinyurl.com/w2ryj8hd for further details.

Overview

- SIAM FM21 hosted the first student programming challenge which is sponsored by MathWorks
- $>$100 students partook in a two month competition to optimize a portfolio under transaction costs and market impact
- We have 7 finalist teams[3] presenting their solutions 9-10:30am EST, June 3rd.
- The four winning teams will be announced after the lightning talks and will receive cash prizes.
- Please submit your questions to the teams in the chat area.

[3] Each team consists of either 2 or 3 student members.

Definitions

- Consider the problem of optimizing a portfolio in $d > 0$ exchange traded stocks over each time period $t = 0, 1, \ldots, T - 1$.
- At each period, the proportional allocation of capital to each stock is represented by the weights

$$w_t \in \Delta^d := \{x \in \mathbb{R}^d : x^i \geq 0 \text{ and } \sum_i x^i = 1\},$$

where each w_t^i represents the proportion of the total capital allocated to stock i at time period t[4]

[4]Note for avoidance of doubt, that the weights are defined in terms of the position size (i.e. number of assets held), u_t^i, as $w_t^i = u_t^i S_t^i / P_t^i$, where the portfolio value $P_t^i = \sum_i^d u_t^i S_t^i$.

Definitions

- Given a sequence of chosen weights $w_{0:T-1} = (w_t)_{t=0}^{T-1}$ historical stock prices $s_{0:T} = (s_t)_{t=0}^{T}$, where for each t, $s_t = (s_t^i)_{i=1}^{d}$ denotes the vector of prices, the total return of the portfolio over the T periods is

$$R_T(w_{0:T-1}) = \prod_{t=0}^{T-1} \left(1 + \sum_{i=1}^{d} (w_t^i \, r_t^i - \eta \, |\Delta u_{t-1}^i|) \right)^+ ,$$

  where $(\,\cdot\,)^+ = \max\{\cdot, 0\}$, $r_t^i = \frac{s_{t+1}^i - s_t^i}{s_t^i}$ and $\Delta u_t^i = u_{t+1}^i - u_t^i = w_{t+1}^i P_{t+1}^i / S_{t+1}^i - w_t^i P_t^i / S_t^i$ with the convention that $\Delta u_{-1}^i = 0$.
- There are two terms for each period t:
  1. The standard definition of the portfolio return in period t
  2. A transaction cost parameter that the portfolio manager must pay each time they rebalance (i.e. change) their portfolio positions, where $\eta > 0$ controls the scale of this cost.

Problem Statement

- Construct a trading strategy which for any fixed $\lambda > 0$ (Risk aversion parameter) and $T > 0$, maximizes the mean-variance objective function

$$\mathcal{L}_T^\lambda(u_{0:T}) = \mathbb{E}[R_T(u_{0:T-1})] - \lambda\,\mathbb{V}[R_T(u_{0:T-1})]\,.$$

- At each time t, the portfolio manager may only only use trading strategies which use historical stock price information in order to decide on positions u_t.
- The data generation process is defined by a market simulator with market impact.
- Teams are judged based on out-of-sample performance of their strategies.[5]

[5] Under an unknown parameterization of the market simulator determined by the committee.

Market Simulator

- Stock prices are assumed to randomly evolve over time and are dependent on how the portfolio is rebalanced.
- Denoting $S_t = (\log s_t^i)_{i=1}^d$, the increments of S_t satisfy the relation

$$S_{t+1} - S_t = \mu + \kappa\,(\Delta u_{t-1}) + M\,\rho_t \ ,$$

  where
  - $\mu \in \mathbb{R}^d$ is an unknown drift vector.
  - $\kappa : \mathbb{R}^d \to \mathbb{R}^d$ is a market impact function which depends on the change in positions, Δu_t, which we define according to

$$\kappa(x_i) = \left(c_i \operatorname{sign}(x_i)|x_i|^{\frac{1}{2}}\right)_{i=1}^d \ ,$$

    for unknown constants $c_i > 0$, and where $\operatorname{sign} : \mathbb{R} \to \{-1, 0, 1\}$ is the sign of a number.
  - $M \in \mathbb{R}^{d\times d}$ is an unknown low-rank matrix
  - $\rho_t = (\rho_t^i)_{i=1}^d$ is a vector of independent and identically distributed random variables with unknown density p satisfying $\mathbb{E}[\rho_t^i] = 0$ and $\mathbb{V}[\rho_t^i] = 1$.

Schedule

Time	Team	Members		
9-9:10	Opening Remarks from Agostino Capponi & the organizers			
9:10-9:15	LSE	Chris Chia	Sandra Ng	
9:15-9:20	NCU	Ning Yen	Min-Syue Chang	Chung-Yu Shih
9:20-9:25	UC Boulder	L. Minah Yang	Danny Kurban	
9:25-9:30	Imperial-Exeter-Oxford	Ben Batten	Henry Elsom	Tom Walshe
9:30-9:40	Q&A			
9:40-9:45	Giessen-KIT	Lukas Gröber	Levin Kiefer	Michael Zheng
9:45-9:50	Sheffield	Georgios Moulantzikos	Vinh Vu	
9:50-9:55	KCL	Haochen Li	Yan Wu	Chunli Liu
9:55-10:10	Q&A			
10:10-10:30	Award ceremony			

Acknowledgements

A special thanks to the following:

- MathWorks for their generous support of the competition
- Eva Donnelly (SIAM) & Kristin O'Neill (SIAM) for assistance with organizing the event
- All the student teams that participated and to the faculty that helped promote the event including Blanka Horvath, Antoine Jacquier, Linda Krauss, Dimitrios Roxanas, Kees Oosterlee.

Mean-Semivariance Optimisation

Sandra Ng [1]   Chris Chia [1]

[1] The London School of Economics and Political Science

May 30, 2021

Methodology: Motivation

$$\mathcal{L}_T^{\lambda} = \mathbb{E}[R_T(w_{0:T-1})] - \lambda \mathbb{V}[R_T(w_{0:T-1})] \quad (1)$$

- **Mean-variance optimisation:** minimising variance directly symmetrically penalizes both downside and **upside** volatilty
- Potentially not desirable as higher upside volatility may be associated with higher mean returns
- Instead, minimize **semivariance**: the variance of negative returns
- Markowitz, Starer, Fram, and Gerber, 2020

Methodology: Implementation

Problem formulation:

Minimise

$$\boldsymbol{n}^T\boldsymbol{n} - \frac{1}{2\lambda}\boldsymbol{\mu}^T\boldsymbol{w} \text{ , subject to } 1^T\boldsymbol{w} = 1 \text{ , } \frac{1}{\sqrt{T}}\boldsymbol{R}\boldsymbol{w} = \boldsymbol{p} - \boldsymbol{n} \text{ , } \boldsymbol{p} \geq 0 \text{ , and } \boldsymbol{n} \geq 0. \tag{2}$$

- Formulate in quadratic programming form; solve using convex optimisation methods in MATLAB
- First trade at the end of $T_w = 100$ periods, subsequently rebalance every $f = 100$ periods

Results: Comparison with alternative strategies

- For fixed $T = 500$ periods, risk aversion $\lambda = 0.1$

	Mean RT	SD RT	Utility
Semivariance	**0.0261**	**0.0076**	**0.0260**
Equal-weighted	0.0116	0.0082	0.0116
Price-weighted	0.0115	0.0080	0.0115
Mean Variance	0.0215	0.0656	0.0211
Ledoit-Wolf	0.0178	0.0670	0.0173
CDaR	0.0179	0.0671	0.0175
cVaR	0.0123	0.0054	0.0123
MAD	0.0107	0.0053	0.0107
Inverse Volatility Weighted	0.0112	0.0071	0.0112
Mean Correlation	0.0116	0.0084	0.0116
Hierarchial	0.0108	0.0071	0.0108

Table: Evaluations of Utility over numerous strategies

Results

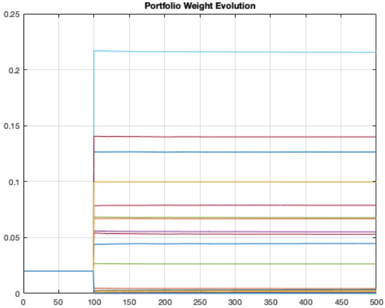


Figure: Portfolio Weight Evolution

Empirical Observations

- Introduces sparse weights
- Convergence to a stable solution in the case of i.i.d. log-returns?
- For $N = 500$ varying small transaction costs and small market impact does not affect result much

Next Steps

- Dynamic position targeting instead of fixed frequency

Trading Strategy with Moving Average and Mean Variance Optimization

SIAM FM21 Programming Challenge
Sponsored by MathWorks

Ning Yen, Min-Syue Chang, Chung-Yu Shih

National Central University, Taiwan

2021/6/3

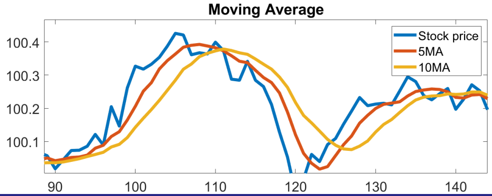
Step 1. **Data collection**
Step 2. **Stocks selection: moving average**

We build the initial without trading (all the weight is 0). The initial length T_i must satisfy the following conditions.

$$T_i = \max\{10 \ \ (\text{for Step 2.}),\ 2 \cdot d \ \ (\text{for Step 3.})\}$$

We select the stock for each time steps by using the long moving average (10MA) and the short moving average (5MA).

$$\text{5MA} \ \leq \ \text{10MA} \Rightarrow \ \text{Bearish. Choose weight as 0}$$



Step 3. **Build weight: mean variance optimization**

We use `Portfolio` in Matlab financial toolbox. This function are base on Markowitz's mean variance optimization (MVO) and expectation conditional maximization (ECM). Given the risk aversion parameter λ, we maximize the return to find the weight w

$$\max_{w} E(R_p), \quad \text{subject to} \quad \sigma_p^2 = \lambda.$$

Where E be the function of expected return

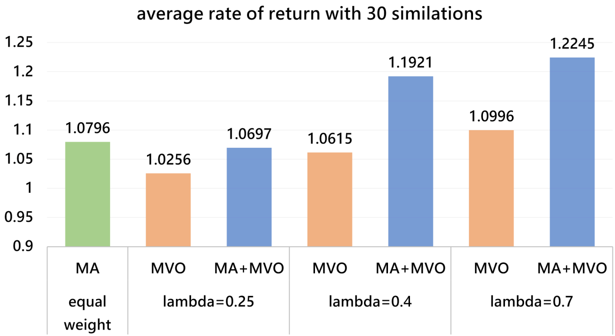
$$E(R_p) = E(w^T r).$$

And using the covariance matrix of returns Σ to estimate the risk, we have

$$\sigma_p^2 = w^T \Sigma w.$$

Results

Set the number of stocks $d = 20$, time grids $T = 4000$. In different risk aversion parameter λ:



Conclusions

- MA is an easy and useful method.
- MVO can take higher risk to get more returns.
- With higher λ, with MVO method can have more return.
- With same λ, MA $+$ MVO have more returns then MVO only.

Picking Winners to Optimize Portfolios

L. Minah Yang
Applied Mathematics
lucia.yang@colorado.edu
https://yangminah.github.io

University of Colorado Boulder

Danny Kurban
Economics
danny.kurban@colorado.edu
https://dannykurban.com

Motivation

https://yangminah.github.io
https://dannykurban.com

- We assume that companies can be characterized as either winners or losers; winners have an upward trend.
  - Winners deliver high return and keeping losers minimizes risk.
- DeMiguel *et al.* (2009, Review of Financial Studies) showed that an equal weight strategy actually outperforms more sophisticated mean-variance optimization strategies for < 6000 periods.
- Predicting future stock prices is difficult and often introduces large estimation errors.

Method Description

Who are the winners?

We tried the following strategies:

- Priciest stocks are the winners. ✔
- Stocks with the highest 1-period returns are the winners. ✘

Suppose there are n winners in a d-sized portfolio.

- Any winner should have more weight than any loser.
  - $\alpha/n > (1-\alpha)/(d-n)$

How to configure the winner basket?

Size:

- Winners have α, losers have $(1-\alpha)$.
  - $\alpha < 1 \rightarrow$ losers get >0. ✔
  - $\alpha = 1 \rightarrow$ losers get 0. ✘

Distribution:

- How is α divided amongst the winners?
  - equal weights : α/n. ✔
  - weighted by price. ✘

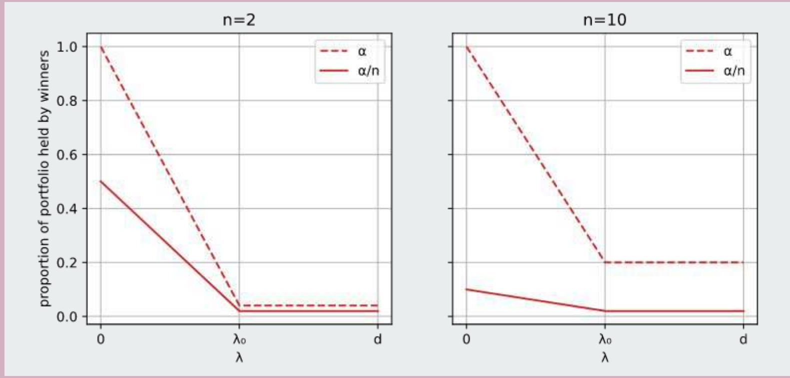
Model

https://yangminah.github.io
https://dannykurban.com

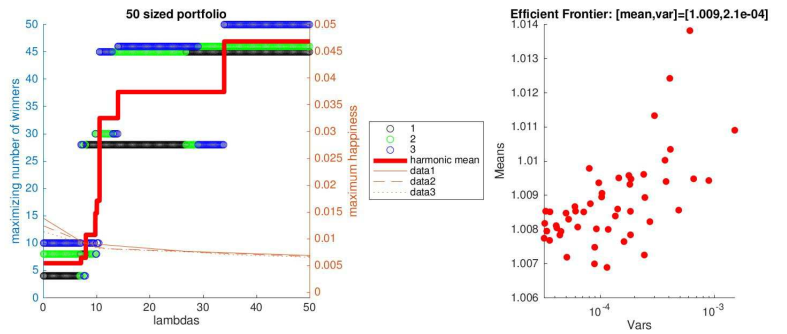
- High λ is risk-averse, and low λ is risky.
- Low α is risk-averse, and high α is risky.
- For $\lambda \geq \lambda_0$, revert to the equal weight strategy.

Given a *fixed* number of winners n, we set the winner basket proportion α to:

$$\alpha = \alpha(\lambda; n, d, \lambda_0) = \begin{cases} 1 + \frac{-1+n/d}{\lambda_0}\lambda, & 0 < \lambda < \lambda_0 \\ n/d, & \lambda \geq \lambda_0. \end{cases}$$



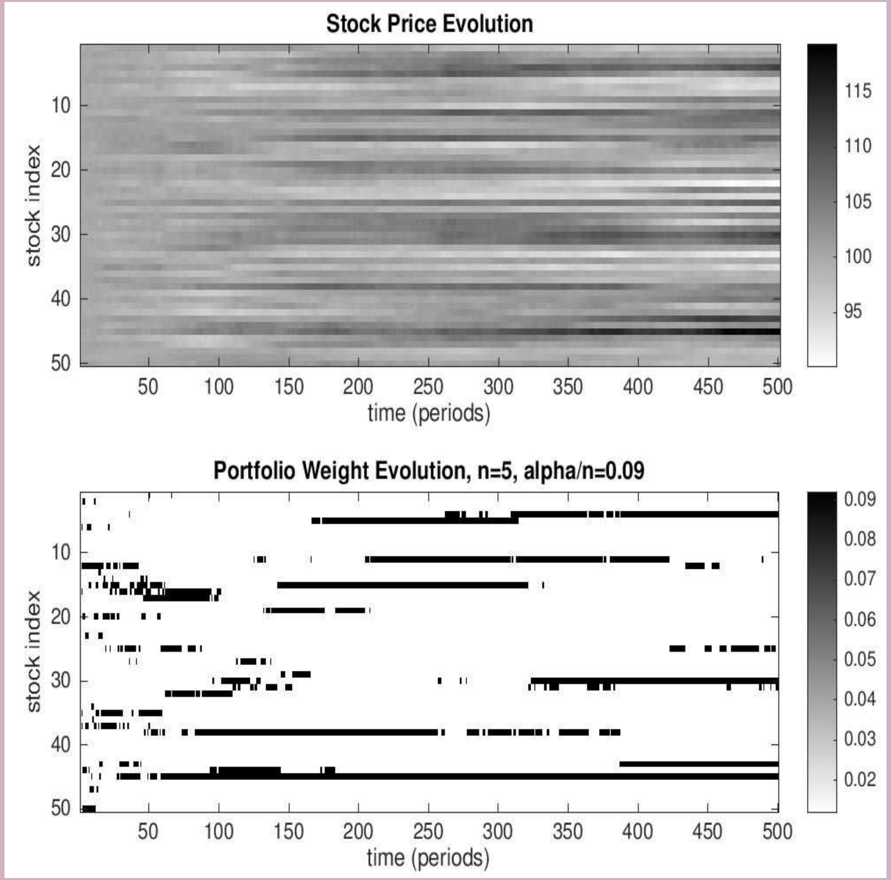
- High n is risk-averse, and low n is risky.
- Data-driven approach: Compute the optimal n with respect to the objective function for various combinations of d (portfolio size) and λ.



$$n = n(\lambda; k, \lambda_0) = 1 + \frac{d-1}{1+\exp(-k(\lambda - \lambda_0))}$$

- k: steepness of logistic function.
- Centered around λ_0.

Results: Example with λ=15



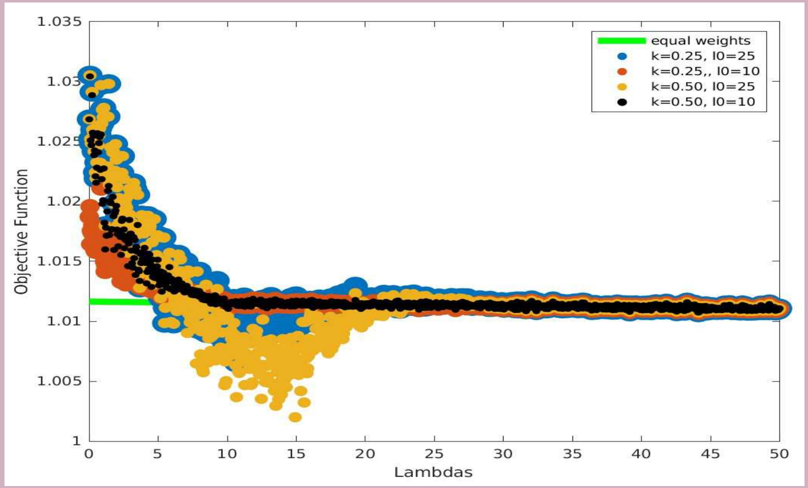
We configured our model with k=0.25 and λ_0=25.

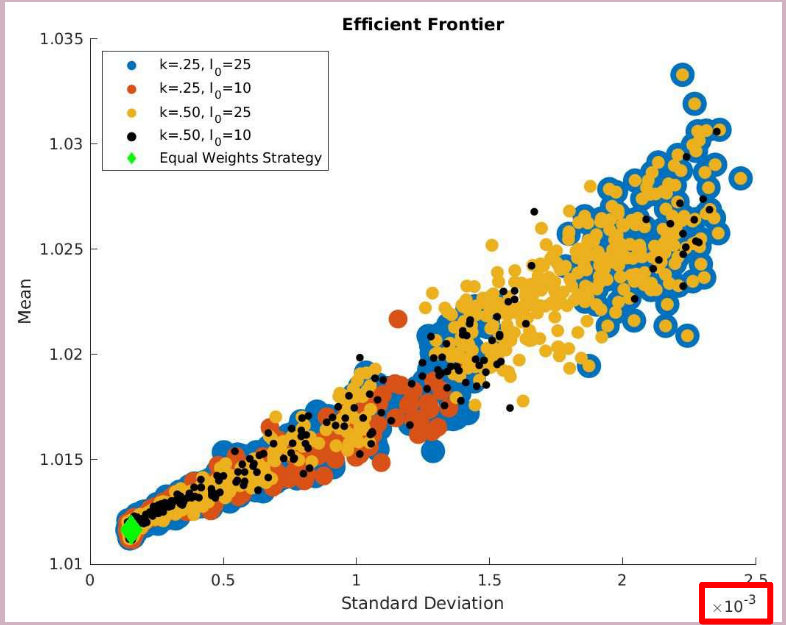
- At λ=15,
  - number of winners: n=5
  - number of losers: d-n=45.
  - proportion of winner basket: α=0.45
  - proportion of loser basket: 1-α=0.55
- High turnover initially.
- Lower turnover after ~150 periods.

https://yangminah.github.io
https://dannykurban.com

Results: $k \in \{0.25, 0.50\}, \quad \lambda_0 \in \{10, 25\}$

- All configurations
  - converge to the equal weights strategy (EWS) at high λ.
  - outperform the EWS at low λ.
- k=0.25,λ_0=25 performs best for wide range of λ.





https://yangminah.github.io
https://dannykurban.com

- All configurations yield higher expected return and variance in comparison to the EWS.
- On average, the gain in expected return more than makes up for the increase in risk.

Portfolio Optimisation with Dynamic Risk Aversion Tuning

Ben Batten[1], Henry Elsom[2], and Tom Walshe[3]

Imperial College London[1]
University of Exeter[2]
University of Oxford[3]

Method - Solution overview

- Strategies are **assessed** based on an **inter-simulation performance** function.

$$\text{Performance} = \text{Mean Return} - \lambda(\text{Variance of Return})$$

- We use **quadratic optimisation** on a constrained **Markowitz model**.
- Gamma is **dynamically tuned** according to three **simulation-state dependant metrics**.
- By **contextualising** performance **metrics**, we **dynamically tune** our optimisation **hyperparameters**.

$$\max_w(\text{score}) = \max_w((\Delta p)^T w - \gamma(w^T(\sigma^2)w + w^T \text{cov}' w))$$

Method - Empirically determined risk metrics

- Risk aversion increases with current return to **preserve profit**.

$$\log_{10}(\gamma_{CR}) = 0.24CR + 0.013$$

- **Monte Carlo** simulation provides an estimate for **expected return**.

$$\gamma_{SR} = 100e^{0.25(CR-SR)+15}$$

- Scale the **user specified** risk aversion parameter, λ.

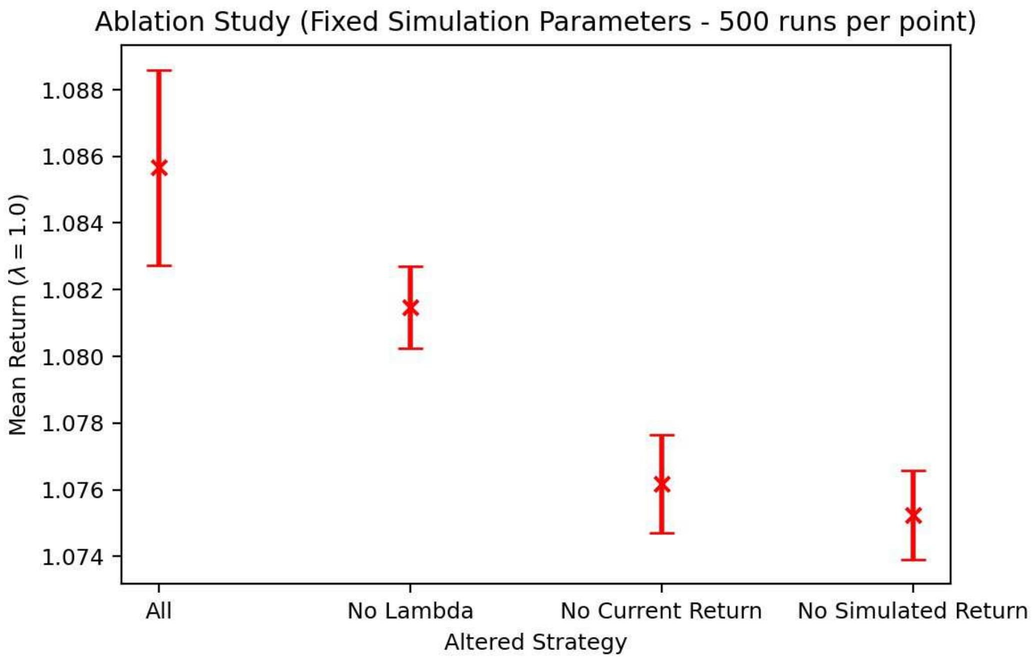
$$\gamma_L = 1.88\lambda^{2.19}$$

$$\gamma = c_1\gamma_L + c_2\gamma_{CR} + c_3\gamma_{SR}$$

CR: Current Return
L: Lambda
SR: Simulated Return

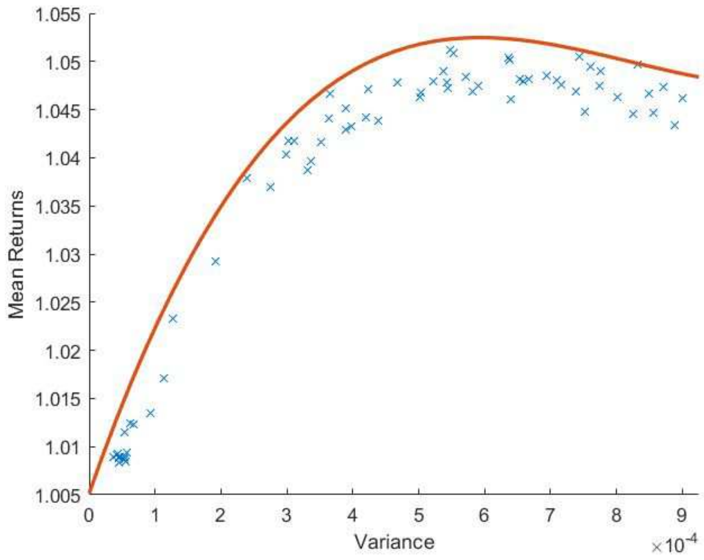
Results - Efficacy of strategy components

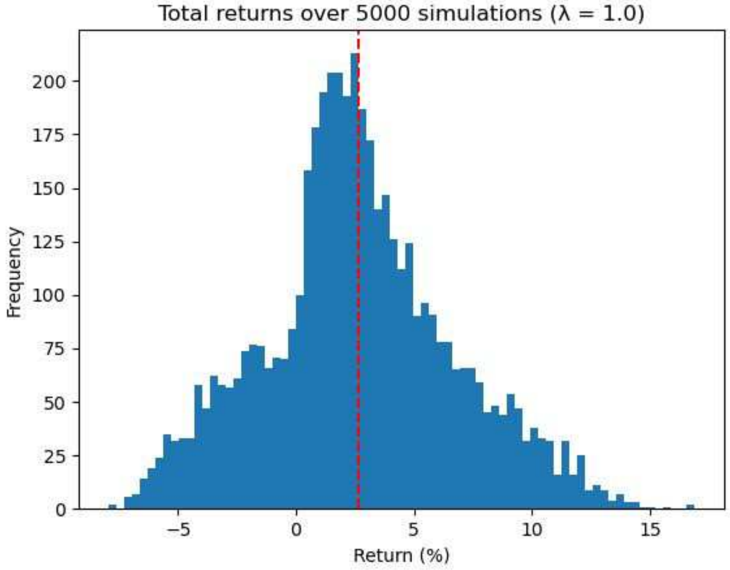
- We perform an **ablation study** on our strategy.
- Test the **validity** of **combining** our three approaches to **dynamic risk aversion**.
- **Combined approach** finds **best balance** for given simulation **state**.



Results - Performance analysis

- Maximum average **return** of **5.2%** (λ = 5.5).
- Average **return** of **2.65%** over 500 steps, randomised parameters (λ = 1.0).





SIAM Student Coding Competition

Flash presentations

Lukas Gröber (Uni Gießen), Levin Kiefer (KIT), Michael Zheng (KIT)

2021-06-03

Basic Approach

- Ideas: Machine learning, Monte Carlo Tree Search, Linear Extrapolation
- Optimizing for best cumulative drift in portfolio worked out the best
- Periods without trading to estimate drift and pairwise covariance
- Initial strategy according to risk aversion $\lambda \rightarrow$ more or less random shares
- Look for "good pairs" $\rightarrow$ no or negative correlation, number of pairs according to λ

Basic Approach

- Higher $\lambda \rightarrow$ longer period with initial strategy
- Heuristics to implement ideas e.g. trading every $\max(\lfloor 50000 \cdot \eta \rfloor, 30)$ periods
- Choice for specific formulas based on testing
- We decided to ignore the reallocation reactions $\kappa \rightarrow$ not enough effects, too difficult of a relation with η
- Reasonable computing time with best performance (compared to our other solutions)

Results

- Machine learning did not converge sufficiently $\rightarrow$ parameters too small (?)
- MCTS took way too much computing power for poor results
- Linear Extrapolation worked surprisingly well
- Conventional estimators with a strategy favoring low correlation and high drift worked the best
- We achieved returns ranging from 0.9263 to 1.1138 (under given parameters)

Results

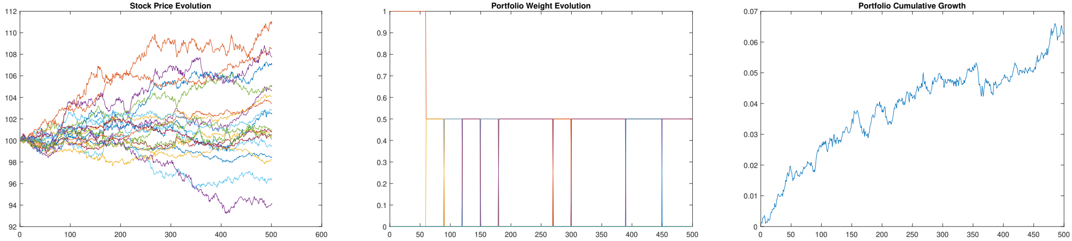
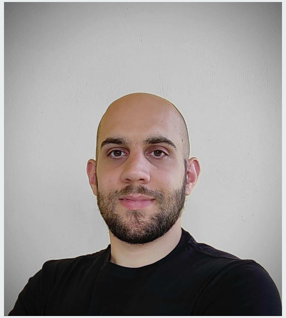


Figure: Example run with default parameters $T = 500, d = 20, \eta = 2 \cdot 10^{-4}, \lambda = 0.25, \ldots$

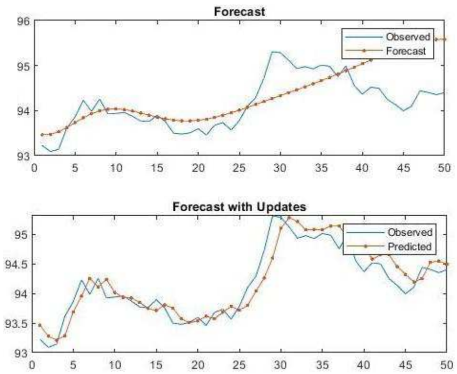
SIAM Financial Competition 2021



Georgios Moulantzikos and Vinh Vu

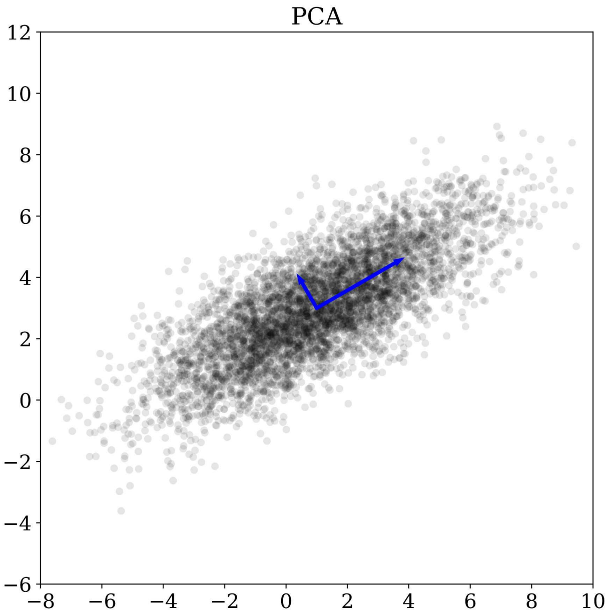
Strategy 1 - LSTM

- **Long short-term memory:** special type of RNN capable of learning long term dependencies. **LSTM** networks are well-suited to making predictions based on time series data.
- **Algorithm:**
    1. Split timeframe into a) first training interval - use equal weights to generate stock prices b) smaller predictions intervals.
    2. When we reach a rebalancing date: train an LSTM network with stock prices up to this date and predict stock prices for the next prediction interval.
    3. Use predicted stock prices to calculate the weights using Markowitz optimization.
- **Comments/Future work:** computational costs, batch vs incremental training, lengths of intervals.



Strategy 2 - PCA

- **Principal component analysis** is an optimal decomposition method to decompose a dataset in the Euclidean space.
  - Filters out noise.
  - Allows trends to be identified.
- **Simple to obtain**
  - Find eigenvalues and eigenvectors of the covariance.
- **Algorithm**
  - Take data from past few timesteps to predict the trend.
  - Weights are allocated proportionally to the most dominant eigenvectors.
- **Risk strategy**
  - Rises the weights to the power of the inverse of the risk parameter.

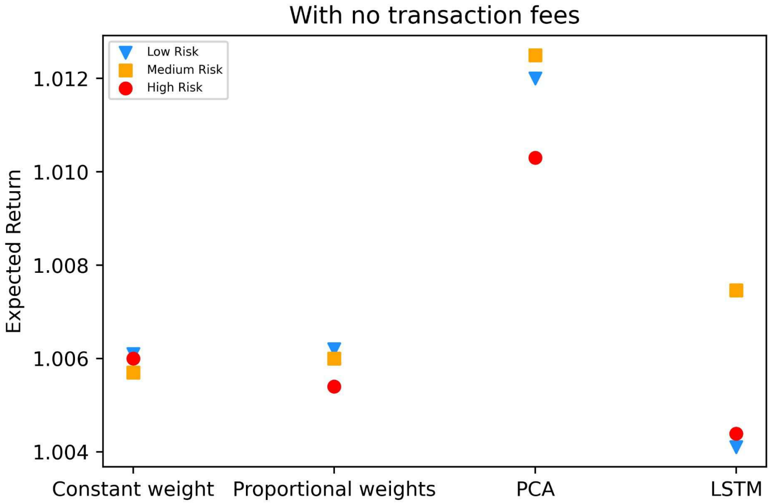


Results Convergence

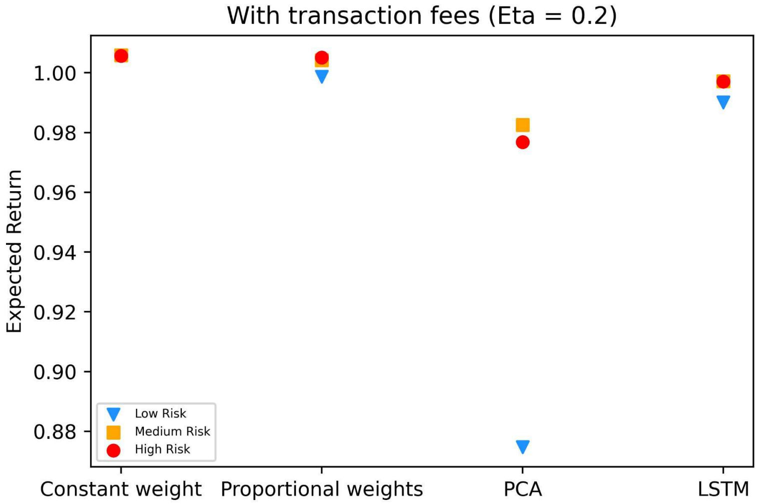
- **Problem** - how do we know if these strategies are effective at predicting stock prices?
  - Need to factor in effects of randomness.
- **Welford's Online algorithm** - We can use this to calculate the expected returns of a strategy.
  - This gives us an expected returns for our strategies after considering the effects of randomness.
  - If the moving average of the second term is less than our convergence criteria of 1e-8, we assume the solution has converged.

$$\overline{x}_{n+1} = \overline{x}_n + \frac{\overline{x}_n - x}{n}$$

Results



With no transaction fees
Low Risk
Medium Risk
High Risk
Expected Return
1.012
1.010
1.008
1.006
1.004
Constant weight
Proportional weights
PCA
LSTM



With transaction fees (Eta = 0.2)
Expected Return
1.00
0.98
0.96
0.94
0.92
0.90
0.88
Low Risk
Medium Risk
High Risk
Constant weight
Proportional weights
PCA
LSTM

Black-Litterman Model & Long Short-Term Memory Network

Haochen Li
King's College London

Chunli Liu
King's College London

Yan Wu
Beijing Normal University

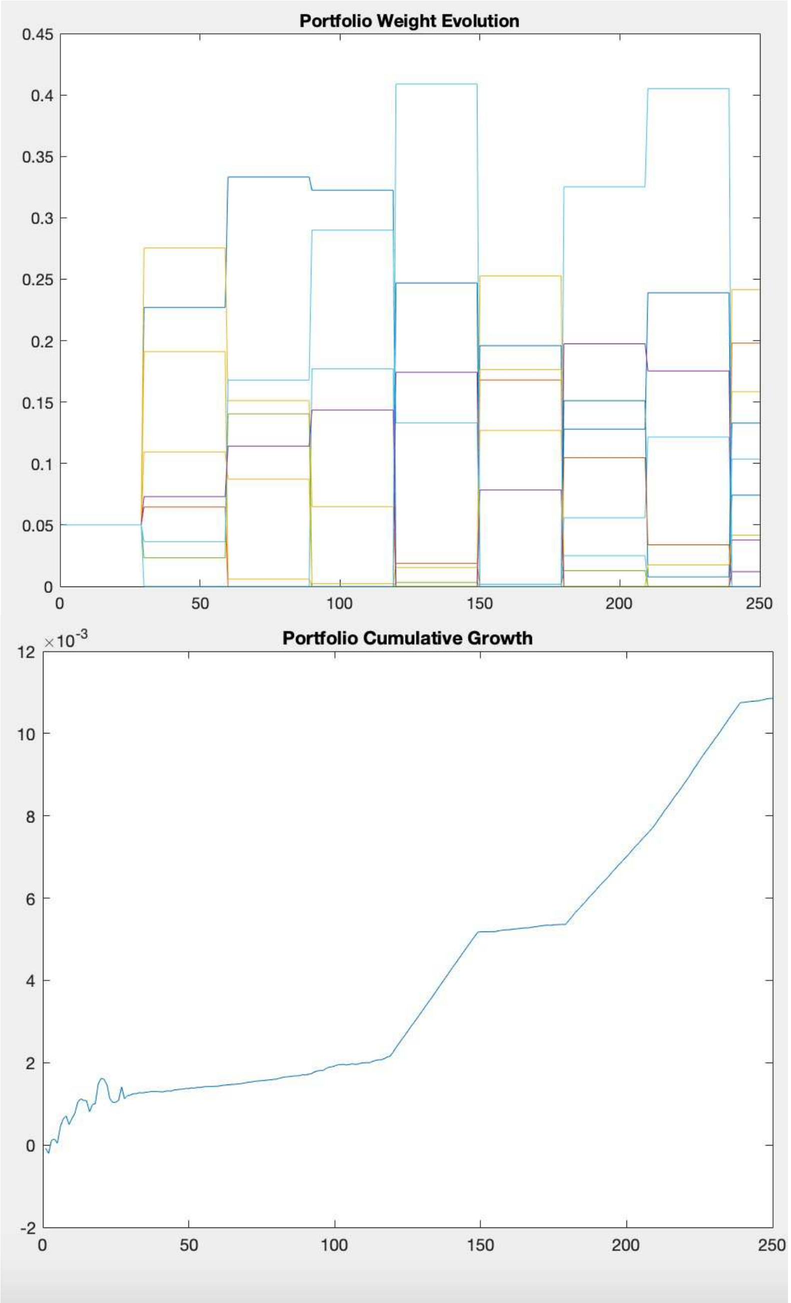
Black-Litterman Model (BL model)

- Portfolio optimization based on Markowitz's Modern Portfolio Theory
- Incorporate subjective views of the investor instead of relying only on historical asset returns
- Views: Weigh highly on the stocks with less volatility and higher expected return predicted by the predictor
- Predictor: moving average regression

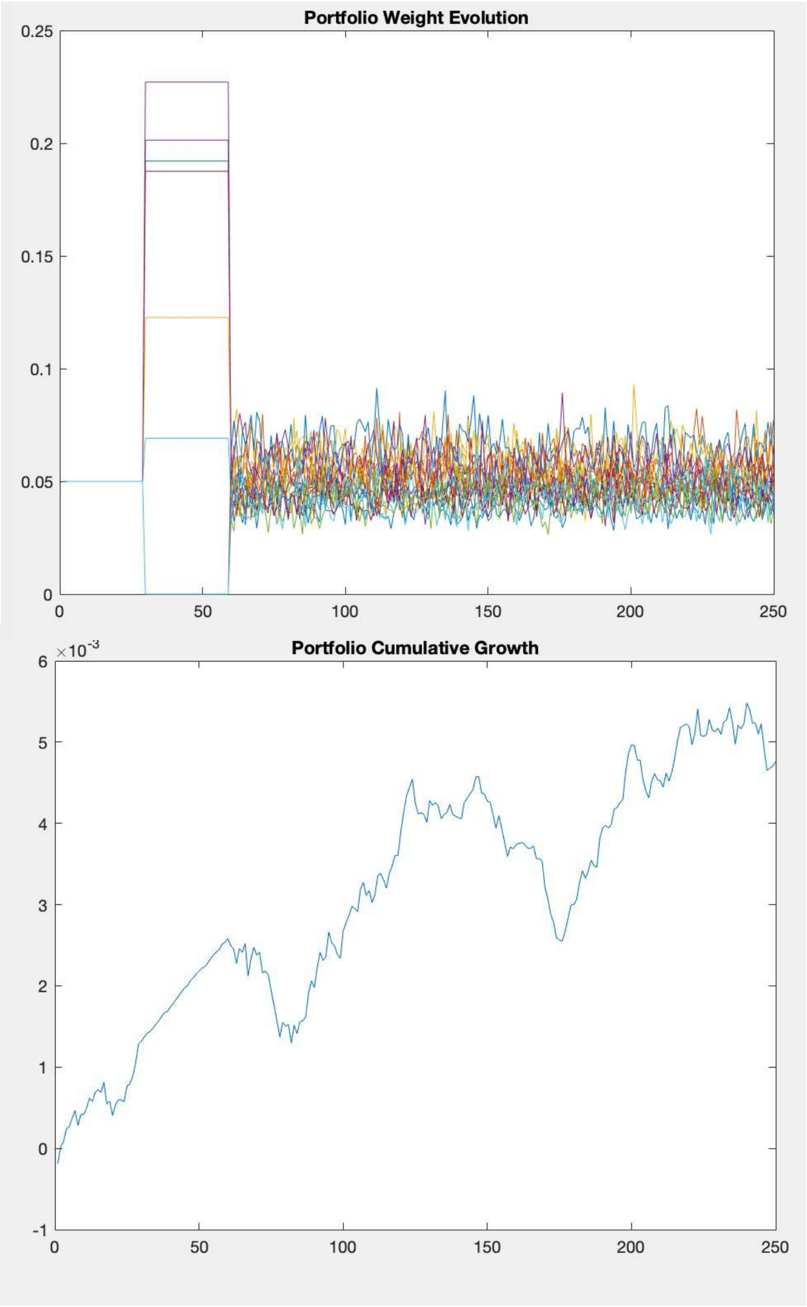
Long Short-Term Memory Network (LSTM)

- A type of Recurrent neural network (RNN)
- Good at learning, processing, and classifying sequential data
- Trained by backpropagation through time
- Weigh more on the historical trend of each asset instead of recent fluctuations

BL model

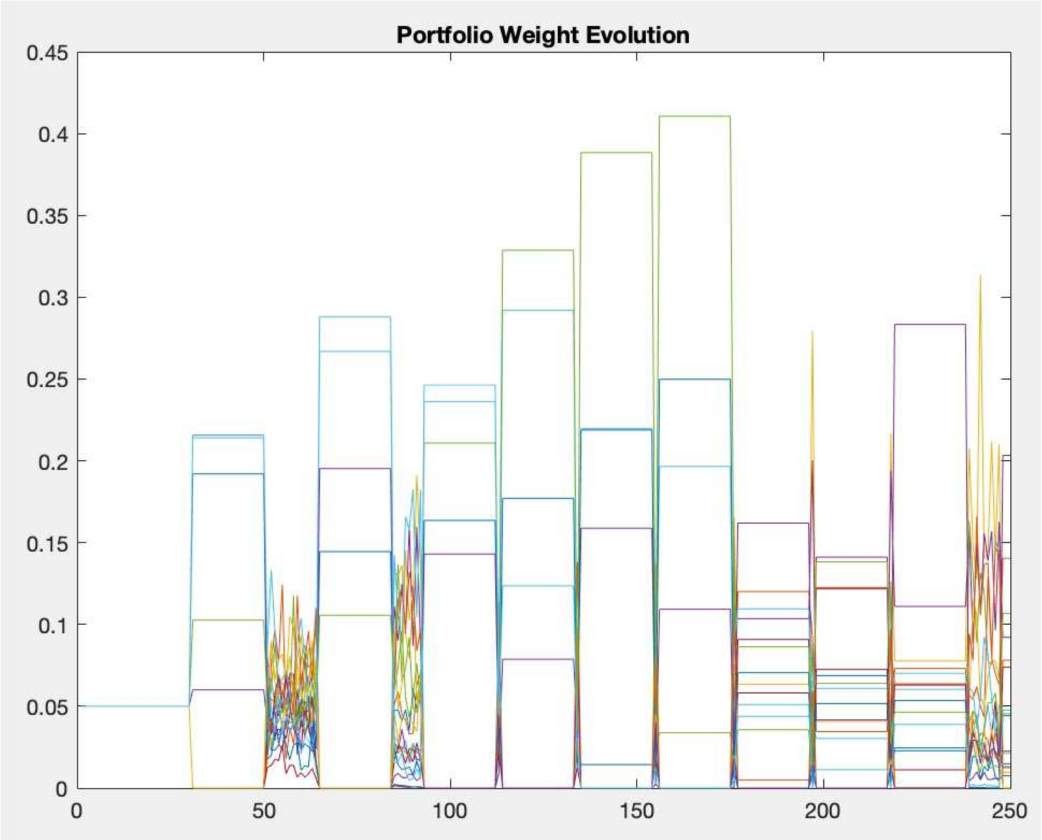


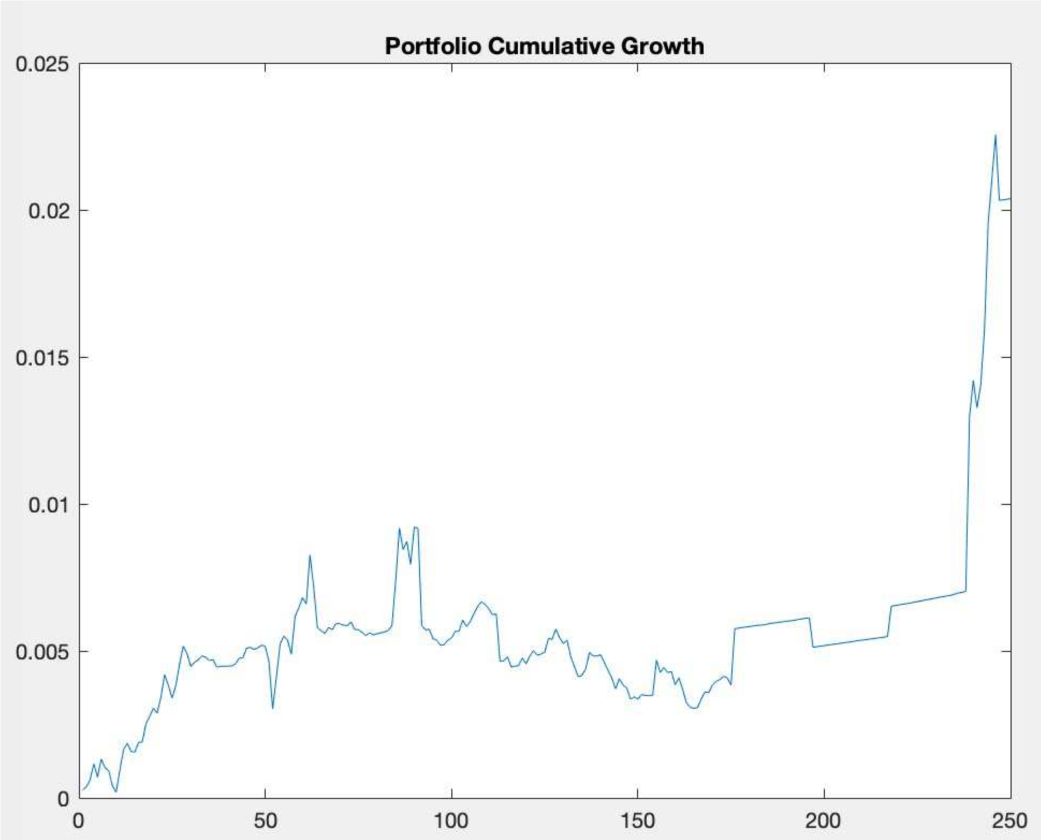
LSTM



Incorporation

- BL model: more stable, lower return, LSTM: more potential, unstable
- Set threshold on the maximum drawdown
- BL model to keep the lower limit, LSTM to fight for the upper limit

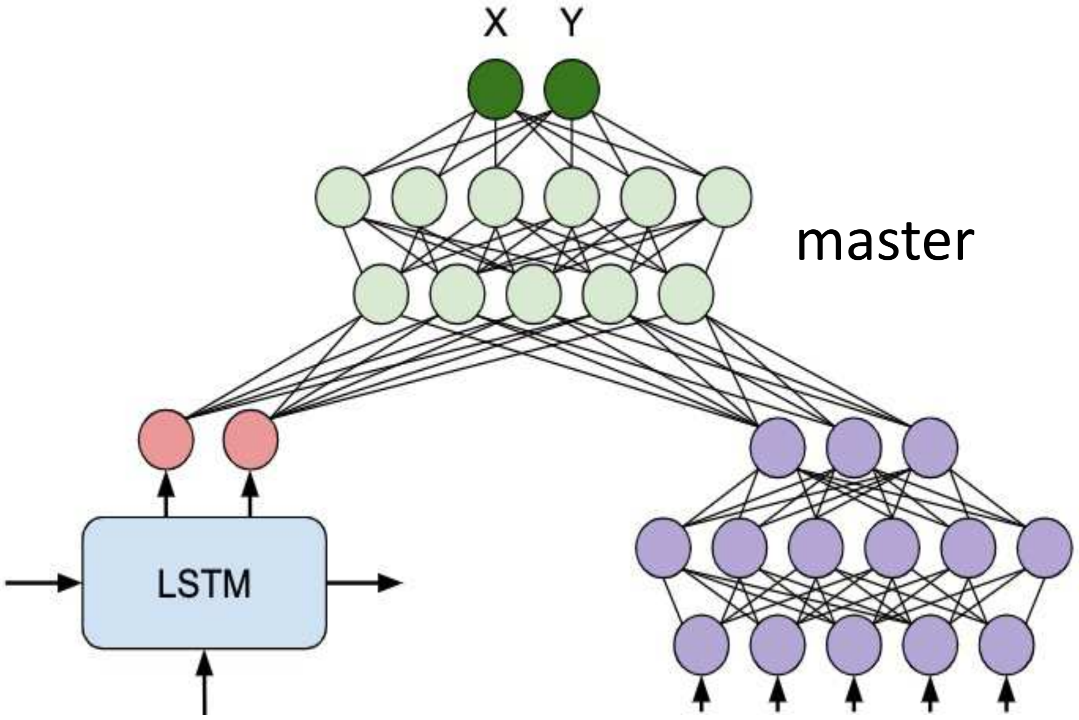




Future works

- Framework of multimodal machine learning
- Master: multi-layer neural network with fully connected layers
- Use LSTM and BL model simultaneously with weights controlled by master

weights of models



master

BL model