Reinforcement Learning in Quanti
Wealth & Investment Management (QWIM) - Proximal Policy Optimization (PPO)

Melissa Atmaca, Sid Bhatia, Kevin Lochbihler, Alvin Radoncic

STEVENS
INSTITUTE OF TECHNOLOGY
1870

1. Introduction

Reinforcement Learning in QWIM

Introduction

- **Revolutionizing Investment Landscape with AI**: This paper underscores the transformative role of advanced AI technologies, particularly Deep Reinforcement Learning (DRL), in reshaping the investment industry, focusing on the innovative application of Proximal Policy Optimization (PPO) in Quantitative Wealth & Investment Management (QWIM).
- **PPO's Dual-Network Approach for Dynamic Strategies**: Our approach leverages PPO's state-of-the-art dual-network architecture, comprising actor and critic models, enabling effective learning and adaptation of investment strategies. This is demonstrated through the agent's ability to dynamically adjust portfolio allocations across various assets, including ETFs and stocks.
- **Achieving Strong Risk-Adjusted Returns**: The research highlights significant improvements in portfolio value with a strong, reproducible Sharpe Ratio, indicating the PPO agent's capability to generate solid risk-adjusted returns. This success is attributed to a sophisticated reward structure based on the Sharpe Ratio, ensuring a balance between profit maximization and risk management.
- **Advancing QWIM through DRL Innovations**: The paper contributes substantially to the AI-driven investment management field, showcasing DRL's potential in crafting intelligent, adaptive, and high-performing investment strategies. It suggests a paradigm shift in investment management, where embracing complexity leads to innovation, resulting in more efficient, transparent, and profitable strategies.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

2. Data Analysi

Reinforcement Learning in QWIM

Data

Methodology

- Conducted Exploratory Data Analysis (EDA) on daily log returns of 8 diverse ETFs spanning equities to corporate bonds, and introduced the analysis of certain financial metrics to help dictate potential investment decisions.
- The time window is intended from Jan 1st, 2021 to December 31st, 2022 with adjusting for differing ETF inception dates.
- ETFs: iShares Russell 1000 Growth (IWF), iShares Russell 1000 Value (IWD), iShares Russell 2000 Growth (IWO), iShares MSCI Emerging Markets (EEM), iShares MSCI Japan (EWJ), iShares 0-5 Year High Yield Corporate Bond (SHYG), iShares MSCI USA Momentum Factor (MTUM).

Ticker	Row #	Mean	Std	Min	25%	50%	75%	Max
IWD	5534	0.000262	0.0125	-0.124	-0.0044	0.000692	0.00589	0.121
IWF	5534	0.00272	0.0131	-0.125	-0.00518	0.000807	0.00636	0.116
IWO	5534	0.000255	0.0159	-0.135	-0.00777	0.001037	0.00889	0.104
EEM	4964	0.000319	0.0178	-0.176	-0.00795	0.000971	0.00908	0.205
EWJ	5534	0.00085	0.0136	-0.110	-0.00681	0.000365	0.00727	0.147
SHYG	2317	0.00012	0.0041	-0.043	-0.00011	0.000212	0.00148	0.053
MTUM	2444	0.00048	0.0124	-0.132	-0.00443	0.001114	0.00633	0.101

Results

- All of the datasets are affirmed to display stationarity upon conducting an Augmented-Dickey Fuller test. This sort of data allows us to train the RL model with more ease.
- The sharpe ratio for each ETF was computed upon extracting the mean and standard deviation from descriptive statistics. As financial jargon, they represent expected returns and risk.
- Overall, none of the ETFs produced sharpe ratios that would be considered remarkable while making an investment decision.

ETF	Expected Return	Risk	Sharpe Ratio
IWD	0.000262	0.0125	0.02085
IWF	0.00272	0.0131	0.02079
IWO	0.000255	0.0159	0.01605
EEM	0.000319	0.0178	0.0179
EWJ	0.00085	0.0136	0.00625
SHYG	0.00012	0.0041	0.02796
MTUM	0.00048	0.0124	0.03862

STEVENS
INSTITUTE OF TECHNOLOGY
1870

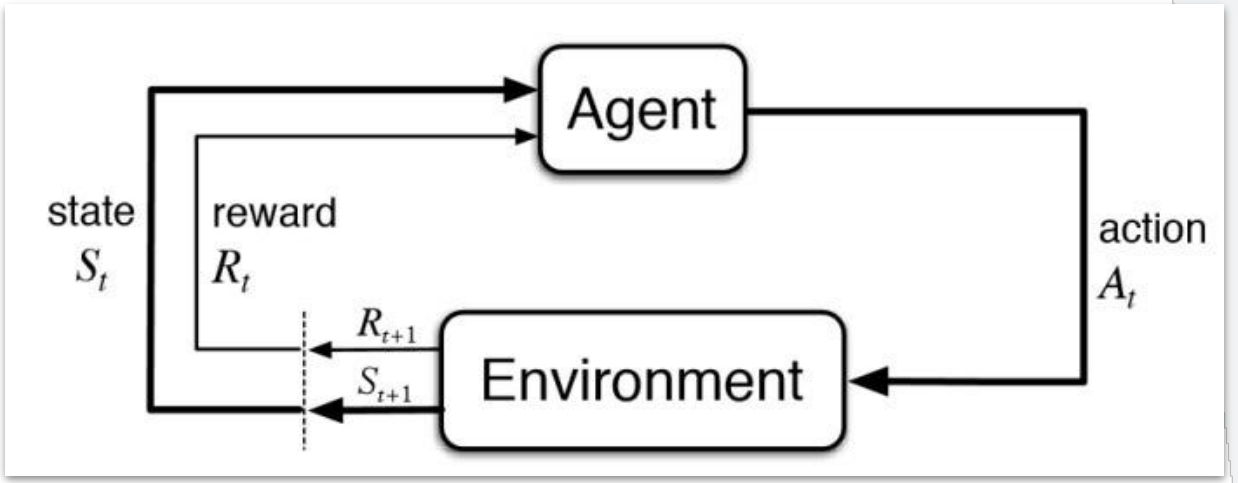
3. Overview of R

Reinforcement Learning in QWIM

Reinforcement Learning

Overview

- Reinforcement Learning (RL) is a branch of machine learning where an agent learns decision-making by interacting with the environment.
  - Agent performs actions, receives rewards or penalties that collectively develop an optimal strategy towards maximizing the reward
- A significant challenge in RL is balancing exploration, where the agent tries new actions to discover potentially better strategies, and exploitation, where it leverages known information to gain rewards.
- RL Approaches:
  - Model-based: Agent has an explicit model of the environment for planning.
  - Model-free: Agent learns directly from experience without an explicit model.
- Value Function
  - Estimates the expected returns from a given state or state-action pair.
  - Crucial in assessing the desirability of states and guiding the agent towards beneficial actions.



RL is an **iterative process**

STEVENS
INSTITUTE OF TECHNOLOGY
1870

4. Overview of PP

Reinforcement Learning in QWIM

Proximal Policy Optimization

Overview

- Proximal Policy Optimization (PPO) is particularly well-suited for tasks involving continuous action spaces, such as portfolio optimization.
- This characteristic makes it an ideal choice for financial applications where the decision space often includes a range of potential investment allocations.
  - A key feature of PPO is its focus on stable policy updates. In the context of financial applications, this stability is vital as abrupt or significant changes in investment strategies can pose substantial financial risks.
- Additionally, PPO's sample efficiency is a notable advantage, especially beneficial in scenarios where data spans over long time windows (Schulman, et al; 2017).
- This efficiency allows for more effective learning and adaptation from historical market data, making PPO a robust choice for developing sophisticated investment strategies in the dynamic and often unpredictable realm of finance.

Proximal Policy Optimization

Model Architecture

- Within the PPO architecture, there are two primary models: the Actor and the Critic. The Actor model suggests actions based on the current policy, denoted as $\pi_\theta(a_t/s_t)$, where θ are the parameters of the model.
  - The Critic model, on the other hand, evaluates the potential value of the state transitions using the value function $V_\mu(s_t)$, with μ representing its parameters. This evaluation helps to gauge the quality of the actions proposed by the Actor.
- The algorithm uses mini-batch samples from the collected experiences to update the models in an incremental fashion. Key to the PPO method is the advantage estimate, $\boldsymbol{\hat{A}_t}$, which informs the Actor model of the relative value of the actions taken compared to a baseline, guiding it towards more beneficial actions.
  - The Critic model utilizes the temporal difference error, δ_t, to refine its value function estimates, which reflects the difference between predicted and actual rewards.
- both the Actor and Critic models are iteratively improved using stochastic gradient descent (SGD). The Actor is updated by the gradient of the clipped objective, $\boldsymbol{\nabla L^{CLIP}(\theta)}$, while the Critic is updated by the gradient of the value loss, $\boldsymbol{\nabla L^{V}(\mu)}$.

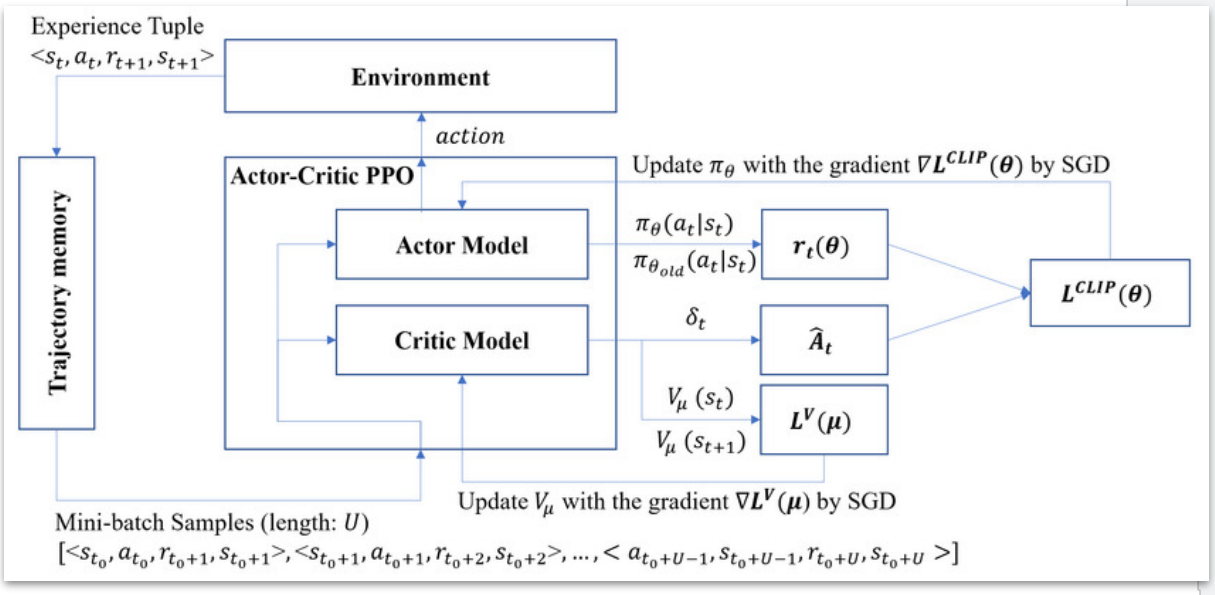


Figure 2 - Actor-Critic Proximal Policy Optimization (Actor-Critic PPO) Process

- A distinctive aspect of PPO is the policy ratio, $\boldsymbol{r_t(\theta)}$, which is used within the PPO clipping objective to moderate policy updates, ensuring that new policies do not deviate excessively from previous ones, thus maintaining training stability.
  - The objective function $\boldsymbol{L^{CLIP}(\theta)}$ employs this ratio, applying a clipping mechanism to the policy gradient objective to prevent disruptive updates to the policy.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

5. Methodology

Reinforcement Learning in QWIM

Methodology

PPO.py

- **Objective**: PPO is a type of reinforcement learning algorithm designed for optimizing decision-making policies. It aims to balance exploration (trying new actions) with exploitation (using known information to maximize reward).
- **Architecture**: The PPO class in *PPO.py* defines separate actor and critic neural networks. The actor network determines the action probabilities, while the critic estimates the value of being in a given state.
  - The **actor network** takes the current state as input and outputs a probability distribution over possible actions. It's trained to select actions that maximize expected rewards.
  - The **critic network** evaluates the value of being in a given state, assisting the actor in policy improvement. It predicts the expected sum of rewards (state value) from the current state.
- **Training**: The PPO algorithm involves calculating policy loss and value loss to update the actor and critic networks. The policy loss encourages the actor to adopt better policies, while the value loss helps the critic in accurate value estimation.

Methodology

PortfolioOptimization.py

- **Objective:** This script uses PPO for optimizing a financial portfolio. The goal is to allocate assets in a way that maximizes the portfolio's expected return for a given level of risk.
- **Data Handling:** Data for various ETFs (Exchange-Traded Funds) is fetched from Yahoo Finance.
  - The data is cleaned and normalized (Z-score normalization) to make it suitable for training the PPO agent.
- **Environment Setup:** A custom environment for the PPO algorithm is defined, including state space (historical price data), action space (asset allocations), and the reward function (based on portfolio returns).
- **Agent Training:** The PPO agent is trained over several episodes, where in each episode, the agent makes decisions (asset allocations), observes rewards (portfolio returns), and updates its policy.
- **Evaluation:** The trained agent is evaluated on a separate test dataset to assess its performance. Metrics like the Sharpe Ratio (measure of risk-adjusted return) are used for evaluation.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

6. Results

Reinforcement Learning in QWIM

Results (Large)

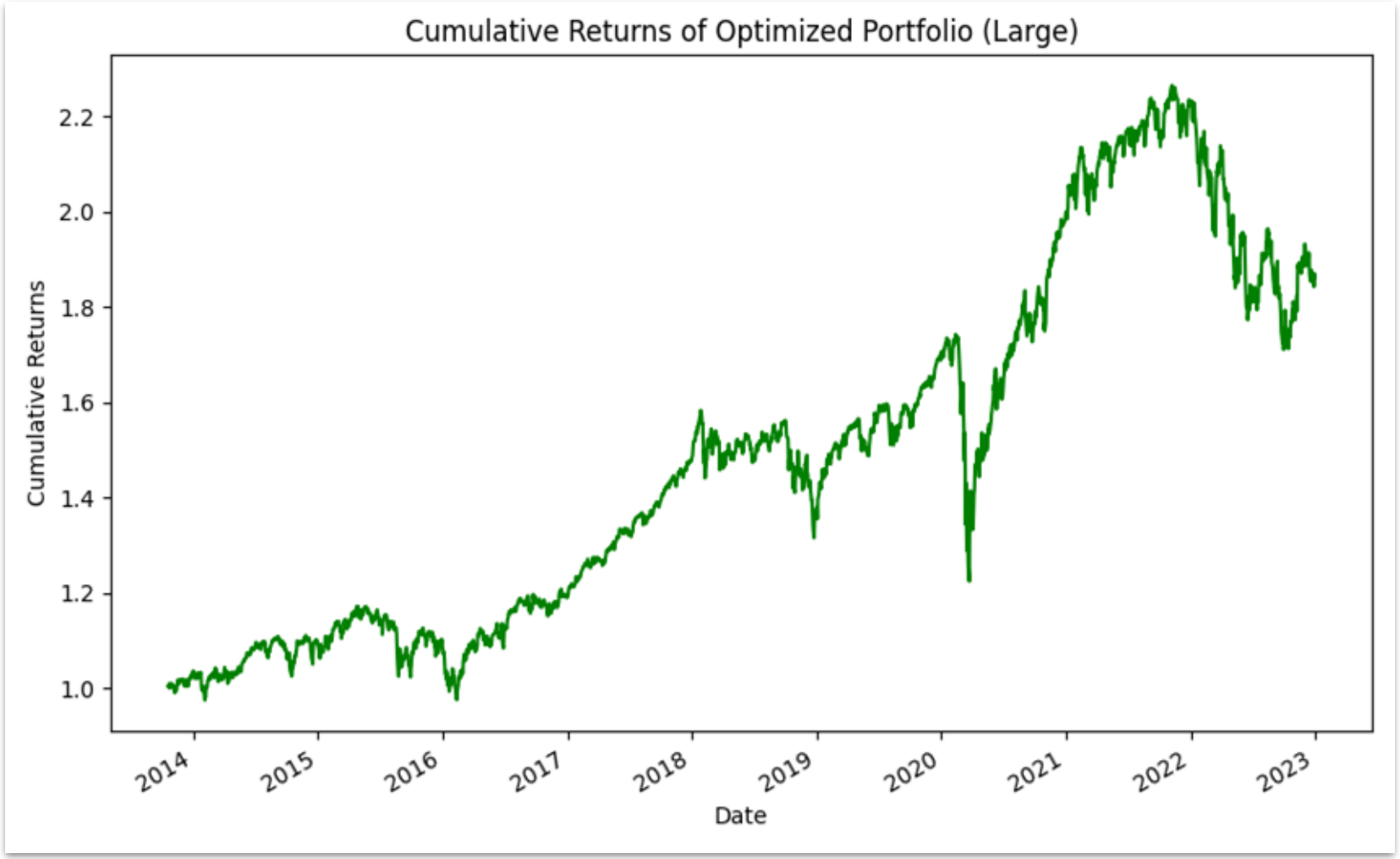


Figure 3 - Cumulative Returns of Optimized Large PPO Portfolio

Parameter	Value
Episodes	100
Actor Learning Rate	0.001
Critic Learning Rate	0.001
Clip Ratio	0.2
Training Interval	10

ETFs Used: **IWF, EEM, SHYG, MTUM, IWD, EWJ**
Average Sharpe Ratio (Train, Test): **1.58, 1.55**

- The portfolio has demonstrated consistent growth, with periods of gains, especially noticeable before the downturn around mid-2020 due to the **Covid**.
- The average Sharpe Ratios for both training and testing periods are relatively **decent** at **1.58** and **1.55**, least performing of all.
- The inclusion of multiple ETFs in the portfolio and the model's optimization strategy, contributed to a diversified investment approach, **reducing unsystematic risk**.

Results (Medium)

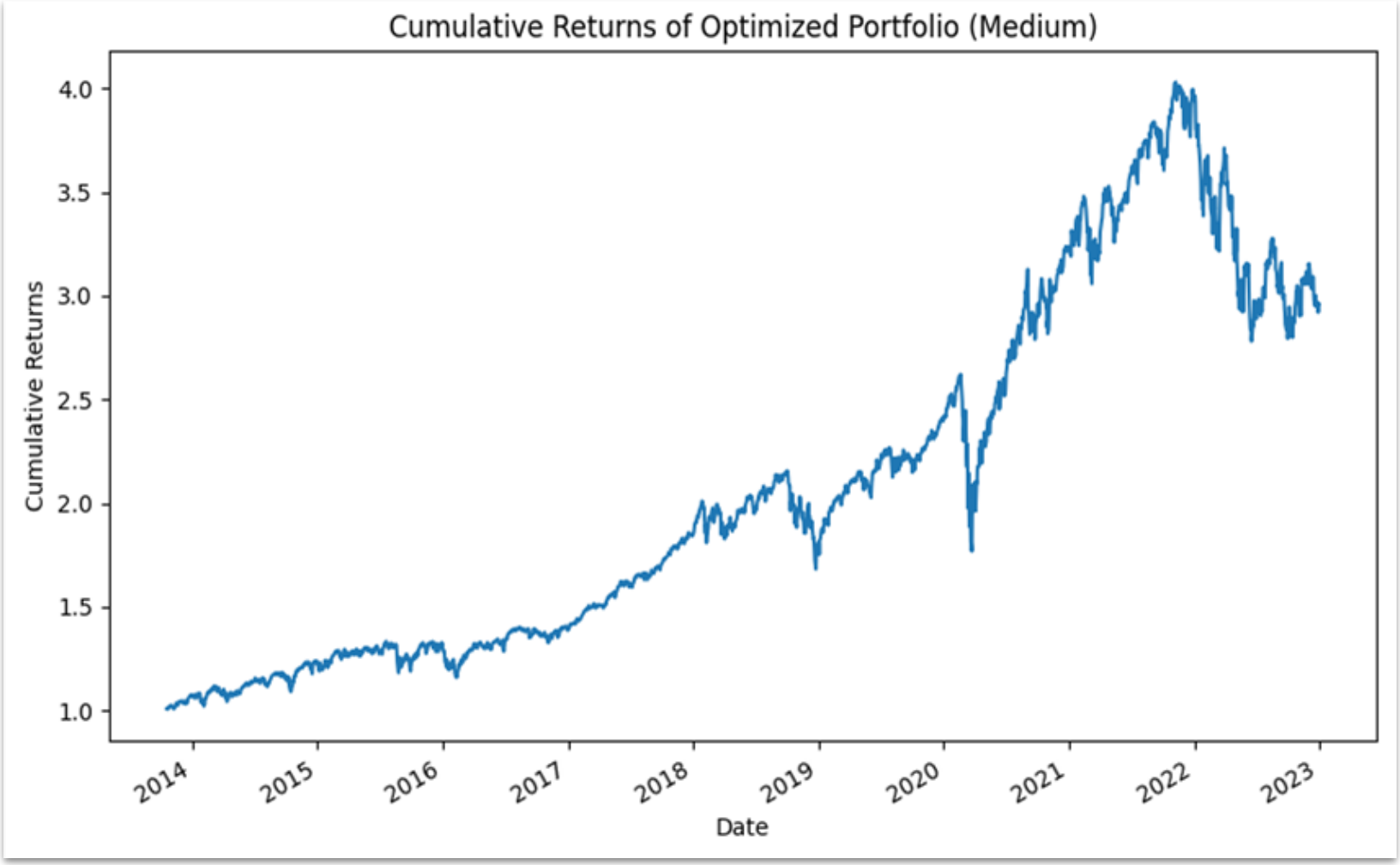


Figure 4 - Cumulative Returns of Optimized Medium PPO Portfolio

Parameter	Value
Episodes	100
Actor Learning Rate	0.001
Critic Learning Rate	0.001
Clip Ratio	0.2
Training Interval	10

ETFs Used: **IWF, EEM, SHYG, MTUM**
Average Sharpe Ratio (Train, Test): **1.75, 1.73**

- The growth trajectory of the portfolio is quite strong, with an evident peak in performance around the year **2022**, followed by a decline.
- The average Sharpe Ratios are quite **high**, with **1.75** for the training period and **1.73** for the testing period, indicating a very good risk-adjusted performance.
- With a medium-size portfolio, transaction fees are **reduced** than the large size with a better overall Sharpe.

Results (Small)

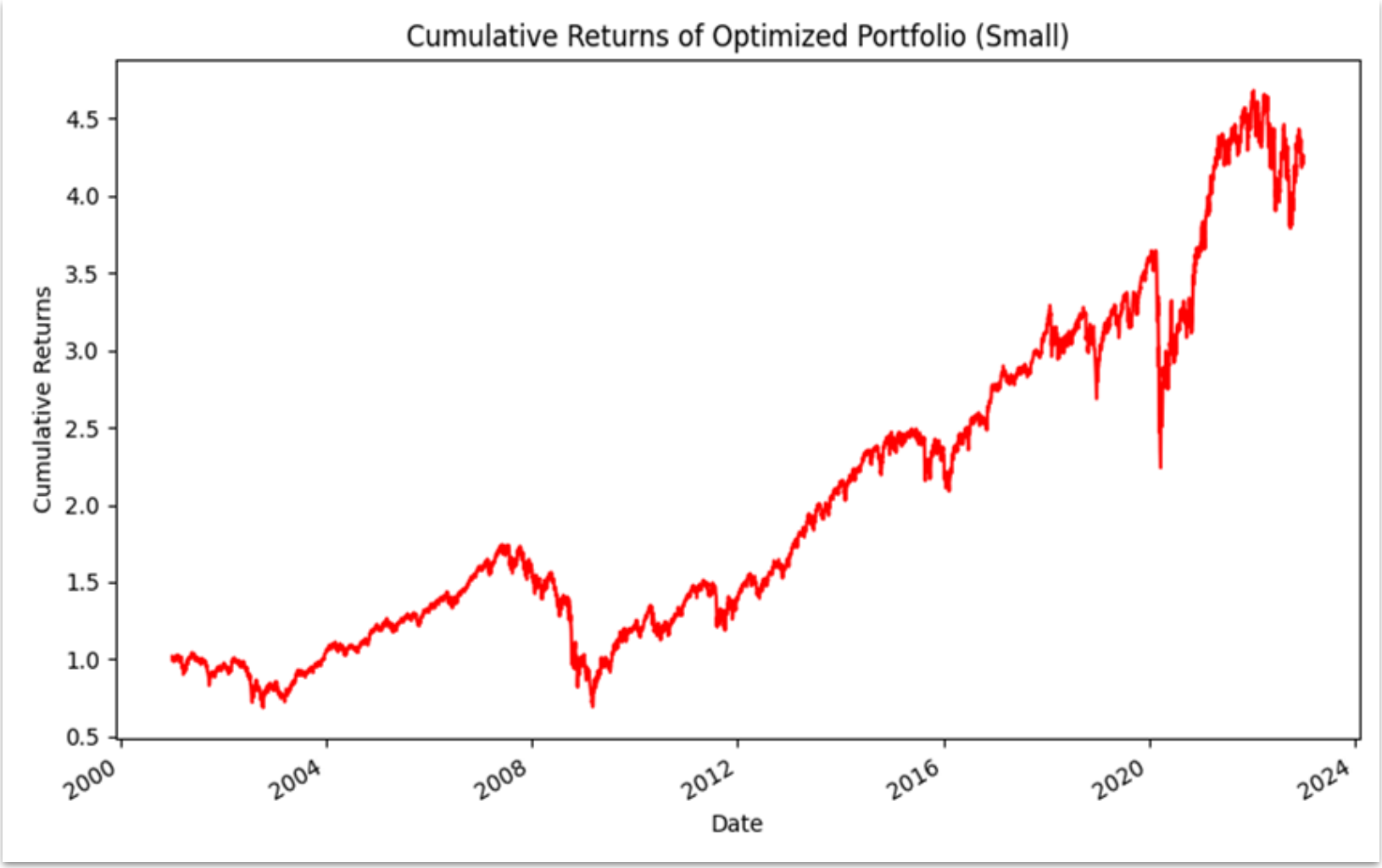


Figure 5 - Cumulative Returns of Optimized Small PPO Portfolio

Parameter	Value
Episodes	100
Actor Learning Rate	0.001
Critic Learning Rate	0.001
Clip Ratio	0.2
Training Interval	10

ETFs Used: **IWD, EWJ**
Average Sharpe Ratio (Train, Test): **2.03, 2.01**

- There is a visible long-term upward trend in the portfolio's performance, and a notable increase around **2016** leading up to a peak before a slight pullback (Covid).
- The average Sharpe Ratios are very **high**, at **2.03** for the training period and **2.01** for the testing period, strongest out of all.
- Given the time frame, the portfolio has weathered various **market cycles/regimes**, including the early 2000s recession, the 2008 financial crisis, etc.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

7. Next Steps

Reinforcement Learning in QWIM

Next Steps

To Be Implemented

- **Data Expansion and Diversity**: Incorporate a broader range of financial instruments, including bonds, commodities, and perhaps cryptocurrencies, to diversify the data set and test the algorithm's adaptability across different asset classes.
- **Hyperparameter Tuning**: Experiment with different settings for learning rates, clip ratios, and training intervals to fine-tune the model's performance. This could involve techniques like grid search or random search for hyperparameter optimization.
- **Algorithmic Enhancements**: Explore advanced variants of PPO or other reinforcement learning algorithms that could potentially offer better performance, especially in volatile market conditions.
- **Feature Engineering**: Enhance the model's input features by including macroeconomic indicators or sentiment analysis data to give the model a more comprehensive view of the market.
- **Incorporating Transaction Fees, Market Liquidity, and Other Market Realities**: Adapt the model to account for more transaction costs, market liquidity, and other real-world trading factors. This involves simulating transaction fees to reflect the cost of executing trades, considering the impact of large orders on market prices (market impact), and accounting for the liquidity of different assets.

STEVENS
INSTITUTE OF TECHNOLOGY
1870

8. References

Reinforcement Learning in QWIM

References

- Cong, Lin William, Ke Tang, Jingyuan Wang, and Yang Zhang. 2022. "AlphaPortfolio: Direct Construction Through Deep Reinforcement Learning and Interpretable AI."
- Das, Sanjiv, and Subir Varma. 2020. "Dynamic Goals-Based Wealth Management Using Reinforcement Learning."
- Benhamou, Eric, David Saltiel, Sandrine Ungari, Abhishek Mukhopadhyay, and Jamal Atif. 2020. "AAMDRL: Augmented Asset Management with Deep Reinforcement Learning."
- Guan, Mao, and Xiao-Yang Liu. 2021. "Explainable Deep Reinforcement Learning for Portfolio Management: An Empirical Approach."
- Das, Sanjiv R., Daniel Ostrov, Aviva Casanova, Anand Radhakrishnan, and Deep Srivastav. 2021. "Optimal Goals-Based Investment Strategies For Switching Between Bull and Bear Markets."
- Bauman, Tessa, Bruno Gašperov, Stjepan Begušić, and Zvonko Kostanjčar. 2023. "Deep Reinforcement Learning for Robust Goal-Based Wealth Management."
- Wu, Bo, and Lingfei Li. 2023. "Reinforcement Learning for Continuous-Time Mean-Variance Portfolio Selection in a Regime-Switching Market."
- Nakagawa, Kei, Masaya Abe, and Junpei Komiyama. 2020. "RIC-NN: A Robust Transferable Deep Learning Framework for Cross-sectional Investment Strategy."

STEVENS
INSTITUTE OF TECHNOLOGY
1870

9. Appendix

Reinforcement Learning in QWIM

STEVENS
INSTITUTE OF TECHNOLOGY
1870

10. Literature Review

Reinforcement Learning in QWIM

AAMDRL: Augmented Asset Management with Deep Reinforcement Learning

By Eric Benhamou, David Saltiel, Sandrine Ungari, Abhishek Mukhopadhyay, Jamal Atif (2020)

Abstract

- The authors use **Deep Reinforcement Learning (DRL)** to create an augmented asset manager bot for asset management.
- They aimed to use **trading bots** to illustrate how DRL can tackle this challenge.
- They used data from a noisy and self-adapting environment with sequential, non-stationary, and non-homogeneous observations.
- Methods included using augmented state, implementing a one period lag between observations and actions, developing a new repetitive train test method called walk forward analysis, similar to cross validation for time series.
- Their proposed method, AAMDRL, achieves **superior returns and lower risk** for an augmented asset manager.

Numerical Results

- For **baseline**, the authors used Risky asset (MBCI world index), follow the winner strategy, follow the loser strategy, and the Markowitz theory.
- The DRL model was implemented with augmented states and without.
- The **DRL model with augmented states** performed the best over 3 categories: return, Sortino ratio, and Sharpe ratio.

	3 Years			
	return	Sortino	Sharpe	max DD
Risky asset	10.27%	0.34	0.38	- 0.34
DRL	**22.45%**	**1.18**	**1.17**	-0.27
Winner	13.19%	0.66	0.72	-0.35
Loser	9.30%	0.89	0.89	**-0.15**
DRL no context	8.11%	0.42	0.47	-0.34
Markowitz	-0.31%	-0.01	-0.01	-0.41

	5 Years			
	return	Sortino	Sharpe	max DD
Risky asset	9.16%	0.54	0.57	- 0.34
DRL	**16.42%**	**0.98**	**0.96**	-0.27
Winner	10.84%	0.65	0.68	-0.35
Loser	7.04%	0.78	0.76	**-0.15**
DRL no context	6.87%	0.44	0.47	-0.34
Markowitz	-0.07%	-0.00	-0.00	-0.41

Explainable Deep Reinforcement Learning for Portfolio Management: An Empirical Ap

By Mao Guan and Xiao-Yang Liu (2021)

Abstract

- The authors compare methods of **deep reinforcement learning (DRL)** with **classical machine learning models** in portfolio management.
- The authors introduce a new method to understand the strategies of DRL agents in portfolio management. It involves using coefficients from a linear model in hindsight as reference feature weights.
- They assess the prediction power by calculating **linear correlations** between the feature weights of the DRL agent and the reference feature weights to check if the DRL has learned to focus on the most important environment features.
- The approach is evaluated on a portfolio management task using Dow Jones 30 constituent stocks from 2009 to 2021, and shows that the **DRL agent exhibits stronger** multi-step prediction power than machine learning methods.

Numerical Results

- The **DRL models** used were Proximal Policy Optimization (PPO) and Advantage Actor Critic (A2C).
- The **ML models** used were Decision Tree (DT), Linear Regression (LR), Random Forest (RF), and Support Vector Machine (SVM).
- The **DRL models outperformed** the classical ML models and the DJIA over the trading period with excellent return, volatility, Sharpe ratio, and correlation coefficients over the multi-step predictions.

(2020/07/01-2021/09/01)	PPO	A2C	DT	LR	RF	SVM	DJIA
Annual Return	**35.0%**	34 %	10.8%	17.6%	6.5%	26.2%	31.2%
Annual Volatility	**14.7%**	14.9 %	40.1%	42.4%	41.2 %	16.2 %	14.1 %
Sharpe Ratio	**2.11**	2.04	0.45	0.592	0.36	1.53	2.0
Calmar Ratio	**4.23**	4.30	0.46	0.76	0.21	2.33	3.5
Max Drawdown	-8.3%	**-7.9%**	-23.5%	-23.2%	-30.7 %	-11.3 %	-8.9 %
Ave. Corr. Coeff. (single-step)	0.024	0.030	**0.068**	0.055	0.052	0.034	N/A
Ave. Corr. Coeff. (multi-step)	**0.09**	0.078	-0.03	-0.03	-0.015	-0.006	N/A

AlphaPortfolio: Direct Construction Through Deep Reinforcement Learning and Interp

By Lin Cong, Ke Tang, Jingyuan Wang, Yang Zhang (2022)

Abstract

- They use **deep reinforcement learning** to optimize portfolio management, an alternative to traditional supervised-learning methods.
- They introduce attention-based neural network models tailored for financial big data, which interact with market states and train without labels.
- Their "AlphaPortfolio" models demonstrate:
  - Strong out-of-sample results, e.g., **Sharpe Ratio** above **2.0** and over **13%** risk-**adjusted alpha** with monthly re-balancing.
  - Robustness across varied market conditions and when excluding certain stocks.
- The models can accommodate factors like transaction costs, state interactions, and different objectives.
- Through polynomial-feature-sensitivity analysis, they identify key factors influencing investment performance, including their dynamics.
- The study emphasizes the value of deep reinforcement learning in finance and introduces "economic distillation" for model interpretation.

Numerical Results

- AlphaPortfolio forms a monthly L/S portfolio based on winner scores' decile rankings.
- “q_n” denotes nth NYSE size percentile
- Performance metrics (annualized) are in columns (1)-(3): Return, Std.Dev., Sharpe ratio.
- Adjusted portfolio returns using models are in columns (4)-(9): CAPM, various Fama-French models, Stambaugh-Yuan, and Hou-Xue-Zhang.

	AP Performance				AP Excess Alpha					
	(1)	(2)	(3)		(4)	(5)	(6)	(7)	(8)	(9)
Firms	All	$> q_{10}$	$> q_{20}$	Factor Models	All		$> q_{10}$		$> q_{20}$	
					$\alpha(\%)$	R^2	$\alpha(\%)$	R^2	$\alpha(\%)$	R^2
Return(%)	17.00	17.09	18.06	CAPM	13.9***	0.005	12.2***	0.088	14.0***	0.102
Std.Dev.(%)	8.48	7.39	8.19	FFC	14.2***	0.052	13.4***	0.381	14.7***	0.465
Sharpe	2.00	2.31	2.21	FFC+PS	13.7***	0.054	12.3***	0.392	13.3***	0.480
Skewness	1.42	1.73	1.91	FF5	15.3***	0.12	13.8***	0.426	14.7***	0.435
Kurtosis	6.35	5.70	5.97	FF6	15.6***	0.128	14.5***	0.459	15.8***	0.516
Turnover	0.26	0.24	0.26	SY	17.4***	0.037	15.8***	0.332	17.0***	0.394
MDD	0.08	0.02	0.02	Q4	16.0***	0.121	15.0***	0.495	16.2***	0.521

Dynamic Goal-Based Wealth Management Using Reinforcement Learning

By Sanjiv Das and Subir Varma (2020)

Abstract

- The research introduces an RL (Reinforcement Learning) algorithm tailored for dynamically optimal goal-based portfolios, which aim to adaptively achieve specific financial objectives.
- The RL-derived solution aligns with what one would achieve using **dynamic programming**, a method known for breaking complex problems into simpler subproblems.
- This RL approach is **model-free**, meaning it doesn't rely on predefined market assumptions. Instead of using backward methods like dynamic programming, it employs forward simulation to project future scenarios for decision-making.
- Beyond the specific algorithm, the paper provides a brief overview of various RL methods, giving readers a broader context of reinforcement learning's landscape.
- An example application in the paper demonstrates RL's potential in addressing finance challenges, especially those with path-dependency and large state spaces, which represent the intricate and vast scenarios often seen in financial contexts.

Numerical Results

- Initial portfolio value is W(0) = 100 with a target of 200 over a 10-year horizon.
- Utilized **15** portfolios, based on a mean vector of returns (M) and a covariance matrix.
- Algorithm parameters were **α = 0.10** and **γ = 1**; zero infusions assumed
- Runtime varies for 50K epochs (~1.5 minutes) and 100K epochs (~3 minutes); DP = 0.5 sec

MODEL INPUTS

$$M = \begin{bmatrix} 0.05 \\ 0.10 \\ 0.25 \end{bmatrix}; \quad \sum = \begin{bmatrix} 0.0025 & 0 & 0 \\ 0 & 0.04 & 0.02 \\ 0 & 0.02 & 0.25 \end{bmatrix}$$

	Portfolios							
	0	1	2	3	4	5	6	7
μ	0.0526	0.0552	0.0577	0.0603	0.0629	0.0655	0.0680	0.0706
σ	0.0485	0.0486	0.0493	0.0508	0.0529	0.0556	0.0587	0.0623

	Portfolios						
	8	9	10	11	12	13	14
μ	0.0732	0.0757	0.0783	0.0809	0.0835	0.0860	0.0886
σ	0.0662	0.0705	0.0749	0.0796	0.0844	0.0894	0.0945

MODEL OUTPUTS

ϵ	No. of epochs	$V[W(0), t = 0]$
DP solution	1	0.72
0.10	50K	0.65
0.10	100K	0.65
0.20	50K	0.69
0.20	100K	0.71
0.25	100K	0.71
0.30	50K	0.72
0.30	100K	0.72
0.40	50K	0.73
0.40	100K	0.77
0.40	200K	0.75
0.40	500K	0.71

Reinforcement Learning for Continuous-Time Mean-Variance Portfolio Selection in a Regime-Switching Market

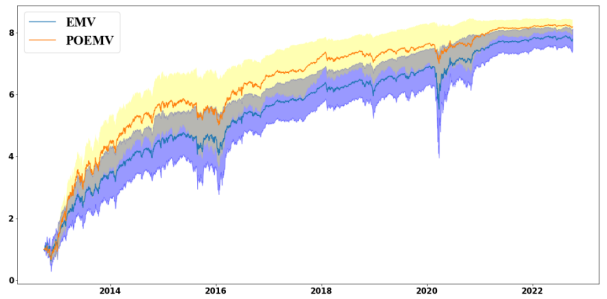
By Bo Wu, Lingfei Li (2023)

Abstract

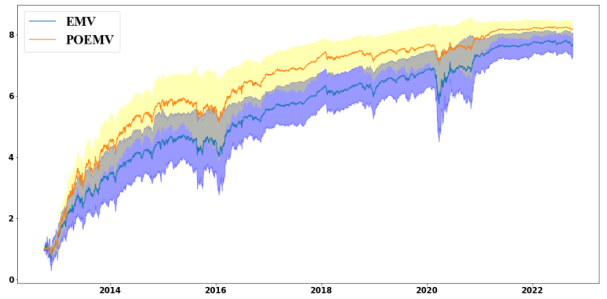
- The study proposes a **r**einforcement learning approach for solving the continuous-time mean-variance portfolio selection problem in an uncertain market with unobservable regime switches.
- To encourage exploration and learning, they create a stochastic control problem with an **entropy-regularized mean-variance objective**.
- The optimal value function and policy are represented semi-analytically and involve solutions to two linear parabolic partial differential equations.
- These representations are used to parameterize the value function and policy, approximating the unknown solutions with polynomials for learning.
- An actor-critic reinforcement learning algorithm is developed to learn the optimal policy, involving market regime probability estimation, policy evaluation, and policy gradient updates.
- Empirical results demonstrate the algorithm's superiority in long-term investment scenarios compared to classical control approaches and other reinforcement learning methods.

Numerical Results

- Developed an actor-critic RL algorithm for continuous-time MV investment in a regime-switching market where the drift of the risky asset price is modulated by an unobservable Markov chain.
- Target level for **the mean value was 8** which all models fell relatively close to. However, they differed substantially in variance of terminal wealth.
- Despite POEMV's excellent performance, the sharpe ratio produced is rather unusual and may be a result of data anomalies that were failed to be considered.



(a) Daily rebalancing



(b) Monthly rebalancing

	Mean	Variance	Sharpe ratio
POEMV	8.19	0.224	15.38
EMV	7.79	0.453	10.09
Classical	8.15	1.537	5.77

(a) Daily rebalancing

	Mean	Variance	Sharpe ratio
POEMV	8.19	0.284	13.49
EMV	7.69	0.597	8.66
Classical	8.22	10.081	2.27

(b) Monthly rebalancing

Table 4: Mean and variance of the terminal wealth and the 10-year Sharpe ratio of three algorithms

A Robust Transferable Deep Learning Framework for Cross-sectional Investment Strategy

By Kei Nakagawa, Masaya Abe, and Junpei Komiyama (2020)

Abstract

- Stock return predictability is a significant area of research, with "factor" statistics proposed to explain stock return dynamics.
- Despite the increasing popularity of machine learning in stock return prediction, predicting stock returns remains challenging, and many investors still rely on intuition.
- The challenge is to create an investment strategy that is consistent over time with **minimal human intervention**.
- The proposed framework, Ranked Information Coefficient Neural Network (RIC-NN), is a deep learning approach with three key features: nonlinear multi-factor analysis, stopping criteria based on ranked information coefficient (rank IC), and deep transfer learning across multiple regions.
- RIC-NN aims to enhance stock return prediction by incorporating these novel ideas, facilitating more reliable and automated decision-making in investment strategies.
- The authors conducted a comprehensive evaluation of their approach based on the stocks in different Morgan Stanley Capital International (MSCI) indices; namely the North American region and the Asia Pacific region.

Numerical Results

- The annualized return is the excess return (Alpha) against the average return of all stocks in the universe, the risk (tracking error; TE) is calculated as the standard deviation of Alpha and risk-normalized return is measured by information ratio (IR).
- Its identical neural network architecture and RankIC stopping value consistently generated favorable returns in diverse market conditions.
- RIC-NN demonstrated superior performance when compared to off-the-shelf machine learning methods in experimental comparisons.

Table 1: Experimental Results of Long portfolio in MSCI North America. Bold characters indicate the best ones among each category. Alpha measures return, TE measures risk, and IR measures a risk-normalized return.

Long	Linear	Nonlinear			
	LASSO	RF	NN	RIC-NN	RIC-NN (TF from AP)
Alpha	0.62%	0.79%	0.82%	**1.23%**	1.20%
TE	5.40%	5.14%	4.48%	**4.14%**	4.43%
IR	0.11	0.13	0.18	**0.30**	0.27

Table 2: Experimental Results of Long portfolio in MSCI Pacific.

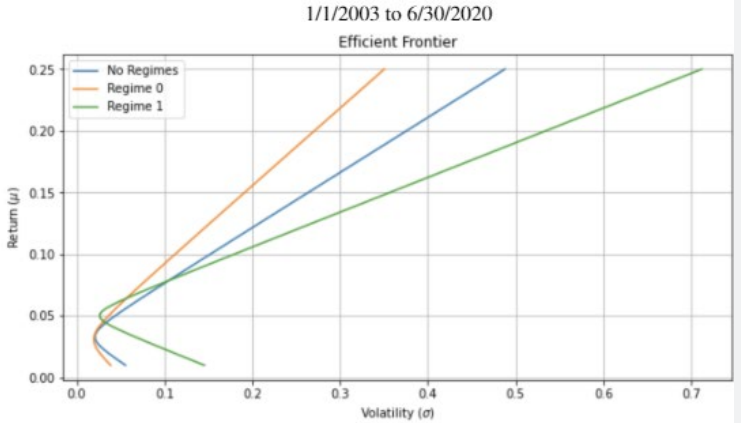
Long	Linear	Nonlinear			
	LASSO	RF	NN	RIC-NN	RIC-NN (TF from NA)
Alpha	5.35%	3.79%	4.34%	5.25%	**5.78%**
TE	5.17%	5.75%	4.18%	4.20%	**3.95%**
IR	1.04	0.66	1.04	1.25	**1.46**

Optimal Goals-Based Investment Strategies For Switching Between Bull and Bear Markets

By Sanjiv R. Das, Daniel Ostrov, Aviva Casanova, Anand Radhakrishnan, Deep Srivastav (2021)

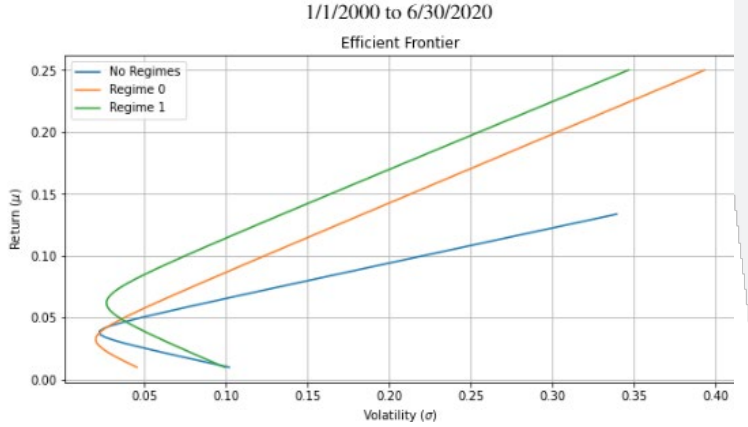
Abstract

- The objective is to achieve a target level of wealth (i.e. **Goals-Based Investing**) over the long-term investment horizon where market regimes may switch (i.e. from a bull market to a bear market, and vice versa)
- They observe that investors cognizant of regime switches perform better than those recognizing one regime, however this is entirely dependent on the former being able to predict the current regime with significant confidence
- With the use of **dynamic programming**, **Bayesian techniques** and data from recent history, it is shown that investors may simply perform better under the assumption of a single-regime (meaning investing with no regard to regimes)



Numerical Results

- Constructs efficient frontiers of a portfolio containing three index-based assets representing U.S. bonds, U.S. stocks, and international stocks
- This effectively creates the returns distribution of a variety of funds (15) within different regimes and regime assumptions
- Visually determined the superiority of the two-regime strategy under the aforementioned caveat, else it loses to the one-regime strategy
- An alternative strategy of mixing optimal funds based on weighting by the probability of being in each regime underperforms selection based on one regime strictly



Deep Reinforcement Learning for Robust Goal-Based Wealth Management

By Tessa Bauman, Bruno Gašperov, Stjepan Begušić, Zvonko Kostanjčar (2023)

Abstract

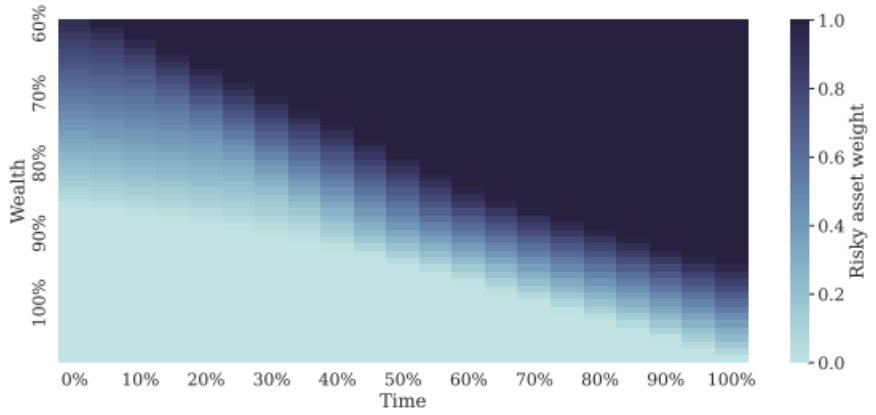
- Demonstrates the superior performance of RL over several standard Goal-Based Wealth Management implementations
- Makes use of both historical and simulated (via **bootstrapping**) to substantiate results
- Balances capital preservation and portfolio growth over time with a simple two asset model (pairing of stock market index and a risk-free asset)
- Models the underlying interactions between the agent and environment as a **Markov Decision Process**, where the actions are the shifting portfolio weights and the reward function is binary

Numerical Results

- RL outperforms Deterministic Glide Path, Merton's Constant, Variance Budgeting, and Dynamic Programming within all three testing methods
- The **RL policy** (visualization of the state space and selected actions in each state) follows wealth management intuition and the properties of the Glide Path investment strategy

Table 1: Comparison of results - DRL vs benchmarks

	Historical data	Simulated data (window size)			Bootstrapped data (block size)		
		36	{24, 36, 48}	60	1	{1,2,3}	{4,5,6}
DG	54.3%	70.5%	70.3%	69.7%	79.2%	76.9%	75.1%
MC	67.0%	71.8%	72.3%	67.0%	80.7%	77.4%	74.6%
VB	68.6%	74.1%	74.1%	70.6%	87.2%	83.5%	80.1%
DP	74.4%	73.7%	73.8%	74.7%	88.9%	84.4%	80.7%
RL	**77.5%**	**76.7%**	**76.4%**	**75.3%**	**90.8%**	**86.3%**	**82.3%**



STEVENS
INSTITUTE OF TECHNOLOGY
1870

11. RL Algorithm

Reinforcement Learning in QWIM

Reinforcement Learning

Algorithms

- **Q-Learning**
  - Model-free
  - Aims to find the best action in each state by learning the optimal Q-values (Bellman Equation)
  - Flexible, but at risk of being inefficient (scalability issues, inefficient with similar states, slow convergence.
- **Deep Q Networks**
  - Deep neural networks (NN) to approximate Q values for better scalability
  - Deep learning allows for sophisticated pattern recognition, and the algorithm can be stabilized (experience replay and fixed Q-targets)
  - Added complexity, resource-intensity, and overestimation risks
- **Policy Gradient Methods**
  - Directly learns and updates the policy function (which action the agent should take)
  - Better convergence and can better handle high-dimensional data
  - May exhibit high variance (slower convergence) and sensitivity to initial parameters

Reinforcement Learning

Algorithms

- **Actor-Critic Methods**
  - Two models: an actor that decides actionand a critic that evaluates the action
  - Balance between value-based and policy-based RL approaches, helping to reduce variance
  - Requires higher model maintenance and complex training
- **Proximal Policy Optimization**
  - Conservatively updates policy for the sake of stable learning
  - Strikes a balance between sample efficiency and complexity
  - Complex hyperparameter tuning, added computational cost, and exploitation risk
- **Monte Carlo Tree Search**
  - Builds trees to simulate the value of many future actions
  - Ideal for sequential decision-making (i.e. multi-step investments)
  - While addressing exploration-exploitation well, this is not ideal for high-dimensional continuous spaces and inadequate computational resources
  - Heavily reliant on ideal simulation

STEVENS
INSTITUTE OF TECHNOLOGY
1870

12. Code

Reinforcement Learning in QWIM

```python
def train(self, states: np.ndarray, actions: np.ndarray, rewards: np.ndarray, next_states: np.ndarray, dones: np.ndarray):
    """
    Implements the training loop for the PPO agent.

    Args:
        states (np.ndarray): Batch of states.
        actions (np.ndarray): Batch of actions taken.
        rewards (np.ndarray): Batch of rewards received.
        next_states (np.ndarray): Batch of next states.
        dones (np.ndarray): Batch of done flags (boolean) indicating the end of an episode.
    """

    # Convert numpy arrays to TensorFlow tensors.
    states = tf.convert_to_tensor(states, dtype=tf.float32)
    actions = tf.convert_to_tensor(actions, dtype=tf.int32)
    rewards = tf.convert_to_tensor(rewards, dtype=tf.float32)
    next_states = tf.convert_to_tensor(next_states, dtype=tf.float32)
    dones = tf.convert_to_tensor(dones, dtype=tf.float32)

    # Calculate the value of current and next states.
    values = self.critic(states)
    next_values = self.critic(next_states)

    # Calculate discounted rewards and advantages.
    target_values = self.calculate_discounted_rewards(rewards, dones)
    advantages = self.calculate_advantages(rewards, values, next_values, dones)

    # Update the actor and critic networks.
    with tf.GradientTape() as actor_tape, tf.GradientTape() as critic_tape:
        # Calculate current policy probabilities.
        current_probs = self.actor([states, advantages, self.actor(states)])

        # Calculate policy loss.
        p_loss = self.policy_loss(advantages, self.actor(states), actions, current_probs, self.clip_ratio)

        # Recompute critic values and calculate value loss.
        values = self.critic(states)
        v_loss = self.value_loss(values, target_values)

    # Compute gradients and update actor network.
    actor_grads = actor_tape.gradient(p_loss, self.actor.trainable_variables)
    self.actor_optimizer.apply_gradients(zip(actor_grads, self.actor.trainable_variables))

    # Compute gradients and update critic network.
    critic_grads = critic_tape.gradient(v_loss, self.critic.trainable_variables)
    self.critic_optimizer.apply_gradients(zip(critic_grads, self.critic.trainable_variables))
```

Figure 4 - Train Method in PPO.py

```python
def evaluate_agent(self, test_data: pd.DataFrame, ppo_agent: PPO, risk_free_rate: float = 0.0):
    """
    Evaluates the PPO agent using the testing dataset.

    Args:
        test_data (pd.DataFrame): The testing dataset.
        ppo_agent (PPO): The trained PPO agent.
        risk_free_rate (float): The risk-free rate for calculating risk-adjusted returns.
    """

    current_index = 0
    total_reward = 0
    portfolio_values = []
    negative_returns = []
    risk_free_rate = self.get_current_risk_free_rate()

    while current_index + self.state_window < len(test_data):
        state_data = test_data.iloc[current_index:current_index + self.state_window]
        state = state_data['Normalized_Close'].values.flatten()

        action = ppo_agent.select_action(state)
        next_index = current_index + 1
        next_state, reward, done = self.execute_action(action, current_index, test_data)

        total_reward += reward
        current_index = next_index

        portfolio_value = 1 + total_reward
        portfolio_values.append(portfolio_value)

        # Track negative returns for Sortino Ratio.
        if reward < risk_free_rate:
            negative_returns.append(reward - risk_free_rate)

        if done:
            break

    final_portfolio_value = portfolio_values[-1]
    average_return = np.mean(portfolio_values)
    std_dev = np.std(portfolio_values)

    sharpe_ratio = (average_return - risk_free_rate) / std_dev if std_dev != 0 else 0

    # Calculate the Sortino Ratio.
    downside_deviation = np.sqrt(np.mean(np.square(negative_returns))) if negative_returns else 0
    sortino_ratio = (average_return - risk_free_rate) / downside_deviation if downside_deviation != 0 else 0

    print(f'Final Portfolio Value: {final_portfolio_value}')
    print(f'Average Return: {average_return}')
    print(f'Sharpe Ratio: {sharpe_ratio}')
    print(f'Sortino Ratio: {sortino_ratio}')
```

Figure 5 - Evaluate Agent Method in PortfolioOptimization.py

STEVENS
INSTITUTE OF TECHNOLOGY
1870

THANK YOU

Stevens Institute of Technology
1 Castle Point Terrace, Hoboken, NJ 07030