Portfolio Optimization using Reinforcement Learning

Lavanya Yogendra Lahane
Indian Institute of Technology, Ropar

April 20, 2025

Abstract

This report presents a reinforcement learning approach to portfolio optimization, comparing the performance of an RL agent against traditional benchmark strategies including the Sortino ratio-optimized portfolio and the equally weighted $\frac{1}{N}$ portfolio. The methodology leverages historical daily stock price data from ten major Indian companies spanning the period from 2004 to 2024.

The core of the approach involves training a Proximal Policy Optimization (PPO) agent to learn optimal asset allocation strategies that maximize risk-adjusted returns. To simulate a realistic trading environment and ensure out-of-sample evaluation, a rolling window technique is adopted. In this setup, the agent is trained on a moving window of 126 trading days (approximately six months), and its performance is evaluated over the subsequent 21 trading days (approximately one month). After each evaluation period, the window is advanced by 21 days to form the next training-testing split.

This rolling window framework mimics how strategies would adapt to evolving market conditions and provides a robust way to assess the generalization performance of the agent across different market regimes.

1 Problem Setup

Let there be N assets and a time horizon T. At each time step $t \in \{0, 1, ..., T\}$, the agent observes a state s_t and selects a portfolio weight vector $\mathbf{w}_t \in R^N$ such that:

$$\sum_{i=1}^{N} w_{t,i} = 1, \quad w_{t,i} \geq 0$$

The environment then returns a reward r_t, commonly defined based on the Sharpe ratio or cumulative return.

2 Data and Preprocessing

We use daily price data from Yahoo Finance for 10 major Indian stocks from 2004 to 2024. Log returns are computed using:

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

Missing values are dropped, and the data is normalized to ensure stationarity.

3 Environment Design

We define a custom `gym.Env` for portfolio optimization. The key components are:

- **State:** A window of past k days of returns, normalized
- **Action:** Softmax-transformed weights over N assets
- **Reward:** Difference of sortino ratio over the holding period
- **Step Function:** Advances the window by y days (holding period)

Reward Formulation

At each evaluation step, the agent receives a reward based on its outperformance relative to two benchmarks:

1. Equal-weighted portfolio (S_{eq})
2. Mean-variance optimized portfolio (S_{mv})

The reward function is defined as:

$$R_t = S_{\text{agent}} - \max\left(S_{\text{eq}}, S_{\text{mv}}\right)$$

where:

- R_t is the reward at time step t,
- S_{agent} is the Sortino ratio of the RL agent's portfolio,
- S_{eq} and S_{mv} are the Sortino ratios of the equal-weighted and mean-variance portfolios respectively.

4 Algorithm

We use Proximal Policy Optimization (PPO) from the `stable-baselines3` library. PPO updates the policy π_θ by maximizing the clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = E_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $\hat{A}_t$ is the advantage estimate.

Baseline Algorithms

- **1/N Portfolio:** Equal weights
- **Mean-Variance Optimization:** Uses `PyPortfolioOpt` to solve:

$$\max_{\mathbf{w}} \quad \mu^\top \mathbf{w} - \frac{\lambda}{2}\mathbf{w}^\top \Sigma \mathbf{w}$$

5 Rolling Window Strategy

We implement a rolling window training/testing strategy:

- Train over $x = 126$ days
- Test/hold for $y = 21$ days
- Roll forward by y days

The portfolio weights returned by PPO are applied over the next y days, and cumulative returns are tracked.

6 Results

Portfolio	Average Cumulative Return
RL Agent (PPO)	19381.026%
Equal-Weighted (1/N)	213.467%
Mean-Variance	284.590%

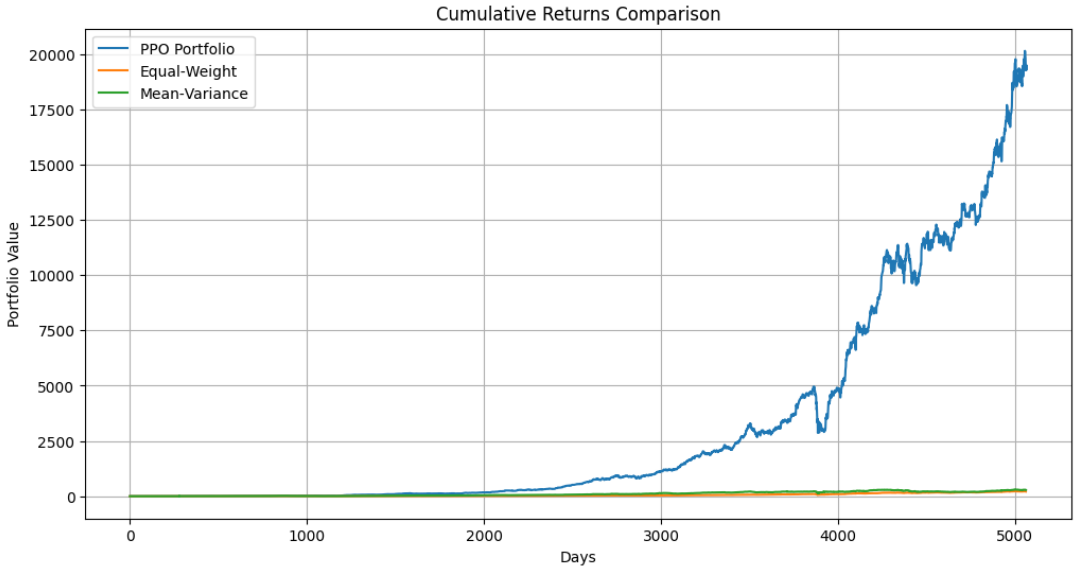

Figure 1: Comparison of Portfolio Values Over Time