STAT 4830 Week 3 Deliverable
Portfolio Maximization
NAPPERS

Problem Statement

This project develops and studies an optimization-based pipeline for constructing equity portfolios that aim to outperform standard benchmarks out of sample. At a high level, the decision variable at each rebalancing time t is a vector of portfolio weights $w_t \in \mathbb{R}^{N_t}$ over a set of tradable assets, chosen using only information available at time t and then held over the next period. The immediate goal for Week 3 is to instantiate and test this pipeline on a CRSP-style monthly stock panel (with fields such as `date`, `permno`, `ret`, `prc`, and `vol`), but the design is intended to generalize to other return datasets and universes. The quantity being optimized is the trade-off between expected portfolio return and risk under realistic investment constraints. Given estimates of conditional expected returns $\mu_t \approx \mathbb{E}_t[r_{t+1}]$ and a conditional covariance matrix Σ_t constructed from historical data, the baseline optimization problem chooses w_t by solving

$$\max_{w_t} \ \mu_t^\top w_t - \gamma\, w_t^\top \Sigma_t w_t, \qquad \text{s.t. } \sum_i w_{t,i} = 1,\ w_{t,i} \geq 0,$$

so that portfolios are fully invested and long-only. Success is defined in terms of strictly out-of-sample performance relative to clear benchmarks such as an equal-weighted portfolio or a market index. Several practical and methodological constraints shape the problem. All experiments must be feasible on a single CPU machine, which limits the size of the asset universe and the complexity of hyperparameter searches in early iterations. All estimation and evaluation must respect temporal ordering to avoid look-ahead bias, and the optimization must be robust to noisy, low–signal-to-noise return data. The initial CRSP-style panel provides a realistic environment in which to test the pipeline, but the project is exposed to standard risks: overfitting to a particular historical window, hidden data leakage through improper panel handling, unstable or extreme optimized weights, and the possibility that apparent statistical gains do not survive when compared to simple benchmark strategies or when extended to new datasets.

Technical Approach

The project treats portfolio construction as a sequence of constrained optimization problems, one at each rebalancing time t. Given a cross section of N_t assets with estimated conditional expected returns $\mu_t \approx \mathbb{E}_t[r_{t+1}] \in \mathbb{R}^{N_t}$ and a conditional covariance matrix $\Sigma_t \in \mathbb{R}^{N_t \times N_t}$ computed from past returns, the baseline allocation problem is

$$\max_{w_t} \ \mu_t^\top w_t - \gamma\, w_t^\top \Sigma_t w_t - \kappa \|w_t - w_{t-1}\|_1, \qquad \text{s.t. } \sum_{i=1}^{N_t} w_{t,i} = 1,\ \ w_{t,i} \geq 0\ \forall i.$$

The quadratic term penalizes portfolio variance, while the ℓ_1 turnover penalty $\kappa\|w_t - w_{t-1}\|_1$ serves as a simple proxy for transaction costs and discourages excessive trading between periods. The constraints enforce full investment and long-only positions. This formulation is designed to be generic: although the initial implementation uses a CRSP-style monthly equity panel to construct (μ_t, Σ_t), the same optimization problem can be applied to any return dataset where conditional moments can be estimated. The optimization problem is solved numerically in PyTorch using gradient-based methods. To handle the simplex constraints in a differentiable way, portfolio weights are parameterized via unconstrained logits $v_t \in \mathbb{R}^{N_t}$ and mapped to feasible weights using a softmax transformation,

$$w_t = \mathrm{softmax}(v_t),$$

which guarantees $w_{t,i} \geq 0$ and $\sum_i w_{t,i} = 1$ by construction. Given estimated μ_t and Σ_t, the objective is implemented as a differentiable function of v_t, and gradients are obtained automatically via `autograd`. The baseline optimizer is a simple first-order method (e.g., gradient descent) with modest step sizes and iteration budgets. At each rebalancing date in a backtest, the pipeline (i) updates estimates (μ_t, Σ_t) from historical returns, (ii) solves the optimization problem to obtain w_t, and (iii) records the realized portfolio return $w_t^\top r_{t+1}$ using next-period returns from the panel. Early experiments are constrained to be CPU-feasible, with limited universe size and short backtest windows to keep iteration cycles fast. This staged design prioritizes an auditable optimization pipeline before introducing more complex models, alternative risk measures, or explicit asset-selection mechanisms.

Initial Results and Next Steps

The baseline pipeline was implemented in PyTorch and tested on the `sp500_monthly` dataset, using a fixed universe of large, liquid stocks and a 24-month rolling window to estimate (μ_t, Σ_t) at each rebalancing date. For every month in the backtest, the code (i) constructs historical windows of returns from the panel, (ii) estimates conditional mean returns and covariance, (iii) solves the constrained mean–variance optimization problem with a turnover penalty, and (iv) records the realized next-month portfolio return $w_t^\top r_{t+1}$. The implementation runs end-to-end on CPU in seconds, with full-investment and nonnegativity constraints satisfied numerically to tolerance and results reproducible under fixed random seeds. Out-of-sample performance of the optimized portfolio is very close to that of an equal-weighted benchmark rebalanced monthly. Over the backtest, the optimized portfolio attains an average monthly return of approximately 1.19%, an annualized volatility of about 16.9%, and an annualized Sharpe ratio of roughly 0.85, with a cumulative return near 1,760% and a maximum drawdown of about -46%. The equal-weighted benchmark exhibits essentially the same profile: mean monthly return 1.19%, annualized volatility 16.9%, Sharpe ratio 0.85, cumulative return around 1,775%, and a similar drawdown. Diagnostics on the weight paths explain this similarity: the average ℓ_1 distance between the optimized weights w_t and the equal-weight vector is only about 0.007 (on a scale where the maximum possible distance is 2), and the average turnover per rebalance is roughly 0.00023, meaning that only about 0.02% of the portfolio is traded each month. In other words, with simple rolling-mean signals and a turnover penalty, the optimizer learns a solution that is almost equal-weight and almost buy-and-hold. From an implementation perspective, this is reassuring: the pipeline behaves sensibly on real data, the optimization routine is stable, and sanity checks on constraints, shapes, and reproducibility all pass. At the same time, the lack of material outperformance highlights the limitations of the current signal and risk model and motivates more expressive specifications in subsequent iterations. The next development cycle will focus on making the optimization problem more informative and

stress-testing the robustness of the pipeline. On the technical side, an immediate priority is to explore the role of key hyperparameters: systematically varying the risk-aversion parameter γ and the turnover penalty κ (including the limiting case $\kappa = 0$) to see when, if at all, the optimizer deviates meaningfully from the equal-weight solution and how this affects Sharpe ratio, drawdowns, and turnover. In parallel, the estimation of (μ_t, Σ_t) will be refined: for example, replacing simple rolling means with basic momentum-style signals for μ_t and applying shrinkage methods to Σ_t to reduce covariance estimation error. These changes are designed to give the optimizer a stronger cross-sectional signal while maintaining computational simplicity. A key technical challenge is to balance responsiveness against trading costs: too little regularization leads to noisy, unstable portfolios, while too much regularization collapses the solution back to equal-weight. Questions for course staff include best practices for time-series cross-validation in portfolio settings (e.g., rolling vs. expanding windows, nested backtests) and diagnostic tools for detecting optimizer instability or hidden leakage in panel data. In later iterations, the project may consider alternative risk measures (such as downside risk or CVaR) or more explicit selection mechanisms, but for now the focus is on solidifying a leak-free, interpretable mean–variance framework and understanding how optimization design and moment estimation interact with the empirical properties of the `sp500_monthly` panel.

Self-Critique Guidlines