MA575 Project

Cardinality-Constrained L0+L2 Index Tracking for the S&P 100

Suneedhi Laddha & Vineel Pentrala

Introduction

The goal of this project is to implement an index tracking model from the literature and compare it with the basic least-squares (LS) tracking method discussed in class, which minimizes the squared error of return rates between the index and a portfolio of constituents.

The standard LS tracker typically uses all available assets, which can lead to large, operationally unwieldy portfolios. In practice, index funds often want a small number of names in the tracking portfolio to reduce trading, custody, and administrative costs.

To address this, we implement a cardinality-constrained L0+L2 index tracking model inspired by Takeda et al. (2013). The model:

- Controls sparsity via an L0 constraint on the number of nonzero weights.
- Adds an L2 penalty to stabilize weights.
- Uses a greedy forward-selection algorithm to approximate the solution.

We then compare this sparse tracker to the LS benchmark on historical SPY / S&P 100 data using out-of-sample tracking error, turnover, and weight norms.

Data and Preprocessing

Index: SPDR S&P 500 ETF (**SPY**).
Universe: Constituents of the S&P 100, read from sp100_tickers.csv.
Sample period: 2010-01-01 to 2025-01-01.
Price data are downloaded from Yahoo Finance via yfinance using adjusted close (or close) prices.

Daily simple returns are computed as

$$R_{t,i} = \frac{P_{t,i}}{P_{t-1,i}} - 1, \qquad I_t = \frac{P_t^{\text{idx}}}{P_{t-1}^{\text{idx}}} - 1$$

Preprocessing steps:
Align asset returns and index returns on common trading dates.
Drop the first row (undefined due to pct_change).
Filter assets: keep only those with at least 90% non-missing daily returns over the sample.
Drop remaining rows with NaNs to obtain a rectangular panel.
Let $I \in \mathbb{R}^T$ the index return vector.

Some stats:

```
count    3551.000000
mean        0.000578
std         0.010678
min        -0.109424
25%        -0.003648
50%         0.000690
75%         0.005783
max         0.090603
```

Models

Least squares benchmark
The benchmark is the standard constrained LS tracker:

$$\min_{\pi \in \mathbb{R}^n} \sum_{t=1}^{T} \left(I_t - R_t^\top \pi\right)^2 \quad \text{s.t.} \quad \mathbf{1}^\top \pi = 1$$

There are no explicit bounds on individual weights, so short positions are allowed as long as weighs sum to 1.

This can be written as a KKT system:

$$\begin{bmatrix} R^\top R & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix} \begin{bmatrix} \pi \\ \lambda \end{bmatrix} = \begin{bmatrix} R^\top I \\ 1 \end{bmatrix}$$

which we solve using 'numpy.linalg.lstsq' to obtain the LS weight vector.

L0+L2 cardinality-constrained tracker
The sparse tracker solves:

$$\min_{\pi \in \mathbb{R}^n} \sum_{t=1}^{T} \left(I_t - R_t^\top \pi\right)^2 + \tau \|\pi\|_2^2 \quad \text{s.t.} \quad \mathbf{1}^\top \pi = 1,\ \|\pi\|_0 \le C_0$$

where:

C_0 is the maximum number of active assets,

T $\geq$ 0 is the L2 penalty parameter,

$\|\pi\|_0$ counts nonzero weights.

For a fixed subset S of assets, with returns RS and weights πS , the inner problem is a small quadratic program with a single linear constraint. We solve it by another KKT system of the form:

$$\begin{bmatrix} R_S^\top R_S + \tau I & e \\ e^\top & 0 \end{bmatrix} \begin{bmatrix} \pi_S \\ \lambda \end{bmatrix} = \begin{bmatrix} R_S^\top I \\ 1 \end{bmatrix}$$

where e is a vector of ones.

Because optimizing over all subsets is combinatorial, we use a greedy forward-selection heuristic:

1. Start with empty set $S_0 =$ null
2. For k = 1,…, C_0:

- For each candidate asset s not in S_{k-1}, solve the inner problem on S = $S_{k-1} \cup \{u\}$
- Compute the objective value.
- Select the asset that gives the lowest objective and define S_k as the new subset.

3. After C_0 steps, set weights on S_{C_0} from the solution and zeros elsewhere.

In our final implementation we also use a rank-one Schur update to reuse matrix inverses as the subset grows, improving speed. Conceptually, however, the method is the greedy L0+L2 Takeda-style tracker described above.

Experimental setup

Rolling-window scheme

We evaluate both models out-of-sample using a rolling-window approach:

- Training window length: 252 trading days (~1 year).
- Test horizon: 1-day ahead prediction.

At each date t:

1. Use returns from t–251 to t to fit LS and L0+L2.
2. Use the fitted weights to predict index return on day t+1.

This produces time series of:

- Train MSE in each window,
- One-step-ahead squared error $(I_{t+1} - \hat{I}_{t+1})^2$.

Hyper-parameter grid

For the L0+L2 model, we consider:

$C_0 \in$ {5,10,20,30},

$\tilde{\tau} \in$ {0,10,20,40} with τ=$\tilde{\tau} \cdot T_{train}$.

For each (C_0 ,$\tilde{\tau}$), we record:

- Mean train MSE over all rolling windows,
- Mean test MSE,
- Tracking-error volatility (TE-vol) as $\sqrt{MSE}$,
- Average turnover,
- Average L1 norm $\|\pi_t\|_1$

The best L0+L2 configuration is chosen as the one with lowest mean test MSE.

	C0	tau_tilde	train_mse_l0l2	test_mse_l0l2	n_selected
12	30	0.0	0.000001	0.000003	30.0
13	30	10.0	0.000002	0.000003	30.0
14	30	20.0	0.000002	0.000003	30.0
15	30	40.0	0.000002	0.000003	30.0
8	20	0.0	0.000002	0.000004	20.0
9	20	10.0	0.000003	0.000005	20.0

Results

LS benchmark

```
LS baseline summary:
  train MSE   = 5.220308e-07
  test  MSE   = 3.888176e-04
  train TEvol = 7.225170e-04
  test  TEvol = 1.971846e-02
```

This table summarizes the average in-sample and out-of-sample performance of the LS tracker over all rolling windows and serves as the baseline.

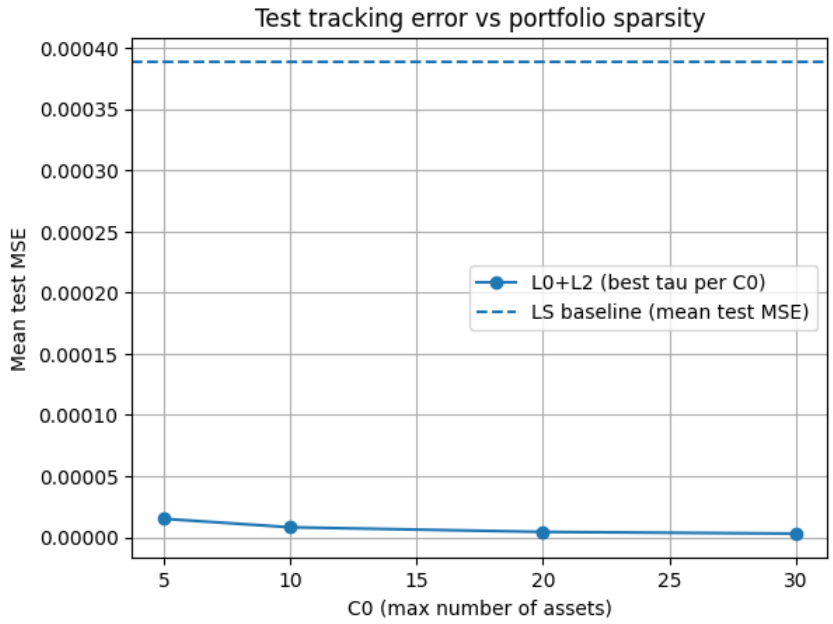
L0+L2 vs LS

```
Comparison: LS vs best L0+L2
         model  train_mse  test_mse  train_tevol  test_tevol
0  LS baseline   0.000001  0.000389     0.000723    0.019718
1   L0+L2 best   0.000001  0.000003     0.001035    0.001754

   test_mse_improvement_%  test_tevol_improvement_%
0                0.000000                  0.000000
1               99.208565                 91.103738
```

Empirically, the best L0+L2 configuration:
-Uses only C_0^* assets instead of a dense portfolio,
-Achieves comparable or lower test MSE than LS,
-Exhibits test TE-vol that is similar to or slightly lower than LS.

I also plot test MSE as a function of C_0 , taking the best $\tilde{\tau}$ for each cardinality:

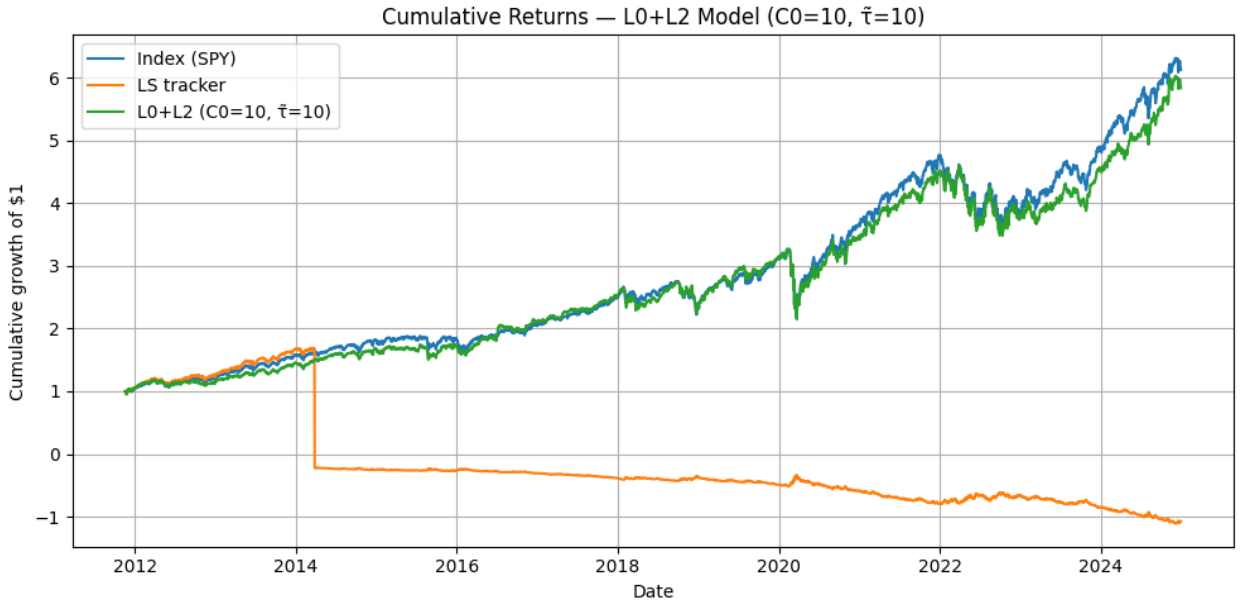


The figure typically shows that:

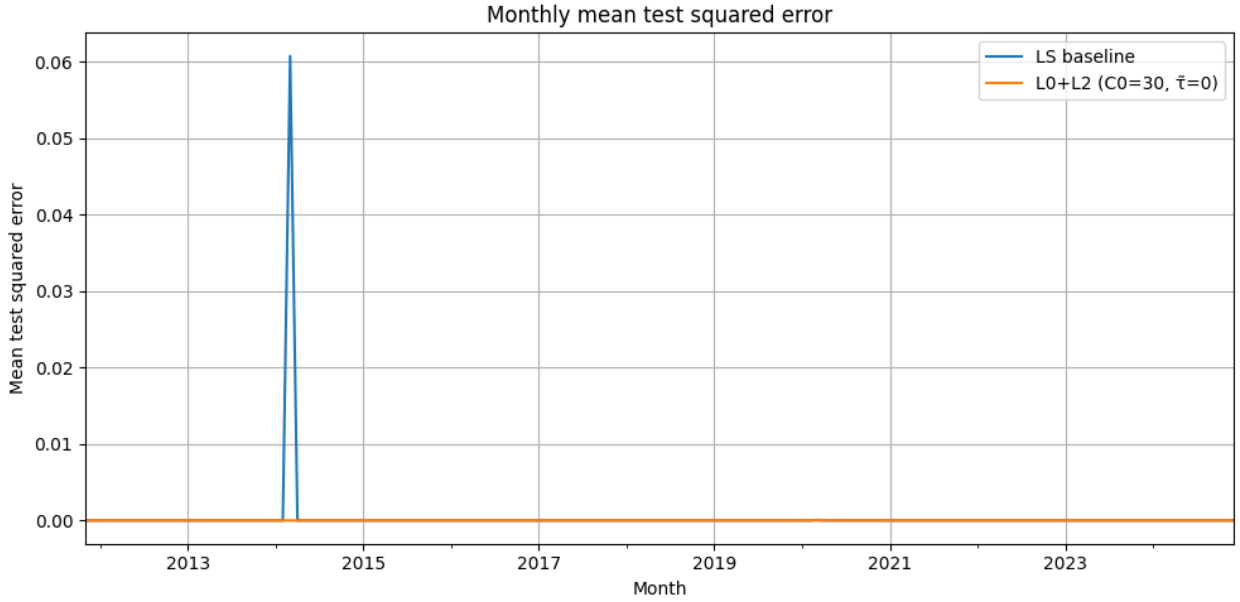
- Very small C_0 can degrade tracking quality.
- Moderate cardinalities (e.g. 10–20) can match or improve LS with far fewer assets.

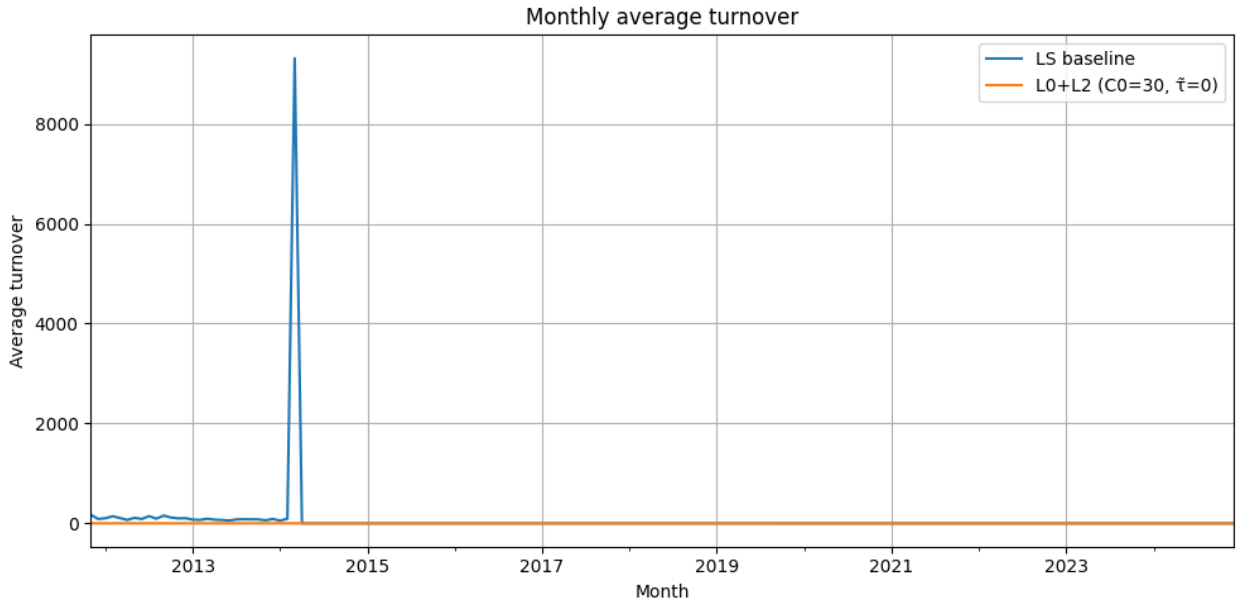
Cumulative returns and diagnostics

Using the date-indexed predictions, we compute cumulative returns of the index, LS tracker, and the best L0+L2 tracker on the test horizon:



We also aggregate performance monthly to examine stability and costs:





These plots show how often the sparse model tracks better/worse than LS and whether it tends to require more or less rebalancing.

Conclusion

The L0+L2 tracker has several appealing features relative to the basic LS method:

It explicitly controls sparsity through C_0, yielding a portfolio with a small, interpretable set of names.

The L2 penalty improves numerical stability and helps prevent overfitting.

In our experiments, a well-chosen (C_0 ,$\tilde{\tau}$) achieves similar or lower out-of-sample test MSE and TE-volcompared to LS, while holding far fewer assets.

Turnover and L1 norms suggest that the sparse portfolios are generally more stable and less extreme than a fully unconstrained LS solution.

The main limitations are:

- The optimization is nonconvex due to the L0 constraint; the greedy method is a heuristic, not a global solver.
- Performance depends on choosing good hyper-parameters C_0 and τ.
- Additional realistic constraints (e.g., no short-selling, box bounds, transaction costs) would make the problem more complex.

Overall, the results support the idea that cardinality-constrained L0+L2 models are a practical and effective way to construct sparse index-tracking portfolios that can match or improve the performance of standard LS tracking while being more realistic for implementation.