Index Tracking with Differentiable Asset Selection

Yu Zheng
Southwestern University of Finance
and Economics
zhengyu@swufe.edu.cn

Yunpeng Li
University of Surrey
yongxin.yang@surrey.ac.uk

Qiuhua Xu
Southwestern University of Finance
and Economics
xuqh@swufe.edu.cn

Timothy Hospedales
University of Edinburgh
t.hospedales@ed.ac.uk

Yongxin Yang*
University of Surrey
yongxin.yang@surrey.ac.uk

ABSTRACT

Partial index tracking aims to replicate the performance of a given benchmark index with a *small number* of its constituents. It can be formulated as a sparse regression problem, but remains challenging due to several practical constraints, especially the fixed number of assets in the portfolio. In this paper, we propose a differentiable relaxation for asset selection, such that we can construct a portfolio with exactly K assets, where the objective function can be optimised efficiently via vanilla gradient descent. Our method is backtested with S&P 500 index data from 2002 to 2020. Empirical results demonstrate that our model achieves excellent tracking performance compared with some widely used approaches.

CCS CONCEPTS

• Computing methodologies → Artificial intelligence.

KEYWORDS

Index tracking, portfolio optimisation

ACM Reference Format:

Yu Zheng, Yunpeng Li, Qiuhua Xu, Timothy Hospedales, and Yongxin Yang. 2020. Index Tracking with Differentiable Asset Selection. In *ACM International Conference on AI in Finance (ICAIF '20)*, October 15–16, 2020, New York, NY, USA. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3383455.3422516>

1 INTRODUCTION

Passive investing has attracted much more attention over the last decade. Products such as Exchange-Traded Funds (ETFs) have become more popular, as they are often more transparent and offer lower management fees [2, 3]. Index tracking is a popular passive investment strategy, which attempts to construct a portfolio that replicates the performance of a given benchmark index. The most straightforward way is to hold the full constituents of the benchmark index in exactly the same proportions. This is referred to as full replication. However, full replication has a few disadvantages

[6, 35]: (i) holding hundreds of assets (e.g., 500+ for S&P 500) results in high administrative and transaction costs (ii) when communicating with potential clients, it is less comprehensible compared to a product with a small amount of, say 50, assets. These issues motivate the problem of replicating a benchmark index using a subset of its constituents. This task is known as partial replication, and is the focus of this work.

Partial replication approaches have been well studied in the financial literature [4, 8, 9, 38]. This problem can be formulated as a constrained regression problem: minimising the tracking error of the constructed portfolio against the benchmark index, where the portfolio is a weighted sum of index constituents, and the weight comes from the regression weight vector. The constraints reflect business requirements, for example, non-negativity on the weight vector corresponds to a long only strategy. The most challenging constraint is the cardinality, which caps the number of assets to hold. It turns out that the cardinality constraint is hard to handle when another two constraints, namely non-negativity and sum-to-one, are also applied, because the efficient ℓ_1 regularised regression (LASSO) is no longer feasible (ℓ_1 norm is always one in this case). Alternative methods are proposed due to the fundamental non-negativity and sum-to-one constraints as solutions that can not meet them are impractical in industry. Representative approaches to constrained regression include fractional norm [15], mix-integer programming formulation [30], and (meta-) heuristics such as evolutionary algorithms [33].

In this paper, we propose a novel approach to partial index tracking, which can select *exactly* K assets while meeting non-negativity and sum-to-one constraints. We emphasise that there is a great practical need to fix the number of assets, as it eases product design. The key ingredient of our approach is a differentiable asset selection module, which builds on recent developments of differentiable operators (e.g., sort, rank, top_k) from the optimal transport community [27, 36]. The final objective in our model can be optimised by vanilla gradient descent efficiently, and it scales well when the number of features/assets increases.

The paper is organised as follows. Sec. 2 briefly reviews the literature in index tracking and differentiable operators. Sec. 3 is the core methodology of this work. In Sec. 4, we evaluate the proposed method and some baselines using both synthetic data and real S&P500 data. Sec. 5 summarises the paper and points out some future directions.

*Corresponding author.

2 RELATED WORK

2.1 Index Tracking

Partial index tracking can be divided into two sub-problems: (i) asset selection: which assets to hold in portfolio; (ii) capital allocation: what proportion of capital to allocate to each of the selected assets. Previous studies can be roughly grouped according to whether these two problems are solved sequentially or jointly.

The first group addresses these problems in a sequential manner. The first stage is to select a small number of assets from the index constituents according to some measures, e.g., the time-series similarity to the benchmark index [14, 16], replicating other factors beyond prices/returns [11], co-integration that exploits a long-run equilibrium relationship to the benchmark index [1]. In the second stage, index tracking is usually formulated as a constrained regression problem, where the weights of portfolio are determined by the optimal weight vector for the regression problem. This group of methods is usually easy to implement, as the asset selection is performed separately to tracking, and it is easy to control the number of assets selected by the first stage since they are usually about ranking according to certain statistics. The second stage only involves non-negativity and sum-to-one constraints, which can be solved by quadratic programming easily. However, the downside is also clear: since asset selection and capital allocation are carried out separately, there is no mechanism to ensure that the first stage helps the second stage. It is possible that, even if we selected assets that are similar to benchmark index, the regression problem would still under-fit when other constraints involved.

The second group considers asset selection and capital allocation jointly, and this is usually realised by enforcing sparsity on the weight vector w of a regression problem. As mentioned earlier, ℓ_1 regularisation is not applicable due to the presence of the non-negativity and sum-to-one constraints. Thus we have to either handle the cardinality constraint directly, i.e., through ℓ_0 regularisation [6], or seek for the help of a fractional norm [15]. However, either ℓ_0 or ℓ_p (where $0 < p < 1$) pose serious challenges in terms of optimisation. Even worse, if we want a fixed number of assets, e.g., $\|w\|_0 = K$, the strength of regularisation has to be tuned such that $\|w\|_0 = K$ is met. In fact, this is non-trivial, as increasing the strength of regularisation by a small margin may not necessarily reduce the value of $\|w\|_0$ due to the fact that the problem itself is heavily non-smooth. Alternatively, we can employ meta-heuristic and heuristics approaches, such as genetic algorithms [17, 23, 25], Tabu search [17], and simulated annealing [9, 38]. However, this family of methods usually requires a long computational time, esp. when the number of all candidate assets is large. Finally, one can choose to introduce some auxiliary binary variables to explicitly select assets and form a mixed-integer programming problem [8, 30, 37], but this needs commercial optimisation solvers, such as CPLEX, MOSEK, XPRESS, or GUROBI.

2.2 Differentiable Operators

The proposed method is similar to [10], which approximates the discontinuous counting function by a differentiable surrogate. In our model, we approximate the TopK function instead. Recently, differentiable approximations to certain discontinuous functions, such as sorting [13, 18], ranking [7, 13], and TopK [22, 39], have

become topical in machine learning, especially the deep learning community. Deep learning models are usually trained by gradient-based optimisers, which means all modules in the model must be differentiable. However, some elementary functions such as sorting are non-differentiable, which stops them from being integrated into an end-to-end deep learning pipeline. Some successful attempts have been made to find differentiable proxies for those functions, and proven to improve the model performance, e.g., differentiable k -nearest neighbours (k NN) outperforms vanilla k NN [28]. The underlying techniques usually originate from operations research (OR), such as optimal transportation problems. In this work, we aim to provide a self-contained explanation and focus on the application of differentiable TopK to index tracking. For a more comprehensive reviews, refer to [27, 29].

3 METHODOLOGY

3.1 Notation

Assume that we have the returns of candidate assets in a matrix $X \in \mathbb{R}^{N \times D}$ where each column is an asset, and each row refers to a timestep. The returns of benchmark index is denoted as $y \in \mathbb{R}^N$. The vector of asset weights is $w \in \mathbb{R}^D$. $\|\cdot\|_0$ stands for the ℓ_0 norm for a vector, which is the number of non-zero elements in that vector. $\text{diag}(\cdot)$ constructs a diagonal matrix from the input vector. $\lceil \cdot \rceil$ converts floating point numbers to integer by rounding up. Unless explicitly stated otherwise, vectors refer to column vectors throughout the paper.

3.2 Problem Setting

With the above notions, the simplest form of index tracking is the least-squared regression,

$$\min_w \|Xw - y\|_2^2 \quad (1)$$

In practice, there are some constraints on w , namely,

Non-negativity Requiring a long-only strategy corresponds to the constraint $w_i \geq 0, \forall i$

Sum-to-one Requiring full utilization of capital requires $\sum_i w_i = 1$

Thus, Eq. 1 is modified to be a constrained regression problem,

$$\min_{w \geq 0, \sum_i w_i = 1} \|Xw - y\|_2^2 \quad (2)$$

Eq. 2 can be handled by off-the-shelf quadratic programming (QP) solvers, as it has a standard QP form.

$$\begin{aligned} \min_w \quad & \frac{1}{2} w^T P w + q^T w \\ \text{subject to:} \quad & Gw \leq h \text{ and } Aw = b \end{aligned} \quad (3)$$

where $P = 2(X^T X)$, $q = -2X^T Y$, $G = -I$, $h = \mathbf{0}$, $A = \mathbf{1}^T$, and $b = 1$.

To achieve *partial* index tracking – and thus define a portfolio that is easy to maintain and has low transaction cost – a sparse solution is required. However, one quickly finds that it is very challenging to enforce sparsity in the solution of w , because the commonly used ℓ_1 regularisation (i.e., LASSO regression) can not be applied here, since $\|w\|_1$ is always one. Furthermore, in practice the number of assets (i.e., the number of non-zero elements in w)

must often be set to a fixed number K . To meet this constraint, the optimisation problem becomes

$$\min_{w \geq 0, \sum_i w_i = 1, \|w\|_0 = K} \|Xw - y\|_2^2 \quad (4)$$

Eq. 4 is NP-hard because of its combinatorial nature [24]. The main contribution of this work is a reformulation of Eq. 4 such that it can be optimised by vanilla gradient descent, and the key concept is *differentiable asset selection*, which we detail in the next section.

3.3 Differentiable Asset Selection

First, we introduce a re-parametrisation of w ,

$$w_i = \frac{1}{\sum_i \exp(\tilde{w}_i) z_i} \exp(\tilde{w}_i) z_i, \quad z_i \in \{0, 1\} \quad (5)$$

Then we can easily verify that all three constraints in Eq. 4 are met for any $\tilde{w}_i \in \mathbb{R}$ as long as $\sum_i z_i = K$. The binary vector $z = [z_1, z_2, \dots, z_D]$ defines our asset selection. Now we can define the following optimisation problem with the re-parametrised w as a function of $\tilde{w} \in \mathbb{R}^D$ and $z \in \{0, 1\}^D$.

$$\min_{\tilde{w}, z} \|Xw - y\|_2^2 \quad \text{s.t.} \quad \sum_i z_i = K \quad (6)$$

Eq. 6 is sometimes referred to as mixed integer quadratic programming (MIQP). Instead of solving MIQP by some commercial solvers, we propose an alternative approach that relaxes the discontinuity caused by z .

The key idea is to re-parametrise z as,

$$z = \text{TopK}(s), \quad s \in \mathbb{R}^D \quad (7)$$

where $\text{TopK}(\cdot)$ is an operator that maps the K largest entities in the vector to 1 and discards the remaining as 0. The technical challenge will be how to implement $\text{TopK}(\cdot)$ differentiably, and we introduce how to achieve this using Optimal Transport next.

3.3.1 Introducing Differentiable TopK by Optimal Transport. Consider the following toy example,

$$\begin{aligned} s &= [-0.5, \quad 1.7, \quad 0.3, \quad 0.8, \quad 1.1] \\ z = \text{Top3}(s) &= [0, \quad 1, \quad 0, \quad 1, \quad 1] \end{aligned}$$

where the Top3 result is because the values $\{1.7, 0.8, 1.1\}$ are the largest entities in s . Such a TopK operator can be easily implemented using a sorting algorithm such as Quicksort [20], but classic sorting algorithms are usually non-differentiable.

To implement a *differentiable* TopK operator, we will formulate it as an optimal transport (OT) problem [7, 13, 29, 39] and solve the OT problem using Sinkhorn-Knopp's fixed point iteration [12, 21, 32].

An Example: Following the canonical optimal transport terminology of dirt transport: In the above example, we have 5 (derived from D) units of dirt in a 1-dimensional space, and each unit's location is specified by s . We also have two pits at $[t_1, t_2]$ where $t_1 < t_2$, e.g., $[0, 1]$, to which we can move the dirt. The capacity of these pits is 2 and 3 respectively (derived from $[D - K, K]$). The objective is to move all dirt to the pits with the smallest total cost. As illustrated in Fig. 1, the solution is simple: we move the dirt located at $\{-0.5, 0.3\}$ to 0 and the dirt located $\{1.7, 0.8, 1.1\}$ to 1.

Figure 1: An illustrative example of OT. Here we have 5 units of dirt located at $[-0.5, 0.3, 0.8, 1.1, 1.7]$. Two pits with capacity 2 and 3 are located at $[0, 1]$. The optimal transport is illustrated by grey arrows.

TopK via OT: Formally, we formulate the TopK operator as the solution to the following optimal transport problem,

$$\begin{aligned} \text{TopK}(s) &= P \cdot [0, 1] \\ \text{where } P &= \underset{P \in \{0,1\}^{D \times 2}}{\text{argmin}} \sum_{i=1}^D \sum_{j=1}^2 P_{ij} (s_i - t_j)^2 \\ \text{s.t. } P \cdot [1, 1] &= \underbrace{[1, 1, \dots, 1]}_D \\ P^T \cdot \underbrace{[1, 1, \dots, 1]}_D &= [D - K, K] \end{aligned} \quad (8)$$

where P specifies the movement plan, and its second column (obtained by $P \cdot [0, 1]$) indicates what move to t_2 .

We can prove that the solution of the above optimal transportation is the TopK operator by contradiction. First, we sort s such that

$$s_{[1]} < s_{[2]} < \dots < s_{[D-K]} < s_{[D-K+1]} < \dots < s_{[D]} \quad (9)$$

where $s_{[i]}$ is the i -th smallest element in s . If the TopK operator matches with optimal transportation, it suggests that we move $\{s_{[1]}, s_{[2]}, \dots, s_{[D-K]}\}$ to t_1 and $\{s_{[D-K+1]}, s_{[D-K+2]}, \dots, s_{[D]}\}$ to t_2 . We can tell that any perturbation to it will lead to a sub-optimal solution, because if we swap $s_{[i]}$ and $s_{[j]}$, where $s_{[i]} \in \{s_{[1]}, s_{[2]}, \dots, s_{[D-K]}\}$ and $s_{[j]} \in \{s_{[D-K+1]}, s_{[D-K+2]}, \dots, s_{[D]}\}$, the relative cost will become $(s_{[j]} - t_1)^2 + (s_{[i]} - t_2)^2$ instead of $(s_{[i]} - t_1)^2 + (s_{[j]} - t_2)^2$, and the difference is always positive.

$$\begin{aligned} &((s_{[j]} - t_1)^2 + (s_{[i]} - t_2)^2) - ((s_{[i]} - t_1)^2 + (s_{[j]} - t_2)^2) \\ &= -2s_{[j]}t_1 - 2s_{[i]}t_2 + 2s_{[i]}t_1 + 2s_{[j]}t_2 \\ &= 2(s_{[j]} - s_{[i]})(t_2 - t_1) > 0 \end{aligned} \quad (10)$$

A Continuous Relaxation: Note that Eq. 8 is still a discrete optimisation problem, but we can obtain a smoothed version by (1) relaxing P_{ij} from $\{0, 1\}$ to $[0, 1]$ (2) introducing an entropic regularisation to encourage a high-entropy solution, e.g., $P_i = [0.9, 0.1]$ is preferred to $P_i = [0.99, 0.01]$.

The OT problem with entropic regularisation is,

$$P = \underset{P \in [0,1]^{D \times 2}}{\text{argmin}} \sum_{i=1}^D \sum_{j=1}^2 P_{ij} (s_i - t_j)^2 - \frac{1}{\lambda} H(P_{ij}) \quad (11)$$

subject to the same constraints. Here $H(P_{ij}) = -P_{ij} \log(P_{ij})$ is the entropy term. λ controls the trade-off between the smoothness and the optimisation efficiency. Large λ leads to nearly binary valued solutions, but it also takes longer running time to converge because the gradients tend to zero. On the other hand, small λ eases the

Figure 2: Visualisation of P matrices with different λ values. $s = [-0.5, 1.7, 0.3, 0.8, 1.1]$, $t = [0, 1]$, and $K = 3$

optimisation, but overly smooth solution is less desired since it can not tell us which assets to select confidently. The effect of different λ values for our illustrative toy problem above is illustrated in Fig. 2.

The regularised OT problem in Eq. 11 can be solved via Sinkhorn-Knopp’s fixed point iteration, as summarised in Alg. 1.

Algorithm 1 Sinkhorn-Knopp’s fixed point iteration

Require: s, t, λ, K
 $D \leftarrow \text{length}(s)$
 $C_{ij} \leftarrow |s_i - t_j|$
 $P_{ij} \leftarrow \exp(-\lambda C_{ij}^2)$
 $a \leftarrow [1, 1, \dots, 1]$
 $b \leftarrow [D - K, K]$
repeat
 $P \leftarrow P \cdot \text{diag}(\frac{1}{P \cdot a})$ ▷ row-wise normalisation
 $P \leftarrow \text{diag}(\frac{1}{P \cdot b}) \cdot P$ ▷ column-wise normalisation
until P does not change
 $P \leftarrow \text{diag}(a) \cdot P \cdot \text{diag}(b)$
return P

The complexity of the algorithm above is $O(DM)$ where M is the number of iterations needed for convergence, and it is usually higher than the sorting based solution at cost of $O(D \log D)$. However, it has the crucial property that no non-differentiable operations are used in the whole process. This means that we can use automatic differentiation to compute the gradient of z w.r.t. s .

3.4 Final Objective Function

With the derivation above, we can write down the final objective function for differentiable partial index tracking,

$$\begin{aligned} \min_{\tilde{w}, s} \|X\tilde{w} - y\|_2^2 \\ w_i = \frac{1}{\sum_i \exp(\tilde{w}_i) z_i} \exp(\tilde{w}_i) z_i \\ z_i = [\text{TopK}(s)]_i \end{aligned} \quad (12)$$

The variables to optimise, i.e., $\tilde{w}$ and s , are unbounded, and the objective is fully differentiable, thus we can apply any first-order method for optimisation, such as gradient descent.

The final outstanding problem is that z vector is not truly binary since we employ a smoothed version of TopK operator. Here we

discuss two options to alleviate this problem: (i) Straight-through gradient estimator [5], i.e., we use the smoothed TopK operator to compute z , and we binarise it by $z \leftarrow z > z_{[D-K]}$ in the forward pass ($z_{[D-K]}$ is the $(D-K)$ -th smallest element in z), but we still use the ‘soft’ z in the backward pass. Note that this is basically to set the gradient of binarisation operation as one; (ii) When optimisation is complete, retrieve the selected assets by sorting s , and run the QP version (Eq. 3) with reduced X where only selected assets are present. In fact, these two options are orthogonal, so we employ both in this work.

4 EXPERIMENTS

4.1 Toy Problem

We first evaluate our method in a toy problem built as follows: (1) we randomly generate K assets’ log returns from the standard Gaussian distribution. Each asset’s log return is an N -dimensional column vector, and stacking them into a matrix we get an $N \times K$ matrix $X_0 = [x_1, x_2, \dots, x_K]$. (2) we obtain the return of the benchmark index, denoted as y , by taking the column-wise average of X . We can immediately find that, if we were to do a regression for (X_0, y) , the perfect solution would be $[\frac{1}{K}, \frac{1}{K}, \dots, \frac{1}{K}]$. (3) we duplicate the i -th column of X_0 (i.e., x_i) R_i times, where R_i is a random integer drawn from $[\lceil \frac{D}{2K} \rceil, \lceil \frac{2D}{K} \rceil]$ uniformly, and add them into a new matrix X . This results in a matrix sized $N \times (R_1 + R_2 + \dots + R_K)$. (4) Finally, we add some random noise to both X and y .

4.1.1 Performance and Analysis. We can tell that the solution to the regression problem for (X, y) should be very sparse, and the perfect w should contain exactly K non-zero elements, and their values should be equal to $\frac{1}{K}$.

Not surprisingly, we do not get a sparse solution if we do not enforce sparsity. For example, if we run the QP solver (Eq. 3) for this toy problem, we will get good yet dense w vector (Fig. 3 Left). In practice, this leads to higher transaction costs.

In contrast, both the MIQP solver (Eq. 6) and our method (Eq. 12) recover the sparse solution with exactly K assets when the ground truth K is given (Fig. 3 Middle and Right). Note that, since x_i ’s in X_0 are repeated many times, choosing any of the available duplicates is equally good. The copies of a certain column of X_0 are coloured the same in Fig. 3.

4.1.2 Running Time. Since both the MIQP solver and the proposed method can solve this toy problem perfectly, we are interested in their efficiency, as measured by running time. We start from the smallest problem by setting $K = 10$, $N = 1000$, and $D = 500$, and progressively increase (K, N, D) by a complexity multiplier α_i , where $\alpha = [1.0, 1.25, 1.5, 1.75, 2.0]$.

We measure the wall clock time from building the optimisation problem to finding a solution, using the same stopping criteria: for 100 consecutive iterations, the standard deviation of evaluations of the objective function is smaller than 0.01. Empirically, we found this criteria guarantees that toy problem is solved for all cases in α . The proposed method is implemented using PyTorch [26], and we deactivate GPU acceleration for fair comparison. For the same multiplier, we run the algorithms 10 times.

Figure 3: Toy problem. Different colours correspond to different columns (assets) in X_0 . $K = 10$, $N = 1000$, and $D = 500$. Left to right: QP solution (Eq. 3), MIQP solution (Eq. 6), and the proposed method. While QP produces a sub-optimal dense solution, both the MIQP solver and our method recover the correct sparse solution with exactly $K = 10$ equally weighted assets.

Figure 4: Running time of the MIQP solver and the proposed method for the toy problem. Y-axis is log time in seconds. X-axis is the problem size multiplier for (K, N, D) . For every multiplier, we present a boxplot over 10 runs.

As we can see in Fig. 4, the running time of the MIQP solver grows exponentially w.r.t. the complexity multiplier, but the proposed method grows (almost) linearly, which coincides our analysis about complexity in Sec. 3.3.

4.2 S&P 500 Tracking

To evaluate the proposed method for a real problem, we track the S&P500 index using its own constituents.

4.2.1 Baselines. We will compare the proposed method with three baselines that are commonly used in practice.

- (1) **Forward Selection:** We fit the regression problem with the non-negativity and sum-to-one constraint using QP solver (i.e., Eq. 3), then select one stock with the largest weight value w_i , and record it in a selected stock list. In the next round of fitting, we train the model without the selected stock. We do this repeatedly until the size of the selected stock list reaches K . Finally, we fit the constrained regression problem using QP with these K stocks only.
- (2) **Backward Selection:** It works similarly as forward selection above, but we get rid of one stock with the smallest w_i value, and re-fit the model in each round until the number of the remaining stocks is K .
- (3) **MIQP** We use a commercial solver, namely, Gurobi [19], to solve the MIQP problem in Eq. 6, and this results in a solution that meets all three constraints including $\|w\|_0 = K$. It may use a branch-and-bound search method internally.

4.2.2 Data Preparation. The pricing data is obtained from the Center for Research in Security Prices (CRSP), a well recognised

provider for high-quality historical market data. We use the adjusted daily closing prices (for dividends, splits etc).

To simulate the real trading system, we also take into account transaction fees, for which we choose a flat-fee pricing model, \$5.00 per trade, quoted by TradeStation, a popular online stock brokerage firm in the US.

4.2.3 Rebalancing. More frequent rebalancing usually reduces tracking error, but it also results in higher transaction fees. As a passive investment strategy, we choose to rebalance the portfolio quarterly, and apply this to all compared methods.

The evaluation period is from 2002-01-01 to 2020-01-01, which includes some interesting events like 2008 global financial crisis. Backtesting is performed with a standard sliding window model: by the end of each quarter, e.g., on 2002-03-31, we rebalance the portfolio w.r.t. the weights produced by the regression model for three-year historical data, i.e., 1999-03-31 to 2002-03-31, which means we buy the stocks suggested by the model, and then hold them, until the next rebalance day, i.e., 2002-06-31. This procedure is carried out quarterly until the final rebalance day 2019-09-31, which is used to test for 2019-10-01 to 2020-01-01. Note that, we employ an industry standard simulator for backtesting, where it accounts for all factors such as trading costs, stock splits, dividends, and avoids survival bias and information leakage by rigorous data processes.

4.2.4 Performance metric. The objective is to track the index as accurately as possible for all methods. Thus, the most direct performance measure is (out-of-sample) tracking error between the benchmark index, i.e., S&P500 index and the constructed portfolio. Given that we use log returns, the absolute or squared error is usually at the scale of 10^{-3} or 10^{-6} , we report percentage error instead.

Apart from tracking error, we calculate six statistics of interest: (i) annual return, (ii) annual volatility (iii) max drawdown (iv) Sharpe ratio [31] (v) Sortino ratio [34] (vi) Calmar ratio [40]. They are defined as follows:

Annual return also known as annualised return, defined as $\left(\frac{\text{Ending Value}}{\text{Beginning Value}}\right)^{\frac{1}{\text{Years}}} - 1$.

Annual volatility is computed by $\sigma_d \sqrt{252}$ where σ_d is the daily standard deviation (of log returns) and 252 is the number trading days every year.

Max Drawdown is the maximum observed loss from a peak to a trough, which can be calculated by $\frac{\text{Trough Value} - \text{Peak Value}}{\text{Peak Value}}$.

Figure 5: Equity curve for the index and trackers. Rows from top to bottom compare different partial replication trackers: Forward-selection, Backward-selection, MIQP, Proposed method. Columns evaluate different portfolio sizes $K \in \{30, 40, 50\}$ from left to right. In every subplot, the top section (two curves) shows the index and the tracker, and bottom section shows the percentage tracking error $\frac{\hat{y}-y}{y}$.

Sharpe ratio is a risk-adjusted measure, defined by the ratio of excess return (portfolio return subtracts risk-free rate) and the standard deviation of excess return.

Sortino ratio is similar to Sharpe ratio, but it is return divided by downside standard deviation rather than the entire standard deviation (upside and downside).

Calmar ratio is similar to Sharpe ratio as well, but it is return divided by max drawdown.

From the perspective of measuring the performance of a given portfolio, those statistics have their own properties, e.g., we prefer higher Sharpe ratio, lower max drawdown, etc. However, as our main goal is to track the benchmark index, they are less important.

E.g., if the benchmark index is of high volatility, we prefer a more volatile portfolio that tracks the index accurately, rather than a less volatile portfolio that fails in terms of the tracking error. We expect an index tracking model to match these statistics of the target index. But where a given level of mismatch exists, we would rather see it move in the ‘good’ way according to these statistics.

4.2.5 Performance and Analysis. The optimisation problem itself is heavily non-convex. Thus the MIQP solver and our proposed method exhibit some randomness in the results due to initialisation. Instead of fixing a random seed, we choose to run each experiment

Figure 6: Portfolio statistics with various methods (colour indicators match Fig. 5). First row: Annual return, annual volatility, max drawdown. Second row: Sharpe ratio, Sortino ratio, Calmar ratio. In each plot x-axis spans various portfolio size K values. Note that the plot is based on normalised error: $\frac{s - s_I}{|s_I|}$ where s_I is the statistics of benchmark index. F, B, M, P stands for Forward selection, Backward selection, MIQP, Proposed method respectively.

10 times and report the average. A more rigorous study for our method can be found in Sec. 4.2.6

Fig. 5 shows all models' performance with different choice of $K \in \{30, 40, 50\}$, from which we can tell: (i) The proposed method has the lowest tracking error for $K = 30$ and $K = 40$. Forward select performs best when $K = 50$. Note that, all methods have the same transaction costs since the number of traded stocks the same, i.e., K . Thus, the difference can be only explained by the contribution of model itself. (ii) Even though the proposed method has a bigger tracking error when $K = 50$, it is biased positively (outperforming the index). (iii) The tracking performance and the choice of K are not strongly correlated, and this might be caused by the narrow range of K .

In terms of other statistics, the proposed method has comparable performance to the alternatives (Fig. 6). For annual volatility and max drawdown, forward selection and backward selection show some advantages respectively, i.e., smaller mismatch and preferred direction. However, for the risk-adjusted measures, which are more meaningful in practice, our method outperforms others.

4.2.6 Sensitivity to random initialisations. Since our method is non-deterministic, we study the consistency of its performance by running it 100 times with different initialisations. The out-of-sample performance with $K = 50$ in Fig. 7 demonstrates that the good tracking performance is consistent, with the small errors caused by outperforming the target index.

5 CONCLUSION

We present a novel approach for partial index tracking where the number of assets is fixed. It is derived from a re-parametrisation of weight vector for capital allocation and binary masks for asset selection. By employing a differentiable solver for the TopK operator, the proposed model can be trained efficiently by gradient descent. Our method is easy to implement with modern deep

Figure 7: Sensitivity to initialisation. Out-of-sample performance of 100 runs when $K = 50$. Orange line is the mean and the shadow area covers $1 \times$ standard deviation. This shows that the proposed method is consistently effective. Note that the errors are accumulated so the shadow area becomes larger as time progresses.

learning frameworks such as PyTorch, TensorFlow, and provides comparable or even better performance than some commercial MIQP solvers, while scaling much better in terms of computation cost. For future work, we will explore more applications such as enhanced index tracking, and study the generation of the auxiliary variable.

Acknowledgement: This work has been supported by the Financial Innovation Center of the Southwestern University of Finance and Economics and the Key Laboratory of Financial Intelligence and Financial Engineering of Sichuan Province.

Disclaimer: All authors are faculty. Neither graduate students nor small animals were hurt while producing this paper.

REFERENCES

- [1] Carol Alexander and Anca Dimitriu. 2005. Indexing and statistical arbitrage. *The Journal of Portfolio Management* 31, 2 (2005), 50–63.
- [2] Kenechukwu Anadu, Mathias Krutli, Patrick McCabe, Emilio Osambela, and Chae Hee Shin. 2018. The shift from active to passive investing: potential risks to financial stability? Federal Reserve Bank of Boston Working Paper.
- [3] Brad Barber and Terrance Odean. 2000. Trading is hazardous to your wealth: the common stock investment performance of individual investors. *Journal of Finance* 55, 2 (2000), 773–806.
- [4] John E Beasley, Nigel Meade, and T-J Chang. 2003. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research* 148, 3 (2003), 621–643.
- [5] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR abs/1308.3432* (2013).
- [6] K. Benidis, Y. Feng, and D. P. Palomar. 2018. Sparse Portfolios for High-Dimensional Financial Index Tracking. *IEEE Transactions on Signal Processing* 66, 1 (2018), 155–170.
- [7] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. 2020. Fast Differentiable Sorting and Ranking. [arXiv:stat.ML/2002.08871](https://arxiv.org/abs/2002.08871)
- [8] N.A. Canakgoz and J.E. Beasley. 2009. Mixed-integer programming approaches for index tracking and enhanced indexation. *European Journal of Operational Research* 196, 1 (2009), 384–399.
- [9] T-J Chang, Nigel Meade, John E Beasley, and Yazid M Sharaiha. 2000. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research* 27, 13 (2000), 1271–1302.
- [10] Thomas F Coleman, Yuying Li, and Jay Henniger. 2006. Minimizing tracking error while restricting the number of assets. *Journal of Risk* 8, 4 (2006), 33.
- [11] Francesco Corielli and Massimiliano Marcellino. 2006. Factor based index tracking. *Journal of Banking & Finance* 30, 8 (2006), 2215–2233.
- [12] Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NIPS*.
- [13] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. 2019. Differentiable Ranking and Sorting using Optimal Transport. In *NIPS*.
- [14] Christian Dose and Silvano Cincotti. 2005. Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications* 355, 1 (2005), 145–151.
- [15] Bjorn Fastrich, Sandra Paterlini, and Peter Winker. 2014. Cardinality versus-q-norm constraints for index tracking. *Quantitative Finance* 14, 11 (2014), 2019–2032.
- [16] Sergio M Focardi and Frank J Fabozzi. 2004. A methodology for index tracking based on time-series clustering. *Quantitative Finance* 4 (2004), 417–425.
- [17] Fernando García, Francisco Guijarro, and Javier Oliver. 2017. Index tracking optimization with cardinality constraint: a performance comparison of genetic algorithms and tabu search heuristics. *Neural Computing and Applications* (2017).
- [18] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *ICLR*.
- [19] LLC Gurobi Optimization. 2020. Gurobi Optimizer Reference Manual. <http://www.gurobi.com>
- [20] C. A. R. Hoare. 1962. Quicksort. *Comput. J.* 5, 1 (1962), 10–16.
- [21] Philip A. Knight. 2008. The Sinkhorn–Knopp Algorithm: Convergence and Applications. *SIAM J. Matrix Anal. Appl.* 30, 1 (2008), 261–275.
- [22] Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. In *ICML*.
- [23] Qian Li, Linyan Sun, and Liang Bao. 2011. Enhanced index tracking based on multi-objective immune algorithm. *Expert Systems with Applications* 38, 5 (2011), 6101–6106.
- [24] Y. Nesterov and A. Nemirovskii. 1994. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics.
- [25] He Ni and Yongqiao Wang. 2013. Stock index tracking by Pareto efficient genetic algorithm. *Applied Soft Computing* 13, 12 (2013), 4519–4535.
- [26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NIPS*.
- [27] Gabriel Peyré and Marco Cuturi. 2019. Computational Optimal Transport. *Foundations and Trends® in Machine Learning* 11, 5–6 (2019), 355–607.
- [28] Tobias Plötz and Stefan Roth. 2018. Neural Nearest Neighbors Networks. In *NIPS*.
- [29] F. Santambrogio. 2015. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Springer International Publishing.
- [30] Andrea Scozzari, Fabio Tardella, Sandra Paterlini, and Thiemo Krink. 2013. Exact and heuristic approaches for the index tracking problem with UCITS constraints. *Annals of Operations Research* 205 (2013), 235–250.
- [31] William F. Sharpe. 1994. The Sharpe Ratio. *The Journal of Portfolio Management* 21, 1 (1994), 49–58.
- [32] Richard Sinkhorn and Paul Knopp. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.* 21, 2 (1967), 343–348.
- [33] Amparo Soler-Dominguez, Angel A. Juan, and Renatas Kizys. 2017. A Survey on Financial Applications of Metaheuristics. *Comput. Surveys* 50, 1 (2017).
- [34] Frank A. Sortino and Lee N. Price. 1994. Performance Measurement in a Downside Risk Framework. *The Journal of Investing* 3, 3 (1994), 59–64.
- [35] O. Strub and P. Baumann. 2018. Optimal construction and rebalancing of index-tracking portfolios. *European Journal of Operational Research* 264, 1 (2018), 370–387.
- [36] C. Villani. 2008. *Optimal Transport: old and new*. Vol. 338. Berlin, Germany: Springer Science & Business Media.
- [37] Meihua Wang, Chengxian Xu, Fengmin Xu, and Hongang Xue. 2012. A mixed 0–1 LP for index tracking problem with CVaR risk constraints. *Annals of Operations Research* 196, 1 (2012), 591–609.
- [38] Maria Woodside-Oriakhi, C Lucas, and John E Beasley. 2011. Heuristic algorithms for the cardinality constrained efficient frontier. *European Journal of Operational Research* 213, 3 (2011), 538–550.
- [39] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. 2020. Differentiable Top-k Operator with Optimal Transport. [arXiv:cs.LG/2002.06504](https://arxiv.org/abs/2002.06504)
- [40] Terry W. Young. 1991. Calmar Ratio: A Smoother Tool. *Modern Trader* 20, 12 (1991).