Differentiable Multi-Period Portfolio Optimization (DMPO)

Firstname1 Lastname1 [* 1] **Firstname2 Lastname2** [* 1 2] **Firstname3 Lastname3** [2] **Firstname4 Lastname4** [3]
Firstname5 Lastname5 [1] **Firstname6 Lastname6** [3 1 2] **Firstname7 Lastname7** [2] **Firstname8 Lastname8** [3]
Firstname8 Lastname8 [1 2]

Abstract

This document provides a basic paper template and submission guidelines. Abstracts must be a single paragraph, ideally between 4–6 sentences long. Gross violations will trigger corrections at the camera-ready phase.

1. Introduction

Dynamic portfolio optimization presents a fundamental challenge in quantitative finance, requiring agents to balance expected returns against risks and transaction costs in a stochastic environment. While classical Mean-Variance frameworks (Markowitz, 1952) provide theoretical foundations, they often lack the flexibility to adapt to changing market regimes. Conversely, deep Reinforcement Learning (RL) has emerged as a powerful tool for learning complex, path-dependent strategies directly from data (Jiang et al., 2017). However, a critical limitation of standard model-free RL algorithms (e.g., PPO, DDPG) is their inability to strictly enforce hard constraints—such as turnover limits or cardinality constraints—during both exploration and deployment. Conventional approaches typically rely on reward shaping (soft penalties), which offers no guarantees of feasibility and often leads to suboptimal convergence due to conflicting objectives.

To bridge this gap, we introduce **Differentiable Multi-Period Portfolio Optimization (DMPO)**, a hybrid framework that integrates deep sequence modeling with differentiable convex optimization. Unlike standard RL agents that directly output portfolio weights, DMPO learns to generate a *market view* vector, which parameterizes a quadratic programming (QP) layer embedded within the policy network. By leveraging implicit differentiation through the Karush-Kuhn-Tucker (KKT) conditions, our method permits end-to-end training via gradient ascent while guaranteeing rigorous satisfaction of turnover constraints at every time step. Our extensive experiments demonstrate that DMPO achieves zero constraint violations and superior risk-adjusted returns compared to traditional benchmarks.

[*]Equal contribution [1]Department of XXX, University of YYY, Location, Country [2]Company Name, Location, Country [3]School of ZZZ, Institute of WWW, Location, Country. Correspondence to: Firstname1 Lastname1 <first1.last1@xxx.edu>, Firstname2 Lastname2 <first2.last2@www.uk>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

2. Methodology

We formulate the portfolio management problem as a constrained Markov Decision Process (CMDP) and propose a neural architecture that explicitly embeds a convex optimization solver as the final layer of the policy network.

2.1. Problem Formulation

Consider a financial market with N assets over discrete time steps $t = 0, \dots, T$. Let $p_t \in \mathbb{R}_+^N$ denote the asset prices and $r_t \in \mathbb{R}^N$ denote the vector of asset returns at time t. The portfolio state is represented by the allocation vector $w_t \in \Delta^N$, where $\Delta^N = \{w \in \mathbb{R}^N \mid \mathbf{1}^\top w = 1, w \succeq 0\}$ is the standard simplex.

We define the state space $\mathcal{S}$ as a tuple $s_t = (X_t, w_{t-1})$, where $X_t \in \mathbb{R}^{k \times N}$ represents the trailing window of historical market features (e.g., returns) of length k, and w_{t-1} is the portfolio weight vector from the previous step. The action a_t corresponds to the new portfolio weights w_t.

The agent aims to maximize the cumulative risk-adjusted return subject to a hard turnover constraint. The turnover $\mathcal{T}_t$ is defined as the L_1-norm of the rebalancing vector:

$$\mathcal{T}_t = \|w_t - w_{t-1}\|_1. \tag{1}$$

Crucially, we impose a strict limit δ on turnover (e.g., $\delta = 0.1$) to control transaction costs and market impact.

2.2. The DMPO Architecture

The core innovation of DMPO is the decoupling of *market view generation* (learned via Neural Networks) and *constraint enforcement* (solved via Convex Optimization). The policy π_θ consists of two distinct modules:

2.2.1. 1. View Generation Network

We employ a Long Short-Term Memory (LSTM) network to extract temporal dependencies from the market history X_t. The network outputs a latent representation, which is then projected to a "view" vector $\hat{\mu}_t \in \mathbb{R}^N$:

$$\begin{aligned} h_t &= \text{LSTM}(X_t; \theta_{\text{enc}}), && (2) \\ \hat{\mu}_t &= \text{Linear}(h_t \oplus \phi(w_{t-1}); \theta_{\text{head}}), && (3) \end{aligned}$$

where $\oplus$ denotes concatenation and $\phi(\cdot)$ is a feature embedding of the current position. Unlike traditional expected return vectors, $\hat{\mu}_t$ is an unconstrained latent parameter learned end-to-end to maximize the RL objective.

2.2.2. 2. Differentiable QP Layer

Instead of treating $\hat{\mu}_t$ as the action, we feed it into a differentiable optimization layer. The final action w_t is obtained by solving the following Quadratic Program (QP):

$$\begin{aligned} w_t^* = \underset{w \in \mathbb{R}^N}{\arg\min} \quad & \frac{1}{2}\|w\|_2^2 - \hat{\mu}_t^\top w && (4a) \\ \text{s.t.} \quad & \mathbf{1}^\top w = 1, && (4b) \\ & w \succeq 0, && (4c) \\ & \|w - w_{t-1}\|_1 \leq \delta. && (4d) \end{aligned}$$

Here, the objective function (4a) encourages the portfolio to align with the learned signal $\hat{\mu}_t$ while the regularization term $\frac{1}{2}\|w\|_2^2$ prevents extreme concentration and ensures the problem is strictly convex. The constraint (4d) strictly enforces the turnover limit.

2.3. End-to-End Learning via Implicit Differentiation

The optimization problem (4) defines an implicit mapping $w_t^* = \mathcal{P}(\hat{\mu}_t, w_{t-1})$. To train the upstream parameters θ using standard policy gradient methods (e.g., PPO), we require the Jacobian $\partial w_t^* / \partial \hat{\mu}_t$.

Since w_t^* is the solution to a convex QP, it satisfies the KKT optimality conditions $F(z^*, \psi) = 0$, where z^* contains the primal and dual optimal variables, and $\psi = \{\hat{\mu}_t, w_{t-1}\}$ are the layer parameters. By the Implicit Function Theorem, we compute the gradients by solving a linear system involving the KKT matrix:

$$\frac{\partial z^*}{\partial \psi} = -\left(\frac{\partial F}{\partial z^*}\right)^{-1} \frac{\partial F}{\partial \psi}. \quad (5)$$

In practice, we utilize efficient batched solvers (e.g., OSQP) combined with the `cvxpylayers` framework to perform this backward pass efficiently on GPUs. This ensures that the RL agent receives meaningful gradients indicating how to adjust the view $\hat{\mu}_t$ to achieve better long-term rewards while respecting physical constraints.

During training, we employ Proximal Policy Optimization (PPO). To ensure exploration within the feasible region, we apply parameter-space noise to $\hat{\mu}_t$ prior to the QP layer, rather than adding noise to the output weights w_t. This guarantees that every exploratory action taken by the agent remains strictly feasible.

References

Jiang, Z., Xu, D., and Liang, J. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

Markowitz, H. Portfolio selection. *The journal of finance*, 7 (1):77–91, 1952.

A. You *can* have an appendix here.

You can have as much text here as you want. The main body must be at most 8 pages long. For the final version, one more page can be added. If you want, you can use an appendix like this one.

The `\onecolumn` command above can be kept in place if you prefer a one-column appendix, or can be removed if you prefer a two-column appendix. Apart from this possible change, the style (font size, spacing, margins, page numbering, etc.) should be kept the same as the main body.