Reinforcement Learning and Portfolio Allocation: Challenging Traditional Allocation Methods[⋆]

Matus Jan Lavko[a,b], Tony Klein[c,d], Thomas Walther[b,e]

[a] *Imperial College London, UK*
[b] *Utrecht School of Economics, Utrecht University, The Netherlands*
[c] *Queen's Management School, Queen's University Belfast, UK*
[d] *Department of Econometrics, Statistics, and Applied Economics, University of Barcelona, Spain*
[e] *Faculty of Business and Economics, Technische Universität Dresden, Germany*

Abstract

We test the out-of-sample trading performance of model-free reinforcement learning (RL) agents and compare them with the performance of equally-weighted portfolios and traditional mean-variance (MV) optimization benchmarks. By dividing European and U.S. indices constituents into factor datasets, the RL-generated portfolios face different scenarios defined by these factor environments. The RL approach is empirically evaluated based on a selection of measures and probabilistic assessments. Training these models only on price data and features constructed from these prices, the performance of the RL approach yields better risk-adjusted returns as well as probabilistic Sharpe ratios compared to MV specifications. However, this performance varies across factor environments. RL models partially uncover the nonlinear structure of the stochastic discount factor. It is further demonstrated that RL models are successful at reducing left-tail risks in out-of-sample settings. These results indicate that these models are indeed useful in portfolio management applications.

Keywords: Asset Allocation, Reinforcement Learning, Machine Learning, Portfolio Theory, Diversification

JEL classification: G11, C44, C55, C58

[⋆]We thank the participants of the European Alternative Finance Research Conference 2022 in Utrecht, the Cardiff Fintech Conference 2022, and the Finance and Innovation Symposium 2022 in Hanoi for valuable comments and suggestions.

[∗]Corresponding Author, Mail: t.klein@qub.ac.uk.

1. Introduction and literature review

Advances in computing and engineering bring about intriguing challenges in the realm of nonlinear modeling problems that are unfeasible to solve analytically and require more complex methods. The advent of reinforcement learning and stochastic control has been deployed with great success onto the board games problems, notably TD-Gammon (Tesau & Tesau, 1995) or later ventures of the IBM DeepMind, where the models achieve a high level of winning rates against other programs as well as human professional players[1] (Silver et al., 2016). Growing digitization and automationin the asset management industry provides a natural ground for experiments with stochastic control because of suitable mathematical specifications of the problem. Numerous attempts to apply reinforcement learning have been conducted with promising results especially using direct reinforcement or inverse reinforcement methods (Moody et al., 1998), but until recently problems associated with training brittleness and high level of dimensionality did not allow for deep networks to be utilized during the process as the learning process proved often to be unstable.

In the model-free representation of the environment—in contrast to the model-based—there are two main approaches that allow for the use of deep learning, often combined in parallel. In Q-learning, an experience buffer storing transition states has improved training significantly (Charpentier et al., 2020). In Policy Optimization methods, various parallelism techniques as in Schulman et al. (2017) or other tweaks to the architecture[2] have been presented in order to address the issues described earlier. These algorithms that are on the forefront of research have proven to be useful for benchmark problems and given their promising results this provides a motivation to test them on a domain problem in finance—the Portfolio Allocation problem.

Portfolio construction is an important problem in finance, with an objective to construct an optimally performing portfolio. Traditionally, it has been proposed that this

[1]Such as the IBM Deep Blue playing Garry Kasparov, a world champion in 1996-1997. The symbolic win of Deep Blue in 1997 was a landmark victory for the computer over a champion player.

[2]Gradient clipping, Monte Carlo sampling or delayed networks are among many proposed ideas as will be described later.

construction consists in two-steps, with a parametric approach to the estimation of moments and optimizing over a grid of feasible portfolios (Markowitz, 1952). It has been established that due to high dimensionality and short sample histories, the estimation is unstable (Merton, 1980), and various methods have been proposed to stabilize this process such as robust specifications with "uncertainty structures" (Goldfarb & Iyengar, 2003), Bayesian approaches (Aguilar & West, 2000) or incorporating equilibrium returns and views about asset returns (Black & Litterman, 1992). These have proven to have ad hoc advantages but often fail due to estimation errors as described in DeMiguel et al. (2009).

The problem of nonlinear effects and interaction effects present in the structure of the cross-section of the returns result in ambiguity and a "zoo" of factors attempting to explain the difference in risk premia (Harvey et al., 2016). Our work is hence tangent to the problem of asset pricing in the field of machine learning applications in the domain space. Recovering the Stochastic Discount Factor[3] for asset pricing models using nonlinear effects (Gu et al., 2020) or regularization approaches (Kozak et al., 2020), and further review of deep learning using an array of firm-specific and macro features under a no-arbitrage machine learning problem specification (Chen et al., 2019) show that the SDF is well approximated by linear factor models, although there are robustness advantages to using machine learning methods.

Although the goal is not retrieving the SDF explicitly, an application of this approach is used for actively managing exposures and risks, found implicitly by the model and incorporated into the decision making within the RL framework with various constraints. This can provide benefits to algorithmic trading strategies complementing traditional tools. A similar effort in Cong et al. (2020) based on a complex model-based multi-armed bandit RL system strives to make RL more interpretable for investing through considering logits as scoring metrics for allocation. In this paper, a different approach is taken based on a coarser price history used for training and no use of firm-specific or macro variables

[3]SDF is defined as the transformation of a Stochastic Discount Factor portfolio lying on the capital market line such that the expected excess return is zero. For more general definition see Danthine & Donaldson (2014).

in order to conduct a benchmark study similar to the method described in DeMiguel et al. (2009).

The contribution of this work to existing literature is as manifold as the models presented herein. Firstly, we offer a comprehensive overview and introduction to the application of reinforcement learning on problems of portfolio optimization and asset allocation. Secondly and in the context of this distinct set of problems, we introduce state-of-the-art algorithms of learning, in particular reinforcement learning, and outline their pseudocode and implementations specifically for applications in the domain of Finance. As these learning algorithms are applied for a wide range of real-life utilization, ranging from applications in medicine, autonomous control, to robotics, specifying them explicitly for asset allocation problems offers a thorough overview of their potential to academics and practitioners in Finance alike.

Thirdly, we demonstrate how different reinforcement learning algorithms perform for several types of equity categories that are clustered by a Carhart four factor model. These equities are constituents of the S&P 500 and its European equivalent, the Bloomberg 500 index. We compare this performance with traditional allocation methods, such as global minimum variance portfolios, equal risk contribution, and factor-constrained portfolios. We provide robust evidence that the model-free learning approaches outperform mean-variance approaches. Further, we show that performance of these RL-generated portfolio is more stable indicating that they are capturing nonlinearities of the stochastic discount factor. However, we also find that this outperformance is not universal across different equity sets if the $1/N$ portfolio is benchmarked. For some datasets, RL-portolio outperform while for a small selection of subsamples, they underperform.

Lastly, and of the highest relevance to praciticioners, we show that RL-generated allocation strategies reduce left-tail risks in out-of-sample exercises, translating to robust hedging benefits of these RL approaches. This, in turn, offers another layer of applicability and motivation as well as justification of the use of RL systems.

Further, we attempt to explain how and why these RL-generated portfolios perform better and aim to explain how the actions of portfolio rebalancing are derived from the reinforcement learning algorithms. This benefits the general direction of *explainable Artificial Intelligence*, in which actions based on AI should be justifiable and explainable (Adadi & Berrada, 2018, Rudin, 2019).

The remainder of this paper is structured as follows. Section 2 outlines the general framework of asset allocation optimization. It further introduces the reinforcement learning approaches with direct application to these asset allocation problems. Section 3 provides an overview of the raw equity and ETF data. This section also presents the clustering of our raw datasets and outlines how the equity subsampling is executed. Section 4 specifies how models are trained and outlines the rolling testing approach for performance comparison. Section 5 presents and discusses our results. This work concludes in Section 6.

2. Methodology

2.1. General framework

Consider an environment $\mathcal{E}$ consisting of N securities that are contingent on a partially observed continuous state space[4] $\mathcal{S}$ of which $\mathbf{w}_{n,t+1} \subset \mathcal{S}$ are the weights of a portfolio that constitute actions $\mathbf{a}_t$ with state $\mathbf{s}_t := \mathbf{p}_t$, where $\mathbf{p}_t$ is the respective price vector at time t. A portfolio in time t is a unique set of actions $\mathbf{a}_{t-1}$ and is determined by a reward $r(\mathbf{a}_t, \mathbf{s}_t)$ that we wish to maximize in $[t_0, T)$, where for each t a trade takes place such that

$$\mathbf{a}_{t-1} = \mathbf{w}_t = \begin{pmatrix} w_{1,t} \\ \vdots \\ w_{N,t} \end{pmatrix} \quad , \quad \sum_{i=1}^{N} w_{i,t} = 1 \quad , w_{i,t} \in \mathbb{R}^+ . \tag{1}$$

Given a portfolio $\mathbf{w}_t$ and state $\mathbf{s}_t$, we define a simple arithmetic return in the form of $r_t = \frac{s_t}{s_{t-1}} - 1$, and derive the simple portfolio return and variance for an N-dimensional

[4]Assuming that a particular observation of state $\mathbf{s}_t$ does not describe the state of the environment as modelled by a multivariate stochastic process.

random variable $\mathbf{r}$ as

$$\boldsymbol{\mu} := \mathbb{E}[\mathbf{r}] \approx \mathbb{E}\begin{pmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_N \end{pmatrix} = \begin{pmatrix} \frac{1}{T}\sum_{t=1}^{T} r_{1,t} \\ \vdots \\ \frac{1}{T}\sum_{t=1}^{T} r_{N,t} \end{pmatrix}, \tag{2}$$

$$\sigma^2 := \mathbb{E}[(\mathbf{r} - \mathbb{E}[\mathbf{r}])^2] \ , \mathrm{Var}[\mathbf{r}_i] = \frac{1}{T-1}\sum_{i=1}^{T}(r_t - \mu_r)^2, \tag{3}$$

from which we obtain the return and volatility of a portfolio as:

$$\mu_p = \mathbf{w}^T\boldsymbol{\mu} \ \ , \sigma_p^2 = \mathbf{w}^T\boldsymbol{\Sigma}\mathbf{w},$$

where $\boldsymbol{\Sigma}$ is an element-wise expansion of the left expression in Eq. (3).

Having defined these estimated quantities,[5] we would like to maximize the reward function across the equidistant grid $t_0, \ldots, T$ with distances h. We choose the Sharpe Ratio as reward function, which reads

$$\mathbf{w} = \underset{\mathbf{w}\in\mathcal{G}}{\mathrm{argmax}}\sqrt{h}\frac{\mathbb{E}[\mathbf{r_{t,t+h}}]}{\sqrt{\mathrm{Var}[\mathbf{r}_{t,t+h}]}}, \tag{4}$$

where we satisfy the set of affine constraints $\mathcal{G}$ as defined in Meucci (2005).

This general maximization problem is typically handled with the use of quadratic programming. However, we may exploit this formulation and rewrite the problem without any loss of generality such that it is broadly applicable to the reinforcement learning setting

$$\max_{\Theta}\sum_{t=1}^{T}\mathbb{E}[\gamma r(\mathbf{a}_t, \mathbf{s}_t)], \tag{5}$$

where γ stands for a time-discounting factor and Θ the parametric expression of the non-linear models used to estimate the value or policy functions depending on the architecture of the model used.

[5]Note that there are many options one might take in order to pursue less variance in the estimates as outlined in Elton et al. (2006).

2.2. Portfolio Allocation As An Optimization Problem

In a seminal work of Markowitz (1952) the problem of asset allocation is defined such that given a smooth utility function of an investor $U(w)$, the investor wishes to solve the following problem:

$$\begin{aligned} \min \quad & \mathbf{w}^T \Sigma \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^T \boldsymbol{\mu} = \rho_p,\ \rho_p \in \mathbb{R}^+ \\ & \mathbf{w}^T \mathbf{1} = 1, \end{aligned} \tag{6}$$

for a target return ρ_p. It appears that this problem has a closed form solution under the constraint that Σ is a full-rank non-singular matrix, where we formulate a Lagrangian:[6]

$$\mathcal{L}(\mathbf{w}, \lambda, \delta) = \frac{1}{2}\mathbf{w}^T \Sigma \mathbf{w} - \lambda(\mathbf{w}^T \boldsymbol{\mu} - \rho_p) - \delta(\boldsymbol{\mu}\mathbf{1} - 1),$$

with first order conditions:

$$\begin{aligned} \Sigma \mathbf{w} - \lambda \boldsymbol{\mu} - \delta \mathbf{1} &= 0 \\ \boldsymbol{\mu}\mathbf{1} &= \rho_p \\ \mathbf{w}^T \mathbf{1} &= 1 \end{aligned}$$

or equivalently:

$$\mathbf{w} = \lambda \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \delta \boldsymbol{\Sigma}^{-1} \mathbf{1}.$$

Using constraints from Eq. (6), we may re-parameterize the problem (Bianchi, 2019) as:

$$\begin{aligned} \boldsymbol{\mu}^T \mathbf{w} &= \lambda \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \delta \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \mathbf{1} \\ \mathbf{1}^T \boldsymbol{\mu} &= \lambda \mathbf{1}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu} + \delta \mathbf{1}^T \boldsymbol{\Sigma}^{-1} \mathbf{1}, \end{aligned}$$

[6]Note that we divide by 2 for convenience.

and which yields a solution

$$\mathbf{w}^* = \rho_p \underbrace{\frac{1}{G}(\mathbf{1}^T\mathbf{\Sigma}^{-1}\mathbf{1}\mathbf{\Sigma}^{-1}\boldsymbol{\mu} - \mathbf{1}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}\mathbf{\Sigma}^{-1}\mathbf{1})}_{\substack{\vec{v} \\ n\times 1}} + \underbrace{\frac{1}{G}(\boldsymbol{\mu}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}\mathbf{\Sigma}^{-1}\mathbf{1} - \mathbf{1}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu}\mathbf{\Sigma}^{-1}\boldsymbol{\mu})}_{\substack{\vec{s} \\ n\times 1}}, \quad (7)$$

where $G = (\boldsymbol{\mu}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu})(\mathbf{1}^T\mathbf{\Sigma}^{-1}\mathbf{1}) - (\mathbf{1}^T\mathbf{\Sigma}^{-1}\boldsymbol{\mu})^2$. This solution is of great significance to the argument since $\vec{v}$ and $\vec{s}$ are not based on changing a free parameter. Two important observations follow.

Firstly, the vectors $\vec{v}$ and $\vec{s}$ from Eq. (7) span the entire set of feasible mean-variance portfolios as their linear combination. This is shown by considering q as a minimum variance portfolio resulting from the solution of Eq. (6). If there exists an α such that $\alpha\vec{v} + (1-\alpha)(\vec{v}+\vec{s})$ the weights of q are well defined. Consider an $\alpha = 1 - \rho_p$, then it follows that $(1-\rho_p)\vec{s} + \rho_p(\vec{s}+\vec{s}) = \vec{s} + \vec{v}\rho_p$. Secondly, We generate the set of feasible portfolios as affine combinations of any two distinct portfolios, which follows directly from the existence of α.

The efficient set of portfolios forms the efficient frontier, however only portfolios above the global minimum variance portfolio such as the efficient portfolio—defined as the first-order contact of the capital market line[7] and the frontier—are interesting to our application since they give the maximum return for the least amount of risk.

Naturally, a question of more complex optimization objectives arises which yields different solutions and characteristic portfolios. These objectives include the common quadratic risk utility, the global minimum variance portfolio (GMV), equal risk contribution, and factor-constrained portfolios. We refer the reader to Appendix A for their formal definitions as these specifications constitute the benchmarks in this study. These are all versions of quadratic programs with convex constrains due to the fact that they cannot be easily reduced to a set of linear equations and solved as we have demonstrated with Eq. (6). One of the constraints is for example enforcing an element-wise constraint $\mathbf{w} \succeq 0$ known as the long-only constraint. Commonly, a gradient-based method is used

[7]This is the line which under the assumption of a Capital Asset Pricing Model contains all possible portfolios. See Sharpe (1964).

for this type of general conic problem such as the SQLSP solver (Kraft, 1988).

2.3. Reinforcement Learning Paradigm

Learning a pattern from data is traditionally a process that does not feature a feedback[8] from the system as in a typical supervised learning setting. Reinforcement learning tries to address this issue of a dynamic environment by internalizing a framework of states, actions and values that are endogenous and often feature a lot of noise. This allows for a more holistic view of the system to be developed as some environments tend to have highly nonlinear structures.

In Fig. 1 we see a general setting of such a problem, where a sequence of states and rewards is passed on through the agent, and a policy drives a set of actions that are fed back into the environment. This interaction with the environment uses the Markov Decision Process[9] as a framework, establishing the features of an artificial intelligence problem (Sutton & Barto, 2018).

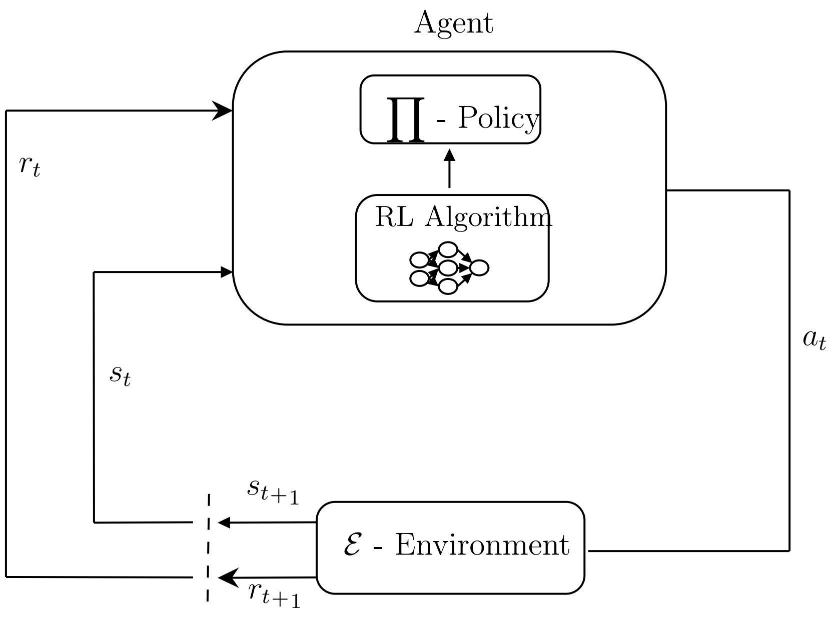


Figure 1: Simplified flow chart of a reinforcement learning control flow.

[8]We define this feedback as a signal being sent back to the model in a circular fashion.

[9]Although we are arguably dealing with a Partially Observed MDP in this work, all of the definitions are applicable without the assurance of convergence in optimality. See chapter 9 in Poole & Mackworth (2017)

2.3.1. Markov Decision Process

In what follows, we define a Markov Decision Process (MDP) as a 5-tuple characterizing a dynamic decision problem of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where

- $\mathcal{S}$ is a countable set of states, where $s_t \in \mathcal{S}\colon \mathbb{P}[s_t \mid s_{t-1}] = \mathbb{P}[s_t \mid \bigcup_{j=1}^{t-1} s_j]$, satisfies the Markov Property.
- $\mathcal{A}$ is a countable set of actions.
- $\mathcal{P}$ is a transition matrix $\mathcal{P}\colon \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$. A transition probability $s \to s'$ is thus $\mathcal{P}_{s,s'} = \mathbb{P}[s' \mid s, a] = \sum_{r \in \mathcal{R}} \mathbb{P}[s', r \mid s, a]$.
- $\mathcal{R}$ is a reward generating function $\mathcal{R}\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
- γ is a scalar discounting the value of $r(s_t, a_t)$, while the agent maximizes $r(s_{t+1}, a_{t+1})$.

An agent typically features a mechanism that determines a policy π, according to a learned experience with the use of a model. This policy is then evaluated using a value function, or a Q-function in order to improve the iteration of the algorithm and learn the dynamics of the environment.

In this vein, we define a *policy* π as a mapping $\pi\colon \mathcal{S} \to \mathcal{A}$. An optimal[10] policy is one such that given a value function $V^\pi(s)$, there is no π' with $V^\pi(s) \leq V^{\pi'}(s)$, $\forall \pi' \nsim \pi$. Further, we define a *value function* as a function that determines the expected return from a state s under π, which reads

$$V^\pi(s) := \mathbb{E}_\pi[R_t \mid s_t] = \mathbb{E}_\pi\Big[\sum_{j=0}^{\infty} \gamma^j r_{t+j+1} \mid s_t\Big], \ \forall s_t \in \mathcal{S} \tag{8}$$

The *Q-function* is then defined as a function that defines the value of taking action a_t, given a state s_t under a policy π as

$$Q^\pi(s) := \mathbb{E}_\pi[R_t \mid s_t, a_t] = \mathbb{E}_\pi\Big[\sum_{j=0}^{\infty} \gamma^j r_{t+j+1} \mid s_t, a_t\Big] \tag{9}$$

[10]It can be shown that a stationary MDP always converges in the limiting scenario, see Poole & Mackworth (2017, Chapter 9).

These functions defined in Eq. (8) and Eq. (9) are interconnected in a sense that they can be defined in terms of each other in a recursive manner (Poole & Mackworth, 2017, p. 59). This fundamental relationship in reinforcement learning is known as the Bellman Equation and allows for a solution of the MDP through an approach involving a dynamic programming setup, where current and subsequent states share a relation that can be iterated upon to reach an optimal point.

From the previous two definitions, we observe that R_t can be decomposed to

$$R_t = r_{t+1} + \gamma\Big[\sum_{j=1}^{\infty}\gamma^j r_{t+j+1}\Big].$$

This is the current reward and sum of all subsequent rewards discounted accordingly. Hence, we obtain the Bellman recursive relation for $V^{\pi}(s)$ as:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\Big[r_{t+1} + \gamma\Big(\sum_{j=1}^{\infty}\gamma^j r_{t+j+1}\Big) \mid s_t\Big]. \tag{10}$$

The expectation operator can be further expanded as:

$$\sum_{a\in\mathcal{A}}\pi(a \mid s)\sum_{s'\in\mathcal{S},\, r\in\mathbb{R}}\mathbb{P}(s', r \mid s \cup a)\Big[r + \gamma V^{\pi}(s')\Big].$$

We may think about the expression above as the logic of the reinforcement learning system, where given a state s, all of the possible and attainable states s' return a reward r based on some dynamics given by the probability distribution we are summing upon. Thus, the value function provides a solution to Eq. (10) and the ultimate goal becomes to learn $V^{\pi}(s)$, or $Q^{\pi}(a, s)$.[11]

2.3.2. Solution to the MDP

In order to solve an MDP we need to find a policy π^* that satisfies the relation $\pi^* \geq \pi'$ which is measured by the value functions and their respective sufficient relation for optimality, $V^{\pi^*}(s) \geq V^{\pi}(s)$. We do not prove the existence of such policy, but assume

[11]The reasoning we have developed may be applied to the Q-function in a similar fashion depending on the optimization objective of the algorithm.

that there is always a policy that satisfies this inequality and is optimal. This gives rise to the optimization problem we face in the form of

$$\begin{aligned} V^*(s) &= \max_\pi V^\pi(s), \\ Q^*(s,a) &= \max_\pi Q^\pi(s,a). \end{aligned} \tag{11}$$

Notice that we may rewrite $Q^*(s,a)$ in terms of $V^*(s)$ as:

$$Q^*(s,a) = \mathbb{E}_\pi \Big[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t, a_t \Big]. \tag{12}$$

Therefore, we arrive at a solution by maximizing the above expression such that an optimal policy is derived from the optimal value function as described in (Poole & Mackworth, 2017, p. 404):

$$\begin{aligned} V^*(s) &= \max_a Q^*(s,a), \\ \pi^*(s) &= \operatorname*{argmax}_a Q^*(s,a). \end{aligned} \tag{13}$$

There are two general approaches that we are considering in the model-free setup that provide a numerical way of tackling the problem. Note that the algorithms used in the empirical experiment depend on these techniques to a smaller or larger extent and combine or extend the ideas described in this chapter.

2.3.3. Q-learning

One of the methods combining a dynamic programming approach and Monte Carlo sampling of expected values in the solution of Eq. (12) is Q-learning. It uses an off-policy method, a method that tries to improve the policy that is different from the policy used to generate the actions. Estimation of the reward for possible future actions is done without following any sort of greedy update—in contrast to on-policy methods. In order to understand the generic Q-learning algorithm we define the TD error as an error resulting from an update in the estimated value of s and the more optimal estimate following from the recurrence relation $r_{t+1} + \gamma V(s_{t+1})$ as

$$\delta_t \doteq r_{t+1} + \gamma V(s_{t+1}) - V(s_t). \tag{14}$$

This error is relevant because as noted in Sutton & Barto (2018, Chapter 6) it results from a guess of another bootstrap. Although this might be a questionable approach, it provides us with a way to describe more complex settings without a model of the environment which is advantageous in problems of high-dimensional state and action spaces such as portfolio allocation problems.

In Q-learning the TD error is reformulated for the Q-function that attempts to directly estimate the optimal Q-function, $Q^*(s,a)$. It feeds from the grid of values—also known as Q-table—chooses the best one and updates the Q-table with an estimate using the TD error. Pseudo-code of a general algorithm for solving the portfolio allocation problem is outlined in Appendix B, in Fig. B.11.

2.3.4. Policy Gradient

In contrast to learning a value function and an action-value estimate, Policy Gradient attempts to learn a policy that is parameterized by a mapping $f(X) \rightarrow \boldsymbol{\theta}\colon \pi \sim \pi_\theta$, often in the form of an Artificial Neural Network or a Recurrent Neural Network (Silver et al., 2016). Value functions are thus not involved in the process of selecting an action, although it might still play a role in helping to estimate $\pi(a\mid, s, \boldsymbol{\theta})$. In order to estimate this function one needs to perform a stochastic update with regard to the gradients of an error typically in the form of

$$\mathcal{J}(\boldsymbol{\theta})\colon \boldsymbol{\theta_{t+1}} \leftarrow \boldsymbol{\theta}_t + \alpha\nabla\mathcal{J}(\boldsymbol{\theta}_t). \tag{15}$$

In the process of solving the optimization objective to learn $\boldsymbol{\theta}$, the gradient of $\mathcal{J}(\boldsymbol{\theta})$ is calculated as

$$\begin{aligned}\mathcal{J}(\boldsymbol{\pi_\theta}) &= \mathbb{E}_{\pi_\theta}[\mathcal{R}] \\ &= \int_{\mathbf{s}\in\mathcal{S}} P^{\pi_\theta}(\mathbf{s})\, ds \int_{a\in\mathcal{A}} \pi_\theta(\mathbf{s},\mathbf{a})\mathcal{R}\, da.\end{aligned} \tag{16}$$

The gradient of $\boldsymbol{\theta}$ then reads

$$\begin{aligned}\nabla\mathcal{J}(\boldsymbol{\pi_\theta}) &= \int_{\mathbf{s}\in\mathcal{S}} P^{\pi_\theta}(\mathbf{s})\, ds \int_{a\in\mathcal{A}} \nabla\pi_\theta(\mathbf{s},\mathbf{a})\mathcal{R}\, da \\ &= \int_{\mathbf{s}\in\mathcal{S}} P^{\pi_\theta}(\mathbf{s})\, ds \int_{a\in\mathcal{A}} \pi_\theta(\mathbf{s},\mathbf{a})\nabla\ln[\pi_\theta(\mathbf{s},\mathbf{a})]\mathcal{R}\, da.\end{aligned} \tag{17}$$

This important result allows us to derive the policy gradient that we want to optimize as described in Mandic (2019). We closely follow Sutton & Barto (2018) and Poole & Mackworth (2017) to obtain a policy gradient. From the result in Eq. (17), extending the one-step MDP to a multi-step MDP, for a suitable continuous differentiable policy $\pi(s,a)$ and the expectation of future rewards at each step $\mathcal{J}(\boldsymbol{\theta})$, the **Policy Gradient** reads

$$\nabla\mathcal{J}(\boldsymbol{\pi_\theta}) = \mathbb{E}_{\pi_\theta}\Big[\nabla\ln[\pi_\theta(\mathbf{s},\mathbf{a})]Q^{\pi}(\mathbf{s},\mathbf{a})\Big]. \tag{18}$$

There are potential benefits from optimizing the policy gradient instead of the typical value function approach as it has been shown to converge more easily in some cases. We refer to Necchi (2016) for an extensive analysis of policy gradients and their derivation. A general setup of a Policy Gradient algorithm with an actor and critic network is found in Appendix B, Fig. B.12. Note that simpler versions of the algorithm without TD error updates exist such as REINFORCE or other episodic Monte Carlo methods as described in (Sutton & Barto, 2018, Chapter 13), but due to problems with sequential data[12] we are not considering their application for our use case.

[12]These algorithms are not well equipped to deal with non-iid sampling of the episodes that we use for learning.

2.4. Model-Free Algorithms in Continuous State-Action Space

This section briefly describes the workings of the algorithms used for the benchmark study. All of them are classified as model-free due to their ability to learn the dynamics of $\mathcal{E}$ without prior modeling on the transition matrix $\mathcal{P}$. We choose these algorithms given their ability to learn in a setting with limited information in contrast to linear models that often suffer from sampling issues (Aue & Horváth, 2013) and structural breaks (Andreou & Ghysels, 2002).

2.4.1. Actor Critic (A2C)

Actor Critic or A2C is a class of methods that address a problem with estimating the baseline function $b_t(s_t)$ for $r_{t+1} \mid s_t$ and stabilize the training of the algorithm. During the update on line 7 in algorithm B.12 an iterative step $\alpha Q(\mathbf{s}, \mathbf{a}) \nabla_\theta \ln \pi(\mathbf{a}, \mathbf{s})$ is taken in order to estimate $\nabla_\theta \mathbb{E}[R_t \mid s_t, a_t]$. It has been proposed in Williams (1992) that the variance of this estimate can be reduced without introducing any bias by subtracting a baseline function estimated by the value function. Therefore the update becomes:

$$\begin{aligned} \alpha(Q(\mathbf{s}, \mathbf{a}) - b_t(s_t)) \nabla_\theta \ln \pi(\mathbf{a}, \mathbf{s}) \\ b_t(s_t) \approx V^\pi(s). \end{aligned} \tag{19}$$

The expression $(Q(\mathbf{s}, \mathbf{a}) - b_t(s_t))$ of Eq. (19) is commonly known as the advantage, where the baseline is known as the critic and the policy function as the actor (Mnih et al., 2016). The idea as illustrated in Fig. 2 is to estimate transitions of $\mathcal{P}$ not only at the first stage of the transition but also involving subsequent states.[13]

2.4.2. Deep Deterministic Policy Gradient (DDPG)

Instead of modelling $\pi(\cdot \mid \mathbf{s}, \theta)$ as a probability distribution, a deterministic decision function ρ: $\mu^\rho(\mathbf{s} \mid \theta) \to [0, 1]$ is postulated, mapping states to actions in a one-to-one

[13]For more detailed treatise on this multi-episodic approach and proof of the Policy Gradient theorem in this setting see chapter 13.5-13.6 in Sutton & Barto (2018).

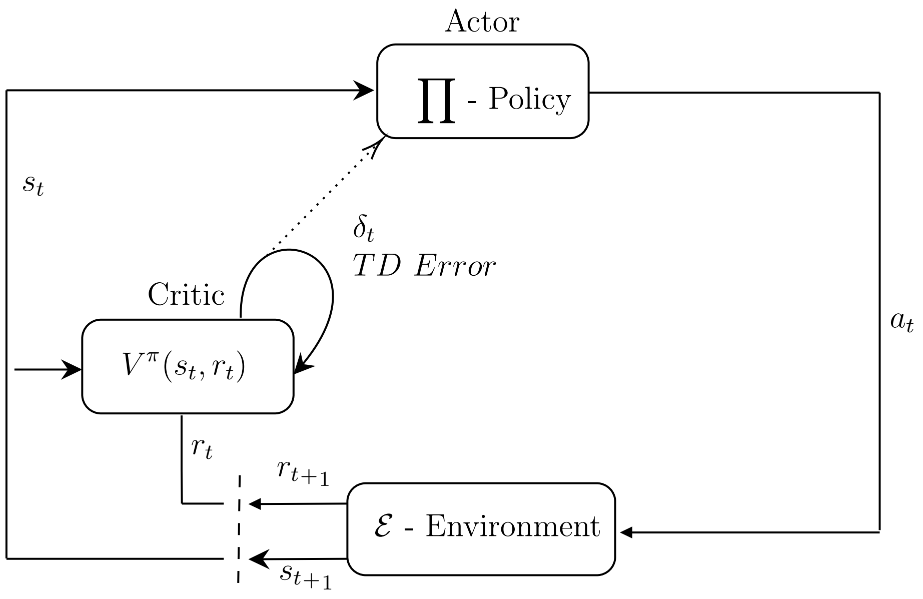


Figure 2: Simplified flow chart of an on-policy actor critic (A2C) control flow.

relation. From the result in Eq. (17) we redefine the loss function as

$$\mathcal{J}(\boldsymbol{\theta}) = \int_{s \in \mathcal{S}} \tilde{\mu}(\mathbf{s}) Q(\mathbf{s}, \mu(s))\ ds \tag{20}$$

where $\tilde{\mu}(\mathbf{s}) = \int_{\mathcal{S}} \sum_{k=1}^{\infty} \gamma^{k-1} \mu_0(s)\mu(s', k)\ ds$ is a discounted distribution defined recursively as in Weng (2018). Hence, a gradient can be taken and an update performed as

$$\begin{aligned} \nabla \mathcal{J}(\boldsymbol{\theta}) &= \int_{s \in \mathcal{S}} \tilde{\mu}(\mathbf{s}) \nabla_a Q(\mathbf{s}, \mu(s)) \nabla_\theta \mu(s)\ ds \\ \boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t + \alpha \nabla_a Q(s_t, a_t) \nabla_\theta \mu(\mathbf{s}). \end{aligned} \tag{21}$$

It is apparent that a deterministic policy likely does not introduce enough exploration[14] in the algorithm. Deep Deterministic Policy Gradient (DDPG) tries to address this by inducing exploration with a noise process[15] $\mathcal{N}$ to the deterministic policy such that

$$\acute{\mu}(s) = \mu(\boldsymbol{s}) + \mathcal{N}.$$

[14]In other words that it does not estimate the Q-function well enough due to insufficient noise and converges to a locally optimal specification. See chapter 11.3 in Poole & Mackworth (2017).

[15]Often in the form of a generic Ornstein-Uhlenbeck process x_t: $dx_t = \theta(\mu - x_t)dt + \sigma dB_t$, where B_t is a standard Brownian motion.

DDPG is usually learned in mini-batches[16] using an experience replay to obtain an approximately i.i.d. sample. It also performs a soft-update in order to stabilize the learning of the actor and critic networks (Mnih et al., 2013). A general pseudo-code for the DDPG algorithm as presented in Lillicrap et al. (2016) is outlined in Appendix B, in Fig. B.13.

2.4.3. Proximal Policy Optimization (PPO)

Given the difficulty with the TD error approach, sometimes the gradient ascent algorithm results in very large policy updates that might destabilize the optimization process and problems in convergence might arise. Circumventing this, an additional constraint is proposed to stabilize this learning. From Eq. (17) and Eq. (19), we observe that a typical actor critic maximizes:

$$\mathcal{J}(\theta) = \mathbb{E}[\nabla_\theta \ln \pi(a_t)(Q(s_t, a_t) - b_t(s_t))]. \tag{22}$$

The so-called Trust Region methods such as TRPO redefine the surrogate objective function as a ratio of the current policy function function and the old policy function (Schulman et al., 2017) and introduce a Kullback-Leibler divergence constraint[17] such that the maximization problem is rewritten as:

$$\begin{aligned} \max \quad & \mathbb{E}\Big[\frac{\pi(a_t)}{\pi_{old}(a_t)}(Q(s_t, a_t) - \hat{b_t}(s_t)\Big] \\ \text{s.t.} \quad & \mathbb{E}[D_{KL}(\pi_{old} \parallel \pi)] \leq \eta. \end{aligned} \tag{23}$$

PPO is an improvement over TRPO that addresses the relative difficulty of training this sort of specification by introducing a clipped surrogate objective function without the KL

[16]By splitting the learning data and computing gradient on a subset to optimize the learning during gradient descent optimization.

[17]This is a measure of relative entropy between two distributions defined as $D_{KL}(Q \parallel G) = \int_{\mathbb{R}} P(x) \ln(\frac{Q(x)}{G(x)})dx$ for distributions defined on the same probability space.

constraint in the form of

$$\begin{aligned}
\mathcal{U}^{original} &= r(\theta)\big[Q(s_t,a_t) - \hat{b}_t(s_t)\big], \\
\mathcal{U}^{clipped} &= \text{clip}\big(r(\theta), 1-\epsilon, 1+\epsilon\big)\big[Q(s_t,a_t) - \hat{b}_t(s_t)\big], \\
\mathcal{J}^{clipped}(\theta) &= \mathbb{E}\Big[\min\big\{\mathcal{U}^{original}, \mathcal{U}^{clipped}\big\}\Big],
\end{aligned} \tag{24}$$

where $r(\theta) = \frac{\pi(a_t)}{\pi_{old}(a_t)}$. The first term in the expectation in Eq. (24) is simply the same expression as in Eq. (6), and the clip function bounds the policy updates within a interval, where ϵ is a hyperparameter. Finally, a lower bound of this expression is taken to ensure a conservative update (Schulman et al., 2017, p. 3). Optimizing this loss ensures that objective function is more stable and the surrogate objective does not deviate too much when $r(\theta)$ becomes large.

2.4.4. Soft Actor Critic (SAC)

Soft Actor Critic features a model that encourages more exploration in order to ensure more efficient and stable update to π_θ by reformulating the optimization problem in terms of maximum entropy (Haarnoja et al., 2018). It is an off-policy algorithm[18] as illustrated in Fig. 3, as the agent performs updates based on some sort of a greedy policy in contrast to on-policy method, where TD error is directly connected to the agent's current policy as in Fig. 2. Consider Eq. (16) in which we define the expected sum of rewards as the objective we wish to maximize through gradient ascent. In SAC this objective is generalized to

$$\mathcal{J}(\pi) = \mathbb{E}[r(s_t,a_t) + \phi\mathcal{H}(\pi(\cdot \mid s_t))], \tag{25}$$

where ϕ determines the importance of entropy[19] $\mathcal{H}$ against the reward function in the optimization problem by allowing more noise as it gets bigger.

In the soft update, the Bellman recursive relation is defined in order to satisfy Eq. (25)

[18]Note that SAC, DDPG and TD3 are off-policy algorithms – although an asynchronous version of A2C could be formulated off-policy. PPO and A2C are on-policy algorithms so they are learning the value function for one policy as they follow it online.

[19]As defined in (Goodfellow et al., 2016) to be the negative expectation of the log probability distribution.

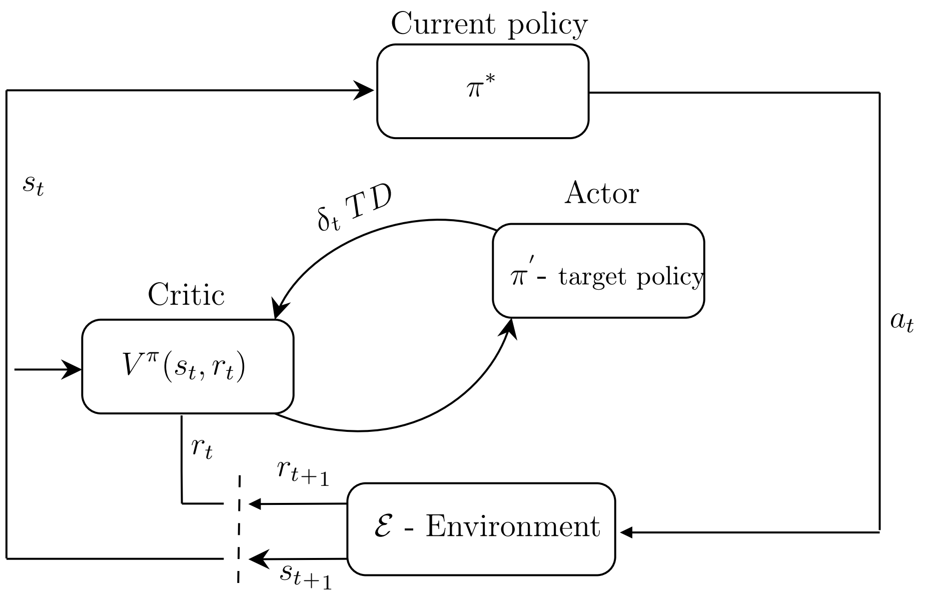


Figure 3: Simplified flow chart of an off-policy actor critic (SAC) control flow.

as

$$
\begin{aligned}
Q(\mathbf{s}_t, \mathbf{a}_t) &= r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}[V(\mathbf{s}_{t+1})], \\
V(\mathbf{s}_{t+1}) &= \mathbb{E}_\pi[Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \phi \ln \pi(\mathbf{a}_{t+1})].
\end{aligned} \tag{26}
$$

Similarly to the actor critic setup, two networks are trained corresponding to the value function and the Q-function with the use of stochastic gradients as derived in Haarnoja et al. (2018):

$$
\begin{aligned}
\mathcal{J}_{V(\mathbf{s_t})}(\theta^V) &= \mathbb{E}\Big[\frac{1}{2}\big(V_{\theta^V}(\mathbf{s}_t) - \mathbb{E}[Q(\mathbf{s}_t, \mathbf{a}_t) \ln \pi_{\theta^V}(a_t)]\big)^2\Big] \\
\mathcal{J}_{Q(\mathbf{s}_t,\mathbf{a_t})}(\theta^Q) &= \mathbb{E}\Big[\frac{1}{2}\big(Q(\mathbf{s}_t, \mathbf{a}_t) - (r(\mathbf{s_t}, \mathbf{a}_t) + \gamma \mathbb{E}_{\mu(\mathbf{s})}[V_{\tilde{\theta}^V}(\mathbf{s}_{t+1})])\big)^2\Big],
\end{aligned} \tag{27}
$$

where the outer expectation is taken under states and actions sampled from an experience replay, and the inner expectation in the soft Bellman residual $\mathcal{J}_{Q(\mathbf{s}_t,\mathbf{a_t})}(\theta^Q)$ under the deterministic marginal $\mu(\mathbf{s})$ as in Eq. (20). In addition, a target networks trick is used to stabilize learning by taking updates based on an exponential moving average of the estimated value function[20] as denoted by $V_{\tilde{\theta}^V}$.

[20]This trick has been explored as a way to improve stability in the Deep Q-learning setup, see Wang et al. (2015).

Finally, a policy is updated using an optimization step based on the distance between the new policy function and a transformed Q-function as:

$$\pi = \underset{\pi}{\operatorname{argmin}}\, D_{KL}\Big[\pi(\cdot \mid \mathbf{s}_t)\Big| \frac{exp(Q^{old}(\mathbf{s}_t, \cdot)}{Z^{old}(\mathbf{s}_t)}\Big], \tag{28}$$

where $Z^{old}(\mathbf{s}_t)$ constitutes a partition function that normalizes the distribution.[21]. SAC combines the advantages from DDPG, Actor Critic and maximum entropy frameworks to achieve more consistent performance, outperforming PPO and DDPG on benchmark problems (Haarnoja et al., 2018, p. 8). Although its multi-stage optimization process is arguably more complex, it is suggested that the soft policy approach possibly leads to better asymptotic performance.

2.4.5. Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed DDPG algorithm is another modification of the Q-learning algorithm with the use of deep networks for function approximation in a deterministic setting. Because of the large variance in the estimates of the Q-function and the resulting overestimation (Fujimoto et al., 2018), a couple of tricks are proposed in order to stabilize the learning and restrict the propagation of the error resulting from this bias in the greedy selection of the actions in the Bellman recursion.

TD3 uses two networks for selecting an action and estimating the Q-value, where the target network is used for estimation and the current network for action selection. These networks according to Fujimoto et al. (2018) are not necessarily independent due to slow updates and so the decoupling is altered by introducing an upper bound of the less biased estimate by the more biased one as follows:

$$L_{critic} = \frac{1}{N}\sum_i (r_i + \gamma \min_{\boldsymbol{\omega}} Q(s_{i+1}, \mu_1(s_{i+1})) - Q(s_i, a_i))^2, \tag{29}$$

where $\boldsymbol{\omega}$ is the set of weights learned by the critic networks. Given this loss, the target and current networks are updated using a soft delayed update similar to the SAC in order to

[21]The notion of partition function and Boltzmann distribution are concepts from information theory, see MacKay (2003, Chapter 3)

stabilize the learning (Dankwa & Zheng, 2019, p. 1). Furthermore, a regularizing method for training the critic network is introduced with the idea of forcing similar actions to have similar values and smooth the value estimate as in (Fujimoto et al., 2018, p. 6). Hence a clipped noise term is added to the sampled action from an experience replay as described in step 7 of the TD3 algorithm presented in Appendix B, in Fig. B.14.

3. Data

3.1. Raw Data

We collect raw daily open, high, low, and close (OHLC) price and volume data for all equities in the S&P 500 index and its European counterpart compiled by Bloomberg—the Bloomberg 500 index (Bloomberg, 2020). In addition, we further collect a sector dataset according to the S&P sector classification, extracted from nine SPDR sector exchange traded funds (SPDR) and two iShares ETF's, namely the Telecommunications (iShares, b) and Real Estate (iShares, a) funds. Due to computational constraints and a hypothesis of a change in the market regime, the dataset is restricted to start from 15 October 2008 (t_0) and runs until 1 January 2021 with a total of 2974 daily observations. This includes the estimation period for the sample covariance matrix which is used as a feature during the training, effectively pushing the training by 252 trading days starting on 2 February 2009. In order to prevent survivorship bias and to ensure completeness of the sample, securities that do not have at least two years of data before t_0 are purged, resulting in a dataset of 479 securities listed in Europe and 487 U.S. listed stocks.

For estimating the factor model, we use factors constructed by Bloomberg, aggregating by monthly return on these factors that are constructed using the methodology in Fama & French (1992). These are the suitably defined risk premia across the section of the assets in our factor model. In addition, a risk-free proxy is used for computing performance metrics and excess returns as the 10-year rate on U.S. Treasury bills. In what follows, we outline the clustering process.

3.2. Factor Clustering

Training on the whole index is computationally expensive for such a large set of securities. Additionally, it is not of practical relevance as the resulting portfolios would contain a very large number of stocks. To divide the datasets into clusters, we take an approach based on a linear time-series factor model that is specified as follows:

$$\mathbf{E}[R_{t,i}] - R_t^f = \alpha_{t,i} + \sum_{j=1}^{n_f} \underbrace{\beta_i(\mathbf{E}[F_{t,j}] - R_t^f)}_{\text{systematic risk}} + \underbrace{\epsilon_{t,i}}_{\text{intrinsic risk}}, \tag{30}$$

where it is assumed that intrinsic risks can be hedged such that $\mathbf{E}[\epsilon_{t,i}] = 0$ for a model with j number of factors across i assets such that the no-arbitrage condition holds.[22] For the purpose of the experiment, this rather strict assumption might be arbitrarily relaxed as it is not an exercise in retrieving the SDF, but simply good practice to keep it in mind given the fact that it affects the form of our training datasets.

We follow the four-factor asset pricing model of Carhart (1997), where a filtration $\mathcal{F}\colon F_j \in \{MKT_t, SMB_t, HML_t, MOM_t\}$ refers to the set of risk premia describing the investment universe $\mathcal{E}$. These well-known and widely-used factors are (1) $\mathbf{MKT}_t$, referring to market index returns, (2) $\mathbf{SMB}_t$ referring to the *Small Minus Big* factor that bases on a difference between proxy portfolios of small companies and big companies formed on size and book-to-market ratio or other value indicators, (3) $\mathbf{HML}_t$, the *High Minus Low* factor that bases on a difference between value and a growth portfolios formed in the same way, and (4) $\mathbf{MOM}_t$, *Momentum*, formed as an equal-weighted portfolio of the difference in the best performing securities in terms of lagged simple returns and the worst performing ones.

Assuming that the Generalized Gauss-Markov theorem holds, we retrieve $\hat{\beta}_{i,j}$ by using a least-squares estimate (Kempthorne, 2013).[23] Given the possibility of substitution effects among the factors, we employ a shrinkage method that has the advantages in re-

[22]This lemma implies an existence of an SDF which is the main object of interest in the asset pricing literature and is the building block of no-arbitrage pricing models (Danthine & Donaldson, 2014).

[23]Note that under the normality assumption on the distribution of residuals, it can be proven that this is also the MLE estimator. See chapter 14 in Rice (2006) for proof.

ducing estimation errors out of sample as in Kozak et al. (2020). Although in such a low-dimensional model this could reduce performance of the SDF as a pricing kernel, this approach is taken in order to improve the stability of the estimates as the parameters are later used for sorting. Over a period of 36 months, we estimate a linear model using the L_1 and L_2 penalties[24] and solve the optimization problem in the unconstrained Lagrangian form:

$$(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}) = \underset{a,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{y}_i - a\mathbf{1} - \mathbf{F}b)\|_2^2 + \lambda_1 \sum_{j=1}^{n_f} \|b_j\|_1 + \lambda_2 \sum_{j=1}^{n_f} \|b_j\|_2^2.$$

After obtaining the estimates, the datasets are constructed using quantile sorting of the coefficients $\hat{\beta}_j$. This β-quantile is defined as

$$Q_{\boldsymbol{\beta}}(p) := \inf\{\beta \in \mathbb{R} \colon p \leq F(x)\},$$

where $F_{\boldsymbol{\beta}}(X) = \mathbb{P}(\beta \leq x)$ is the empirical CDF obtained from the factor model.[25] we summarize the resulting sorted datasets in Table 1 together with their corresponding quantiles.

Dataset	Quantile Range	N
Growth 50	$\{Q_{\boldsymbol{\beta}^{\text{HML}}}(0.1)\}$	50
Momentum 50	$\{Q_{\boldsymbol{\beta}^{\text{MOM}}}(0.9)\}$	50
Size 50	$\{Q_{\boldsymbol{\beta}^{\text{SMB}}}(0.1)\}$	50
Growth + Size 100	$\{Q_{\boldsymbol{\beta}^{\text{HML}}}(0.1)\} \cup \{Q_{\boldsymbol{\beta}^{\text{SMB}}}(0.1)\}$	100
Momentum + Size 100	$\{Q_{\boldsymbol{\beta}^{\text{MOM}}}(0.9)\} \cup \{Q_{\boldsymbol{\beta}^{\text{SMB}}}(0.1)\}$	100
Sector ETF	N/A	11

Table 1: Overview of the resulting datasets of the β-quantile sorting.

We construct these sorted portfolios for both the S&P 500 and BE500 indices, resulting in ten quantile-sorted datasets plus the sector ETF dataset. This provides for means to test the algorithms on both North American and European equities with a single-factor

[24]Where we use the convention of a norm being defined as a notion of distance in the L^p space for a real valued vector $\mathbf{x}$: $\|\mathbf{x}\|_\mathbf{p} := (\sum_\mathbf{j} |\mathbf{x_j}|^\mathbf{p})^{\frac{1}{\mathbf{p}}}$.

[25]Note that the definition of the quantile does not require any smoothness or continuity properties to be required of $F(x)$.

and multi-factor approach, where the idea is to test if the model is able to capture hedging opportunities within different factors—or sectors in the sector ETF dataset. Notably, the constituents of these sorted portfolios remain fixed for remainder of the sample, ensuring homogeneity in sub samples during learning and training and the trading exercise, which bases its asset allocation decisions on the learned models.

4. Model Training and Testing Windows

4.1. Model Training

Training neural networks is often a challenging task given the specifics of stochastic gradient descent optimization techniques. It is also quite computationally expensive and implemented in cloud computing.[26] A softer grid search has been employed on all of the models in order to search for optimal hyper-parameters as summarized which are given in Appendix C. This has been done in order to ensure that a stable configuration of the model is deployed on the data given the difference in the underlying data processes of our datasets.

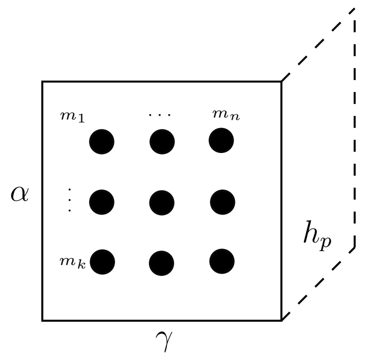
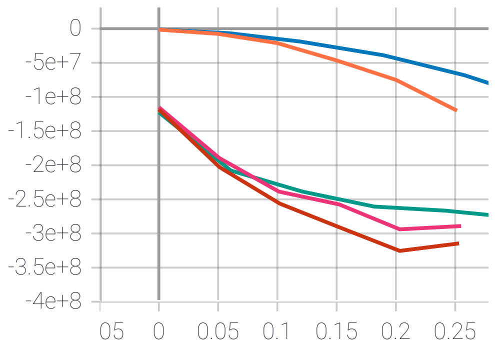


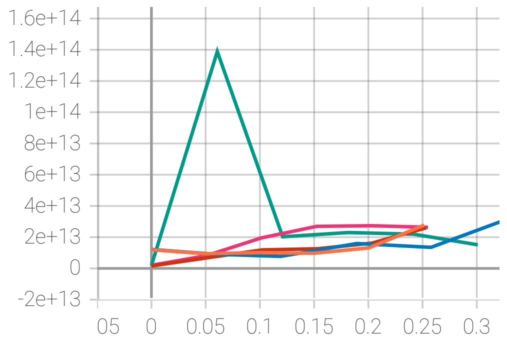
Figure 4: An exemplary outline of the initial state of a grid search procedure.

The grid search as illustrated Fig. 4 initializes an equidistant grid of hyperparameters which differs by the model architecture. Most commonly, the learning rate α and the discount factor γ are optimized with model-specific parameters such as batch size or entropy coefficient in SAC. Note that hyperparameter $h_p \in \{\mathcal{H}\}$ is increasing the dimensions of

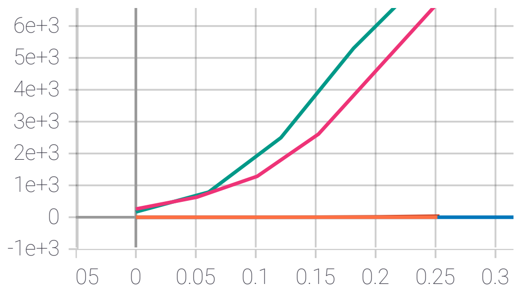
[26]The models are trained using Google's cloud service Colab Pro that allowed to parallelize training on a NVIDIA Tesla P100 GPU. For the models themselves, we use a high-level package that is built on PyTorch (Liu & Gong, 2020) and provided an interface to use the Stable Baselines library (Hill et al., 2018) that provides stable implementations of all state-of-the-art algorithms based on OpenAI infrastructure.



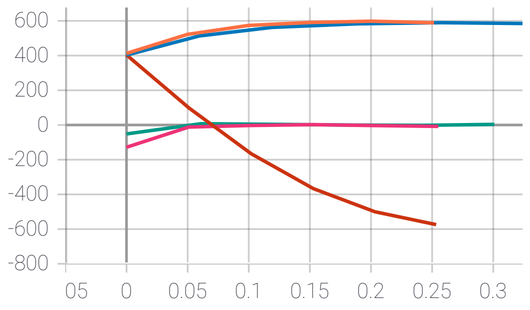
(a) Actor loss



(b) Critic loss



(c) Entropy coefficient



(d) Entropy loss

Figure 5: Sample SAC grid search log

the grid, resulting in $\mathcal{O}(n^{dim(\mathcal{H})})$ complexity. Thus, it is computationally expensive to optimize hyperparameters in this way as there are $k \times n \times (h_1 \times \ldots \times h_p)$ number of models to train and evaluate.[27] It is possible to initialize the grid randomly, but since we are interested in the magnitude of parameters rather than a precise configuration, the evenly-spaced approach is taken. For example, optimizing the parameter α typically takes into consideration an array of possible parameters $(\ldots, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2})$.

Obtaining training statistics for all of the model runs allows for examination of the relevant metrics and selection of the most stable model. Fig. 5 exemplarily visualizes the development of metrices across parameter variants of a search over SAC models for the SPX Growth 50 dataset. Since it is desirable that the networks learn the Q-function—or other learned functions—in a stable fashion, we would like to minimize the critic and actor loss such that the curve is smooth. Critic loss is a standard MSE loss and we want it to converge towards a particular quantity, whereas the actor loss is a negated version of the

[27]We have trained approximately 4-12 configurations in the grid for each of the 11 datasets—depending on training complexity—of which a full soft search has taken approximately two days for each of the datasets resulting in a total of 24 training days for the search to be conducted.

Critic loss (Lapan, 2018, ch. 12) and we would like it to smoothly decrease. Similarly, we would like the entropy loss—or any other endogenous loss function specific to the model—to be stabilized and decrease in an orderly fashion. From Fig. 5, we see that the best candidate is the model drawn in red, as other candidate models such as the one drawn in pink suffer from stability issues. We would therefore pick this model to be our globally optimal model within our soft search space and use the parameters associated with it in the testing. These hyperparameter configurations are presented in Appendix C.

4.2. Rolling Test

Having searched for an optimal model, weights $\boldsymbol{w}^*$ are obtained for each t in the testing set that starts on 1 January 2019 and ends on 1 January 2021. These weights are the weights suggested by the RL agents as optimal for each day, and we use them for rebalancing the testing portfolios. Note that the model is trained with daily data, but the rebalancing period is set to be longer. We follow the convention of *monthly* rebalancing (DeMiguel et al., 2009), which is what is reported, but also conduct tests for weekly, quarterly and semi-annual rebalancing as robustness checks. The rolling test as illustrated in Fig. 6 simply executes trades after a period of h trading days given a vector of optimal weights, repeating this process until T. For the benchmark models, this implies a rolling estimation of necessary parameters $(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ and solving the optimization problem at each t_n with the result being another set of optimal weights simulating the test portfolio. For the factor constrained approach as described in appendix Appendix A, a new factor model is estimated on a rolling basis.

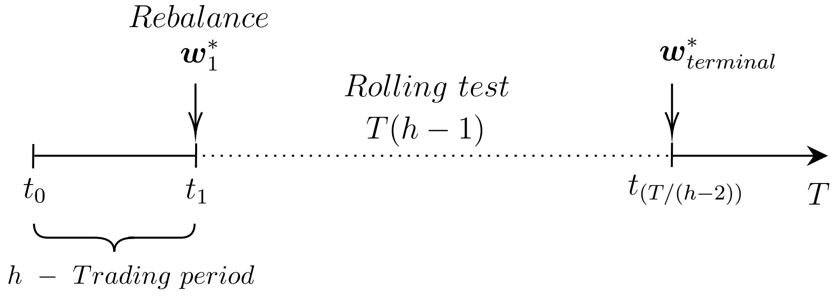


Figure 6: Rolling test diagram

Given this procedure, we obtain returns for each of the T/h discrete trading periods as

$$\mathbf{r}_t = \boldsymbol{w}^*_{t-1} \cdot \mathbf{p}_{[t,t+h)} - \kappa \Delta \boldsymbol{w}^*_{t-1} \cdot \mathbf{p}_{[t,t+h)}, \tag{31}$$

where we adjust for transaction costs with $\kappa = 0.001$ that is subtracted as a fixed amount scaling, $\Delta\mathbf{w}^*$, which represents the trades that would have been executed. It is important to note that this transaction cost adjustment is implemented during the training of the RL agents as well, in order to prevent over-trading and reduce the effective number of bets the model is taking. Subsequently, these returns are aggregated geometrically as $\prod_{i=1}^{T}(1 + r_i)$, $\forall r \in \mathbf{r}_{test}$, and a cumulative performance of the test portfolio is obtained. From these returns, we compute the performance and risk measures and other results as presented in the next section.

5. Empirical results and discussion

5.1. Single-Factor Equity Sets

Results for the single factor datasets are detailed in Table 2 and Table 3. Firstly, we focus on the results for the North American equity datasets. For the S&P 500 single-factor datasets in Table 2, we find that all of the algorithms except A2C outperform the naive equal-weight portfolio on a risk-adjusted basis in the Growth 50 dataset. None of them, however, are able to beat this benchmark on the Momentum and Size datasets. Comparing the performance to the mean-variance approaches, we see that these portfolios consistently beat the equal risk contribution as well as the minimum variance portfolio in the case of SAC and DDPG. Interestingly, none of the algorithms beat the factor constrained portfolio in terms of performance.

In terms of skewness, the RL agents perform reasonably well at reducing left-tail risk uniformly in the Growth and Momentum factor, but struggle with the Size dataset. Although all of them beat the mean-variance benchmarks in the Size factor set, negative skewness remains larger than the equal-weight portfolio. There are no large deviations in terms of drawdowns from the equal-weight portfolio, although generally the RL models deflate the drawdowns slightly. This is true particularly for the SAC that is the only model that is able to decrease drawdown across all of the three factor datasets, and hence offers significant risk-reducing potential.

	Growth 50				**Momentum 50**				**Size 50**			
	r^{ann}	SR_α	Max DD	$skew(r)$	r^{ann}	SR_α	Max DD	$skew(r)$	r^{ann}	SR_α	Max DD	$skew(r)$
A2C	37.1%	1.34	-32.8%	-0.35	25.9%	0.95	-35.5%	-0.75	2.54%	0.03	-57.8%	-0.73
DDPG	36.4%	**1.30**	-34.0%	-0.48	24.4%	0.94	-32.5%	-0.80	3.8%	0.05	-59.9%	-0.59
PPO	35.0%	**1.27**	-34.6%	-0.53	26.2%	0.98	-33.5%	-0.77	1.0%	-0.01	-59.6%	-0.81
SAC	37.6%	**1.40**	-32.3%	-0.52	26.6%	1.00	-32.6%	-0.77	4.2%	0.07	-57.3%	-0.69
TD3	38.3%	**1.41**	-32.%	-0.37	26.0%	0.98	-33.8%	-0.66	2.3%	0.02	-57.9%	-0.48
EW	36.8%	1.36	-33.1%	-0.45	27.4%	1.04	-33.5%	-0.76	5.2%	0.09	-57.5%	-0.54
MV	25.2%	1.13	-22.3%	-0.06	22.4%	0.97	-28.6%	-1.04	3.1%	0.05	-41.7%	-0.86
ER	27.8%	1.19	-26.7%	-0.59	19.3%	0.84	-29.6%	-0.94	-2.3%	-0.14	-44.5%	-0.94
FC	53.1%	1.67	-28.4%	-0.45	43.8%	1.46	-31.3%	-0.61	38.9%	1.04	-41.5%	0.10
MSR					46.2%	1.70	-30.4%	-0.91				

Table 2: Annualized monthly returns, Sharpe ratios, maximum draw down, and skewness for the out of sample tests of the Reinforcement Learning agents, the naive $1/N$ equal weights portfolio, and the Minimum Variance, Equal Risk Contribution, Factor Constrained, and Maximum Sharpe Ratio portfolios for the S&P 500 single factor datasets.

	Growth 50				**Momentum 50**				**Size 50**			
	r^{ann}	SR_α	Max DD	$skew(r)$	r^{ann}	SR_α	Max DD	$skew(r)$	r^{ann}	SR_α	Max DD	$skew(r)$
A2C	17.6%	**0.90**	-26.6%	-1.95	5.3%	0.11	-51.0%	-0.31	-4.9%	**-0.21**	-46.6%	-0.49
DDPG	18.2%	**0.92**	-27.2%	-1.88	7.8%	**0.19**	-51.0%	-0.37	-5.1%	**-0.21**	-46.8%	-0.47
PPO	17.5%	**0.85**	-27.6%	-2.03	6.6%	**0.15**	-51.4%	-0.33	-5.9%	-0.24	-47.4%	-0.40
SAC	19.5%	**0.97**	-27.1%	-1.78	9.5%	**0.25**	-51.1%	-0.51	-5.2%	**-0.22**	-45.2%	-0.41
TD3	18.0%	**0.87**	-27.7%	-1.99	4.8%	0.09	-51.5%	-0.27	-6.2%	-0.25	-46.6%	-0.63
EW	17.0%	0.84	-27.4%	-1.96	5.1%	0.11	-51.3%	-0.25	-5.6%	-0.23	-46.6%	-0.58
MV	8.2%	0.44	-21.7%	-1.37	-2.4%	-0.16	-44.1%	-1.07	-8.2%	-0.47	-35.1%	-1.28
ER	6.1%	0.30	-21.8%	-1.56	0.1%	0.96	-41.2%	-0.72	13.3%	-0.49	-36.5%	-1.59
FC	42.2%	1.68	-28.9%	-0.92	24.8%	0.93	-38.5%	-0.97	15.1%	0.44	-45.7%	-0.4
MSR	42.24%	1.95	-23.5%	-0.44	25.1%	-0.06	-38.5%	-1.03	-8.6%	0.39	-45.7%	-0.39

Table 3: Annualized monthly returns, Sharpe ratios, maximum draw down, and skewness for the out of sample tests of the Reinforcement Learning agents, the naive $1/N$ equal weights portfolio, and the Minimum Variance, Equal Risk Contribution, Factor Constrained, and Maximum Sharpe Ratio portfolios for the Bloomberg 500 single factor datasets.

Fig. 7 provides evidence that the 5% VaR is very similar for the equal-weight benchmark and the RL models, where only the SAC and TD3 perform better in the Momentum and Growth factors, but the problems with the Size dataset remains. In terms of the mean-variance approaches, both feature a lower VaR than RL for the Size factor. Minimum variance and equal risk parity also outperform on the Growth and Momentum factors, but we find that all of the RL models outperform the factor constrained specification in the Momentum factor. For A2C, SAC an TD3, better performance is noted for the Growth 50 dataset with a slightly lower VaR than the benchmark.

For the European equities, the RL algorithms seem to be more successful at outperforming the equal-weight in the performance measures. All of them yield a slightly better

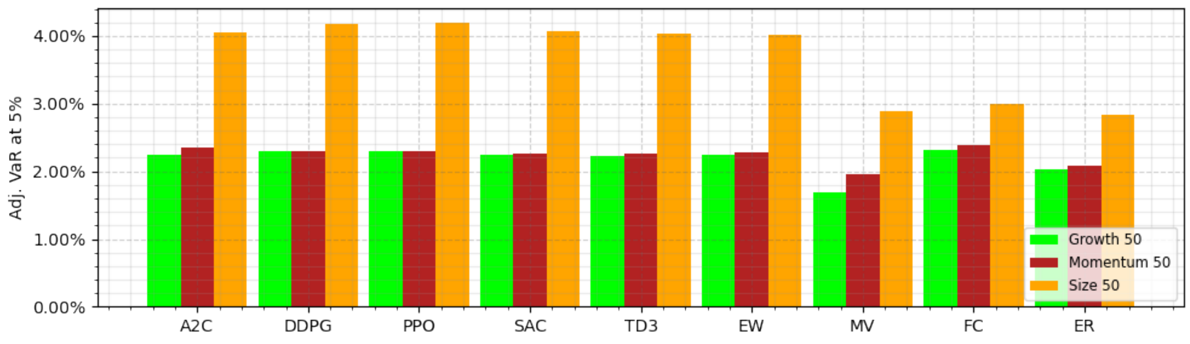


Figure 7: Cornish-Fisher Adjusted Value at Risk for the S&P 500 equity sets.

performance on the Growth 50 dataset with SAC beating the EW Sharpe by 11%. For the Momentum and Growth factors, TD3 was less successful, but SAC and DDPG still yield a significant outperformance. All of the algorithms outperform the minimum variance portfolio, but fall short when compared to the linear factor constrained portfolio. An interesting anomaly happens with the Momentum factor, where the equal parity outperforms not only the RL models but also all of the mean-variance benchmarks and the MSR which often results in a portfolio that is largely overfit and very concentrated.[28]

Assessing the distributional properties of the result, RL seems to increase the negative skew in the Growth dataset, where all of the models outperform the benchmark, suggesting that they are taking more risky bets, effectively negating any risk-reducing behavior. In the Momentum and Size dataset, however, they reduce negative skewness and beat the mean-variance benchmark as well as the EW benchmark in the Size factor. Although the drawdown remains to be around the same level as the EW benchmark, this shows that there could be benefits from reducing the left-tail risk. In contrast to the U.S. equities, RL algorithms perform better in terms of risk measures what is presented in Fig. 8 as the difference between the VaR of RL algorithms and the FC mean-variance optimization. For the Growth factor dataset, all of the RL algorithms perform better than the factor model. In the Size factor, this result is not as pronounced, although notably the SAC algorithm outperforms the FC and the EW. Momentum factor shows that the mean-variance outperforms with respect to VaR measures, although SAC yields better results

[28]Since this is a result of a greedy search over the efficient frontier, it often results in a small number of positions being very overweight due to unstable moments estimates.

than the EW again.

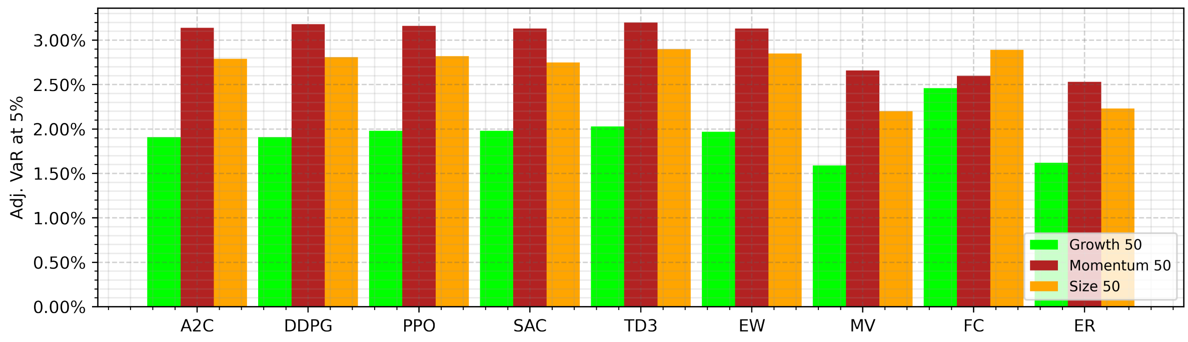


Figure 8: Cornish-Fisher Adjusted Value at Risk for the Bloomberg 500 equity sets.

5.2. Multi-Factor Equity Sets

In the two-factor datasets, where we combine Growth and Momentum factors with the Size factor in order to learn a hedging strategy, RL performs better on the U.S securities than the European-listed stocks. Hedging Growth with Size, SAC and PPO stands out with better SR_α. In terms of risk, SAC is able to reduce the drawdown slightly, albeit skewness remains to be about the same level. These two models also beat MV, but are unable to beat the linear factor as well as the equal parity portfolios. Notably, the ER portfolio is able to reduce the DD of the test strategy by more than 13% from the EW benchmark.

For the Momentum combined with Size equity set, TD3 seems to perform well at reducing the left-tail risk as well as outperforming the EW benchmark. Similarly, SAC seems to perform just as good as EW and PPO and DDPG also perform considerably well with DDPG increasing the SR_α by 21%, while still reducing drawdowns slightly. With regards to the mean-variance benchmarks, DDPG outperforms MV and ER but falls short of the linear model again.

Looking at the results tested on the European stocks, the RL agents perform rather poorly with only TD3 and A2C outperforming the EW benchmark on Growth & Size and Momentum & Size, in that order. However, it does so with problems to learn a good hedging strategy, as we see in the case of TD3 resulting in inflated negative skew. These results show that EW portfolio is superior to mean-variance approaches too as only the

	Growth + Size 100				**Momentum + Size 100**			
	r^{ann}	SR_α	Max DD	$skew(r^{ann})$	r^{ann}	SR_α	Max DD	$skew(r^{ann})$
A2C	19.8%	0.55	-45.9%	-0.75	17.6%	0.51	-44.5%	-0.95
DDPG	20.1%	0.58	-43.6%	-0.86	20.6%	**0.63**	-42.1%	-0.89
PPO	22.0%	**0.63**	-44.2%	-0.83	18.8%	**0.55**	-44.5%	-0.87
SAC	21.8%	**0.66**	-42.5%	-0.82	17.4%	0.51	-43.6%	-0.88
TD3	21.1%	0.61	-45.0%	-0.80	18.6%	**0.54**	-43.3%	-0.73
EW	21.3%	0.62	-43.9%	-0.83	17.9%	0.52	-43.9%	-0.88
MV	16.1%	0.58	-32.6%	-0.56	13.1%	0.41	-39.0%	-1.00
ER	16.3%	0.68	-29.4%	-0.77	14.4%	0.60	-29.8%	-1.05
FC	23.0%	0.70	-41.2%	-0.52	21.6%	0.65	-40.4%	-0.76
MSR					20.6%	0.62	-40.4%	-0.77

Table 4: Annualized monthly returns, Sharpe ratios, maximum draw down, and skewness for the out of sample tests of the Reinforcement Learning agents, the naive $1/N$ equal weights portfolio, and the Minimum Variance, Equal Risk Contribution, Factor Constrained, and Maximum Sharpe Ratio portfolios for the S&P 500 multi-factor datasets.

factor model beats the EW in both of the cases. In contrast to this, the RL approach works better for all of the models except for PPO—where after inspecting the trading pattern, transaction costs probably cut from the performance considerably. Both the MV and ER approaches perform worse with ER, failing quite hard in the Momentum-Size dataset. In general, the RL approach underperforms but not by a wide margin, and seems to have quite stable results not deviating from the EW portfolio to a large extent.

	Growth + Size 100				**Momentum + Size 100**			
	r^{ann}	SR_α	Max DD	$skew(r^{ann})$	r^{ann}	SR_α	MAX DD	$skew(r^{ann})$
A2C	4.2%	0.11	-38.5%	-1.20	1.2%	**-0.01**	-47.1%	-0.50
DDPG	4.2%	0.11	-37.0%	-1.26	-0.9%	-0.08	-47.1%	-0.38
PPO	2.0%	0.02	-39.5%	-1.64	-1.0%	-0.08	-47.3%	-0.60
SAC	4.6%	0.12	-37.8%	-1.25	-0.9%	-0.08	-48.8%	-0.58
TD3	5.7%	**0.17**	-37.3%	-1.38	-0.3%	-0.06	-47.4%	-0.41
EW	5.0%	0.14	-37.4%	-1.27	0.9%	-0.02	-47.2%	-0.48
MV	1.2%	-0.01	-33.5%	-1.31	-3.9%	-0.2	-42.9%	-0.78
ER	2.6%	0.07	-23.7%	-1.64	-6.4%	-0.39	-36.0%	-1.48
FC	13.6%	0.46	-39.0%	-1.48	3.4%	0.06	-46.9%	-0.67
MSR	11.5%	0.38	-38.6%	-1.49	3.2%	0.06	-46.9%	-0.67

Table 5: Annualized monthly returns, Sharpe ratios, maximum draw down, and skewness for the out of sample tests of the Reinforcement Learning agents, the naive $1/N$ equal weights portfolio, and the Minimum Variance, Equal Risk Contribution, Factor Constrained, and Maximum Sharpe Ratio portfolios for the Bloomberg 500 multi-factor datasets.

In the sector ETF dataset we test for the ability for the models to hedge across the sectors, as there is a different and broader set of exposures for the model to learn. The results show that the simple A2C has been the most successful to learn these paterns, with all of the other models having a relatively good performance as well. DDPG and SAC have been able to reduce the DD and negative skewness the most, suggesting that they were able to capture some of the pricing structure of these complex[29] ETF's. Minimum Variance and Equal Risk deliver lower risk-adjusted returns, but again we see that the factor constrained strategy is superior. Achieving a Sharpe close to the MSR portfolio, it is able to reduce the skew by almost a half while pushing the drawdowns down by 5%.

	Sector ETF			
	r^{ann}	SR_α	Max DD	$skew(r)$
A2C	16.5%	**0.58**	-36.1%	-0.78
DDPG	12.9%	0.44	-36.3%	-0.66
PPO	12.5%	0.42	-37.6%	-0.92
SAC	13.1%	0.46	-35.3%	-0.73
TD3	14.8%	**0.50**	-37.0%	-0.77
EW	13.7%	0.47	-36.9%	-0.77
MV	8.6%	0.33	-27.7%	-0.41
ER	8.1%	0.26	-34.4%	-0.70
FC	25.1%	0.87	-31.2%	-0.39
MSR	25.1%	0.88	-31.6%	-0.68

Table 6: Annualized monthly returns, Sharpe ratios, maximum draw down, and skewness for the out of sample tests of the Reinforcement Learning agents, the naive $1/N$ equal weights portfolio, and the Minimum Variance, Equal Risk Contribution, Factor Constrained, and Maximum Sharpe Ratio portfolios for the Sector ETF dataset.

5.3. Inference and Robustness

In order to look at the out of sample stability of the results, we would like to run many trials on different samples and ideally obtain some sort of a bootstrap result, or arrive at a more robust result. This is due to the results being representative of just one instance of a sequence of random variables of prices. Because of the complexity of training and optimizing RL models, however, such a Monte Carlo experiment is hard to conduct. Hence, in addition to the monthly rolling test, a weekly, quarterly and semi-annual windows have been tested and checked for large divergences that would suggest a high turnover of the strategy rendering the model useless for practical purposes. Referring

[29]Complex from the standpoint of an asset pricing equation as there are many risk premia defining an ETF.

to Fig. 9, we observe that most of the rankings of the algorithms in terms of performance propagate through these windows. Notably, the PPO algorithm seems to deviate more across the windows with more unstable profile due to over-trading as confirmed by the animations we produced for the rebalancing process in the tests. This has been the case with all of the datasets,[30] which is interesting due to the PPO's construction and conservative updates mechanism.

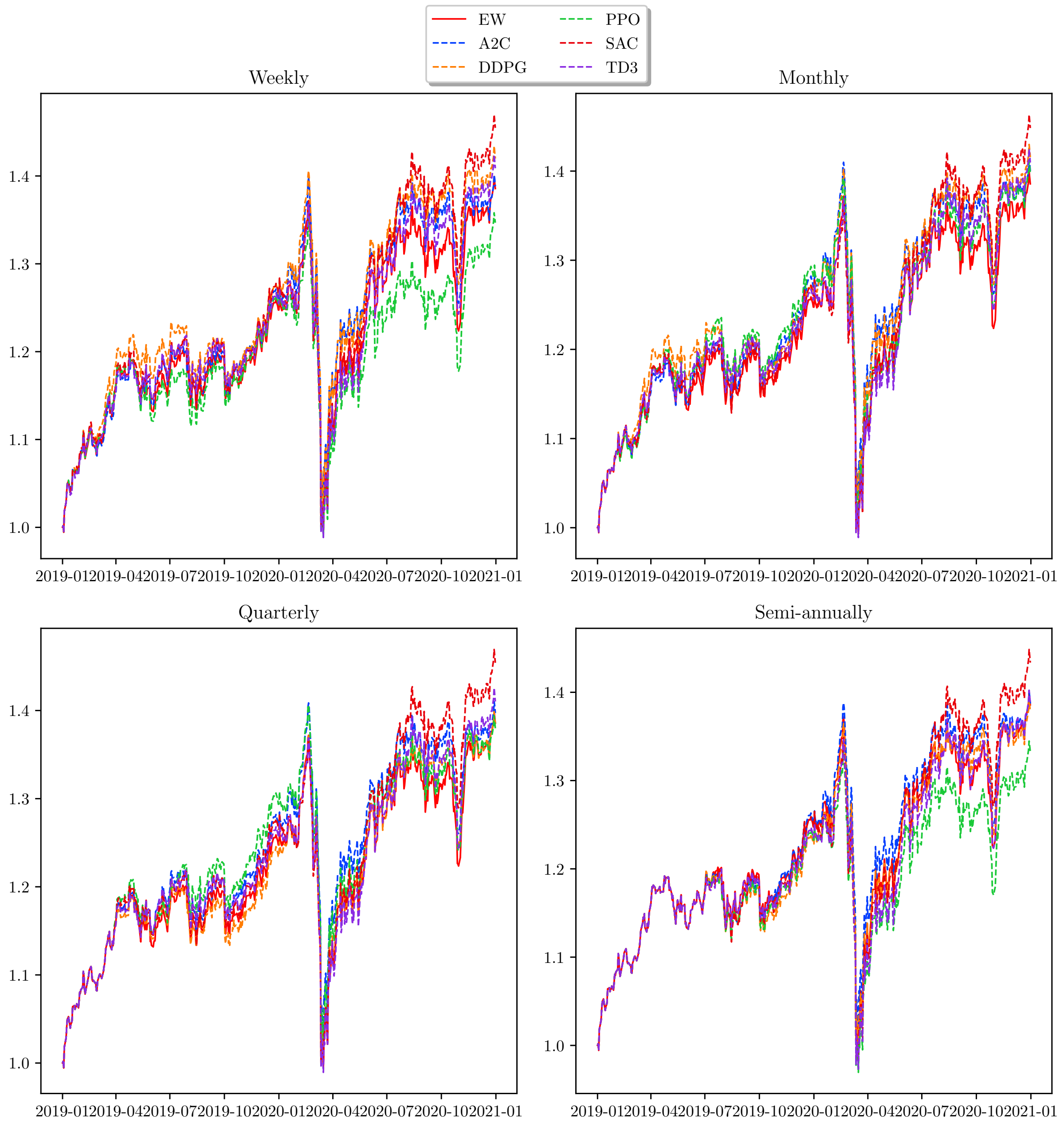


Figure 9: BE500 Growth 50 Rolling Tests Cumulative Return

[30]Even if the ranking changed slightly, a clear winner and a clear loser algorithm has emerged. This suggests using a voting ensemble of the models.

Another issue, that is often times overlooked when such rolling tests are conducted is the large variance of the Sharpe Ratio estimator. It is important to realize that all of the reported parameters such as the sample statistical moments are estimators and they have the same distributional properties of a parameter. Therefore, we need a probabilistic approach to assess the relative performance of the models. Given the non-normal features of the observed distribution of returns we need a way to account for the third and fourth moment as they might lead to a large inflation of the sample Sharpe Ratio due to the fact that any observed Sharpe Ratio can be expressed as a combination of two Gaussians with different parameters (Harvey et al., 2016, p. 6). This fact is used to prove that the parameter follows a normal distribution and therefore we formulate a test statistic in the form of

$$\hat{\mathrm{PSR}}(SR_\alpha) = z\left\{\frac{(\hat{SR} - SR_\alpha)\sqrt{T-1}}{\sqrt{1 - \hat{\gamma}_3\hat{SR} + \frac{\hat{\gamma_4}-1}{4}SR_\alpha}}\right\}, \tag{32}$$

where $\hat{SR}$ is the estimate, SR_α is the reference SR, $\hat{\gamma}_3$, $\hat{\gamma}_4$ are the skewness and kurtosis estimates and $z\{\cdot\}$ is the Standard Normal cdf. We call this the Probabilistic Sharpe Ratio (Harvey et al., 2016, p. 9), and essentially we interpret it as the probability that the observed Sharpe ratio is higher than the reference. This statistic is a function of T as well, so it requires larger returns in order to establish statistical significance.

Using Eq. (32), we construct a hypothesis test of the form

$$\begin{aligned} H_0: &\quad SR_{test} = \omega, \\ H_1: &\quad SR_{test} > \omega, \end{aligned}$$

with a rejection threshold of $1 - P(\omega \leq \hat{SR}_{test}) > \alpha$, for a statistical significance level α. This framework can be used for deflating a reported Sharpe ratio, but we are rather interested in comparing the PSR on a relative level. For example, from left side of Fig. 10 we observe that setting $\omega = 0$, the so-called skill-less level, all of the algorithms together with the benchmark pass the 95% significance level and with this level of confidence we conclude that the population $SR_{test} > 0$. In addition, we see that based on this probabilistic assessment, SAC and TD3 have a larger probability of rejecting the null, which is desirable. The right side of Fig. 10 suggests that if we run this test on an array of different possible SR, the majority of the algorithms fall short of $\alpha = 10\%$ somewhere between 0 and 0.05. Such an inverted sigmoid is present in all of the datasets, albeit with different cutoffs for statistical significance.

To compare the models, we decided to use a relative percentage marginal for a given test where $\omega = 0$. Note that we have established that the function mapping the $P(SR_{test} > \omega | H_0) \rightarrow SR^*$ does not yield a linear relationship, and therefore it is hard to compare the magnitude of the marginals across the datasets as these functions feature different domain sets. It is, however, a good proxy for seeing how the models compare against the benchmark.

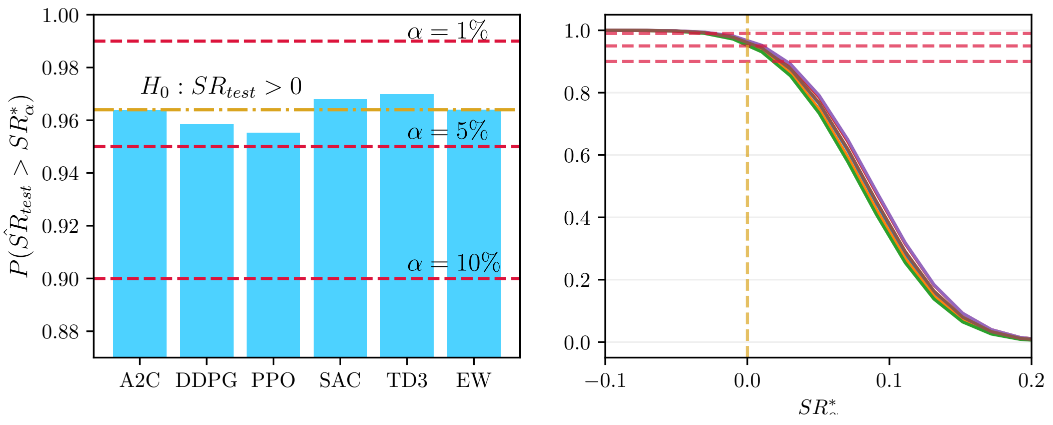


Figure 10: PSR test on SPX Growth 50

Dataset	A2C	DDPG	PPO	SAC	TD3
SPX Growth 50	-0.02	-0.55	-0.88	**0.40**	**0.59**
SPX Momentum 50	-1.82	-2.01	-1.20	**0.02**	-1.10
SPX Size 50	-3.42	-2.28	-5.71	-1.12	-3.99
SPX Growth + Size 100	-2.66	-1.52	**0.37**	**1.43**	-0.36
SPX Momentum + Size 100	-0.45	**4.28**	**1.23**	-0.42	**0.88**
BE500 Growth 50	**1.49**	**1.97**	**0.19**	**3.18**	**0.74**
BE500 Momentum 50	**0.01**	**4.46**	**2.24**	**7.69**	-1.13
BE500 Size 50	**1.10**	**1.10**	-0.53	**0.56**	-1.09
BE500 Growth + Size 100	-1.67	-1.67	-6.78	-1.11	**1.65**
BE500 Momentum + Size 100	**0.57**	**3.43**	-3.44	-3.42	-2.29
Sector ETF	**4.54**	-1.31	-2.29	-0.43	**1.3**

Table 7: PSR pct. marginal over EW for $H_0 : SR_{test} > 0$

The results show in Fig. 10 suggest, that the algorithms perform better on the Growth and Momentum factors as evident from the positive marginals. In the case of the European equities, most of the algorithms except for TD3 in the Momentum dataset outperformed the EW benchmark. Generally, they seemed to struggle more with the U.S stocks. The only algorithm that performed relatively well on U.S as well as European single factor data was the SAC. Notably, for the multi-factor datasets the simpler A2C and DDPG have performed better than the more complex architectures such as PPO, SAC and TD3. This suggests that different architectures have advantages in learning in different systems.

6. Conclusions

In this paper, we evaluate and benchmark state of the art Model-Free Reinforcement Learning model architectures against the Occam's Razor approach of equal-weight portfolio and the traditional mean-variance benchmarks. Evaluating the performance of these models on the European and American equities within a range of different contexts such as the single and multi-factor datasets, a complementary approach to portfolio management is proposed. This is done without the need to estimate the problematic parameters such as expected returns or the introduction of specific constraints to the SDF. Notably, it has been achieved using a sparse information set consisting only of price data and features transformed thereof.

The main conclusions of this work are summarized in four points. Firstly, we conclude that the model-free architectures have outperformed the Mean-Variance approaches as demonstrated by the results and the probabilistic assessment of the return series. The resulting portfolios were more stable as a consequence of the models successfully capturing a part of the nonlinear structure in the SDF. This ability to capture the mispricings has not been, however, as successful as the inclusion of risk-premia in the construction of the conditional mean-variance portfolio. The second conclusion ipso facto implies that the factor model has outperformed the RL approach in agreement with the findings of Chen et al. (2019). Third, we note that the performance against the $\frac{1}{N}$ portfolio is largely tied and it is inconclusive to say that these algorithms outperform this benchmark in this setting. Although notably in some contexts such as the European Growth and Momentum datasets it has performed very well, in others, such as the Size datasets in both U.S. and

EU equities, it fell short. The most consistent model architecture has turned out to be the SAC which hints at the inclusion of entropy within the model-free setting to be an important part of learning such nonlinear structures in a limited data setting as it introduces a lot of exploration into the framework. Fourth, we conclude that there are potential hedging benefits in using the RL approach as we have demonstrated that the models are successful at reducing left-tail risks out of the sample. This is certainly subject to the construction of a more complex decision system that would potentially make use of an ensemble model as different architectures performed well in different subsets of the data.

This paper is complementary to the current literature applying Machine Learning and Deep Learning in the Reinforcement Learning context and demonstrates the usefulness of such methods. Questions about interpretability as well as the use of coarser data frequency are subject to further research. Similarly, different asset classes and exposures should be tested to draw more general conclusions about the applicability of this approach. Since this is a rapidly developing field, an increasing number of papers considering similar applications are expected to be written in financial applications. New models from the field of computer science should be further considered for the Portfolio Allocation problem. As such, this paper provides a comprehensive review of the performance of the current state-of-the-art algorithms and provides a study similar to DeMiguel et al. (2009) in the context of Reinforcement Learning methods. Its use is of direct interest to portfolio managers as an alternative approach to construct and support investment strategies with an actionable output that consists of optimal weights similar to the Mean-Variance approach.

References

Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138–52160. doi:10.1109/ACCESS.2018.2870052.

Aguilar, O., & West, M. (2000). Bayesian dynamic factor models and portfolio allocation. *J. Bus. Econ. Stat.*, *18*, 338–357. URL: https://www.tandfonline.com/action/journalInformation?journalCode=ubes20. doi:10.1080/07350015.2000.10524875.

Andreou, E., & Ghysels, E. (2002). Detecting multiple breaks in financial market volatility dynamics. *J. Appl. Econom.*, *17*, 579–600. doi:10.1002/jae.684.

Aue, A., & Horváth, L. (2013). Structural breaks in time series. *J. Time Ser. Anal.*, *34*, 1–16. URL: http://doi.wiley.com/10.1111/j.1467-9892.2012.00819.x. doi:10.1111/j.1467-9892.2012.00819.x.

Bianchi, S. (2019). Lectures on Intermediate Financial Economics.

Black, F., & Litterman, R. (1992). Global Portfolio Optimization. *Financ. Anal. J.*, *48*, 28–43. URL: https://www.tandfonline.com/doi/abs/10.2469/faj.v48.n5.28. doi:10.2469/faj.v48.n5.28.

Bloomberg (2020). Bloomberg professional. Available at: Subscription Service (Accessed: 27th December 2020).

Carhart, M. M. (1997). On persistence in mutual fund performance. *J. Finance*, *52*, 57–82. URL: http://doi.wiley.com/10.1111/j.1540-6261.1997.tb03808.x. doi:10.1111/j.1540-6261.1997.tb03808.x.

Charpentier, A., Élie, R., & Remlinger, C. (2020). Reinforcement Learning in Economics and Finance. arXiv:2003.10014.

Chen, L., Pelgery, M., & Zhuz, J. (2019). Deep learning in asset pricing, . doi:10.2139/ssrn.3350138. arXiv:1904.00745.

Cong, L., Tang, K., Wang, J., & Zhang, Y. (2020). AlphaPortfolio for Investment and Economically Interpretable AI. *SSRN Electron. J.*, . URL: https://papers.ssrn.com/abstract=3554486. doi:10.2139/ssrn.3554486.

Dankwa, S., & Zheng, W. (2019). Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent. *ACM Int. Conf. Proceeding Ser.*, . doi:10.1145/3387168.3387199.

Danthine, J.-P., & Donaldson, J. B. (2014). *Intermediate financial theory*. academic press.

DeMiguel, V., Garlappi, L., & Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy? *Rev. Financ. Stud.*, *22*, 1915–1953. doi:10.1093/rfs/hhm075.

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *J. Mach. Learn. Res.*, *17*, 1–5. arXiv:1603.00943.

Elton, E. J., Gruber, M. J., & Spitzer, J. (2006). Improved estimates of correlation coefficients and their impact on optimum portfolios. *Eur. Financ. Manag.*, *12*, 303–318. doi:10.1111/j.1354-7798.2006.00322.x.

Fama, E. F., & French, K. R. (1992). The cross-section of expected stock returns. *the Journal of Finance*, *47*, 427–465.

Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *35th Int. Conf. Mach. Learn. ICML 2018*, *4*, 2587–2601. URL: http://arxiv.org/abs/1802.09477. arXiv:1802.09477.

Goldfarb, D., & Iyengar, G. (2003). Robust portfolio selection problems. *Math. Oper. Res.*, *28*, 1–38. URL: https://pubsonline.informs.org/doi/abs/10.1287/moor.28.1.1.14260. doi:10.1287/moor.28.1.1.14260.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gu, S., Kelly, B., & Xiu, D. (2020). Empirical Asset Pricing via Machine Learning. *Rev. Financ. Stud.*, *33*, 2223–2273. URL: https://academic.oup.com/rfs/article/33/5/2223/5758276. doi:10.1093/rfs/hhaa009.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *35th Int. Conf. Mach. Learn. ICML 2018*, *5*, 2976–2989. URL: http://arxiv.org/abs/1801.01290. arXiv:1801.01290.

Harvey, C. R., Liu, Y., & Zhu, H. (2016). ... and the Cross-Section of Expected Returns. *Rev. Financ. Stud.*, *29*, 5–68. URL: http://faculty.fuqua. doi:10.1093/rfs/hhv059.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., & Wu, Y. (2018). Stable baselines. https://github.com/hill-a/stable-baselines.

iShares (a). iShares U.S. Real Estate ETF | IYR | US Class. URL: https://www.ishares.com/us/products/239520/ishares-us-real-estate-etf.

iShares (b). iShares U.S. Telecommunications ETF | IYZ. URL: https://www.ishares.com/us/products/239523/ishares-us-telecommunications-etf.

Kempthorne, P. (2013). *Factor Models MIT 18.S096 Lecture 15*. Technical Report MIT.

Kozak, S., Nagel, S., & Santosh, S. (2020). Shrinking the cross-section. *J. financ. econ.*, *135*, 271–292. doi:10.1016/j.jfineco.2019.06.008.

Kraft, D. (1988). A Software Package for Sequential Quadratic Programming. *Tech. Rep. Dfvlr-Fb*, *88*, 33.

Lapan, M. (2018). *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc.*. International Conference on Learning Representations, ICLR. URL: `https://goo.gl/J4PIAz`. `arXiv:1509.02971`.

Liu, T., & Gong, X. (2020). Analyzing time-varying volatility spillovers between the crude oil markets using a new method. *Energy Economics*, *87*, 104711. doi:`10.1016/j.eneco.2020.104711`.

MacKay, D. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

Maillard, S., Roncalli, T., & Teiletche, J. (2011). On the Properties of Equally-Weighted Risk Contributions Portfolios. *SSRN Electron. J.*, . doi:`10.2139/ssrn.1271972`.

Mandic, D. (2019). Reinforcement learning for portfolio management. *arXiv*, . `arXiv:1909.09571`.

Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, *7*, 77–91. doi:`10.2307/2975974`.

Merton, R. C. (1980). On estimating the expected return on the market. *Journal of Financial Economics*, *8*, 323–361. doi:`10.1016/0304-405X(80)90007-0`.

Meucci, A. (2005). *Risk and Asset Allocation*. Springer Finance Textbooks. Springer. URL: `https://books.google.sk/books?id=Qc8KWWtUokcC`. doi:`10.1007/3-540-27904-0`.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *33rd Int. Conf. Mach. Learn. ICML 2016*, *4*, 2850–2869. URL: `http://arxiv.org/abs/1602.01783`. `arXiv:1602.01783`.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning, . URL: `http://arxiv.org/abs/1312.5602`. `arXiv:1312.5602`.

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). *Performance Functions and Reinforcement Learning For Trading Systems and Portfolios*. Technical Report.

Necchi, P. (2016). *Policy Gradient Algorithms for the Asset Allocation Problem*. Master's thesis Politecnico Di Milano.

Poole, D. L., & Mackworth, A. K. (2017). *Artificial Intelligence*. Cambridge University Press. URL: `https://www.cambridge.org/core/product/identifier/9781108164085/type/book`. doi:`10.1017/9781108164085`.

Rice, J. A. (2006). *Mathematical statistics and data analysis*. Nelson Education.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*, 206–215. doi:`10.1038/s42256-019-0048-x`.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. URL: `https://arxiv.org/abs/1707.06347v2`. `arXiv:1707.06347`.

Sharpe, W. F. (1964). Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk. *The Journal of Finance*, *19*, 425–442. URL: `http://www.jstor.org/stable/2977928?origin=crossref`. doi:`10.2307/2977928`.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–489. URL: `https://www.nature.com/articles/nature16961`. doi:`10.1038/nature16961`.

SPDR (). Fund Finder - State Street Global Advisors. URL: `https://www.ssga.com/nl/en{_}gb/institutional/etfs/fund-finder?g=assetclass{%}3Aequity!noLabel*Sectors{%}7Cmarket-cap*Large{%}5EMid`.

Sutton, R., & Barto, A. (2018). *Reinforcement Learning: An Introduction*. (2nd ed.). MIT Press.

Tesau, C., & Tesau, G. (1995). Temporal Difference Learning and TD-Gammon. *Commun. ACM*, *38*, 58–68. URL: `https://dl.acm.org/doi/10.1145/203330.203343`. doi:`10.1145/203330.203343`.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling Network Architectures for Deep Reinforcement Learning. *33rd Int. Conf. Mach. Learn. ICML 2016*, *4*, 2939–2947. URL: `http://arxiv.org/abs/1511.06581`. `arXiv:1511.06581`.

Weng, L. (2018). Policy Gradient Algorithms. URL: `https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html{#}sac`.

Williams, R. J. (1992). *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*. Technical Report. URL: `https://link.springer.com/content/pdf/10.1007/BF00992696.pdf`.

Appendix A. Mean-Variance Optimization

Quadratic Risk Utility

Quadratic utility is the most common formulation of the portfolio allocation problem with a non negative parameter λ controlling for the level of risk-aversion determining the efficient frontier of optimal portfolios. In the first stage of solving this problem we solve:

$$\begin{aligned} \min \ & \lambda \mathbf{w}^T \Sigma \mathbf{w} - \boldsymbol{\mu} \mathbf{w}, \ \lambda \in \mathbb{R}^+ \\ \text{s.t.} \ & \mathbf{w}^T \mathbf{1} = 1 \\ & \mathbf{w} \succeq 1. \end{aligned}$$

In the second step we choose a satisfaction measure which conforms the definition of a reward function $r(a_t, s_t)$ such as the sample Sharpe Ratio $\frac{\hat{\mu}_{test} - r_f}{\hat{\sigma}_{test}}$ and search over the set of optimal portfolios such that:

$$\mathbf{w}^* = \underset{\lambda}{\operatorname{argmax}} \, (r(a_t, s_t)).$$

We search for $\mathbf{w}^*$ greedily since λ is a monotonically increasing function and thus we obtain a global maximum.

Global Minimum Variance

The advantage of a GMV portfolio is that there is no need for an estimate of $\boldsymbol{\mu}$ as in the standard mean-variance formulation. We are able solve for this portfolio either analytically or using a simple convex program that reduces to the case where the least amount of risk is taken on:

$$\begin{aligned} \min \ & \mathbf{w}^T \Sigma \mathbf{w} \\ \text{s.t.} \ & \mathbf{w}^T \mathbf{1} = 1 \\ & \mathbf{w} \succeq 1. \end{aligned}$$

Equal Risk Contribution

Risk contribution portfolios look at the optimization problem from a perspective of marginal volatilities of each of the assets and their contribution to the overall risk of the portfolio. An Equal Risk Contribution portfolio forces each of the assets to contribute an equal amount of risk such that the weights adjust the volatility contribution to be at parity with other assets.

We can define the volatility of a portfolio as $\sigma_p = \sqrt{\mathbf{w}^T \Sigma \mathbf{w}}$ and for each asset $i \in \mathcal{E}$ the marginal risk contribution:

$$\sigma_i = \frac{\partial \sigma_p}{\partial w_i} \Longrightarrow \sigma_p = \sum_i^N \sigma_i, \qquad \text{(by Euler's theorem)}$$

since σ_p is a homogeneous function of degree one expressible as:

$$\sigma_p(w) = \sum_{i=1}^{N} w_i \frac{\partial \sigma_i}{\partial w_i}.$$

For a complete proof see Maillard et al. (2011). Expressing the above equation in a vector form we obtain the convex objective we can minimize:

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} \left\{ \frac{\sigma_p}{N} - w_i \cdot \frac{\sum_{i \in \mathcal{E}} w}{\sigma_p} \right\} \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{1} = 1 \\ & \mathbf{w} \succeq 1. \end{aligned}$$

Factor Constrained Portfolio

Given a time-series factor model of the form 30 we can estimate the covariance matrix as:

$$\hat{\Sigma} = \mathbf{F} \Sigma_{\mathbf{F}} \mathbf{F}^T + \hat{\boldsymbol{\psi}},$$

where $\mathbf{F}$ is the vector of factor loadings, $\Sigma_{\mathbf{F}}$ the sample covariance of the factor loadings and $\hat{\psi}$ is the diagonal matrix of the form $\text{diag}(\hat{\sigma}_1, \ldots, \hat{\sigma}_N)$. We can then solve for a factor constrained portfolio that limits the exposure on a particular factor by the vector υ. We use the value of 0.2 or 20% as a limit of the portfolio exposure for a particular factor. Given the fact that we only use long-only portfolio it is hard to decrease this limit as there might not be such a solution that would satisfy this constraint. We express the problem as a maximization of the objective as described in Diamond & Boyd (2016):

$$\begin{aligned} \max \quad & \mu^T \mathbf{w} - \gamma(\upsilon \hat{\Sigma} \upsilon + \mathbf{w}^T \hat{\boldsymbol{\psi}} \mathbf{w}) \\ \text{s.t.} \quad & \mathbf{w}^T \mathbf{1} = 1 \\ & \mathbf{F}^T \mathbf{w} = \upsilon \\ & \mathbf{w} \succeq 1. \end{aligned}$$

Appendix B. Pseudo Code

Figure B.11 Q-learning algorithm for the portfolio allocation problem.

```
    Input: investment universe $\mathcal{E}$, price history $\mathbf{P}$, feature matrix $\mathbf{X}$
    Output $Q^*(s,a) \to w^*$
 1: procedure Q-LEARNING PROCEDURE
 2:     Initialize $Q(s,a)\ \forall\ s \in \mathcal{S}$ with $Q(T,\cdot) = 0,\ \alpha \in [0,1]$
 3:     repeat
 4:         for $t = 0, \dots, T$ do
 5:             select an action $\mathbf{a'_t}$ based on a policy $a'_t = \max Q(s,a)$.
 6:             take action $\mathbf{a'_t}$ and observe $\{r, s'\}$
 7:             update the Q-table using Eq. (14):
 8:             $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_a Q(s',a) - Q(s,a)]$
 9:             if $T$ mod $h_{retrain} = 0$ then Update network parameters $\tilde{\theta} \leftarrow \hat{\theta}$
10:             end if
11:         end for
12:     until $T > T_{max}$ or terminal.
13: end procedure
```

Figure B.12 Policy Gradient algorithm for the portfolio allocation problem.

```
    Input: investment universe $\mathcal{E}$, price history $\mathbf{P}$, feature matrix $\mathbf{X}$
    Output $\pi^*(s,a) = w^*$
 1: procedure POLICY GRADIENT UPDATE
 2:     Initialize cache and parameters $\{C\},\ \boldsymbol{\theta}_0,\ \alpha \in [0,1]$
 3:     repeat
 4:         for $t = 0, \dots, T$ do
 5:             observe $\{\mathbf{s}_t, \mathbf{r}_t\}$
 6:             obtain action (by sampling from MDP) or $\pi$
 7:             update actor network parameters $\theta \leftarrow \theta + \alpha Q(\mathbf{s},\mathbf{a}) \nabla_\theta \ln \pi(\mathbf{a},\mathbf{s})$
 8:             cache rewards and gradients in $\{C\}$
 9:             update critic network with TD error (if present)
10:         end for
11:     until convergence.
12: end procedure
```

Figure B.13 DDPG algorithm

```
    Initialize actor and critic networks $Q(s, a \mid \theta),\ \mu(s \mid \theta_\mu)$ with random weights
    along with target networks $Q', \mu', \theta_Q, , \theta_\mu$ respectively.
    Initialize experience replay cache $\mathcal{C}$
 1: procedure
 2:     for $t = 0, \dots, S$ do
 3:         Initialize a noise process $\mathcal{N}$, Observe $s_0$
 4:         for $t = 0, \dots, T$ do
 5:             select action $a_t = \mu(s_t) + \mathcal{N}$
 6:             observe $r_t \mid s_t$ and $s_{t+1}$ and store the array in the replay cache
 7:             sample from the cache to obtain a minibatch $(s_i, a_i, r_i, s_{i+1}) \overset{iid}{\sim} \{\mathcal{C}\}$
 8:             update critic by minimizing:
                    $L = \frac{1}{N}\sum_i (r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}) \mid \theta^{Q'})) - Q(s_i, a_i))^2$
 9:             update actor using the sampled policy gradient:
                    $\nabla_{\theta^\mu} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a) \nabla_{\theta^\mu} \mu(s)$
10:             perform a soft-update of weights for:
                    $Q',\ \mu'\colon \theta^{target} \leftarrow \tau\theta^{net} + (1 - \tau)\theta^{target}$
11:         end for
12:     end for
13: end procedure
```

Figure B.14 TD3 algorithm

```
    Initialize actor and critic networks $Q(s, a \mid \theta),\ \mu(s \mid \theta_\mu)$ with random weights
    along with target networks $Q',\ \mu'\ \theta_Q, \theta_\mu$ respectively
    initialize experience replay cache $\mathcal{C}$
 1: procedure
 2:     for $t = 0, \dots, T$ do
 3:         initialize a noise process $\mathcal{N}$, Observe $s_0$
 4:         select action $a_t = \mu(s_t) + \mathcal{N}$
 5:         observe $r_t \mid s_t$ and $s_{t+1}$ and store the array in $\mathcal{C}$
 6:         sample from the cache to obtain a minibatch $(s_i, a_i, r_i, s_{i+1}) \overset{iid}{\sim} \{C\}$
 7:         smooth the target policy $\tilde{a}_t = \pi(s)_t + \epsilon,\ \epsilon = \text{clip}(\mathcal{N}(0, \sigma), -c, c)$, where
              $c$ is in the neighbourhood of $a_t$
 8:         update the critic nets according to the loss in Eq. (29) using $\tilde{a}_t$
 9:         if $t$ mod $h_{retrain}$ then
10:             update actor weights using the sampled policy gradient:
                    $\nabla_{\theta^\mu} \approx \frac{1}{N}\sum_i \nabla_a Q(s, a) \nabla_{\theta^\mu} \mu(s)$
11:             update target actor and critic networks with a soft update
12:         end if
13:     end for
14: end procedure
```

Appendix C. Model Hyperparameters

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.01	0.001	0.0001	0.005
γ		0.001		0.1	
batch		512	512	256	512
steps	60K	80K	60K	60K	60K

SPX Growth 50

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.01	0.001	0.0001	0.01
γ		0.001		0.1	
batch		512	512	256	512
steps	60K	80K	60K	60K	60K

SPX Momentum 50

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.01	0.01	0.0001	0.005
γ		0.001		0.1	
batch		256	256	256	512
steps	60K	80K	60K	60K	60K

SPX Size 50

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.001	0.01	0.0001	0.005
γ		0.001		0.1	
batch		256	256	256	512
steps	60K	80K	60K	60K	60K

BE 500 Growth 50

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.001	0.01	0.0001	0.005
γ		0.001		0.1	
batch		256	256	128	256
steps	60K	80K	60K	60K	60K

BE500 Momentum 50

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.001	0.0001	0.001	0.005
γ		0.001		0.1	
batch		256	512	256	512
steps	60K	80K	60K	60K	60K

BE 500 Size 50

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.001	0.001	0.0001	0.005
γ		0.001		0.1	
batch		256	256	256	256
steps	60K	80K	60K	60K	60K

SPX Growth + Size 100

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.01	0.001	0.001	0.005
γ		0.001		0.1	
batch		512	128	512	512
steps	60K	80K	60K	60K	60K

SPX Momentum + Size 100

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.001	0.001	0.0001	0.005
γ		0.001		0.1	
batch		256	256	256	256
steps	60K	80K	60K	60K	60K

BE500 Growth + Size 100

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.01	0.001	0.001	0.005
γ		0.001		0.1	
batch		512	128	512	512
steps	60K	80K	60K	60K	60K

BE500 Momentum + Size 100

	A2C	PPO	DDPG	SAC	TD3
α	0.01	0.001	0.001	0.0001	0.005
γ		0.001		0.1	
batch		512	128	256	512
steps	60K	80K	60K	60K	60K

Sector ETF