Achieving Mean–Variance Efficiency by Continuous-Time Reinforcement Learning

Yilie Huang
Yanwei Jia
yh2971@columbia.edu
yj2650@columbia.edu
Department of Industrial Engineering and Operations Research, Columbia University
New York, New York, USA

Xun Yu Zhou
Department of Industrial Engineering and Operations Research & Data Science Institute, Columbia University
New York, New York, USA
xz2574@columbia.edu

ABSTRACT

We conduct an extensive empirical analysis to evaluate the performance of the recently developed reinforcement learning algorithms by Jia and Zhou [11] in asset allocation tasks. We propose an efficient implementation of the algorithms in a dynamic mean-variance portfolio selection setting. We compare it with the conventional plug-in estimator and two state-of-the-art deep reinforcement learning algorithms, deep deterministic policy gradient (DDPG) and proximal policy optimization (PPO), with both simulated and real market data. On both data sets, our algorithm significantly outperforms the others. In particular, using the US stocks data from Jan 2000 to Dec 2019, we demonstrate the effectiveness of our algorithm in reaching the target return and maximizing the Sharpe ratio for various periods under consideration, including the period of the financial crisis in 2007-2008. By contrast, the plug-in estimator performs poorly on real data sets, and PPO performs better than DDPG but still has lower Sharpe ratio than the market. Our algorithm also outperforms two well-diversified portfolios: the market and equally weighted portfolios.

CCS CONCEPTS

• **Applied computing** → *Economics*; **Decision analysis**; • **Theory of computation** → **Reinforcement learning**.

KEYWORDS

portfolio choice, dynamic mean-variance analysis, reinforcement learning, actor-critic, policy gradient

ACM Reference Format:
Yilie Huang, Yanwei Jia, and Xun Yu Zhou. 2022. Achieving Mean–Variance Efficiency by Continuous-Time Reinforcement Learning. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561760

1 INTRODUCTION

In this paper, we revisit the asset allocation problem in a dynamically traded market where a small investor aims to achieve mean–variance efficiency in a finite investment horizon through reinforcement learning (RL, for short) techniques. Pioneered by the introduction of the mean–variance analysis framework for a static asset allocation problem in Markowitz [21], this problem soon became one of the central topics in the classical modern portfolio theory and the mainstream of quantitative investment. However, despite significant theoretical implications, practical implementation of the mean–variance efficient portfolio is challenging. First, most practical applications of mean–variance analysis are restricted to static settings to this day [15]. Yet, adopting such static strategies myopically often becomes inefficient from the dynamic perspective. Second, accurately estimating the moments of asset returns for inherently non-stationary financial markets is notoriously difficult, especially for the expected returns [20, 22]. The computed portfolios (by the analytical solution) to the mean–variance problem in the static setting are already sensitive to such estimation errors [1–3], and become much worse for the dynamic one. Therefore, overcoming the difficulty of those errors and sensitivity and achieving mean–variance efficiency in the dynamic environment remain largely an open question.

Due to the recent development in machine learning and optimization techniques, there is a trending shift in the practice of dynamic decision-making under uncertainty – RL-based approaches have become more popular and recognized in many different application domains. The research questions of this paper are: how do we apply RL techniques for dynamic portfolio optimization problems, and how do they perform compared with conventional methods and general-purpose RL algorithms, both in theory (with simulated data and known ground truth) and in practice (with real market data)?

Our main contributions are two-fold. First, we devise RL algorithms for the asset allocation problem under a continuously traded market for a small investor who is a price-taker with the mean-variance preference. We discuss its implementation based on the framework and theoretical analysis for general continuous-time decision-making problems in Jia and Zhou [11, 12], Wang et al. [37] and various ways to enhance its performance in the particular context of portfolio problems. These techniques include offline pre-training, off-policy learning, mini-batch, and incorporating constraints in terms of leverage and rebalancing frequency.

Second, we provide a thorough comparative study between the proposed algorithm and various popular methods using multiple

ICAIF '22, November 2–4, 2022, New York, NY, USA

ACM ISBN 978-1-4503-9376-8/22/10…$15.00
https://doi.org/10.1145/3533271.3561760

common performance metrics on the simulated data and the U.S. stocks data from 2000 to 2019 (and the 1990-1999 period is burnt-in for training). In particular, we compare our algorithm with two state-of-the-art, general-purpose RL algorithms for continuous action space – the deep deterministic policy gradient (DDPG) algorithm [17, 33] and the proximal policy optimization (PPO) algorithm [32]. In the real market data study, besides DDPG and PPO, we use two computation-free benchmarks: the market portfolio (approximated by S&P 500 index, which is the market capitalization-weighted portfolio) and the equally weighted portfolio. Such seemingly naive benchmarks are not only actual choices of many investors but also are shown to be robust and reliable and difficult to beat empirically by DeMiguel et al. [5]. In addition, we also compare with the conventional model-based methods, that is, to statistically estimate the model coefficients and then plug the estimated coefficients into the model-based optimal solution. Our results suggest that the proposed RL method can significantly outperform all the alternative methods.

Despite a long history of the RL research, its application to finance only started recently, in the wake of the booming FinTech revolution and the emerging technology-driven financial services such as algorithmic trading and robo-advising [44]. There have been studies that aim to apply RL in dynamic portfolio optimization [8, 14, 25–30, 34, 36]. More recently, deep neural networks are introduced to incorporate high-dimensional stock-level features to learn policies for trading and hedging [4, 16, 19, 41, 43, 43]. However, these works typically rely on discrete-time MDPs to derive their algorithms and do not have comprehensive comparative empirical studies. To develop a theory for continuous-time RL, Wang et al. [37] propose a general exploratory stochastic control framework with an entropy regularizer to study and devise RL algorithms. With the entropy regularizer, the policy as a probability distribution is incentivized to be more exploratory, that is, to have a higher chance of deviating from the center. Such randomization turns out to be useful in algorithm design as well. Under this framework, Jia and Zhou [11] propose model-free actor–critic policy gradient algorithms for general decision-making problems. We apply a variant of those algorithms for the mean–variance asset allocation problem in this paper, and provide comprehensive comparative experimental (both simulated and empirical) results to validate such a model-free RL approach.

The rest of the paper is organized as follows: We describe the formulation of the portfolio choice problem in Section 2 and discuss the theoretical foundation of RL algorithms in Section 3. The algorithm design is discussed in Section 4. We present numerical results in Section 5 and Section 6 concludes.

2 DESCRIPTION OF PORTFOLIO CHOICE PROBLEMS

We first describe a general continuously traded market. There are $d+1$ assets, and we always say the 0-th asset is a risk-free asset with interest rate r, whose price is denoted by S_t^0, satisfying $\mathrm{d}S_t^0 = rS_t^0\mathrm{d}t$. The other d assets are risky assets, whose prices are denoted by $S_t^1, \cdots, S_t^d$. The price movement of risky assets is driven by an exogenous d-dimensional standard Brownian motion $\boldsymbol{W}_t$ and follows

$$\mathrm{d}S_t^i = S_t^i \left[\boldsymbol{\mu}^i \mathrm{d}t + \sum_{j=1}^{d} \boldsymbol{\sigma}^{ij} \mathrm{d}\boldsymbol{W}_t^j \right], \quad i = 1, 2, \cdots, d, \tag{1}$$

where the vector $\boldsymbol{\mu} = (\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \cdots, \boldsymbol{\mu}^d)^\top \in \mathbb{R}^d$ is the instantaneous expected return, and the matrix $\boldsymbol{\sigma} = (\boldsymbol{\sigma}^{ij})_{1\le i\le d, 1\le j\le d} \in \mathbb{R}^{d\times d}$, and $\Sigma = \boldsymbol{\sigma}\boldsymbol{\sigma}^\top \in \mathbb{S}_{++}^d$, a symmetric and positive definite matrix, is the covariance of the returns of risky assets.[1] We assume the above mentioned price processes are all well-defined and adapted to the the filtered probability space $(\Omega, \mathcal{F}, \mathbb{P}; (\mathcal{F}_t)_{t\ge 0})$ satisfying the usual conditions.

Consider an agent with an initial wealth x_0 and a pre-specified investment horizon $T > 0$. The agent is a price taker and can continuously trade in the market with no frictions, e.g., there are no transaction costs, no taxes, no bid–ask spread, no price impact, and no order execution uncertainty. We denote the agent's portfolio choice at time t by $\boldsymbol{a}_t \in \mathbb{R}^d$, where $\boldsymbol{a}_t^i$ stands for the discounted dollar amount (or, equivalently, $\boldsymbol{a}_t^i S_t^0$ in nominal dollar amount) invested in the i-th risky asset at time t for $1 \le i \le d$. The sequence of portfolio choice $\{\boldsymbol{a}_t, 0 \le t \le T\}$ is restricted to be a square-integrable process throughout this paper; see [18] for this regularity condition. We denote by $\{x_t^{\boldsymbol{a}}, 0 \le t \le T\}$ the discounted wealth process of the agent given a portfolio process $\{\boldsymbol{a}_t, 0 \le t \le T\}$,

There are no extra constraints (e.g., the leverage constraints) on agent's portfolio choice as long as it is self-financing. Then the agent's discounted wealth satisfies

$$\mathrm{d}x_t^{\boldsymbol{a}} = (\boldsymbol{\mu} - r\boldsymbol{e}_d)^\top \boldsymbol{a}_t \mathrm{d}t + \boldsymbol{a}_t^\top \boldsymbol{\sigma} \mathrm{d}\boldsymbol{W}_t, \; 0 \le t \le T, \tag{2}$$

with the initial wealth $x_0^{\boldsymbol{a}} = x_0$, where $\boldsymbol{e}_d = (1, \cdots, 1)^\top \in \mathbb{R}^d$ is a d-dimensional vector with all entries 1.

The agent aims to find the mean-variance efficient allocation in this dynamically traded market. As the continuous-time counterpart to the Markowitz problem [21], it is to minimize the variance of the portfolio while achieving a given expected target return:

$$\begin{aligned} &\min_{\boldsymbol{a}_t, 0\le t\le T} \mathrm{Var}\left[x_T^{\boldsymbol{a}}\right] \\ &\text{subject to } \mathbb{E}\left[x_T^{\boldsymbol{a}}\right] = z \end{aligned} \tag{3}$$

where z is the target expected terminal wealth that is pre-specified at $t = 0$ by the agent as part of the agent's preference. Larger z indicates that the agent pursues higher return and hence is less risk-averse.

As a remark, this is not a standard stochastic control problem due to the presence of the variance term in (3). This term causes time-inconsistency so the dynamic programming principle does not apply directly. Zhou and Li [46] introduce the method of using a Lagrange multiplier w to transform the problem into an unconstrained expected quadratic utility maximization problem:

$$\min_{\boldsymbol{a}_t, 0\le t\le T} \mathbb{E}[(x_T^{\boldsymbol{a}} - w)^2] - (w - z)^2 \tag{4}$$

and then find a proper multiplier w to enforce the constraint.

If the market coefficients are known, then standard stochastic control theory [7, 42] can be applied to solve (4). Practically, one

[1] In this paper, for simplicity, we assume these are all unknown constants. Alternatively, they can be unknown deterministic functions of some factors such as stock-specific features or macroeconomic variables.

can compute the optimal policy to (4) by plugging in the estimated model coefficients. This is the classical school of *separation principle* between estimation and optimization [40]. However, it is now well known that it is hard if not impossible to accurately estimate these market coefficients. In particular, the mean parameter is known to be "blurred" [20]. On the other hand, solutions to the problem are extremely sensitive to coefficients, largely due to the need of calculating the inverse of the covariance matrix.

In contrast to the above conventional plug-in estimators, the emerging RL approach focuses on interacting with an unknown (market) environment and learning optimal strategies directly. This approach is often referred to as *model-free.* We emphasize here that a model-free approach does not mean there is no model; indeed, there is a *structural* underlying model for generating data (e.g., a Markov chain, a diffusion process, a jump-diffusion process, among others) but we do not know the model coefficients and do not aim to estimate them. From the theoretical perspective, any provable statement is based upon some assumptions regarding how data is generated. From the implementation perspective, every algorithm takes input data and produces an output (a final solution) regardless of the black box in between, and hence is free from any model as long as data is available.

Lastly, observe that the (discounted) wealth equation (2) can also be written as

$$\mathrm{d}x_t^{\boldsymbol{a}} = \sum_{i=1}^{d} \boldsymbol{a}_t^i \frac{\mathrm{d}S_t^i}{S_t^i} - \boldsymbol{e}_d^\top \boldsymbol{a}_t \frac{\mathrm{d}S_t^0}{S_t^0}, \tag{5}$$

where $\frac{\mathrm{d}S_t^i}{S_t^i}$ and $\frac{\mathrm{d}S_t^0}{S_t^0}$ are the returns of the risky and risk-free assets respectively, and can be observed directly from data without the knowledge of market coefficients. This observation facilitates solving the asset allocation problem in the model-free and data-driven fashion.

3 THE FOUNDATION OF REINFORCEMENT LEARNING ALGORITHMS

The design of an RL algorithm typically involves answering three questions: how to generate trial-and-error policies to interact with the environment for learning, how to evaluate the performance of a given policy, and how to update the policy to improve the performance. We follow the recent works on continuous-time RL [11, 12, 37] to address these three questions respectively.

Note that in the mean–variance problem, there is a multiplier w in (4). It is the usual Lagrange multiplier in the prime-dual approach for the constrained optimization problem.

3.1 Deterministic versus Stochastic Policies

Although the optimal policy ought to be a deterministic mapping from the state space to the portfolio space, the agent takes a randomized portfolio process in order to broaden the interaction with the market. Specifically, the agent searches over the following space of probability density functions:

$$\left\{\pi : (t,x) \mapsto \pi(\cdot|t,x) \in \mathcal{P}(\mathbb{R}^d)\right\},$$

where $\mathcal{P}(\mathbb{R}^d)$ denotes the set of all probability density functions on $\mathbb{R}^d$.[2] At each time t, a portfolio $\boldsymbol{a}_t$ is independently generated from the distribution $\pi(\cdot|t,x_t)$, denoted as $\boldsymbol{a}_t \sim \pi(\cdot|t,x_t)$. Such a rule to generate the choice is called a *stochastic policy.* By contrast, a *deterministic policy* refers to the case where π degenerates into a point mass (a Dirac measure).

Under a stochastic policy π, the dynamics of wealth are no longer (2). Using the notion of relaxed stochastic control [45], Wang et al. [37] propose to use the following dynamics to theoretically describe the "averaged" (over randomization) wealth process under a stochastic policy:

$$\begin{aligned}\mathrm{d}\tilde{x}_t^\pi =& (\boldsymbol{\mu} - r\boldsymbol{e}_d)^\top \boldsymbol{m}(t,\tilde{x}_t^\pi;\pi)\mathrm{d}t \\ &+ \boldsymbol{m}(t,\tilde{x}_t^\pi;\pi)^\top \boldsymbol{\sigma}\mathrm{d}\boldsymbol{W}_t + \sqrt{\langle \Sigma, \boldsymbol{C}(t,\tilde{x}_t^\pi;\pi)\rangle}\mathrm{d}\bar{W}_t,\end{aligned} \tag{6}$$

where $\bar{W}_t$ is an independent standard Brownian motion that represents the randomness of generating choices from a stochastic policy, and $\langle \Sigma, \boldsymbol{C}\rangle = trace(\Sigma^\top \boldsymbol{C})$ stands for the inner product between two matrices. Here $\boldsymbol{m}(t,x;\pi), \boldsymbol{C}(t,x;\pi)$ are the mean and covariance matrix of the distribution $\pi(\cdot|t,x)$ respectively, with

$$\boldsymbol{m}(t,\tilde{x}_t^\pi;\pi) = \int_{\mathbb{R}^d} \boldsymbol{a}\pi(\boldsymbol{a}|t,\tilde{x}_t^\pi)\mathrm{d}\boldsymbol{a},$$

$$\boldsymbol{C}(t,\tilde{x}_t^\pi;\pi) = \int_{\mathbb{R}^d} [\boldsymbol{a} - \boldsymbol{m}(t,\tilde{x}_t^\pi;\pi)][\boldsymbol{a} - \boldsymbol{m}(t,\tilde{x}_t^\pi;\pi)]^\top \pi(\boldsymbol{a}|t,\tilde{x}_t^\pi)\mathrm{d}\boldsymbol{a}.$$

Therefore (6) forms a closed stochastic differential equation system for any given policy π.

To avoid possible degeneracy of a stochastic policy into a deterministic one, we further add an entropy term to regularize the policy and encourage exploration. The entropy regularizer is related to the soft-max approximation and Boltzmann exploration [9, 37, 47]. With this entropy regularization, our problem is reformulated as follows:

$$\begin{aligned}\min_{\pi\in\mathcal{S}(0,x_0)} \mathbb{E}\left[\left(\tilde{x}_T^\pi - w\right)^2 + \gamma\int_0^T\int_{\mathbb{R}^d} \pi(\boldsymbol{a}|t,\tilde{x}_t^\pi)\log\pi(\boldsymbol{a}|t,\tilde{x}_t^\pi)\mathrm{d}\boldsymbol{a}\mathrm{d}t\right] \\ - (w-z)^2,\end{aligned} \tag{7}$$

where $\mathcal{S}(0,x_0)$ is the set of all admissible stochastic policies starting from time 0 and initial wealth x_0, and $\gamma \geq 0$ is called a temperature parameter and a tuning parameter for the algorithm. Larger γ means higher weight will be placed on entropy and hence exploration is more encouraged.

We emphasize that (6) is mainly for the purpose of theoretical analysis on the averaged property of the wealth process under a given stochastic policy π. The actual portfolio processes and the corresponding wealth trajectories will still be generated from (5) with $\boldsymbol{a}_t \sim \pi(\cdot|t,x_t)$, which do not require any knowledge about the market coefficients.

The theoretical formulation (6) leads to useful properties for the RL problem. The first important property is the optimality of normal distributions for stochastic policies, as stated in Lemma 1. The proof of Lemma 1 is parallel to that of Wang and Zhou

[2] The density functions here can be replaced by any distribution functions. We restrict to density functions since they are the most commonly used and compatible with the entropy regularization to be introduced subsequently.

[38, Theorem 1], and hence omitted here. However, the essential reason behind this result is that the entropy regularization leads to a Gibbs measure, whose density function is the exponential of the Hamiltonian scaled by the temperature parameter. The Hamiltonian is a quadratic function of the control (portfolio) variable in the current mean–variance setting; hence the normal distribution.

Lemma 1. *The optimal stochastic policy with the entropy regularizer is a normal distribution.*

This suggests that in the current mean–variance setting the best way for exploration is to generate policies according to normal distributions. Note that normal distributions are easy and efficient to sample. Hence, in the following, it suffices to restrict our attention to the class of normal distributions: $\pi(\cdot|t,x) = \mathcal{N}\left(\cdot|\boldsymbol{m}(t,x;\pi), \boldsymbol{C}(t,x;\pi)\right)$ where $\boldsymbol{m}(t,x;\pi)$ is the mean vector and $\boldsymbol{C}(t,x;\pi)$ the covariance matrix. This, in turn, will subsequently lead to a natural policy approximator.

Given the above parameterization, we can further specify the objective function (7) by calculating the entropy of a multivariate normal distribution:

$$\begin{aligned}\min_{\boldsymbol{m},\boldsymbol{C}} \mathbb{E}&\left[\left(\tilde{x}_T^\pi - w\right)^2 - \frac{\gamma}{2}\int_0^T \log\det \boldsymbol{C}(t,\tilde{x}_t^\pi;\pi)\mathrm{d}t\right] \\ &- \frac{\gamma d T}{2}\log 2\pi e - (w-z)^2.\end{aligned} \tag{8}$$

The theoretical solution to (8) can be characterized by an HJB equation:

$$\begin{aligned}\inf_{\boldsymbol{m},C}&\left\{\frac{\partial J^*}{\partial t} + (\boldsymbol{\mu} - r\boldsymbol{e}_d)^\top \boldsymbol{m}\frac{\partial J^*}{\partial x} + \frac{1}{2}\left[\boldsymbol{m}^\top\Sigma\boldsymbol{m} + \langle\Sigma, C\rangle\right]\frac{\partial^2 J^*}{\partial x^2}\right.\\ &\left.- \frac{\gamma}{2}\left[\log\det C + d\log 2\pi e\right]\right\} = 0,\end{aligned}$$

and the optimality condition is

$$\begin{aligned}\boldsymbol{m}^*(t,x) &= -\frac{\frac{\partial J^*}{\partial x}(t,x;w)}{\frac{\partial^2 J^*}{\partial x^2}(t,x;w)}\Sigma^{-1}(\boldsymbol{\mu} - r\boldsymbol{e}_d)\\ \boldsymbol{C}^*(t,x) &= \frac{\gamma}{\frac{\partial^2 J^*}{\partial x^2}(t,x;w)}\Sigma^{-1},\end{aligned} \tag{9}$$

where $J^*(t,x;w)$ is the optimal value function for problem (8).

Notice that (9) provides the structure of the optimal policy but we are not going to solve J^* using the estimated coefficients even though the analytical solutions to (9) are available (see in Appendix A). Instead, we use RL.

3.2 Policy Evaluation

Policy evaluation is a step to estimate the expected payoff of a given, fixed policy, based on which the agent decides how to update the policy. In our case, it is to estimate the expected payoff (7) for a given stochastic policy π and a given multiplier w. Note it requires learning the whole function J of (t,x), known as the value function, where

$$\begin{aligned}J(t,x;\pi;w) =&\mathbb{E}\left[\left(\tilde{x}_T^\pi - w\right)^2 - \frac{\gamma}{2}\int_t^T \log\det \boldsymbol{C}(s,\tilde{x}_s^\pi;\pi)\mathrm{d}s\middle|\tilde{x}_t^\pi = x\right]\\ &- \frac{\gamma d(T-t)}{2}\log 2\pi e - (w-z)^2.\end{aligned}$$

The celebrated Feynman–Kac formula stipulates that $J(\cdot,\cdot,\cdot;\pi;w)$ solves the following equation:

$$\begin{aligned}\frac{\partial J}{\partial t} &+ (\boldsymbol{\mu} - r\boldsymbol{e}_d)^\top \boldsymbol{m}(t,x;\pi)\frac{\partial J}{\partial x} + \frac{1}{2}[\boldsymbol{m}(t,x;\pi)^\top\Sigma\boldsymbol{m}(t,x;\pi)\\ &+ \langle\Sigma, \boldsymbol{C}(t,x;\pi)\rangle]\frac{\partial^2 J}{\partial x^2} - \frac{\gamma}{2}[\log\det \boldsymbol{C}(t,x;\pi) + d\log 2\pi e] = 0.\end{aligned} \tag{10}$$

Jia and Zhou [12] show that the function J is characterized by two conditions. First, it satisfies a known terminal condition: $J(T,x;\pi;w) = (x-w)^2 - (w-z)^2$. Second, it ensures the following process

$$J(t,\tilde{x}_t^\pi;\pi;w) - \frac{\gamma}{2}\int_0^t [\log\det \boldsymbol{C}(s,\tilde{x}_s^\pi;\pi) + d\log 2\pi e]\mathrm{d}s$$

to be a martingale or, equivalently, the sample trajectory $J(t,x_t^{\boldsymbol{a}};\pi;w) - \frac{\gamma}{2}\int_0^t[\log\det \boldsymbol{C}(s,x_s^{\boldsymbol{a}};\pi) + d\log 2\pi e]\mathrm{d}s$ with $\boldsymbol{a}_s \sim \pi(\cdot|s,x_s^{\boldsymbol{a}})$ to be a martingale.

Suppose we use a class of functions J^θ to approximate J. Note that, for a given policy π and a function approximator J^θ, both $J^\theta(t,x_t^{\boldsymbol{a}};w)$ and $\log\det \boldsymbol{C}(s,x_s^{\boldsymbol{a}};\pi) + d\log 2\pi e$ can be computed by samples. As shown in Jia and Zhou [12], there are several data-driven ways to characterize the martingality of an observable process. One of the most popular and easy-to-implement method is the temporal-difference (TD) learning: namely, to force $\mathrm{d}J^\theta(t,x_t^{\boldsymbol{a}};w) - \frac{\gamma}{2}[\log\det \boldsymbol{C}(t,x_t^{\boldsymbol{a}};\pi) + d\log 2\pi e]\mathrm{d}t$ to be a "martingale difference sequence" so that such a term is orthogonal to any adapted process.

3.3 Policy Gradient

Now we have a method to estimate the value function of any given stochastic policy. To optimize over the feasible policies, we follow the usual gradient-based method in general optimization theory. In our setting, we need to estimate the gradient of the (learned) value function with respect to the policy. As the policy itself is a function and it is impossible to compute the derivative on a function, we seek a finite dimensional parametric approximation. Lemma 1 suggests such an approximation: $\pi^\phi(\cdot|t,x) = \mathcal{N}\left(\cdot|\boldsymbol{m}^\phi(t,x), \boldsymbol{C}^\phi(t,x)\right)$.

It then suffices to consider the gradient of $J(0,x_0;\pi^\phi;w)$ with respective to ϕ: $\frac{\partial}{\partial\phi}J(0,x_0;\pi^\phi;w)$. Jia and Zhou [11] derive the policy gradient representation and find

$$\begin{aligned}\frac{\partial}{\partial\phi}J(0,x_0;\pi^\phi;w) = \mathbb{E}\Bigg[\int_0^T &\left\{\frac{\partial\log\pi^\phi}{\partial\phi}(\boldsymbol{a}_t|t,x_t^{\boldsymbol{a}})\Big[\mathrm{d}J(t,x_t^{\boldsymbol{a}};\pi^\phi;w)\right.\\ &- \frac{\gamma}{2}[\log\det \boldsymbol{C}^\phi(t,x_t^{\boldsymbol{a}}) + d\log 2\pi e]\mathrm{d}t\Big]\\ &\left.- \frac{\gamma}{2}\frac{\partial}{\partial\phi}\log\det \boldsymbol{C}^\phi(t,x_t^{\boldsymbol{a}})\mathrm{d}t\right\}\Bigg].\end{aligned} \tag{11}$$

The right hand side term inside the expectation can be computed using sample trajectories and the known parametric forms π^ϕ, $\boldsymbol{m}^\phi$, $\boldsymbol{C}^\phi$, together with the learned value function replacing the term $\mathrm{d}J(t,x_t^{\boldsymbol{a}};\pi^\phi;w)$, without having to know the market coefficients.

4 ALGORITHM DESIGN

4.1 Implementation Considerations

Since the mean–variance problem has a linear–quadratic nature, we parameterize the value function using a quadratic function in x, with parameters $\theta = (\theta_1, \theta_2, \theta_3) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ by

$$J^{\theta}(t, x; w) = (x-w)^2 e^{-\theta_3 (T-t)} + \theta_2 \left(t^2 - T^2\right) + \theta_1 (t-T) - (w-z)^2. \tag{12}$$

Motivated by the structure of the optimal policy (9), we parameterize the policy with parameters $\phi = (\phi_1, \phi_2, \phi_3) \in \mathbb{R}^d \times \mathbb{S}^d_{++} \times \mathbb{R}$ by[3]

$$\boldsymbol{\pi}^{\phi}(\cdot \mid t, x; w) = \mathcal{N}\left(\cdot \mid -\phi_1 (x - w), \phi_2 e^{\phi_3 (T-t)}\right). \tag{13}$$

The entropy regularization value can then be calculated as

$$\begin{aligned}\mathcal{H}\left(\boldsymbol{\pi}^{\phi}(\cdot \mid t, x; w)\right) &= -\frac{1}{2} \log\left((2\pi e)^d \det\left(\phi_2 e^{\phi_3 (T-t)}\right)\right) \\ &=: \hat{p}(t, \phi).\end{aligned} \tag{14}$$

Now we are equipped with all the necessary ingredients required by our analysis in Section 3 to implement an algorithm. To enhance the performance and to make our algorithm more practical in view of some realistic restrictions, we describe additional techniques and modifications for implementation.

Rebalancing frequency. Our market setting allows trading continuously, meaning that transactions can occur in ultra high frequency. Such a feature may be relevant to problems such as high frequency trading and algorithmic market making, but those problems may have other significant characteristics in terms of liquidity and market microstructure overlooked by our formulation, including market impact, taxes, and transaction costs. Less frequent trading seems to fit better to a small investor as in our setting. Therefore, we will only rebalance our portfolios on specified dates. On the rest of the dates, we hold the assets and do not trade, but the algorithm keeps updating the parameters as if rebalancing took place continuously. Daily or monthly rebalancing appear to be the reasonable frequency with which we can ignore the market impact and liquidity concerns.

Leverage constraint. Putting a constraint on leverage is often regarded as a simple way to improve the performance of an optimization-based portfolio. Jagannathan and Ma [10] show such a constraint is equivalent to shrinking the estimated covariance matrix in a static mean–variance model. Even though adding constraints reduces the search space of feasible policies, it often proves to be an effective way to mitigate estimation errors and stabilize the resulting portfolios. In addition, in real life, an investor is always constrained by leverage limit and hence such a restriction makes the algorithm more practical. We define a leverage constraint ℓ as the maximum proportion of the current total wealth that one can borrow. Specifically, if a portfolio $\boldsymbol{a}_t = (\boldsymbol{a}_t^1, \boldsymbol{a}_t^2, ..., \boldsymbol{a}_t^d)^\top$ is produced by the algorithm, then the leverage-constrained new portfolio is

$$\boldsymbol{a}'_t = \frac{\boldsymbol{a}_t}{\sum_{i=1}^d \boldsymbol{a}_t^i} x_t \ell \mathbb{1}_{\{\sum_{i=1}^d a_t^i > x_t \ell\}} + \boldsymbol{a}_t \mathbb{1}_{\{\sum_{i=1}^d a_t^i \leq x_t \ell\}}. \tag{15}$$

Offline pretrain, mini-batch, and eligibility traces. These are commonly used RL techniques to effectively use data. A pretrain stage means training before the beginning of the actual task using the existing data set. For example, if the backtesting period is from 2000 to 2020, then all the historical data before 2000 can be used for offline learning to have a better initialization. Mini-batch means using multiple samples to estimate their expectation to reduce the variance, e.g., the right hand side of (11). An eligibility trace is a temporary record of the past data and marks the memory that is eligible for undergoing learning changes. The adoption of eligibility traces means to use history-dependent processes as "test functions"[4] to enforce the orthogonality conditions and hence to update the parameters in the value function and the policy. Therefore one still learns through historical information, e.g., the well-known TD(λ) algorithm [35].

Our algorithm is summarized in Algorithm 1.

Algorithm 1 Off-policy TD(λ) Online CTRL Algorithm with Constraints

```
Input
    $\alpha$, $\eta_\theta$, $\eta_\phi$     Learning rates of $w$, $\theta$, $\phi$ respectively
    $d$, $x_0$, $z$        The number of risky assets, initial wealth, target
    $T$, $\Delta t$          Investment horizon and time interval length
    $N$, $M$           The number of iterations, sample average size
    $\gamma$, $\lambda$, $n$        Temperature parameter, weight decay, batch size
    $\ell$, $f$            The max leverage and rebalance frequency

STEP 1:
Initialize parameters $\theta$, $\phi$ and $w$.

STEP 2:
for every iterations do
    for every timesteps do
        Compute next wealth with deterministic action.
        Generate random actions by behavior policy (13).
        Update parameters $\theta$ and $\phi$ using $TD(\lambda)$
    end for
    Update lagrange multiplier $w$
end for

Output
    $\theta$, $\phi$, $w$        Trainable parameters
```

4.2 Relations with Other Algorithms

There is a large body of literature that proposes various algorithms for general RL problems (in discrete time). The most well-known ones are deep Q-network (DQN) [24], deep deterministic policy gradient (DDPG) [17], (asynchronous) advantage actor–critic (A2C, A3C) [23], trust region policy optimization (TRPO) [31], proximal policy optimization (PPO) [32], and soft actor–critic (SAC) [9]. Meanwhile, the classical Q-learning algorithms are demonstrated to be less suitable for continuous controls [6]. Our algorithm belongs

[3] See Appendix A for the form of the theoretical solution, which provides a further reason for our choice of parameterization. Without taking advantage of the theoretical results, parametric forms can be more complicated, e.g., using neural networks, and the performance of the algorithm may vary with different parameterizations. In our implementation, we only choose the simple form presented here.

[4] See [12] for more discussions for this notions and how such choices affect the approximation.

to the class of soft actor–critic (SAC) algorithms for its entropy-regularization and non-degenerate stochastic policies. In addition, the representation of the policy gradient (11) involves the Hamiltonian term (via Itô's formula), which is shown to be the continuous-time counterpart of the advantage function in Jia and Zhou [13]. Therefore, our algorithm can be considered as a combination of SAC and A2C in continuous time. We do not apply the techniques to reuse the sample asynchronously as in experience-replay [39] because of the fluctuation and possible regime changes in financial markets. On the other hand, TRPO and PPO aim to modify and improve the updating of parameters in policy gradient without changing the way to learn the value function or the advantage function; so their idea can be incorporated to our algorithms. Indeed, the representation of policy gradient (11) can be rewritten as:

$$
\begin{aligned}
&\frac{\partial}{\partial \phi} J(0, x_0; \pi^{\phi}; w)\\
=&\frac{\partial}{\partial \phi'}\mathbb{E}\Bigg[\int_0^T \Bigg\{\frac{\pi^{\phi'}(\boldsymbol{a}_t|t, x_t^{\boldsymbol{a}})}{\pi^{\phi}(\boldsymbol{a}_t|t, x_t^{\boldsymbol{a}})}\Big[\mathrm{d}J(t, x_t^{\boldsymbol{a}}; \pi^{\phi}; w)\\
&\qquad - \frac{\gamma}{2}[\log\det \boldsymbol{C}^{\phi}(t, x_t^{\boldsymbol{a}}) + d\log 2\pi e]\mathrm{d}t\Big]\\
&\qquad - \frac{\gamma}{2}\log\det \boldsymbol{C}^{\phi'}(t, x_t^{\boldsymbol{a}})\mathrm{d}t\Bigg\}\Bigg]\Bigg|_{\phi'=\phi}.
\end{aligned}
\tag{16}
$$

Then based on the discussion of Schulman et al. [32], we can update the policy parameters by constructing a surrogate objective function and modifying (16). For example, adding a penalty in terms of the Kullback—Leibler divergence recovers TRPO, or clipping the likelihood ratio recovers PPO. In this paper, as benchmarks for comparison, we select PPO as a representative state-of-the-art algorithm, and DDPG as one that is based on fundamentally different idea from ours.

5 NUMERICAL RESULTS

We use the parameterization described in Section 4.1 for our algorithm, which only takes time and wealth as features. For a fair comparison, in DDPG and PPO, we use the same feature but with deep neural networks to parameterize policy and value function. Moreover, we fix the same rebalancing frequency (daily rebalancing in simulation study and monthly in real market study) and leverage constraint ($\ell = 1$, meaning no borrowing) for all the algorithms/strategies.

5.1 Test with Simulated Data

We simulate a market with 10 stocks whose prices follow (1). The coefficients used for simulation are estimated by actual price data of randomly picked 10 stocks from 2010 to 2020. Then we simulate stock prices for 510 years. The first 10 years are burnt-in to pretrain, and the next 500 years are used for online updating for the RL algorithms. The investment horizon is 1 year. For the algorithms that need an input of a target return, we choose the target annualized return to be 15%, that is, $z = 1.15$. The risk-free interest rate is taken as $r = 0$.

We adopt commonly used metrics to measure the performance, including annualized return, annualized volatility, Sharpe Ratio, Sortino Ratio, Calmar Ratio, maximum drawdown, and recovery time. The results are presented in Table 1.

On the simulated data set, our algorithm (CTRL) significantly outperforms DDPG and PPO in all metrics and performs similarly to the plug-in estimator (CTMV). DDPG and PPO perform even worse than the equally weighted (EW) portfolio. Since prices are simulated from a fixed distribution and coefficients can be consistently estimated with data from a sufficiently long period, it is not surprising that the plug-in estimator is quite efficient.

Table 1: The performance of various algorithms using simulated data in terms of the return (Rtn), volatility (Vol), Sharpe Ratio, Sortino Ratio, Calmar Ratio, maximum drawdown (MDD) and recovery time (RT).

	Rtn	Vol	Sharpe	Sortino	Calmar	MDD	RT
CTRL	15.07%	0.121	1.491	2.942	2.770	0.091	24
DDPG	8.93%	0.143	0.691	1.184	0.878	0.162	94
PPO	12.23%	0.181	0.781	1.478	1.398	0.160	48
CTMV	13.99%	0.119	1.465	3.007	2.903	0.090	21
EW	15.02%	0.198	0.919	1.742	1.607	0.170	46

5.2 Test with Real Market Data

The data we used are downloaded from Wharton Research Data Services (WRDS). The pool of assets for our experiments consists of 300 US stocks that remained listed from 1990 to 2020. Each time, 10 stocks are randomly selected from the pool. Then we backtested the different asset allocation methods on these 10 stocks from 2000 to 2020. Data from 1990 to 1999 are burnt-in for pretrain. We then repeat the experiments 100 times to collect statistics. We compare our algorithm with the conventional plug-in estimator, DDPG, PPO, the equally weighted portfolio, and the market portfolio (MKT). The same metrics as in the simulation study are used for comparing the performance.

Furthermore, the investment horizon is $T = 1$ year, and we backtest the 20 years on a rolling yearly basis. For algorithms that need an input of a target return, we choose the target annualized return to be 15% noting that the annualized return of S&P 500 during our pretrain period 1990 to 2000 was approximately 15%. It corresponds to $z = 1.15$ in our method. Finally, the risk-free interest rate is taken as $r = 0$ as an approximation. In each experiment, we always reset the initial wealth to be $1.

Results from the whole backtesting period is presented in Table 2. Our algorithm outperforms the others in terms of annualized return, Sharpe Ratio, Sortino Ratio, Calmar Ratio and recovery time. Equally weighted portfolio reaches a much higher return and Sharpe Ratio than the market, while DDPG, PPO, and CTMV significantly underperform the market. It is worth mentioning that the model-based method CTMV perform reasonably well on simulated data, but perform the worst on real data, producing almost zero or even negative reutrns.

Next, we divide the backtesting period into two parts to further investigate how different methods adapt to different market conditions. The first period is from 2000 to 2010, shown in Table 3, which is a very volatile period including two major market crashes: the internet bubble in the early 2000s and the financial crisis in 2007-2009. Our algorithm seems to be resilient and delivers reasonably high returns, followed by the equally-weighted portfolio. All the rest three have losses, especially the plug-in estimator. The second period is from 2010 to 2020, shown in Table 4, which has a huge bull run. In this period, our algorithm performs closer but still significantly better than the two naive portfolios in terms of return and Sharpe ratio, while DDPG, PPO, and plug-in estimator far underperform. In both subperiods, CTRL is the best in terms of all the ratios. Its outperformance is especially notable during the bear markets.

Table 2: The comparison of the out-of-sample performance of different allocation methods from 2000 to 2020. The metrics include the return (Rtn), volatility (Vol), Sharpe Ratio, Sortino Ratio, Calmar Ratio, maximum drawdown (MDD) and recovery time (RT). All metrics are averaged across 100 experiments with randomly picked stocks.

	Rtn	Vol	Sharpe	Sortino	Calmar	MDD	RT
CTRL	11.52%	0.212	0.558	0.882	0.197	0.607	464
DDPG	8.51%	0.351	0.246	0.418	0.128	0.680	708
PPO	9.09%	0.298	0.314	0.521	0.141	0.663	765
CTMV	-30.77%	0.740	-0.186	-0.214	-0.272	0.800	952
EW	10.28%	0.211	0.496	0.807	0.188	0.565	547
MKT	5.90%	0.190	0.311	0.494	0.107	0.552	869

Table 3: The comparison of the out-of-sample performance of different allocation methods from 2000 to 2010. The metrics include the return (Rtn), volatility (Vol), Sharpe Ratio, Sortino Ratio, Calmar Ratio, maximum drawdown (MDD) and recovery time (RT). All metrics are averaged across 100 experiments with randomly picked stocks.

	Rtn	Vol	Sharpe	Sortino	Calmar	MDD	RT
CTRL	9.44%	0.258	0.378	0.600	0.164	0.607	158
DDPG	7.16%	0.412	0.178	0.305	0.109	0.680	399
PPO	6.69%	0.347	0.231	0.389	0.121	0.663	436
CTMV	-32.15%	0.841	-0.256	-0.328	-0.289	0.800	308
EW	7.69%	0.244	0.319	0.524	0.142	0.565	221
MKT	-0.90%	0.224	-0.041	-0.066	-0.017	0.552	N/A

Finally, Figure 1 shows the time series of the average wealth during the testing period. We can see that the portfolio produced by our algorithm remains on the top of alternatives constantly for a long period. It also recovers faster than the others after a few draw-downs. The plug-in estimator leads to bankruptcy in some of the experiments, and hence is omitted in this figure.

6 CONCLUSION

This paper studies the problem of high-dimensional mean–variance asset allocation by continuous-time entropy-regularized reinforcement learning. Based on the recently developed continuous-time RL methodology by Jia and Zhou [11, 12], Wang et al. [37], we design algorithms for the problem and conduct extensive numerical studies with both simulated and real market data. In particular, we compare the proposed algorithm with several other algorithms and strategies. Overall, The model-based plug-in estimator produces almost no profit on the real market data. Existing deep reinforcement learning algorithms, such as DDPG and PPO, can improve the conventional plug-in estimator in real market but converge slowly on simulated data. However, with only price data, they still perform worse than naively diversified portfolios. By contrast, our method gets close to the targeted return and generates the highest Sharpe ratio and other popular risk-adjusted performance metrics compared with the equally weighted portfolio and the market portfolio, which are well-known to be robust in performance and effective in diversification. This study shows great potential in applying data-driven RL methods to the practice of asset management.

Table 4: The comparison of the out-of-sample performance of different allocation methods from 2010 to 2020. The metrics include the return (Rtn), volatility (Vol), Sharpe Ratio, Sortino Ratio, Calmar Ratio, maximum drawdown (MDD) and recovery time (RT). All metrics are averaged across 100 experiments with randomly picked stocks.

	Rtn	Vol	Sharpe	Sortino	Calmar	MDD	RT
CTRL	13.69%	0.152	0.951	1.525	0.509	0.299	164
DDPG	10.33%	0.213	0.494	0.835	0.362	0.367	306
PPO	11.95%	0.209	0.574	0.944	0.375	0.323	236
CTMV	10.49%	0.171	0.689	1.133	0.479	0.300	155
EW	13.01%	0.171	0.780	1.263	0.550	0.253	135
MKT	13.10%	0.147	0.887	1.388	0.675	0.193	75

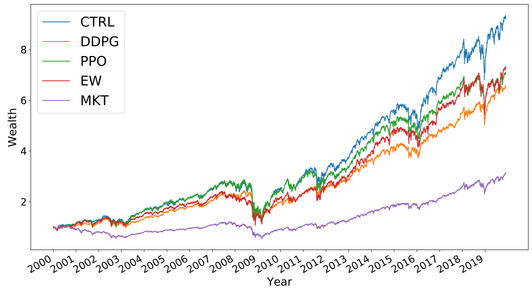

Figure 1: The average wealth trajectory under different allocation methods from 2000 to 2020. The time series plot indicates the average wealth on each date across 100 experiments. The plug-in estimator (CTMV) leads to bankruptcy in some of the experiments, and hence is omitted in this figure.

ACKNOWLEDGMENTS

Zhou gratefully acknowledges financial support through a start-up grant and the Nie Center for Intelligent Asset Management at Columbia University.

REFERENCES

[1] Michael J Best and Robert R Grauer. 1991. On the sensitivity of mean-variance-efficient portfolios to changes in asset means: Some analytical and computational results. *The Review of Financial Studies* 4, 2 (1991), 315–342.

[2] Michael J Best and Robert R Grauer. 1991. Sensitivity analysis for mean-variance portfolio problems. *Management Science* 37, 8 (1991), 980–989.

[3] Mark Britten-Jones. 1999. The sampling error in estimates of mean-variance efficient portfolio weights. *Journal of Finance* 54, 2 (1999), 655–671.

[4] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. 2019. Deep hedging. *Quantitative Finance* 19, 8 (2019), 1271–1291.

[5] Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. 2009. Optimal versus naive diversification: How inefficient is the 1/N portfolio strategy? *The Review of Financial Studies* 22, 5 (2009), 1915–1953.

[6] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. 2016. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*. PMLR, 1329–1338.

[7] Wendell H Fleming and Halil Mete Soner. 2006. *Controlled Markov Processes and Viscosity Solutions*. Vol. 25. Springer Science & Business Media.

[8] Xiu Gao and Laiwan Chan. 2000. An algorithm for trading and portfolio management using Q-learning and Sharpe ratio maximization. In *Proceedings of the International Conference on Neural Information Processing*. 832–837.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 1861–1870.

[10] Ravi Jagannathan and Tongshu Ma. 2003. Risk reduction in large portfolios: Why imposing the wrong constraints helps. *Journal of Finance* 58, 4 (2003), 1651–1683.

[11] Yanwei Jia and Xun Yu Zhou. 2021. Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms. To appear in *Journal of Machine Learning Research; arXiv preprint arXiv:2111.11232* (2021).

[12] Yanwei Jia and Xun Yu Zhou. 2022. Policy evaluation and temporal-difference learning in continuous time and space: A martingale approach. *Journal of Machine Learning Research* 23, 154 (2022), 1–55.

[13] Yanwei Jia and Xun Yu Zhou. 2022. q-Learning in continuous time. *arXiv preprint arXiv:2207.00713* (2022).

[14] Olivier Jin and Hamza El-Saawy. 2016. Portfolio management using reinforcement learning. *Technical Report, Stanford University* (2016).

[15] Jang Ho Kim, Yongjae Lee, Woo Chang Kim, and Frank J Fabozzi. 2021. Mean–Variance Optimization for Asset Allocation. *The Journal of Portfolio Management* 47, 5 (2021), 24–40.

[16] Petter N Kolm and Gordon Ritter. 2019. Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science* 1, 1 (2019), 159–171.

[17] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[18] Andrew EB Lim and Xun Yu Zhou. 2002. Mean-variance portfolio selection with random parameters in a complete market. *Mathematics of Operations Research* 27, 1 (2002), 101–120.

[19] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. 2021. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. In *Proceedings of the Second ACM International Conference on AI in Finance*. 1–9.

[20] Dacid G Luenberger. 1998. *Investment Science.* Oxford University Press: New York.

[21] Harry Markowitz. 1952. Portfolio selection. *Journal of Finance* 7, 1 (1952), 77–91.

[22] Robert C Merton. 1980. On estimating the expected return on the market: An exploratory investigation. *Journal of Financial Economics* 8, 4 (1980), 323–361.

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[25] Ralph Neuneier. 1996. Optimal asset allocation using adaptive dynamic programming. In *Advances in Neural Information Processing Systems*. 952–958.

[26] LA Prashanth and Mohammad Ghavamzadeh. 2013. Actor-critic algorithms for risk-sensitive MDPs. In *Advances in Neural Information processing systems*. 252–260.

[27] LA Prashanth and Mohammad Ghavamzadeh. 2016. Variance-constrained actor-critic algorithms for discounted and average reward MDPs. *Machine Learning* 105, 3 (2016), 367–417.

[28] Gordon Ritter. 2017. Machine learning for trading. *Working Paper. Available at SSRN 3015609* (2017).

[29] Makoto Sato, Hajime Kimura, and Shibenobu Kobayashi. 2001. TD algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence* 16, 3 (2001), 353–362.

[30] Makoto Sato and Shigenobu Kobayashi. 2000. Variance-penalized reinforcement learning for risk-averse asset allocation. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 244–249.

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[33] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*. PMLR, 387–395.

[34] Matthew J Sobel. 1982. The variance of discounted Markov decision processes. *Journal of Applied Probability* 19, 4 (1982), 794–802.

[35] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[36] Aviv Tamar and Shie Mannor. 2013. Variance adjusted actor critic algorithms. *arXiv preprint arXiv:1310.3697* (2013).

[37] Haoran Wang, Thaleia Zariphopoulou, and Xun Yu Zhou. 2020. Reinforcement learning in continuous time and space: A stochastic control approach. *Journal of Machine Learning Research* 21, 198 (2020), 1–34.

[38] Haoran Wang and Xun Yu Zhou. 2020. Continuous-time mean–variance portfolio selection: A reinforcement learning framework. *Mathematical Finance* 30, 4 (2020), 1273–1308.

[39] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224* (2016).

[40] W Murray Wonham. 1968. On the separation theorem of stochastic control. *SIAM Journal on Control* 6, 2 (1968), 312–326.

[41] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. 2020. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–8.

[42] Jiongmin Yong and Xun Yu Zhou. 1999. *Stochastic Controls: Hamiltonian Systems and HJB Equations.* New York, NY: Spinger.

[43] Zihao Zhang, Stefan Zohren, and Stephen Roberts. 2020. Deep reinforcement learning for trading. *The Journal of Financial Data Science* 2, 2 (2020), 25–40.

[44] Xiao-lin Zheng, Meng-ying Zhu, Qi-bing Li, Chao-chao Chen, and Yan-chao Tan. 2019. FinBrain: when finance meets AI 2.0. *Frontiers of Information Technology & Electronic Engineering* 20, 7 (2019), 914–924.

[45] Xun Yu Zhou. 1992. On the existence of optimal relaxed controls of stochastic partial differential equations. *SIAM Journal on Control and Optimization* 30, 2 (1992), 247–261.

[46] Xun Yu Zhou and Duan Li. 2000. Continuous-time mean-variance portfolio selection: A stochastic LQ framework. *Applied Mathematics and Optimization* 42, 1 (2000), 19–33.

[47] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning.. In *AAAI*, Vol. 8. Chicago, IL, USA, 1433–1438.

A THEORETICAL SOLUTIONS TO THE MEAN-VARIANCE PROBLEM

The optimal policy to (8) is

$$\begin{aligned}\pi^*(u|t,x) = \mathcal{N}\Bigg(\boldsymbol{a} \mid &-\Sigma^{-1}(\boldsymbol{\mu} - r\boldsymbol{e}_d)(x - w^*),\\ &\frac{\gamma}{2} e^{(\boldsymbol{\mu}-r\boldsymbol{e}_d)^\top \Sigma^{-1}(\boldsymbol{\mu}-r\boldsymbol{e}_d)(T-t)}\Sigma^{-1}\Bigg),\end{aligned} \tag{17}$$

and the optimal value function is

$$\begin{aligned}J^*(t,x) =&(x-w^*)^2 e^{-(\boldsymbol{\mu}-r\boldsymbol{e}_d)^\top\Sigma^{-1}(\boldsymbol{\mu}-r\boldsymbol{e}_d)(T-t)}\\ &+\frac{\gamma d}{4}(\boldsymbol{\mu}-r\boldsymbol{e}_d)^\top\Sigma^{-1}(\boldsymbol{\mu}-r\boldsymbol{e}_d)(T^2-t^2)\\ &-\frac{\gamma d}{2}\left((\boldsymbol{\mu}-r\boldsymbol{e}_d)^\top\Sigma^{-1}(\boldsymbol{\mu}-r\boldsymbol{e}_d)T-\frac{(T-t)}{d}\log\frac{\det\Sigma}{\pi\gamma}\right)\\ &-(w^*-z)^2,\end{aligned} \tag{18}$$

where the Lagrange multiplier is

$$w^* = \frac{z e^{(\boldsymbol{\mu}-r\boldsymbol{e}_d)^\top\Sigma^{-1}(\boldsymbol{\mu}-r\boldsymbol{e}_d)T}-x_0}{e^{(\boldsymbol{\mu}-r\boldsymbol{e}_d)^\top\Sigma^{-1}(\boldsymbol{\mu}-r\boldsymbol{e}_d)T}-1}. \tag{19}$$