Bridging the gap between Markowitz planning and deep reinforcement learning

Eric Benhamou, David Saltiel, Sandrine Ungari, Abhishek Mukhopadhyay

Bridging the gap between Markowitz planning and deep reinforcement learning

Eric Benhamou [1,2], **David Saltiel** [1,3], **Sandrine Ungari** [4], **Abhishek Mukhopadhyay** [5]
[1] AI Square Connect, France, {eric.benhamou,david.saltiel}@aisquareconnect.com
[2]MILES, LAMSADE, Dauphine university, France, eric.benhamou@lamsade.dauphine.fr
[3] LISIC, ULCO, France, david.saltiel@univ-littoral.fr
[4] Societe Generale, Cross Asset Quantitative Research, UK,
[5] Societe Generale, Cross Asset Quantitative Research, France,
{sandrine.ungari,abhishek.mukhopadhyay}@sgcib.com

Abstract

While researchers in the asset management industry have mostly focused on techniques based on financial and risk planning techniques like Markowitz efficient frontier, minimum variance, maximum diversification or equal risk parity, in parallel, another community in machine learning has started working on reinforcement learning and more particularly deep reinforcement learning to solve other decision making problems for challenging task like autonomous driving, robot learning, and on a more conceptual side games solving like Go. This paper aims to bridge the gap between these two approaches by showing Deep Reinforcement Learning (DRL) techniques can shed new lights on portfolio allocation thanks to a more general optimization setting that casts portfolio allocation as an optimal control problem that is not just a one-step optimization, but rather a continuous control optimization with a delayed reward. The advantages are numerous: (i) DRL maps directly market conditions to actions by design and hence should adapt to changing environment, (ii) DRL does not rely on any traditional financial risk assumptions like that risk is represented by variance, (iii) DRL can incorporate additional data and be a multi inputs method as opposed to more traditional optimization methods. We present on an experiment some encouraging results using convolution networks.

Introduction

In asset management, there is a gap between mainstream used methods and new machine learning techniques around reinforcement learning and in particular deep reinforcement learning. The former methods rely on financial risk optimization and solve the planning problem of the optimal portfolio as a single step optimization question. The latter do not make any assumptions about risk, do a more involving multi-steps optimization and solve complex and challenging tasks like autonomous driving (Wang, Jia, and Weng 2018), learning advanced locomotion and manipulation skills from raw sensory inputs (Levine et al. 2015; 2016; Schulman et al. 2015a; 2017; Lillicrap et al. 2015) or on a more conceptual side for reaching supra human level in popular games like Atari (Mnih et al. 2013), Go (Silver et al. 2016; 2017), StarCraft II (Vinyals et al. 2019), etc ... One of the reasons often put forward for this situation is that asset management researchers have mostly been trained with an econometric and financial mathematics background, while the deep reinforcement learning community has been mostly trained in computer science and robotics, leading to two distinctive research communities that do not interact much between each other. In this paper, we aim to present the various approaches to show similarities and differences to bridge the gap between these two approaches. Both methods can help solving the decision making problem of finding the optimal portfolio allocation weights.

Related works

As this paper aims at bridging the gap between traditional asset management portfolio selection methods and deep reinforcement learning, there are too many works to be cited.

On the traditional methods side, the seminal work is (Markowitz 1952) that has led to various extensions like minimum variance (Chopra and Ziemba 1993; Haugen and Baker 1991), (Kritzman 2014), maximum diversification (Choueifaty and Coignard 2008; Choueifaty, Froidure, and Reynier 2012), maximum decorrelation (Christoffersen et al. 2010), risk parity (Maillard, Roncalli, and Teïletche 2010; Roncalli and Weisang 2016). We will review these works in the section entitled *Traditional methods*.

On the reinforcement learning side, the seminal book is (Sutton and Barto 2018). The field of deep reinforcement learning is growing every day at an unprecedented pace, making the citation exercise complicated. But in terms of breakthroughs of deep reinforcement learning, one can cite the work around Atari games from raw pixel inputs (Mnih et al. 2013; 2015), Go (Silver et al. 2016; 2017), StarCraft II (Vinyals et al. 2019), learning advanced locomotion and manipulation skills from raw sensory inputs (Levine et al. 2015; 2016) (Schulman et al. 2015a; 2015b; 2017; Lillicrap et al. 2015), autonomous driving (Wang, Jia, and Weng 2018) and robot learning (Gu et al. 2017).

On the application of deep reinforcement learning methods to portfolio allocations, there is already a growing interest as recent breakthroughs has put growing emphasis on this method. Hence, the field is growing very rapidly and

survey like (Fischer 2018) are already out dated. Driven initially mostly by applications to crypto currencies and Chinese financial markets (Jiang and Liang 2016; Zhengyao et al. 2017; Liang et al. 2018; Yu et al. 2019; Wang and Zhou 2019; Saltiel et al. 2020; Benhamou et al. 2020b; 2020a; 2020c), the field is progressively taking off on other assets (Kolm and Ritter 2019; Liu et al. 2020; Ye et al. 2020; Li et al. 2019; Xiong et al. 2019). More generally, DRL has recently been applied to other problems than portfolio allocation. For instance, (Deng et al. 2016; Zhang, Zohren, and Roberts 2019; Huang 2018; Théate and Ernst 2020; Chakraborty 2019; Nan, Perumal, and Zaiane 2020; Wu et al. 2020) tackle the problem of direct trading strategies (Bao and yang Liu 2019) handles the one of multi agent trading while (Ning, Lin, and Jaimungal 2018) examine optimal execution.

Traditional methods

We are interested in finding an optimal portfolio which makes the planning problem quite different from standard planning problem where the aim is to plan a succession of tasks. Typical planning algorithms are variations around STRIPS (Fikes and Nilsson 1971), that starts by analysis ending goals and means, builds the corresponding graph and finds the optimal graph. Indeed we start from the goals to achieve and try to find means that can lead to them. New work like Graphplan as presented in (Blum and Furst 1995) uses a novel planning graph, to reduce the amount of search needed, while hierarchical task network (HTN) planning leverages the classification to structure networks and hence reduce the number of graph searches. Other algorithms like search algorithm as A^*, B^*, weighted A^* or for full graph search, branch and bound and its extensions, as well as evolutionary algorithms like particle swarm, CMA-ES are also used widely in AI planning etc.. However, when it comes to portfolio allocation, standard methods used by practitioners rely on more traditional financial risk reward optimization problems and follows rather the Markowitz approach as presented below.

Markowitz

The intuition of Markowitz portfolio is to be able to compare various assets and assemble them taking into account both return and risk. Comparing just returns of some financial assets would be too naive. One has to take into account in her/his investment decision returns with associated risk. Risk is not an easy concept. in Modern Portfolio Theory (MPT), risk is represented by the variance of the asset returns. If we take various financial assets and display their returns and risk as in figure 1, we can find an efficient frontier. Indeed there exists an efficient frontier represented by the red dot line.

Mathematically, if we denote by $w = (w_1, ..., w_l)$ the allocation weights with $1 \geq w_i \geq 0$ for $i = 0...l$, summarized by $1 \geq w \geq 0$, with the additional constraints that these weights sum to 1: $\sum_{i=1}^{l} w_i = 1$, we can see this portfolio allocation question as an optimization.
Let $\mu = (\mu_1, ..., \mu_l)^T$ be the expected returns for our l strategies and Σ the matrix of variance covariances of the l strategies' returns. Let r_{min} be the minimum expected return. The Markowitz optimization problem to solve is to minimize the risk given a target of minimum expected return as follows:

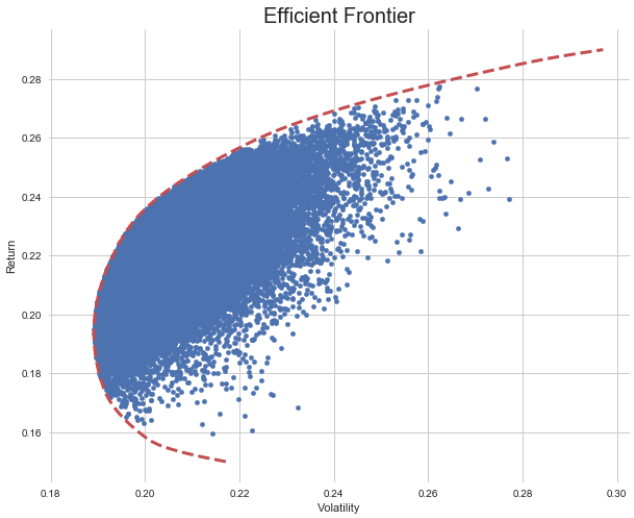

Figure 1: Markowitz efficient frontier for the GAFA: returns taken from 2017 to end of 2019

$$\begin{aligned} &\underset{w}{\text{Minimize}} \quad w^T \Sigma w \\ &\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{1}$$

It is solved by standard quadratic programming. Thanks to duality, there is an equivalent maximization with a given maximum risk σ_{max} for wich the problem writes as follows:

$$\begin{aligned} &\underset{w}{\text{Maximize}} \quad \mu^T w \\ &\text{subject to} \quad w^T \Sigma w \leq \sigma_{max}, \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{2}$$

Minimum variance portfolio

This seminal model has led to numerous extensions where the overall idea is to use a different optimization objective. As presented in (Chopra and Ziemba 1993; Haugen and Baker 1991), (Kritzman 2014), we can for instance be interested in just minimizing risk (as we are not so much interested in expected returns), which leads to the minimum variance portfolio given by the following optimization program:

$$\begin{aligned} &\underset{w}{\text{Minimize}} \quad w^T \Sigma w \\ &\text{subject to} \quad \sum_{i=1...l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{3}$$

Maximum diversification portfolio Denoting by σ the volatilities of our l strategies, whose values are the diagonal elements of the covariance matrix Σ: $\sigma = (\Sigma_{i,i})_{i=1..l}$, we

can shoot for maximum diversification with the diversification of a portfolio defined as follows: $D = \frac{w^T \sigma}{\sqrt{w^T \sum w}}$. We then solve the following optimization program as presented in (Choueifaty and Coignard 2008; Choueifaty, Froidure, and Reynier 2012)

$$\begin{aligned} \underset{w}{\text{Maximize}} \quad & \frac{w^T \sigma}{\sqrt{w^T \sum w}} \\ \text{subject to} \quad & \sum_{i=1\ldots l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{4}$$

The concept of diversification is simply the ratio of the weighted average of volatilities divided by the portfolio volatility.

Maximum decorrelation portfolio Following (Christoffersen et al. 2010) and denoting by C the correlation matrix of the portfolio strategies, the maximum decorrelation portfolio is obtained by finding the weights that provide the maximum decorrelation or equivalently the minimum correlation as follows:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & w^T C w \\ \text{subject to} \quad & \sum_{i=1\ldots l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{5}$$

Risk parity portfolio Another approach following risk parity (Maillard, Roncalli, and Teïletche 2010; Roncalli and Weisang 2016) is to aim for more parity in risk and solve the following optimization program

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & \frac{1}{2} w^T \Sigma w - \frac{1}{n} \sum_{i=1}^{l} \ln(w_i) \\ \text{subject to} \quad & \sum_{i=1\ldots l} w_i = 1, 1 \geq w \geq 0 \end{aligned} \tag{6}$$

All these optimization techniques are the usual way to solve the planning question of getting the best portfolio allocation. We will see in the following section that there are many alternatives leveraging machine learning that remove cognitive bias of risk and are somehow more able to adapt to changing environment.

Reinforcement learning

Previous financial methods treat the portfolio allocation planning question as a one-step optimization problem, with convex objective functions. There are multiple limitations to this approach:

- they do not relate market conditions to portfolio allocation dynamically.
- they do not take into account that the result of the portfolio allocation may potentially be evaluated much later.
- they make a strong assumptions about risk.

What if we could cast this portfolio allocation planning question as a dynamic control problem where we have some market information and needs to decide at each time step the optimal portfolio allocation problem and evaluate the result with delayed reward? What if we could move from static portfolio allocation to optimal control territory where we can change our portfolio allocation dynamically when market conditions changes. Because the community of portfolio allocation is quite different from the one of reinforcement learning, this approach has been ignored for quite some time even though there is a growing interest for the use of reinforcement learning and deep reinforcement learning over the last few years. We will present here in greater details what deep reinforcement is in order to suggest more discussions and exchanges between these two communities.

Contrary to supervised learning, reinforcement learning do not try to predict future returns. It does not either try to learn the structure of the market implicitly. Reinforcement learning does more: it directly learns the optimal policy for the portfolio allocation in connection with the dynamically changing market conditions.

Deep Reinforcement Learning Intuition

As it name stands for, Deep Reinforcement Learning (DRL) is the combination of Reinforcement Learning (RL) and Deep (D). The usage of deep learning is to represent the policy function in RL. In a nutshell, the setting for applying RL to portfolio management can be summarized as follows:

- current knowledge of the financial markets is formalized via a state variable denoted by s_t.
- Our planning task which is to find an optimal portfolio allocation can be thought as taking an action a_t on this market. This action is precisely the decision of the current portfolio allocation (also called portfolio weights).
- once we have decided the portfolio allocation, we observe the next state s_{t+1}.
- we use a reward to evaluate the performance of our actions. In our particular setting, we can compute this reward only at the the final time of our episode, making it quite special compared to standard reinforcement learning problem. We denote this reward by R_T where T is the final time of our episode. This reward R_T is in a sense similar to our objective function in traditional methods. A typical reward is the final portfolio net performance. It could be obviously other financial performance evaluation criteria like Sharpe, Sortino ratio, etc..

Following standard RL, we model our problem to solve with a Markov Decision Process (MDP) as in (Sutton and Barto 2018). MDP assumes that the agent knows all the states of the environment and has all the information to make the optimal decision in every state. The Markov property implies in addition that knowing the current state is sufficient. MDP assumes a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the state action to next state transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and $\mathcal{R}$ is the immediate reward. The goal of the agent is to learn a policy that maps states to the optimal action

$\pi : \mathcal{S} \rightarrow \mathcal{A}$ and that maximizes the expected discounted reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$.

The concept of using deep network is to represent the function that relates dynamically the states to the action called in RL the policy and denoted by $\vec{a}_t = \pi(s_t)$. This function is represented by deep network because of the universal approximation theorem that states that any function can be represented by a deep network provided we have enough layers and nodes. Compared to traditional methods that only solve a one step optimization, we are solving the following dynamic control optimization program:

$$\begin{aligned} \underset{\pi(.)}{\text{Maximize}} \quad & \mathbb{E}[R_T] \\ \text{subject to} \quad & a_t = \pi(s_t) \end{aligned} \tag{7}$$

Note that we maximize the expected value of the cumulated reward $\mathbb{E}[R_T]$ because we are operating in a stochastic environment. To make things simpler, let us assume that the cumulated reward is the final portfolio net performance. Let us write P_t the price at time t of our portfolio, and its return at time t: r_t^P and the portfolio assets return vector at time t: $\vec{r}_t$. The final net performance writes as $P_T/P_0 - 1 = \prod_{t=1}^{T}(1 + r_t^P) - 1$. The returns r_t^P is a function of our planning action a_t as follows: $(1+r_t^P) = 1+\langle \vec{a}_t, \vec{r}_t \rangle$ where $\langle \cdot, \cdot \rangle$ is the standard inner product of two vectors. In addition if we recall that the policy is parametrized by some deep network parameters, θ: $a_t = \pi_\theta(s_t)$, we can make our optimization problem slightly more detailed as follows:

$$\begin{aligned} \underset{\theta}{\text{Maximize}} \quad & \mathbb{E}\left[\prod_{t=1}^{T}(1 + \langle \vec{a}_t, \vec{r}_t \rangle)\right] \\ \text{subject to} \quad & a_t = \pi_\theta(s_t). \end{aligned} \tag{8}$$

It is worth noticing that compared to previous traditional planning methods (optimization 1, 3, 4, 5 or 5), the underlying optimization problem in RL 7 and its rewritting in terms of deep network parameters θ as presented in 8 have many differences:

- First, we are trying to optimize a function π and not simple weights w_i. Although this function at the end is represented by a deep neural network that has admittely also weights, this is conceptually very different as we are optimizing in the space of functions $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that is a much bigger space than simply $\mathbb{R}^l$.
- Second, it is a multi time step optimization at it involves results from time $t = 1$ to $t = T$, making it also more involving.

Partially Observable Markov Decision Process

If there is in addition some noise in our data and we are not able to observe the full state, it is better to use Partially Observable Markov Decision Process (POMDP) as presented initially in (Astrom 1969). In POMDP, only a subset of the information of a given state is available. The partially-informed agent cannot behave optimally. He uses a window of past observations to replace states as in a traditional MDP.

Mathematically, POMDP is a generalization of MDP. POMPD adds two more variables in the tuple, $\mathcal{O}$ and $\mathcal{Z}$ where $\mathcal{O}$ is the set of observations and $\mathcal{Z}$ is the observation transition function $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow [0, 1]$. At each time, the agent is asked to take an action $a_t \in \mathcal{A}$ in a particular environment state $s_t \in \mathcal{S}$, that is followed by the next state s_{t+1} with $\mathcal{P}(s_{t+1}|s_t, a_t)$. The next state s_{t+1} is not observed by the agent. It rather receives an observation $o_{t+1} \in \mathcal{O}$ on the state s_{t+1} with probability $Z(o_{t+1}|s_{t+1}, a_t)$.

From a practical standpoint, the general RL setting is modified by taking a pseudo state formed with a set of past observations $(o_{t-n}, o_{t-n-1}, \ldots, o_{t-1}, o_t)$. In practice to avoid large dimension and the curse of dimension, it is useful to reduce this set and take only a subset of these past observations with $j < n$ past observations, such that $0 < i_1 < \ldots < i_j$ and $i_k \in \mathbb{N}$ is an integer. The set $\delta_1 = (0, i_1, \ldots, i_j)$ is called the observation lags. In our experiment we typically use lag periods like (0, 1, 2, 3, 4, 20, 60) for daily data, where $(0, 1, 2, 3, 4)$ provides last week observation, 20 is for the one-month ago observation (as there is approximately 20 business days in a month) and 60 the three-month ago observation.

Observations

Regular observations There are two types of observations: regular and contextual information. Regular observations are data directly linked to the problem to solve. In the case of an asset management framework, regular observations are past prices observed over a lag period $\delta = (0 < i_1 < \ldots < i_j)$. To normalize data, we rather use past returns computed as $r_t^k = \frac{p_t^k}{p_{t-1}^k} - 1$ where p_t^k is the price at time t of the asset k. To give information about regime changes, our trading agent receives also empirical standard deviation computed over a sliding estimation window denoted by d as follows $\sigma_t^k = \sqrt{\frac{1}{d}\sum_{u=t-d+1}^{t}(r_u - \mu)^2}$, where the empirical mean μ is computed as $\mu = \frac{1}{d}\sum_{u=t-d+1}^{t} r_u$. Hence our regular observations is a three dimensional tensor $A_t = \left[A_t^1, A_t^2\right]$

$$\text{with} \quad A_t^1 = \begin{pmatrix} r_{t-i_j}^1 \ \ldots \ r_t^1 \\ \ldots \ \ldots \ \ldots \\ r_{t-i_j}^m \ \ldots . \ r_t^m \end{pmatrix}, \ A_t^2 = \begin{pmatrix} \sigma_{t-i_j}^1 \ \ldots \ \sigma_t^1 \\ \ldots \ \ldots \ \ldots \\ \sigma_{t-i_j}^m \ \ldots . \ \sigma_t^m \end{pmatrix}$$

This setting with two layers (past returns and past volatilities) is quite different from the one presented in (Jiang and Liang 2016; Zhengyao et al. 2017; Liang et al. 2018) that uses different layers representing closing, open high low prices. There are various remarks to be made. First, high low information does not make sense for portfolio strategies that are only evaluated daily, which is the case of all the funds. Secondly, open high low prices tend to be highly correlated creating some noise in the inputs. Third, the concept of volatility is crucial to detect regime change and is surprisingly absent from these works as well as from other works like (Yu et al. 2019; Wang and Zhou 2019; Liu et al. 2020; Ye et al. 2020; Li et al. 2019; Xiong et al. 2019).

Context observation Contextual observations are additional information that provide intuition about current con-

text. For our asset manager, they are other financial data not directly linked to its portfolio assumed to have some predictive power for portfolio assets. Contextual observations are stored in a 2D matrix denoted by C_t with stacked past p individual contextual observations. Among these observations, we have the maximum and minimum portfolio strategies return and the maximum portfolio strategies volatility. The latter information is like for regular observations motivated by the stylized fact that standard deviations are useful features to detect crisis. The contextual state writes as $C^t = \begin{pmatrix} c_t^1 \ ... \ c_{t-i_k}^1 \\ ... \ ... \ ... \\ c_t^p \ .... \ c_{t-i_k}^p \end{pmatrix}$. The matrix nature of contextual states C_t implies in particular that we will use 1D convolutions should we use convolutional layers. All in all, observations that are augmented observations, write as $O_t = [A_t, C_t]$, with $A_t = [A_t^1, A_t^2]$ that will feed the two sub-networks of our global network.

Action

In our deep reinforcement learning the augmented asset manager agent needs to decide at each period in which hedging strategy it invests. The augmented asset manager can invest in l strategies that can be simple strategies or strategies that are also done by asset management agent. To cope with reality, the agent will only be able to act after one period. This is because asset managers have a one day turn around to change their position. We will see on experiments that this one day turnaround lag makes a big difference in results. As it has access to l potential hedging strategies, the output is a l dimension vector that provides how much it invest in each hedging strategy. For our deep network, this means that the last layer is a softmax layer to ensure that portfolio weights are between 0 and 100% and sum to 1, denoted by $(p_t^1, ..., p_t^l)$. In addition, to include leverage, our deep network has a second output which is the overall leverage that is between 0 and a maximum leverage value (in our experiment 3), denoted by lvg_t. Hence the final allocation is given by $lvg_t \times (p_t^1, ..., p_t^l)$.

Reward

In terms of reward, we are considering the net performance of our portfolio from t_0 to the last train date t_T computed as follows: $\frac{P_{t_T}}{P_{t_0}} - 1$.

Multi inputs and outputs

We display in figure 2 the architecture of our network. Because we feed our network with both data from the strategies to select but also contextual information, our network is a multiple inputs network.

Additionally, as we want from these inputs to provide not only percentage in the different hedging strategies (with a softmax activation of a dense layer) but also the overall leverage (with a dense layer with one single output neurons), we also have a multi outputs network. Additional hyperparameters that are used in the network as L2 regularization with a coefficient of 1e-8.

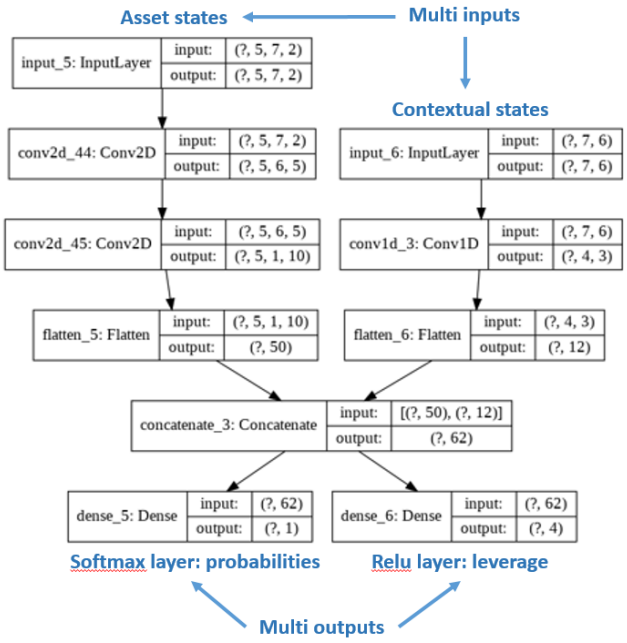

Figure 2: network architecture obtained via tensorflow plot-model function. Our network is very different from standard DRL networks that have single inputs and outputs. Contextual information introduces a second input while the leverage adds a second output

Convolution networks

Because we want to extract some features implicitly with a limited set of parameters, and following (Liang et al. 2018), we use convolution network that perform better than simple full connected layers. For our so called *asset states* named like that because there are the part of the states that relates to the asset, we use two layers of convolutional network with 5 and 10 convolutions. These parameters are found to be efficient on our validation set. In contrast, for the contextual states part, we only use one layer of convolution networks with 3 convolutions. We flatten our two sub network in order to concatenate them into a single network.

Adversarial Policy Gradient

To learn the parameters of our network depicted in 2, we use a modified policy gradient algorithm called adversarial as we introduce noise in the data as suggested in (Liang et al. 2018).. The idea of introducing noise in the data is to have some randomness in each training to make it more robust. This is somehow similar to drop out in deep networks where we randomly perturb the network by randomly removing some neurons to make it more robust and less prone to overfitting. Here, we are perturbing directly the data to create this stochasticity to make the network more robust. A policy is a mapping from the observation space to the action space, $\pi : \mathcal{O} \rightarrow \mathcal{A}$. To achieve this, a policy is specified by a deep network with a set of parameters $\vec{\theta}$. The action is a vector function of the observation given the parameters: $\vec{a}_t =$

$\pi_{\vec{\theta}}(\boldsymbol{o}_t)$. The performance metric of $\pi_{\vec{\theta}}$ for time interval $[0,t]$ is defined as the corresponding total reward function of the interval $J_{[0,t]}(\pi_{\vec{\theta}}) = R\left(\vec{o}_1, \pi_{\vec{\theta}}(o_1), \cdots, \vec{o}_t, \pi_{\vec{\theta}}(o_t), \vec{o}_{t+1}\right)$. After random initialization, the parameters are continuously updated along the gradient direction with a learning rate λ: $\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}})$. The gradient ascent optimization is done with standard Adam (short for Adaptive Moment Estimation) optimizer to have the benefit of adaptive gradient descent with root mean square propagation (Kingma and Ba 2014). The whole process is summarized in algorithm 1.

Algorithm 1 Adversarial Policy Gradient

Input: initial policy parameters θ, empty replay buffer $\mathcal{D}$
repeat
  reset replay buffer
  while not terminal **do**
    Observe observation o and select action $a = \pi_\theta(o)$ with probability p and random action with probability $1-p$,
    Execute a in the environment
    Observe next observation o', reward r, and done signal d to indicate whether o' is terminal
    apply noise to next observation o'
    store (o, a, o') in replay buffer $\mathcal{D}$
    if Terminal **then**
      for however many updates in $\mathcal{D}$ **do**
        compute final reward R
      end for
      update network parameter with Adam gradient ascent $\vec{\theta} \longrightarrow \vec{\theta} + \lambda \nabla_{\vec{\theta}} J_{[0,t]}(\pi_{\vec{\theta}})$
    end if
  end while
until convergence

In our gradient ascent, we use a learning rate of 0.01, an adversarial Gaussian noise with a standard deviation of 0.002. We do up to 500 maximum iterations with an early stop condition if on the train set, there is no improvement over the last 50 iterations.

Experiments

Goal of the experiment

We are interested in planing a hedging strategy for a risky asset. The experiment is using daily data from 01/05/2000 to 19/06/2020 for the MSCI and 4 SG-CIB proprietary systematic strategies. The risky asset is the MSCI world index whose daily data can be found on Bloomberg. We choose this index because it is a good proxy for a wide range of asset manager portfolios. The hedging strategies are 4 SG-CIB proprietary systematic strategies further described below. Training and testing are done following extending walk forward analysis as presented in (Benhamou et al. 2020b; 2020c; 2020a) with initial training from 2000 to end of 2006 and testing in a rolling 1 year period. Hence, there are 14 training and testing periods, with the different testing period corresponding to all the years from 2007 to 2020 and training done for period starting in 2000 and ending one day before the start of the testing period.

Data-set description

Systematic strategies are similar to asset managers that invest in financial markets according to an adaptive, predefined trading rule. Here, we use 4 SG CIB proprietary 'hedging strategies', that tend to perform when stock markets are down:

- Directional hedges - react to small negative return in equities,
- Gap risk hedges - perform well in sudden market crashes,
- Proxy hedges - tend to perform in some market configurations, like for example when highly indebted stocks under-perform other stocks,
- Duration hedges - invest in bond market, a classical diversifier to equity risk in finance.

The underlying financial instruments vary from put options, listed futures, single stocks, to government bonds. Some of those strategies are akin to an insurance contract and bear a negative cost over the long run. The challenge consists in balancing cost versus benefits.

In practice, asset managers have to decide how much of these hedging strategies are needed on top of an existing portfolio to achieve a better risk reward. The decision making process is often based on contextual information, such as the economic and geopolitical environment, the level of risk aversion among investors and other correlation regimes. The contextual information is modeled by a large range of features :

- the level of risk aversion in financial markets, or market sentiment, measured as an indicator varying between 0 for maximum risk aversion and 1 for maximum risk appetite,
- the bond equity historical correlation, a classical ex-post measure of the diversification benefits of a duration hedge, measured on a 1 month, 3 month and 1 year rolling window,
- The credit spreads of global corporate - investment grade, high yield, in Europe and in the US - known to be an early indicator of potential economic tensions,
- The equity implied volatility, a measure if the 'fear factor' in financial market,
- The spread between the yield of Italian government bonds and the German government bond, a measure of potential tensions in the European Union,
- The US Treasury slope, a classical early indicator for US recession,
- And some more financial variables, often used as a gauge for global trade and activity: the dollar, the level of rates in the US, the estimated earnings per shares (EPS).

A cross validation step selects the most relevant features. In the present case, the first three features are selected. The

rebalancing of strategies in the portfolio comes with transaction costs, that can be quite high since hedges use options. Transactions costs are like frictions in physical systems. They are taken into account dynamically to penalise solutions with a high turnover rate.

Evaluation metrics

Asset managers use a wide range of metrics to evaluate the success of their investment decision. For a thorough review of those metrics, see for example (Cogneau and Hübner 2009). The metrics we are interested in for our hedging problem are listed below:

- annualized return defined as the average annualized compounded return,
- annualized daily based Sharpe ratio defined as the ratio of the annualized return over the annualized daily based volatility μ/σ,
- Sortino ratio computed as the ratio of the annualized return overt the downside standard deviation,
- maximum drawdown (max DD) computed as the maximum of all daily drawdowns. The daily drawdown is computed as the ratio of the difference between the running maximum of the portfolio value defined as $RM_T = \max_{t=0..T}(P_t)$ and the portfolio value over the running maximum of the portfolio value. Hence the drawdon at time T is given by $DD_T = (RM_T - P_T)/RM_T$ while the maximum drawdown $MDD_T = \max_{t=0..T}(DD_t)$. It is the maximum loss in return that an investor will incur if she/he invested at the worst time (at peak).

Results and discussion

Overall, the DRL approach achieves much better results than traditional methods as shown in table 1, except for the maximum drawdown (max DD). Because time horizon is important in the comparison we provide risk measures for the last 2 and 5 years to emphasize that the DRL approach seems more robust than traditional portfolio allocation methods.

When plotting performance results from 2007 to 2020 as shown in figure 3, we see that DRL model is able to deviate upward from the risky asset continuously, indicating a steady performance. In contrast, other financial models are not able to keep their marginal over-performance over time with respect to the risky asset and end slightly below the risky asset.

Allocation chosen by models

The reason of the stronger performance of DRL comes from the way it chooses its allocation. Contrarily to standard financial methods that play the diversification as shown in figure 4, DRL aims at choosing a single hedging strategy most of the time and at changing it dynamically, should the financial market conditions change. In a sense, DRL is doing some cherry picking by selecting what it thinks is the best hedging strategy.

In contrast, traditional models like Markowitz, minimum variance, maximum diversification, maximum decorrelation

Table 1: Models comparison over 2 and 5 years

	2 Years			
	return	Sortino	Sharpe	max DD
Risky asset	8.27%	0.39	0.36	- 0.34
DRL	**20.64%**	**0.94**	**0.96**	- 0.27
Markowitz	-0.25%	- 0.01	- 0.01	- 0.43
MinVariance	-0.22%	- 0.01	- 0.01	- 0.43
MaxDiversification	0.24%	0.01	0.01	- 0.43
MaxDecorrel	14.42%	0.65	0.63	- 0.21
RiskParity	14.17%	0.73	0.72	**-0.19**
	5 Years			
	return	Sortino	Sharpe	max DD
Risky asset	9.16%	0.57	0.54	- 0.34
DRL	**16.95%**	**1.00**	**1.02**	- 0.27
Markowitz	1.48%	0.07	0.06	- 0.43
MinVariance	1.56%	0.08	0.06	- 0.43
MaxDiversification	1.77%	0.08	0.07	- 0.43
MaxDecorrel	7.65%	0.44	0.39	- 0.21
RiskParity	7.46%	0.48	0.43	**-0.19**

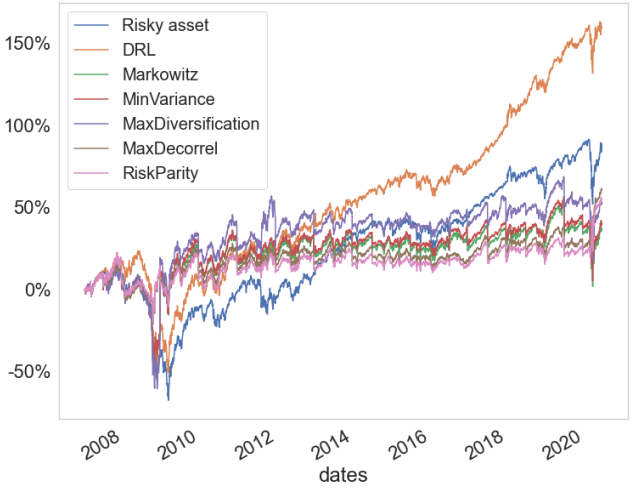

Figure 3: performance of all models

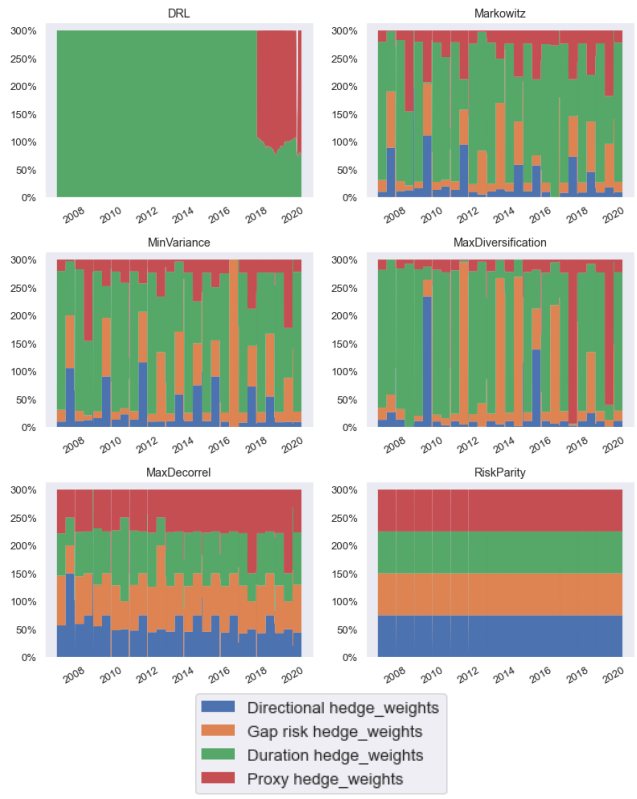

Figure 4: weights for all models

and risk parity provides non null weights for all our hedging strategies and do not do cherry picking at all. They are neither able to change the leverage used in the portfolio as opposed to DRL model.

Adaptation to the Covid Crisis

The DRL model can change its portfolio allocation should the market conditions change. This is the case from 2018 onwards, with a short deleveraging window emphasized by the small blank disruption during the Covid crisis as shown in figure 5. We observe in this figure where we have zoomed over year 2020, that the DRL model is able to reduce leverage from 300 % to 200 % during the Covid crisis (end of February 2020 to start of April 2020). This is a unique feature of our DRL model compared to traditional financial planning models that do not take leverage into account and keeps a leverage of 300 % regardless of market conditions.

Benefits of DRL

As illustrated by the experiment, the advantages of DRL are numerous: (i) DRL maps directly market conditions to actions by design and hence should adapt to changing environment, (ii) DRL does not rely on any traditional financial risk assumptions, (iii) DRL can incorporate additional data and be a multi inputs method as opposed to more traditional optimization methods.

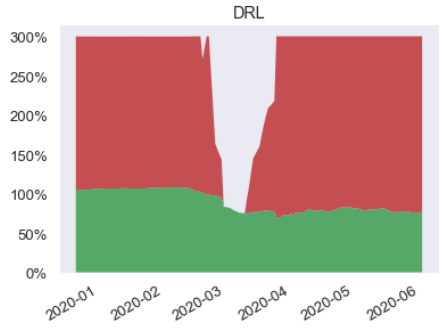

Figure 5: disallocation of DRL model

Future work

As nice as this work is, there is room for improvement as we have only tested a few scenarios and only a limited set of hyper-parameters for our convolutional networks. We should do more intensive testing to confirm that DRL is able to better adapt to changing financial environment. We should also investigate the impact of more layers and other design choice in our network.

Conclusion

In this paper, we discuss how a traditional portfolio allocation problem can be reformulated as a DRL problem, trying to bridge the gaps between the two approaches. We see that the DRL approach enables us to select fewer strategies, improving the overall results as opposed to traditional methods that are built on the concept of diversification. We also stress that DRL can better adapt to changing market conditions and is able to incorporate more information to make decision.

Acknowledgments

We would like to thank Beatrice Guez and Marc Pantic for meaningful remarks. The views contained in this document are those of the authors and do not necessarily reflect the ones of SG CIB.

References

Astrom, K. 1969. Optimal control of markov processes with incomplete state-information ii. the convexity of the loss-function. *Journal of Mathematical Analysis and Applications* 26(2):403–406.

Bao, W., and yang Liu, X. 2019. Multi-agent deep reinforcement learning for liquidation strategy analysis.

Benhamou, E.; Saltiel, D.; Ohana, J.-J.; and Atif, J. 2020a. Detecting and adapting to crisis pattern with context based deep reinforcement learning. *ICPR 2020*.

Benhamou, E.; Saltiel, D.; Ungari, S.; and Mukhopadhyay, A. 2020b. Aamdrl: Augmented asset management with deep reinforcement learning. *arXiv*.

Benhamou, E.; Saltiel, D.; Ungari, S.; and Mukhopadhyay, A. 2020c. Bridging the gap between markowitz planning and deep reinforcement learning. *ICAPS FinPlan workshop*.

Blum, A., and Furst, M. 1995. Fast Planning Through Planning Graph Analysis. In *IJCAI*, 1636–1642.

Chakraborty, S. 2019. Capturing financial markets to apply deep reinforcement learning.

Chopra, V. K., and Ziemba, W. T. 1993. The effect of errors in means, variances, and covariances on optimal portfolio choice. *Journal of Portfolio Management* 19(2):6–11.

Choueifaty, Y., and Coignard, Y. 2008. Toward maximum diversification. *Journal of Portfolio Management* 35(1):40–51.

Choueifaty, Y.; Froidure, T.; and Reynier, J. 2012. Properties of the most diversified portfolio. *Journal of Investment Strategies* 2(2):49–70.

Christoffersen, P.; Errunza, V.; Jacobs, K.; and Jin, X. 2010. Is the potential for international diversification disappearing? Working Paper.

Cogneau, P., and Hübner, G. 2009. The 101 ways to measure portfolio performance. *SSRN Electronic Journal*.

Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; and Dai, Q. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems* 28:1–12.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189.

Fischer, T. G. 2018. Reinforcement learning in financial markets - a survey. *Discussion Papers in Economics 12*.

Gu, S.; Holly, E.; Lillicrap, T.; and Levine, S. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *IEEE International Conference on Robotics and Automation (ICRA)*, 3389–3396.

Haugen, R., and Baker, N. 1991. The efficient market inefficiency of capitalization-weighted stock portfolios. *Journal of Portfolio Management* 17:35–40.

Huang, C. Y. 2018. Financial trading as a game: A deep reinforcement learning approach.

Jiang, Z., and Liang, J. 2016. Cryptocurrency Portfolio Management with Deep Reinforcement Learning. *arXiv e-prints*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization.

Kolm, P. N., and Ritter, G. 2019. Modern perspective on reinforcement learning in finance. *SSRN*.

Kritzman, M. 2014. Six practical comments about asset allocation. *Practical Applications* 1(3):6–11.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2015. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17.

Levine, S.; Pastor, P.; Krizhevsky, A.; and Quillen, D. 2016. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*.

Li, X.; Li, Y.; Zhan, Y.; and Liu, X.-Y. 2019. Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation. In *ICML*.

Liang et al. 2018. Adversarial deep reinforcement learning in portfolio management.

Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *CoRR*.

Liu, Y.; Liu, Q.; Zhao, H.; Pan, Z.; and Liu, C. 2020. Adaptive quantitative trading: an imitative deep reinforcement learning approach. In *AAAI*.

Maillard, S.; Roncalli, T.; and Teïletche, J. 2010. The properties of equally weighted risk contribution portfolios. *The Journal of Portfolio Management* 36(4):60–70.

Markowitz, H. 1952. Portfolio selection. *Journal of Finance* 7:77–91.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M.; Graves, A.; Riedmiller, M.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518:529–33.

Nan, A.; Perumal, A.; and Zaiane, O. R. 2020. Sentiment and knowledge based algorithmic trading with deep reinforcement learning.

Ning, B.; Lin, F. H. T.; and Jaimungal, S. 2018. Double deep q-learning for optimal execution.

Roncalli, T., and Weisang, G. 2016. Risk parity portfolios with risk factors. *Quantitative Finance* 16(3):377–388.

Saltiel, D.; Benhamou, E.; Ohana, J. J.; Laraki, R.; and Atif, J. 2020. Drlps: Deep reinforcement learning for portfolio selection. *ECML PKDD Demo track*.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; and Abbeel, P. 2015a. Trust region policy optimization. In *ICML*.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015b. High-dimensional continuous control using generalized advantage estimation. *ICLR*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *CoRR*.

Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; 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.; and Hassabis, D. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529:484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of go without human knowledge. *Nature* 550:354–.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Théate, T., and Ernst, D. 2020. Application of deep reinforcement learning in stock trading strategies and stock forecasting.

Vinyals, O.; Babuschkin, I.; Czarnecki, W.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J.; Jaderberg, M.; and Silver, D. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature* 575.

Wang, H., and Zhou, X. Y. 2019. Continuous-Time Mean-Variance Portfolio Selection: A Reinforcement Learning Framework. *arXiv e-prints*.

Wang, S.; Jia, D.; and Weng, X. 2018. Deep reinforcement learning for autonomous driving. *ArXiv* abs/1811.11329.

Wu, X.; Chen, H.; Wang, J.; Troiano, L.; Loia, V.; and Fujita, H. 2020. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences* 538:142–158.

Xiong, Z.; Liu, X.-Y.; Zhong, S.; Yang, H.; and Walid, A. 2019. Practical deep reinforcement learning approach for stock trading.

Ye, Y.; Pei, H.; Wang, B.; Chen, P.-Y.; Zhu, Y.; Xiao, J.; and Li, B. 2020. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *AAAI*.

Yu, P.; Lee, J. S.; Kulyatin, I.; Shi, Z.; and Dasgupta, S. 2019. Model-based deep reinforcement learning for financial portfolio optimization. *RWSDM Workshop, ICML 2019*.

Zhang, Z.; Zohren, S.; and Roberts, S. 2019. Deep reinforcement learning for trading.

Zhengyao et al. 2017. Reinforcement learning framework for the financial portfolio management problem. *arXiv*.