# Chapter 1

# Reinforcement Learning

## 1.1 The Reinforcement Learning Problem

## 1.2 Markov Decision Processes

The reinforcement learning problem is modeled using Markov decision processes.

**Definition 1.2.1** (Markov Decision Process)**.** *A Markov decision process (MDP) is a tuple* $<\mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma>$*, where*

*i)* $(\mathbb{S}, \mathcal{S})$ *is a measurable state space.*

*ii)* $(\mathbb{A}, \mathcal{A})$ *is a measurable action space.*

*iii)* $\mathcal{P} : \mathbb{S} \times \mathbb{A} \times \mathcal{S} \to \mathbb{R}$ *is a Markov transition kernel, i.e.*

*a) for every* $s \in \mathbb{S}$ *and* $a \in \mathbb{A}$*,* $B \mapsto \mathcal{P}(s, a, B)$ *is a probability distribution over* $(\mathbb{S}, \mathcal{S})$*.*

*b) for every* $B \in \mathcal{S}$*,* $(s, a) \mapsto \mathcal{P}(s, a, B)$ *is a measurable function on* $\mathbb{S} \times \mathbb{A}$*.*

*iv)* $\mathcal{R} : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ *is a reward function.*

*v)* $\gamma \in (0, 1)$ *is a discount factor.*

The kernel $\mathcal{P}$ describes the random evolution of the system: suppose that at time $t$ the system is in state $s_t$ and that the agent takes action $a_t$, then, regardless of the previous history of the system, the probability to find the system in a state belonging to $B \in \mathcal{S}$ at time $t+1$ is given by $\mathcal{P}(s_t, a_t, B)$, i.e.

$$\mathcal{P}(s_t, a_t, B) = \mathbb{P}\left(S_{t+1} \in B | S_t = s_t, A_t = a_t\right) \tag{1.1}$$

Following this random transition, the agent receives a stochastic reward $R_{t+1}$. The reward function $\mathcal{R}(s_t, a_t)$ gives the expected reward obtained when action $a_t$ is taken in state $s_t$, i.e.

$$\mathcal{R}(s_t, a_t) = \mathbb{E}\left[R_{t+1} | S_t = s_t, A_t = a_t\right] \tag{1.2}$$

At any time step, the agent selects his actions according to a certain policy.

**Definition 1.2.2** (Policy)**.** *A policy is a function $\pi : \mathbb{S} \times \mathcal{A} \to \mathbb{R}$ such that*
*i) for every $s \in \mathbb{S}$, $C \mapsto \pi(s, C)$ is a probability distribution over $(\mathbb{A}, \mathcal{A})$.*
*ii) for every $C \in \mathcal{A}$, $s \mapsto \pi(s, C)$ is a measurable function.*

Intuitively, a policy represents a stochastic mapping from the current state of the system to actions. Deterministic policies are a particular case of this general definition. We assumed that the agent's policy is stationary and only depends on the current state of the system. We might in fact consider more general policies that depends on the whole history of the system. However, as we will see, we can always find an optimal policy that depends only on the current state, so that our definition is not restrictive. A policy $\pi$ and an initial state $s_0 \in \mathbb{S}$ together determine a random state-action-reward sequence $\{(S_t, A_t, R_{t+1})\}_{t\geq 0}$ with values on $\mathbb{S} \times \mathbb{A} \times \mathbb{R}$ following the mechanism described above. We introduce the useful concept of history of an MDP

**Definition 1.2.3** (History)**.** *Given an initial state $s_0 \in \mathbb{S}$ and a policy $\pi$, a history (or equivalently trajectory or roll-out) of the system is a random sequence $H_\pi = \{(S_t, A_t)\}_{t\geq 0}$ with values on $\mathbb{S}\times\mathbb{A}$, defined on some probability space $(\Omega, \mathcal{F}, \mathbb{P})$, such that for $t = 0, 1, \dots$*

$$\begin{cases} S_0 = s_0 \\ A_t \sim \pi(S_t, \cdot) \\ S_{t+1} \sim \mathcal{P}(S_t, A_t, \cdot) \end{cases} \tag{1.3}$$

*we will denote by $(\mathbb{H}, \mathcal{H})$ the measurable space of all possible histories.*

Moreover, we observe that
i) the state sequence $\{S_t\}_{t\geq 0}$ is a Markov process $< \mathbb{S}, \mathcal{P}_\pi >$
ii) the state-reward sequence $\{(S_t, R_t)\}_{t\geq 0}$ is a Markov reward process $< \mathbb{S}, \mathcal{P}_\pi, \mathcal{R}_\pi, \gamma >$
where we denoted

$$\begin{aligned} \mathcal{P}_\pi(s, s') &= \int_{\mathbb{A}} \pi(s, a) \mathcal{P}(s, a, s') da \\ \mathcal{R}_\pi(s) &= \int_{\mathbb{A}} \pi(s, a) \mathcal{R}(s, a) da \end{aligned} \tag{1.4}$$

The goal of the agent is to maximize his expected return.

**Definition 1.2.4** (Return). *The return is the total discounted reward obtained by the agent starting from t*

$$G_t = \sum_{t=0}^{\infty} \gamma^t R_{t+k+1} \tag{1.5}$$

where $0 < \gamma < 1$ is the discount factor. The discount factor models the trade-off between immediate and delayed reward: if $\gamma = 0$ the agent selects his actions in a myopic way, while if $\gamma \rightarrow 1$ he acts in a far-sighted manner. There are other possilble reasons for discounting future rewards. The first is because it is mathematically convenient, as it avoids infinite returns and it solves many convergence issues. Another interpretation is that it models the uncertainty about the future, which may not be fully represented. Finally, the financial interpration is that discounting gives the present value of future rewards. Since the return are stochastic, we consider their expected value.

**Definition 1.2.5** (State-Value Function). *The state-value function $V_\pi : \mathbb{S} \rightarrow \mathbb{R}$ is the expected return that can be obtained starting from a state and following policy $\pi$*

$$V_\pi(s) = \mathbb{E}_\pi \left[ G_t | S_t = s \right] \tag{1.6}$$

where $\mathbb{E}_\pi$ indicates that all the actions are selected according to policy $\pi$. In reinforcement learning, it is useful to consider another function

**Definition 1.2.6** (Action-Value Function). *The action-value function $Q_\pi : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$ is the expected return that can be obtained starting from a state, taking an action and then following policy $\pi$*

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[ G_t | S_t = s, A_t = a \right] \tag{1.7}$$

**Definition 1.2.7** (Optimal State-Value Function). *The optimal state-value function $V_* : \mathbb{S} \rightarrow \mathbb{R}$ is the largest expected return that can be obtained starting from a state*

$$V_*(s) = \sup_\pi V_\pi(s) \tag{1.8}$$

**Definition 1.2.8** (Optimal Action-Value Function). *The optimal action-value function $Q_* : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$ is the largest expected return that can be obtained starting from a state and taking an action*

$$Q_*(s, a) = \sup_\pi Q_\pi(s, a) \tag{1.9}$$

The agent goal is to select a policy $\pi_*$ that maximize his expected return in all possible states. Such a policy is called *optimal.* More formally, we introduce the following partial ordering in the policy space

$$\pi \succeq \pi' \Leftrightarrow V_\pi(s) \geq V_{\pi'}(s) \quad \forall s \in \mathbb{S} \tag{1.10}$$

Then the optimal policy $\pi_* \succeq \pi$, $\forall \pi$.

### 1.2.1 Bellman Equations

### 1.2.2 Risk-Sensitive MDP

## 1.3 Policy Gradient

In policy gradient methods, we directly store and iteratively improve an approximation of the optimal policy. More formally, we consider a parametrized policy $\pi : \mathbb{S} \times \mathcal{A} \times \Theta \to \mathbb{R}$ such that, for every $s \in \mathbb{S}$, $B \in \mathcal{A}$ and a given policy parameter vector $\theta \in \Theta \subseteq \mathbb{R}^{D_\theta}$, $\pi(s, B; \theta) = \pi_\theta(s, B)$ gives the probability of selecting an action in $B$ when the system is in state $s$. This policy is also called an *actor* and methods that directly approximate the policy without exploiting an approximation of the optimal value function are called *actor-only*. We will also see how to combine an approximation of the optimal policy with an approximation of the value function, in what are commonly called *actor-critc* methods. As we have seen in the previous sections, an optimal policy can be derived by simply acting greedily with respect to the optimal action-value function. However, in large or continuous action spaces, this leads to a complex optimization problem that is computationally expensive to solve. Therefore, it can be beneficial to store an explicit estimation of the optimal policy from which we can select actions. Policy gradient methods have other advantages compared to standard value-based approaches

i) these methods have better convergence properties and are guaranteed to converge at least to a local optimum, which may be good enough in practice.
ii) they are effective in high-dimensional or continuous action spaces.
iii) they can learn stochastic policies and not only deterministic ones.
iv) In many applications, the optimal policy has a more compact representation that the value function, so that it might be easier to approximate.

On the other hand, policy gradient methods have a large variance which may hinder the converge speed. We will see in the following sections how different methods address this problem.

### 1.3.1 Basics of Policy Gradient

The general goal of policy optimization in reinforcement learning is to optimize the policy parameters $\theta \in \Theta$ so as to maximize a certain objective function $J : \Theta \to \mathbb{R}$

$$\max_{\theta \in \Theta} J(\theta) \tag{1.11}$$

There are various choices for the objective function.

**Definition 1.3.1** (Start Value)**.** *In an episodic environment, the start value is the expected return that can be obtained starting from the start state* $s^* \in \mathbb{S}$ *and following policy* $\pi_\theta$

$$J_{start}(\theta) = V_{\pi_\theta}(s^*) = \mathbb{E}_{\pi_\theta}\left[G_t | S_t = s^*\right] \tag{1.12}$$

**Definition 1.3.2** (Average Value)**.** *In a continuing environment, the average value is the expected value that can be obtained following policy* $\pi_\theta$

$$J_{avV}(\theta) = \mathbb{E}_{S\sim\mu}\left[V_{\pi_\theta}(S)\right] = \int_{\mathbb{S}} V_{\pi_\theta}(s)\mu(s)ds \tag{1.13}$$

*where* $\mu$ *is a probability distribution over* $(\mathbb{S}, \mathcal{S})$*.*

**Definition 1.3.3** (Average Reward per Time Step)**.** *The average reward per time step is the expected reward that can be obtained over a single time step by following policy* $\pi_\theta$

$$J_{avR}(\theta) = \mathbb{E}_{\substack{S\sim\mu \\ A\sim\pi_\theta}}\left[\mathcal{R}(S, A)\right] = \int_{\mathbb{S}} \mu(s) \int_{\mathbb{A}} \pi_\theta(s, a)\mathcal{R}(s, a)dads \tag{1.14}$$

*where* $\mu$ *is a probability distribution over* $(\mathbb{S}, \mathcal{S})$*.*

Fortunately, the same methods apply to the three formulations. In the following, we will focus on gradient-based and model-free methods that exploit the sequential structure of the the reinforcement learning problem. The idea of policy gradient algorithms is to update the policy parameters using the gradient ascent direction of the objective function

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J|_{\theta=\theta_k} \tag{1.15}$$

where $\{\alpha_k\}_{k\geq 0}$ is a sequence of learning rates. Typically, the gradient of the objective function is not know and needs to be estimated. It is a well-know result from stochastic optimization that, if the gradient estimate is unbiased and the learning rates satisfy the *Robbins-Monro conditions*

$$\sum_{k=0}^{\infty} \alpha_k = \infty \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty \tag{1.16}$$

the learning process is guaranteed to converge at least to a local optimum of the objective function. In the following sections, we describe various methods of approximating the gradient.

### 1.3.2 Finite Differences

### 1.3.3 Monte-Carlo Policy Gradient

Let $h = \{(s_t, a_t)\}_{t \geq 0} \in \mathbb{H}$ be a given trajectory and let us denote by $p_\theta(h) = \mathbb{P}_{\pi_\theta}(H = h)$ the probability of obtaining this trajectory by following policy $\pi_\theta$. Let $G(h)$ denote the expect return obtained on trajectory $h$

$$G(h) = \mathbb{E}\left[G_0 | H_t = h\right] = \sum_{t=1}^{\infty} \gamma^k \mathcal{R}(s_{t-1}, a_{t-1}) \tag{1.17}$$

For simplicity, let us consider the average value objective function which can be rewritten as an expectation over all possible trajectories

$$J_{\mathrm{avV}}(\theta) = \int_{\mathbb{H}} p_\theta(h) G(h) dh \tag{1.18}$$

We can compute its gradient using the likelihood ratio trick

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_{\mathbb{H}} \nabla_\theta p_\theta(h) G(h) dh \\ &= \int_{\mathbb{H}} p_\theta(h) \nabla_\theta \log p_\theta(h) G(h) dh \\ &= \mathbb{E}\left[\nabla_\theta \log p_\theta(H) G(H)\right] \end{aligned} \tag{1.19}$$

where the expectation is taken over all possible trajectories. The crucial point is that $\nabla_\theta \log p_\theta(H)$ can be computed without knowledge of the transition probability kernel $\mathcal{P}$. Indeed

$$p_\theta(h) = \mathbb{P}\left(S_0 = s_0\right) \prod_{t=0}^{\infty} \pi_\theta(s_t, a_t) \mathcal{P}(s_t, a_t, s_{t+1})$$

$$\log p_\theta(h) = \log \mathbb{P}\left(()\, S_0 = s_0\right) + \sum_{t=0}^{\infty} \log \pi_\theta(s_t, a_t) + \sum_{t=0}^{\infty} \log \mathcal{P}(s_t, a_t, s_{t+1})$$

The only term depending on the parameters $\theta$ is the policy term, so that

$$\nabla_\theta \log p_\theta(H) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(s_t, a_t) \tag{1.20}$$

So we do not need the transition model to compute the $\nabla_\theta \log p_\theta(H)$. However, this trick only works if the policy is stochastic. In most cases this is not a big problem, since stochastic policies are needed anyway to ensure sufficient exploration. There are two important classes of stochastic policies that work well in this framework: softmax policies and Gaussian policies [TODO]. Moreover, since

$$\int_{\mathbb{H}} \nabla_\theta p_\theta(h) dh = 0$$

a constant baseline $b \in \mathbb{R}$ can always be added in the gradient formula

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log p_\theta(H)(G(H) - b)\right] \tag{1.21}$$

We will see how this baseline can be chosen in order to minimize the variance of the estimator. In an episodic environment, we can derive an estimate of the objective function gradient by sampling $M$ trajectories $h^{(m)} = \{(s_t^{(m)}, a_t^{(m)})\}_{t=0}^{T^{(m)}}$ from the MDP and by approximating the expected value via Monte Carlo

$$g_{\text{RF}} = \frac{1}{M} \sum_{m=1}^{M} \left[ \sum_{i=0}^{T^{(m)}} \nabla_\theta \log \pi_\theta(s_i^{(m)}, a_i^{(m)}) \right] \left[ \sum_{j=0}^{T^{(m)}} \gamma^j r_{j+1}^{(m)} - b \right] \tag{1.22}$$

This method is known in the literature as the REINFORCE algorithm and is guaranteed to converge to the true gradient at a pace of $O(M^{-1/2})$. In practice, we can obtain an approximation of the gradient using only one sample which leads to a stochastic gradient ascent method

$$g_{\text{SRF}} = \left[ \sum_{i=0}^{T} \nabla_\theta \log \pi_\theta(s_i, a_i) \right] \left[ \sum_{j=0}^{T} \gamma^j r_{j+1} - b \right] \tag{1.23}$$

This method is very easy and works well on many problems. However, the gradient estimate is characterized by a large variance which can hamper the convergence rate of the algorithm. A first approach to adress this issue is to optimally set the benchmark to reduce the estimate variance. [TODO]

### 1.3.4 Policy Gradient Theorem

In the last section, we said that the REINFORCE gradient estimate is characterized by a large variance, which may slow the method's convergence. To improve the estimate, it is sufficient to notice that future actions do not depend on past rewards, unless the policy has been changed. Therefore,

$$\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(S_t, A_t) \mathcal{R}(S_s, A_s)\right] = 0 \qquad \forall t > s \tag{1.24}$$

From this trivial remark, we can derive two estimates of the objective function gradient

$$g_{\text{PG}} = \frac{1}{M} \sum_{m=1}^{M} \sum_{i=0}^{T^{(m)}} \nabla_\theta \log \pi_\theta(s_i^{(m)}, a_i^{(m)}) \left( \sum_{j=i}^{T^{(m)}} \gamma^j r_{j+1}^{(m)} - b \right) \tag{1.25}$$

$$g_{\text{GMDP}} = \frac{1}{M} \sum_{m=1}^{M} \sum_{j=0}^{T^{(m)}} \left[ \sum_{i=j}^{T^{(m)}} \nabla_\theta \log \pi_\theta(s_i^{(m)}, a_i^{(m)}) \right] \left( \gamma^j r_{j+1}^{(m)} - b \right) \tag{1.26}$$

The two estimates are exactly equivalent. This simple trick greatly reduces the estimate variance and this can speed up convergence. These algorithms can all be derived from a more general result: the policy gradient theorem.

**Theorem 1.3.1** (Policy Gradient)**.** *For any differentiable policy $\pi_\theta$, for any of the policy objective functions $J = J_{start}$, $J_{avV}$, $J_{avR}$, $\frac{1}{1-\gamma} J_{avV}$, the policy gradient is*

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\substack{S\sim\mu \\ A\sim\pi_\theta}} \left[ \nabla_\theta \log \pi_\theta(S, A) Q_{\pi_\theta}(S, A) \right] \tag{1.27}$$

The problem is that the action-value function is typically unknown and needs to be approximated. For instance, REINFORCE replaces it with the realized return achieved on a sample trajectory, that are an unbiased estimate of the action-value function. However, the theorem can be used as the starting point to derive many other policy gradient methods that use different approximation of the action-value function.

#### 1.3.4.1 Actor-Critic Policy Gradient

Actor-Critic Policy Gradient employs a *critic*, that is a parametric approximation $\widehat{Q} : \mathbb{S} \times \mathbb{A} \times \Psi \to \mathbb{R}$, where $\Psi \in \mathbb{R}^{D_\psi}$ such that $\widehat{Q}_\psi(s, a) = \widehat{Q}(s, a; \psi) \approx Q_{\pi_\theta}(s, a)$. Therefore these methods maintain two sets of parameters: a *critic* that updates the action-value function parameters $\psi$ and an *actor* that updates the policy parameters $\theta$ in the direction suggested by the critic. More formally, given $\widehat{Q}_\psi$, the current state of the system $s_t$ and the action $a_t$ selected using policy $\pi_\theta$, the policy parameters are updated in the approximated gradient direction

$$\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta \pi_\theta(s_t, a_t) \widehat{Q}_\psi(s_t, a_t) \tag{1.28}$$

On the other hand, the action-value function parameters $\psi$ can be updated using any value-based approach, for instance TD(0). This leads to the QAC algorithm, for which the pseudo code is reported in **??**.

[TODO: TD policy gradient, compatible function approximation, advantage function and actor-critic policy gradient]

### 1.3.5 Natural Policy Gradient

### 1.3.6 Policy Gradient with Parameter Exploration

In REINFORCE, trajectories are generated by sampling at each time step an action from a stochastic policy $\pi_\theta$. This process leads to a large variance in the gradient estimate which can slow down the method convergence. To address this issue, in [48] the authors propose the *policy gradient with parameter-based exploration*, in which the search in the policy space is replaced with a direct search in the model parameter space. In this method, we assume a determinstic policy $F : \mathbb{S} \times \Theta \to \mathbb{A}$ that, given a set of parameters $\theta \in \Theta \subseteq \mathbb{R}^{D_\theta}$, maps a state $s \in \mathbb{S}$ to an action $a = F(s; \theta) = F_\theta(s) \in \mathbb{A}$. However, the policy parameters $\theta$ are random variables distributed according to a probability distribution parametrized by some hyperparameters $\xi \in \Xi \subseteq \mathbb{R}^{D_\xi}$, i.e. $\theta \sim p_\xi(\theta)$. Combining these two hypotheses, we obtain a stochastic policy parametrized by the hyperparameters $\xi$

$$\pi_\xi(s, a) = \pi(s, a; \xi) = \int_\Theta p_\xi(\theta)\delta_{F_\theta(s),a} d\theta \tag{1.29}$$

where $\delta$ denotes the Dirac function. The advantage of this approach is that the policy is deterministic and therefore the actions do not need to be sampled at each time step. It is sufficient to sample the parameters $\theta$ once at the beginning of the period and then generate an entire trajectory using the deterministic policy $F_\theta$, which greatly reduced the gradient estimate variance. The hyperparameters $\xi$ will be updated following the gradient of the expected reward, which can be rewritten as

$$J(\xi) = \int_\Theta \int_\mathbb{H} p_\xi(\theta, h) G(h) dh d\theta \tag{1.30}$$

Hence, using the fact that $h$ is conditionally independent from $\xi$ given $\theta$, so that $p_\xi(\theta, h) = p_\xi(\theta) p_\theta(h)$, and the likelihood trick

$$\begin{aligned}
\nabla_\xi J(\xi) &= \int_\Theta \int_\mathbb{H} \nabla_\xi p_\xi(\theta) p_\theta(h) G(h) dh d\theta \\
&= \int_\Theta \int_\mathbb{H} p_\xi(\theta) p_\theta(h) \nabla_\xi \log p_\xi(\theta) G(h) dh d\theta \\
&= \mathbb{E}\left[\nabla_\xi \log p_\xi(\theta) G(H)\right]
\end{aligned} \tag{1.31}$$

Again, we can subtract a constant baseline $b \in \mathbb{R}$ from the total return

$$\nabla_\xi J(\xi) = \mathbb{E}\left[\nabla_\xi \log p_\xi(\theta) \left(G(H) - b\right)\right] \tag{1.32}$$

By sampling $\theta \sim p_\xi$ and then generating $M$ trajectories $h^{(m)} = \{(s_t^{(m)}, a_t^{(m)})\}_{t \geq 0}$ following the deterministic policy $F_\theta$, this gradient can be approximated via Monte Carlo with

$$g_{\text{PGPE}} = \frac{1}{M} \sum_{m=1}^{M} \nabla_\xi \log p_\xi(\theta) \left[G\left(h^{(m)}\right) - b\right] \tag{1.33}$$

Alternatively, we can use a fully stochastic approximation by sampling a single trajectory and approximating the gradient as

$$g_{\text{PGPE}} = \nabla_\xi \log p_\xi(\theta) \left[ G\left(h^{(m)}\right) - b \right] \tag{1.34}$$

If we assume that all the components of the parameter vector $\theta$ are independent and normally distributed with mean $\mu_i$ and variance $\sigma_i^2$, i.e. $\theta_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, the gradient with respect to the hyperparameters $\xi = (\mu, \sigma)$ is given by

$$\begin{aligned} \frac{\partial \log p_\xi(\theta)}{\partial \mu_i} &= \frac{\theta_i - \mu_i}{\sigma_i^2} \\ \frac{\partial \log p_\xi(\theta)}{\partial \sigma_i} &= \frac{(\theta_i - \mu_i)^2 - \sigma_i^2}{\sigma_i^3} \end{aligned} \tag{1.35}$$

# Bibliography

[1] Agarwal, A., Bartlett, P., and Dama, M. Optimal allocation strategies for the dark pool problem. *arXiv preprint arXiv:1003.2245* (2010).

[2] Almgren, R., and Chriss, N. Optimal execution of portfolio transactions. *Journal of Risk 3* (2001), 5–40.

[3] Bäuerle, N., and Rieder, U. *Markov decision processes with applications to finance.* Springer Science & Business Media, 2011.

[4] Bekiros, S. D. Heterogeneous trading strategies with adaptive fuzzy actor–critic reinforcement learning: A behavioral approach. *Journal of Economic Dynamics and Control 34*, 6 (2010), 1153–1170.

[5] Bertoluzzo, F., and Corazza, M. Testing different reinforcement learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance 3* (2012), 68–77.

[6] Bertoluzzo, F., and Corazza, M. Q-learning-based financial trading systems with applications. *University Ca'Foscari of Venice, Dept. of Economics Working Paper Series No 15* (2014).

[7] Bertoluzzo, F., and Corazza, M. Reinforcement learning for automated financial trading: Basics and applications. In *Recent Advances of Neural Network Models and Applications.* Springer, 2014, pp. 197–213.

[8] Bertsekas, D. P. *Dynamic programming and optimal control*, vol. 1. Athena Scientific, Belmont, 1995.

[9] Bertsekas, D. P., and Tsitsiklis, J. N. *Neuro-Dynamic Programming*, 1 ed. Optimization and neural computation series. Athena Scientific, Belmont, 1996.

[10] Bishop, C. M. *Pattern Recognition and Machine Learning.* Springer, 2006.

[11] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. *Reinforcement learning and dynamic programming using function approximators*, vol. 39. CRC press, 2010.

[12] Casqueiro, P. X., and Rodrigues, A. J. Neuro-dynamic trading methods. *European Journal of Operational Research 175*, 3 (2006), 1400–1412.

[13] Chapados, N., and Bengio, Y. Cost functions and model combination for var-based asset allocation using neural networks. *IEEE Transactions on Neural Networks 12* (2001), 890–906.

[14] Choey, M., and Weigend, A. S. Nonlinear trading models through sharpe ratio maximization. *International Journal of Neural Systems 8* (1997), 417–431.

[15] Chow, Y., Tamar, A., Mannor, S., and Pavone, M. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems* (2015), pp. 1522–1530.

[16] Corazza, M., and Sangalli, A. Q-learning vs. sarsa: comparing two intelligent stochastic control approaches for financial trading.

[17] Cumming, J., Alrajeh, D., and Dickens, L. An investigation into the use of reinforcement learning techniques within the algorithmic trading domain.

[18] Dempster, M. A., and Leemans, V. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications 30*, 3 (2006), 543–552.

[19] Dempster, M. A. H., and Romahi, Y. S. *Intraday FX trading: An evolutionary reinforcement learning approach.* Springer, 2002.

[20] Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems* (2016).

[21] Deng, Y., Kong, Y., Bao, F., and Dai, Q. Sparse coding inspired optimal trading system for hft industry. *IEEE Transactions on Industrial Informatics 11*, 2 (2015), 467–475.

[22] Du, X., Zhai, J., and Lv, K. Algorithm trading using q-learning and recurrent reinforcement learning. *positions 1*, 1.

[23] Elder, T. Creating algorithmic traders with hierarchical reinforcement learning.

[24] Feldkamp, L. A., Prokhorov, D. V., Eagen, C. F., and Yuan, F. Enhanced multi-stream kalman filter training for recurrent networks. In *Nonlinear Modeling*. Springer, 1998, pp. 29–53.

[25] Ganchev, K., Nevmyvaka, Y., Kearns, M., and Vaughan, J. W. Censored exploration and the dark pool problem. *Communications of the ACM 53*, 5 (2010), 99–107.

[26] Gold, C. Fx trading via recurrent reinforcement learning. In *IEEE 2003 IEEE International Conference on Computational Intelligence for Financial Engineering. Proceedings* (2003).

[27] Goodfellow, I., Bengio, Y., and Courville, A. Deep learning. Book in preparation for MIT Press, 2016.

[28] Hastie, T., Tibshirani, R., and Friedman, J. *The elements of statistical learning: data mining, inference, and prediction.* Springer, 2009.

[29] Hendricks, D., and Wilcox, D. A reinforcement learning extension to the almgren-chriss framework for optimal trade execution. In *IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)* (2014), pp. 457–464.

[30] Jaeger, H. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach.* GMD-Forschungszentrum Informationstechnik, 2002.

[31] Kamijo, K., and Tanigawa, T. Stock price pattern recognition-a recurrent neural network approach. In *IEEE 1990 IJCNN International Joint Conference on Neural Networks* (1990).

[32] Kearns, M., and Nevmyvaka, Y. Machine learning for market microstructure and high frequency trading. *High-Frequency Trading–New Realities for Traders, Markets and Regulators* (2013), 91–124.

[33] Konda, V. R., and Tsitsiklis, J. N. Actor-critic algorithms. In *NIPS* (1999), vol. 13, pp. 1008–1014.

[34] Laruelle, S., Lehalle, C.-A., and Pages, G. Optimal split of orders across liquidity pools: a stochastic algorithm approach. *SIAM Journal on Financial Mathematics 2*, 1 (2011), 1042–1076.

[35] Laruelle, S., Lehalle, C.-A., and Pages, G. Optimal posting price of limit orders: learning by trading. *Mathematics and Financial Economics 7*, 3 (2013), 359–403.

[36] Li, H., Dagli, C. H., and Enke, D. Short-term stock market timing prediction under reinforcement learning schemes. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning* (2007), pp. 233–240.

[37] Moody, J., and Saffell, M. Learning to trade via direct reinforcement. *Neural Networks, IEEE Transactions on 12*, 4 (2001), 875–889.

[38] Moody, J., Saffell, M., Liao, Y., and Wu, L. Reinforcement learning for trading systems. In *Decision Technologies for Computational Finance: Proceedings of the fifth International Conference Computational Finance* (2013), vol. 2, Springer Science & Business Media, p. 129.

[39] Moody, J., and Wu, L. Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)* (1997), pp. 300–307.

[40] Moody, J., Wu, L., Liao, Y., and Saffell, M. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting 17* (1998), 441–470.

[41] Nevmyvaka, Y., Feng, Y., and Kearns, M. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning* (2006), ACM, pp. 673–680.

[42] Nocedal, J., and Wright, S. *Numerical optimization.* Springer Science & Business Media, 2006.

[43] O, J., Lee, J., Lee, J. W., and Zhang, B.-T. Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences 176*, 15 (2006), 2121–2147.

[44] Peters, J., and Schaal, S. Policy gradient methods for robotics. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on* (2006), IEEE, pp. 2219–2225.

[45] Peters, J., and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks 21*, 4 (2008), 682–697.

[46] SAAD, E. W., PROKHOROV, D. V., AND WUNSCH, D. C. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks 9*, 6 (1998), 1456–1470.

[47] SEHNKE, F., ET AL. *Parameter exploring policy gradients and their implications*. PhD thesis, Technische Universität München, 2012.

[48] SEHNKE, F., OSENDORFER, C., RÜCKSTIESS, T., GRAVES, A., PETERS, J., AND SCHMIDHUBER, J. Policy gradients with parameter-based exploration for control. In *Artificial Neural Networks-ICANN 2008*. Springer, 2008, pp. 387–396.

[49] SEHNKE, F., OSENDORFER, C., RÜCKSTIESS, T., GRAVES, A., PETERS, J., AND SCHMIDHUBER, J. Parameter-exploring policy gradients. *Neural Networks 23*, 4 (2010), 551–559.

[50] SILVER, D., LEVER, G., HEESS, N., DEGRIS, T., WIERSTRA, D., AND RIEDMILLER, M. Deterministic policy gradient algorithms. In *ICML* (2014).

[51] SUTTON, R. S., AND BARTO, A. G. *Introduction to reinforcement learning*, vol. 135. MIT Press Cambridge, 1998.

[52] SUTTON, R. S., MCALLESTER, D. A., SINGH, S. P., MANSOUR, Y., ET AL. Policy gradient methods for reinforcement learning with function approximation. In *NIPS* (1999), vol. 99, pp. 1057–1063.

[53] SZEPESVÁRI, C. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning 4*, 1 (2010), 1–103.

[54] TAMAR, A., CASTRO, D. D., AND MANNOR, S. Temporal difference methods for the variance of the reward to go. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (2013), pp. 495–503.

[55] TAMAR, A., CHOW, Y., GHAVAMZADEH, M., AND MANNOR, S. Policy gradient for coherent risk measures. In *Advances in Neural Information Processing Systems* (2015), pp. 1468–1476.

[56] TAMAR, A., DI CASTRO, D., AND MANNOR, S. Policy gradients with variance related risk criteria. In *Proceedings of the 29th International Conference on Machine Learning* (2012), pp. 387–396.

[57] Tan, Z., Quek, C., and Cheng, P. Y. Stock trading with cycles: A financial application of anfis and reinforcement learning. *Expert Systems with Applications 38*, 5 (2011), 4741–4755.

[58] Tsay, R. S. *Analysis of financial time series*, vol. 543. John Wiley & Sons, 2005.

[59] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (1990), 1550–1560.

[60] Wiering, M., and Van Otterlo, M. *Reinforcement Learning: State-of-the-Art*, 1 ed., vol. 12 of *Adaptation, Learning, and Optimization.* Springer, 2012.

[61] Williams, R. J., and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural computation 1*, 2 (1989), 270–280.

[62] Yang, S., Paddrik, M., Hayes, R., Todd, A., Kirilenko, A., Beling, P., and Scherer, W. Behavior based learning in identifying high frequency trading strategies. In *IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)* (2012).