Politecnico di Milano
Department of Information, Electronics and Bioengineering
Doctoral Programme in Information Technology

On the Exploitation of Uncertainty to Improve Maximum Expected Value Estimate and Exploration in Reinforcement Learning

Doctoral Dissertation of:
Carlo D'Eramo

Supervisor:
Prof. Marcello Restelli

Tutor:
Prof. Andrea Bonarini

The Chair of the Doctoral Program:
Prof. Marcello Restelli

2019 – XXXI cycle

Abstract

ABSTRACT goes here.

Summary

SUMMARY goes here.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

D	
DL	Deep Learning. 4
DRL	Deep Reinforcement Learning. 4
M	
MAB	Multi-Armed Bandit. 3
MDP	Markov Decision Process. 6, 7
ML	Machine Learning. 2
R	
RL	Reinforcement Learning. 2–5

CHAPTER 1

Introduction

EVERYONE experiences the process of taking decisions during his life. As a matter of fact, drastically the life of an individual can be synthesized in its *perception* of the world and its *interaction* with it. The concepts of perception and interaction might seem quite straightforward to understand: for a human the perception of the world comes from its senses and the interaction comes from its possibility to change its surroundings. On the contrary, these concepts are actually absolutely hard to define and aroused, during the centuries, a strong debate between scientists, biologists, and even philosophers.

1.1 Perception and interaction

We start from the assumption that, by definition, an individual perceive the environment around it and acts on it in order to achieve *goals* expressed by its will. In other words, all the actions made by an individual are done to satisfy its will to obtain something from the world it lives in. This task is naturally performed by humans, but it implies some challenging problems that are hard, or unfeasible, to solve. One of them comes from the intrinsic *uncertainty* of the perception we have of the world around us. Indeed, the perception of the world consists in the interpretation of the information provided by senses, but the process of information retrieval by senses and the mental processes to understand them, inevitably introduce a certain level of noise that distorts the original true information. On the other hand, the interaction with the world deals with the will of the individual to perform actions to change the environment around it, but this apparently simple operation involves complex biologic mechanisms to coordinate the body according to the will and difficulties in the perception of the consequences of the interaction. Moreover, the concept of goal can be unclear and the individual may

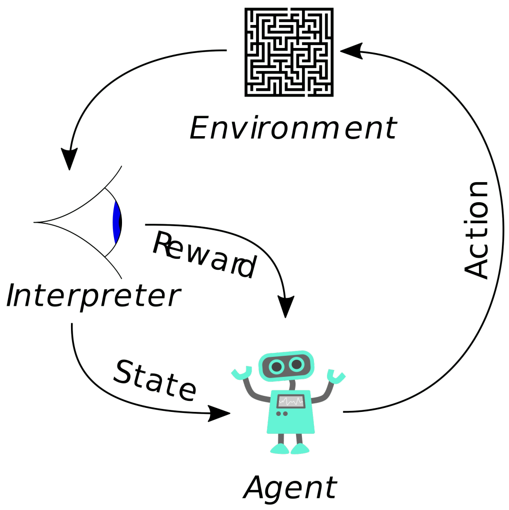

Figure 1.1: *The scheme of a RL model.*

result in performing actions without being sure of what it wants. It is arguable that discussing about the concept of true information and the concept of will requires strong theoretical considerations since they are both hardly definable concepts. For many centuries scientists and philosophers debated about these topics, in particular trying to solve complex problems like the real nature of perceivable things and the concept of free will. However, to make the discussion of these concepts suitable for our purposes throughout all this thesis, we lighten the definition of them to the one provided by common sense.

1.2 Learn how to act with Reinforcement Learning

Reinforcement Learning (RL) [11] is a subfield of Machine Learning (ML) which aims to realize autonomous *agents* able to learn how to act in a certain *environment* in order to maximize an objective function; this is achieved providing the agent with the perception of its *state* in the environment and making it learn the appropriate *action* to perform, where an action can be seen as an atomic operation that brings the agent from a state to another one. The objective function represents a measure of how well the agent is accomplishing its task in the environment and it is usually formalized by a discounted sum of *rewards* obtained after each action (Figure 1.1). The sum is discounted to give more importance to the most recent rewards w.r.t. the ones further in the future. The reward function, i.e. the function returning the reward after each action, is not a concrete part of the environment, but it is often designed by a human which decides whether a certain behavior has to be reinforced (returning a positive reward) or inhibited (returning a negative reward).

1.2.1 Uncertainty in Reinforcement Learning

The major challenge of RL is represented by the uncertainty. In fact, initially, the agent is not provided with the knowledge of how the environment will react to its action, thus it does not know whether an action would be good or not to maximize its objective function. In other words, before trying an action, it does not know if that action will get a positive or a negative reward, and it does not know if that action will let it go to the desired state or not. Thus, the former problem can be seen as uncertainty in the reward function and the latter as uncertainty in the transition (i.e. model) function. In some cases, also the uncertainty in the perception of the current state of the agent is considered, making the problem more complex.

The uncertainty issue results in the need of the agent to try actions in order to improve its knowledge of the environment. This process delays the collection of high rewards, but helps the agent to reduce its uncertainty. However, since the objective function is a sum of discounted rewards where later rewards worth less than recent ones, the agent also needs to learn fast in order to learn to perform the most rewarding actions as soon as possible. The need to explore and the need to *exploit* the actions believed to be good introduces an important problem known as *exploration-exploitation dilemma*.

1.2.2 Balancing exploration and exploitation

The exploration-exploitation dilemma has been broadly studied in the field of Multi-Armed Bandit (MAB), a particular case of the RL problem with a single state [6]. In this problem the goal is to find the sequence of optimal actions, i.e. the sequence of actions that allows to maximize the return. The simplistic setting of the MAB problem allows to theoretically study the balancing of exploratory and exploitative actions, for instance to derive upper confidence bounds on the *regret*, i.e. a measure of the return lost in performing non-optimal actions [1,5,14], and several algorithms to address this problem have been proposed such as UCB1 [2] and Thompson sampling [12].

The RL setting complicates the MAB problem because of the presence of multiple states. This makes the exploration-exploitation dilemma less tractable in terms of complexity and computational feasibility. Indeed, the quality of the actions must now be evaluated for each state, contrarily to the MAB case where the presence of a single state simplifies the problem. This issue is what makes RL so challenging and has been addressed for decades in the literature.

1.3 My research

The strong connection between uncertainty and the exploration-exploitation dilemma is highlighted by the previous considerations and it is intuitive how the effectiveness of a RL algorithm depends on its ability of reducing the uncertainty of the agent in a computationally and data-efficient way. The RL literature contains lots of algorithms and methodologies proposed to make the agent learn a good policy aiming at efficiency; however, despite addressing the reduction of uncertainty via experience, only few of them explicitly exploit uncertainty to learn.

During my Ph.D., I studied ways to develop algorithms that exploit uncertainty since the explicit consideration of uncertainty has been shown to be often helpful in order to improve the performance and efficiency of learning. One of the most common technique to explore is known as ε-greedy and consists in performing, at each state, a random action with probability ε and the action considered to be the best one with probability $1-\varepsilon$. This exploratory policy does not consider the uncertainty of the agent and simply randomly moves it with the drawback of requiring a huge amount of experience to learn effective policies. This is shown especially in recent works on the field of Deep Reinforcement Learning (DRL) [7, 13, 15] which studies the application of Deep Learning (DL) models and methodologies to exploit their strong ability to generalize with the purpose to solve highly complex problems that were unfeasible before. Research on DRL, brought to the realization of groundbreaking works where authors have been able to reach the state-of-the-art in extremely complex games such as Go [8, 10] and chess [9].

The extraordinariness of these results is comparable to the amount of experience required by these algorithms to work. For instance, in [7] the experiments are performed using 50M samples corresponding to three days of computation and weeks of human play. This work does not address the problem of data-efficiency aiming more to other goals (e.g. maximizing the cumulative reward) and for this reason the previously described exploration policy of ε-greedy is used. However, data-efficiency can be pursued exploiting uncertainty in order to balance the knowledge of the agent of already known states and unknown ones, for instance letting it explore unknown states with higher probability. Moreover, the exploitation of uncertainty can also help to improve other aspect involved in learning algorithms which will explained more in details during the thesis.

My Ph.D. research brought to the publication of four conference papers, most of them focused on the previously described topic. I also developed, together with a colleague of mine, a RL Python library which had the initial purpose to facilitate my research, but which has become larger and larger allowing now to do RL research for general purposes. Other works are still ongoing and others have not been accepted for publications, still I think they worth to be mentioned in this thesis anyway. The whole document is composed of seven chapters, with this introduction being the first of them:

- **second** chapter resumes the main concepts of RL starting from the fundamental theory behind it and then giving a description of several methodologies related to this thesis. This chapter has the purpose to provide a general, but useful, overview of what is necessary to understand the following chapters;
- **third** chapter describes three publications I made about ways to exploit uncertainty in the context of Value-Based RL and more in particular in the famous algorithm of Q-Learning;
- **fourth** chapter deals with the description of novel algorithms that exploit uncertainty in order to improve exploration;
- **fifth** chapter extends the previous work to the DRL framework;
- **sixth** chapter provides a description of the RL Python library I developed;
- **seventh** chapter concludes the thesis resuming the previous chapters and providing my considerations about the research I made and the one I think will be interesting to pursue in the following years, by me or someone else!

CHAPTER 2

Preliminaries

RL is intuitively explainable as the process of learning from interaction with the environment, but this hasty explanation is only a very high level description of it; indeed, a more formal way to model the problem is required to properly analyze it. This chapter provides a description of the mathematical framework required to model RL. It also explains a selection of algorithms that are related to the work done in this thesis in order to provide enough knowledge about the literature I dealt with during my years of Ph.D. research.

2.1 Agent and environment

The interaction of an agent inside an environment can be seen as the execution of actions to move itself in the environment and observing the consequences of its actions (Figure 2.1). The temporal progress of the interaction is modeled in a set of discrete time steps $t \in [0, 1, 2, \dots]$ where the agent sees a representation S_t of the environment,

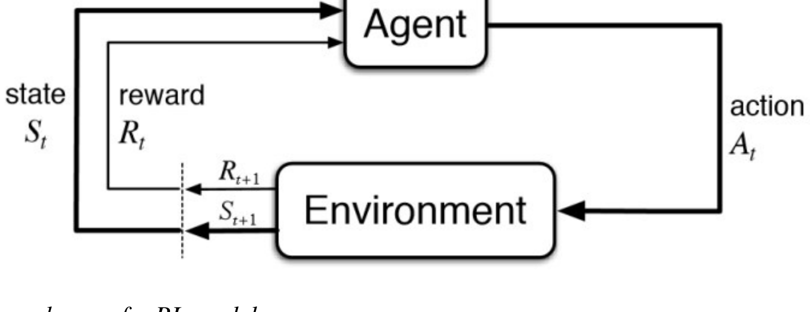

Figure 2.1: *The scheme of a RL model.*

executes an action A_t and observes the new representation of the environment S_{t+1}. The problems about observation and interaction discussed in Chapter 1 are simplified by an explicit selection of data to observe from the environment and of the executable actions. In this way, only the relevant aspect of the sensory data acquired from the environment the agent are used. Together with S_{t+1}, the agent also sees a return R_t which is not given by the environment, but it is a measure considered by the agent to evaluate the convenience of the consequences of the actions it executes. The total number of time steps is called *horizon* H and determines a first taxonomy of problems:

- finite time horizon: $t_i, \forall i \in [0, 1, 2, \dots, H)$;
- infinite time horizon: $t_i, \forall i \in [0, 1, 2, \dots, \infty)$.

Some problems can terminate before reaching the horizon, which happens when the agent reaches special situations called *absorbing* states. These states are usually desirable or catastrophic states when the interaction of the agent with the environment is no more useful or impossible. The set of steps between the start of the interaction to the end is called *episode*.

The interaction of the agent with the environment is performed with the purpose to reach a goal for which the agent has been designed. The way to give the knowledge of the goal to the agent is to provide it with a measure of the quality of its behavior. This measure is called *reward* and is a function usually returning a real scalar value given the observation of the current state of the agent. The goal of the agent is to maximize a measure related to the collected rewards. In an infinite time horizon problem it can be:

- cumulative reward:
$$J = \sum_{t=0}^{\infty} r_t; \tag{2.1}$$
- average reward:
$$J = \lim_{n \to \infty} \frac{\sum_{t=0}^{n} r_t}{n}; \tag{2.2}$$
- discounted cumulative reward:
$$J = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{2.3}$$

The measure in Equation 2.3 uses a real scalar $\gamma \in (0, 1]$, called *discount factor*, which has the purpose to give different importance to rewards w.r.t. the time step they have been collected. If $\gamma = 1$ the equation reduces to 2.1, whereas the smaller it becomes the less the agent cares about rewards far in time.

2.2 Markov Decision Processes

The mathematical framework to study the interaction of the agent with the environment is provided by the theory behind Markov Decision Processes (MDPs). A MDP is defined as a 6-tuple where $< \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma, \mu >$:

- $\mathcal{S}$ is the set of states where the agent can be in the environment;

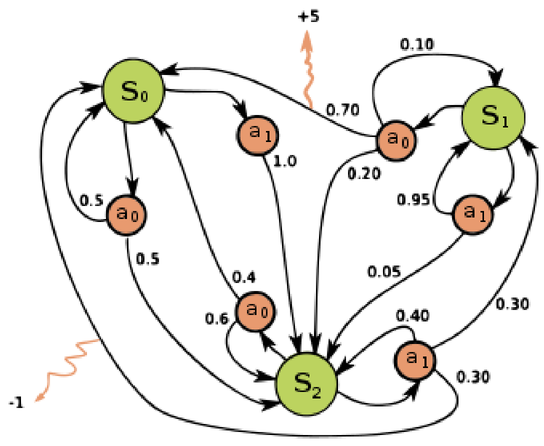

Figure 2.2: ...

- $\mathcal{A}$ is the set of actions that the agent can execute in the environment;
- $\mathcal{R}$ is the set of rewards obtainable by the agent;
- $\mathcal{T}$ is the *transition function* consisting in the probability of reaching a state s' executing action a in state s: $\mathcal{T}(s, a) = P(s'|s, a)$;
- γ is the discount factor;
- μ is the probability of each state to be the initial one: $\mu(s) = P(s_0 = s)$.

A MDP is called *finite*, or *discrete*, if the set of states $\mathcal{S}$ and set of actions $\mathcal{A}$ are finite; it is called *infinite*, or *continuous*, when the set of states $\mathcal{S}$ is infinite and/or the set of actions $\mathcal{A}$ is infinite.

2.2.1 Value functions

Recalling that the goal of the agent is to maximize the cumulative (discounted) reward obtained during an episode, a MDP is considered *solved* when the agent learns the actions to perform in each state which maximizes this measure. The function defining the probability of executing action a in a state s is called *policy*: $\pi(s) = P(a|s)$. An *optimal* policy π^* is the one which, when followed, allows the agent to solve the MDP. Considering the stochasticity in the transition function $\mathcal{T}$ and in the policy π, the expected value of the cumulative discounted reward obtainable following π is called *state value function*:

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k} | S_t = s], \forall s \in \mathcal{S}. \tag{2.4}$$

Then, an optimal policy can be defined also as the one which maximizes the value function of each state:

$$V^*(s) = \max_\pi V_\pi(s), \forall s \in \mathcal{S}. \tag{2.5}$$

Together with the state value function, the *action value function* is defined as:

$$Q_\pi(s,a) = \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k} | S_t = s, A_t = a], \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.6}$$

And subsequently the optimal policy maximizes also the action value function of each state-action tuple:

$$Q^*(s,a) = \max_\pi Q_\pi(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.7}$$

2.3 Solving a MDP

Value functions are the main concept used by several algorithms to address the problem of solving MDPs. In the following, a description of algorithms exploiting value functions to solve MDPs is provided, from the easiest case to the hardest ones.

2.3.1 Dynamic Programming

When the transition function $\mathcal{T}$ and reward function $\mathcal{R}$ of a MDP are known, the full model of the environment is available. This is not the case in many real world problems where an agent does not know where the action would bring it and which return would obtain, but constitutes an interesting scenario to start studying the problem of solving a MDP. The theory behind the solving MDPs with full model available is known under the name of Dynamic Programming (DP) [3,4]. The main concept in the research on DP is the optimal Bellman equation, defined as

$$\begin{aligned} V^*(s) &= \max_a \mathbb{E}[R_t + \gamma V^*(s') | S_t = s, A_t = a, S_{t+1} = s'] \\ &= \max_a \sum_{s'} p(s'|s,a)[r + \gamma V^*(s')] \end{aligned} \tag{2.8}$$

for state value function, and

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}[R_t + \gamma \max_{a'} Q^*(s',a') | S_t = s, A_t = a, S_{t+1} = s'] \\ &= \sum_{s'} p(s'|s,a)[r + \gamma \max_{a'} Q^*(s',a')] \end{aligned} \tag{2.9}$$

for action value function, for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. The optimal Bellman equation serves as a way to derive the optimal policy, but requires the optimal value functions to be known. Usually the optimal value functions are unknown and in order to learn them several algorithms change the Bellman equation in form of an assignment repeated iteratively.

Policy Iteration

The iterative application of the Bellman equation when following a policy π is called *iterative policy evaluation* (Algorithm 1) since it allows to compute the value functions of states and actions w.r.t. the policy π:

$$\begin{aligned} V_{t+1}(s) &= \mathbb{E}_\pi[R_t + \gamma V_t(S_{t+1}) | S_t = s] \\ &= \sum_a \pi(a|s) \sum_s P(s'|s,a)[r + \gamma V_t(s')] \end{aligned} \tag{2.10}$$

Algorithm 1 Iterative Policy Evaluation

```
1: Inputs: policy $\pi$ to evaluate, a small threshold $\theta$ determining the accuracy of the estimate
2: Initialize: $V(s), \forall s \in \mathcal{S}$ arbitrarily, $V(s') = 0$ for all terminal states $s'$
3: repeat
4:    $\Delta \leftarrow 0$
5:    for all $s \in \mathcal{S}$ do
6:       $v \leftarrow V(s)$
7:       $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r + \gamma V(s')]$
8:       $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
9:    end for
10: until $\Delta < \theta$
```

Algorithm 2 Iterative Policy Evaluation

```
1: Initialize: $\pi(s) \in \mathcal{A}$ arbitrarily for all $s \in \mathcal{S}$
2: repeat
3:    Iteration policy evaluation
4:    Policy improvement:
5:    $policy\_stable \leftarrow true$
6:    for all $s \in \mathcal{S}$ do
7:       $old\_a \leftarrow \pi(s)$
8:       $\pi(s) \leftarrow arg\max_a \sum_{s'} P(s'|s,a)[r + \gamma V(s')]$
9:       If $old\_a \neq \pi(s)$, then $policy\_stable \leftarrow false$
10:   end for
11: until policy-stable
```

for all $s \in \mathcal{S}$. It can be shown that the iterative application of the Bellman equation always converges to a single fixed point V_π.

Once the value functions have converged, it is interesting to see if the current policy can be improved in order to make it closer to the optimal one or not. One way to do this consists in considering a state s and an action $a \neq \pi(s)$ and computing

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}[R_t + \gamma V_\pi(S_{t+1})|S_t = s, A_t = a] \\ &= \sum_{s'} P(s'|s,a)[r + \gamma V_\pi(s')]. \end{aligned} \tag{2.11}$$

Whenever $Q_\pi(s,a) > Q_\pi(s,\pi(s))$ it is convenient to update the policy such as $\pi(s) = a$. This procedure is called *policy improvement*. The process of alternating steps of iterative policy evaluation and policy improvement brings to the estimation of the optimal value functions and is resumed in an algorithm called *Policy Iteration* (Algorithm 2).

Value Iteration

The alternation of policy evaluation and policy improvement is a drawback of Policy Iteration which may slowdown the learning. Among other algorithms, the algorithm of *Value Iteration* addresses this problem stopping policy evaluation after only one update of each state value function. The update is different from the one in policy evaluation since it combines the policy evaluation steps and the policy improvement:

$$\begin{aligned} V_{t+1}(s) &= \max_a \mathbb{E}[R_t + \gamma V_t(S_{t+1}) | S_t = s] \\ &= \max_a \sum_s P(s'|s,a)[r + \gamma V_t(s')] \end{aligned} \tag{2.12}$$

for all $s \in \mathcal{S}$. Desirably, Value Iteration maintains the properties of Policy Iteration about convergence to the fixed point corresponding to the optimal value functions.

As stated at the beginning of the section, the previous methods can be applied only when the full model of the MDP is known. However, in most of real cases the full model is not available and the agent must move in the environment in order to understand it.

2.3.2 Reinforcement Learning

Online

Batch

Deep Reinforcement Learning

CHAPTER 3

Maximum Expected Value estimate

CHAPTER 4

Exploration

CHAPTER 5

Deep

CHAPTER 6

Mushroom

CHAPTER 7

Conclusion

Bibliography

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] Richard Bellman. *Dynamic programming*. Courier Corporation, 2013.

[4] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

[5] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[6] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[7] 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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[8] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[9] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[10] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[11] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[12] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[13] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[14] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer, 2005.

[15] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.