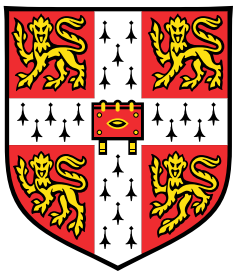
Probabilistic Bellman Consistency in Reinforcement Learning



Luca Biggio

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Robinson College August 2019

Declaration

I, Luca Biggio of Robinson College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

This dissertation contains 14990 words excluding bibliography, photographs and diagrams but including tables, captions, footnotes, appendices and abstract.

We used the following external code for our experiments in Chapter 4:

i) DeepRL-Tutorials as starter implementations for the C51 and DQN algorithms.

ii) The open-source code provided by Clements et al. [2019] for the QR-DQN algorithm implementation.

Luca Biggio
August 2019

Acknowledgements

First, I would sincerely like to thank my supervisor, Prof. Carl Rasmussen, for his precious guidance throughout the various steps of the project. I am very thankful for all the inspiring discussions we had and for his continued encouragement and support.

I would also like to acknowledge all the components of the MLMI staff for the amazing job they did during the academic year. Being part of this course has been one of the best experiences in my life.

Then, a special thank to Riccardo and Chiara who were always there ready to support me in the toughest moments of the course. I will never forget all the moments spent together, without your friendship my experience here would not have been the same.

I would like to thank my brother, Matteo, for being a constant source of inspiration as well as a true friend for me.
I would also like to thank my girlfriend, Camilla, for putting up with me since the good old days of the high school to the years of University. Thank you, for all these happy years together, I feel so lucky to have met you.

Finally, I would like to dedicate this thesis to my loving parents. Without all the sacrifices you have made for me and your constant support I would have never been able to go this far.

Abstract

The main purpose of this work is to provide a thorough analysis of the most recent advances in the quickly evolving field of Distributional Reinforcement Learning (DRL) along with identifying potential fruitful research directions to pave the way to future investigations. The idea of a distributional approach is not new and finds numerous references in the Reinforcement Learning (RL) literature. However, previous attempts were often aimed at tackling very specific problems, such as modelling parametric uncertainty or designing risk-sensitive algorithms.

In contrast, the new approach described in this work is motivated by the ambitious objective of making the RL theoretical framework more principled and general. The achievement of this goal is expected to result in a deeper comprehension of existing RL techniques and in the design of new algorithms capable of outperforming the "classical" ones.

The experimental results obtained by the first fully DRL algorithms have indeed demonstrated that a distributional perspective yields surprising empirical improvements. However, theoretical justifications of the experimental behaviour of these algorithms are still lacking or incomplete. One of the main goals of the present work is to provide a consistent review of the most recent achievements in this field both from the theoretical and algorithmic points of view. A critical evaluation of these results is carried out with particular attention to highlighting the most promising methods for future applications.

The distributional approach described in this project combines elements of "classical" RL and Deep Learning (DL) with concepts of Measure Theory and Advanced Statistics, making a full comprehension of its principles challenging and exciting at the same time. In fact, we will see that part of the success of this approach is due to its multidisciplinary interactions, in particular with DL.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ALE *Arcade Learning Environment*

CNN *Convolutional Neural Network*

D4PG *Distributed Distributional Deterministic Policy Gradient*

DDPG *Deep Deterministic Policy Gradient*

DDQN *Double DQN*

DL *Deep Learning*

DP *Dynamic Programming*

DQN *Deep Q-Network*

DRL *Distributional Reinforcement Learning*

ER-DQN *Expectile-Regression DQN*

GAN *Generative Adversarial Network*

GP *Gaussian Process*

IDS *Information-Directed Sampling*

KDE *Kernel Density Estimation*

KL divergence *Kullback-Leibler divergence*

LSTM *Long Short-Term Memory*

MC *Monte Carlo*

MDP *Markov Decision Process*

ML *Machine Learning*

MuJoCo *Multi-Joint dynamics with Contact*

PILCO *Probabilistic Inference for Learning COntrol*

PPO *Proximal Policy Optimisation*

QR-DQN *Quantile-Regression-DQN*

RL *Reinforcement Learning*

RNN *Recurrent Neural Network*

SGD *Stochastic Gradient Descent*

SimPLe *Simulated Policy Learning*

SL *Supervised Learning*

TD Learning *Temporal Difference Learning*

TRPO *Trust Region Policy Optimization*

UL *Unsupervised Learning*

VI *Variational Inference*

Chapter 1

Reinforcement Learning Background

Reinforcement Learning (RL) is one of the three paradigms that constitute Machine Learning (ML), along with Supervised Learning (SL) and Unsupervised Learning (UL) (Murphy [2012]). The central problem RL deals with consists of describing how an agent ought to take actions in an environment in order to maximise a particular notion of cumulative reward (Francois-Lavet et al. [2018]).

In this introductory chapter, we review some of the fundamental principles of RL and we delineate some of the key ideas explored in this project.

In particular, we introduce the Bellman's equation that is a crucial component of most RL algorithms. Its extension represents the main motivation of a distributional approach to RL, which is the focus of this work, and that is presented in details in Chapter 3.

Furthermore, we briefly review SARSA and Q-Learning, two popular Temporal Difference (TD) learning methods that are then revisited under a Deep Learning (DL) perspective in Chapter 2.

1.1 General Framework: Markov Decision Processes

The typical RL problem can be formalised as an agent that interacts with an environment as illustrated in Fig. 1.1. At each time-step, the agent performs an action. This produces two consequences: (i) the environment changes its internal state and (ii) the agent receives a reward. This agent-environment interaction can be modelled as a Markov Decision Process (MDP) $(\mathscr{X}, \mathscr{A}, R, P, \gamma)$, where:

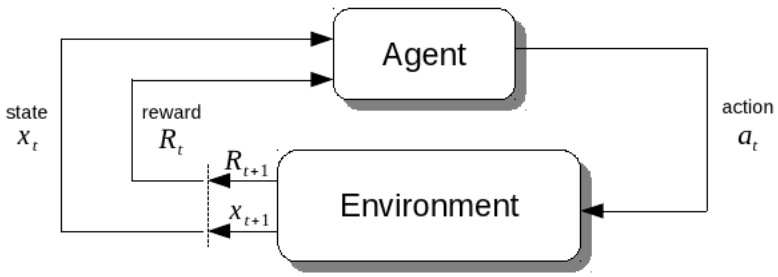


Fig. 1.1 Agent-environment interaction under the MDP assumption, (Sutton and Barto [1998]).

- $\mathscr{X}$ is the state space,
- $\mathscr{A}$ is the action space,
- R is a random variable describing the reward corresponding with a state action pair (x,a), where $x \in \mathscr{X}$ and $a \in \mathscr{A}$.
- $P : \mathscr{X} \times \mathscr{X} \times \mathscr{A} \to [0,1]$ is the *state-transition probability* $P(\cdot|x,a)$,
- $\gamma \in [0,1]$ is a discount factor.

The last ingredient needed to complete the RL basic theoretical formulation is given by the concept of *policy*. A policy $\pi : \mathscr{X} \times \mathscr{A} \to [0,1]$ describes the probability of selecting a particular action $a \in \mathscr{A}$ starting from a state $x \in \mathscr{X}$[1].

RL algorithms aim to find a policy that allows the agent to maximise the rewards it accumulates throughout its interactions with the environment. We clarify this point in the following section.

1.1.1 Bellman's Equations

By iterating the MDP described in Fig. 1.1, the agent accumulates a sequence of rewards that depend on the states and actions it visited along its trajectory of interactions with the environment.

The *return* Z^π is a random variable defined as the discounted sum of the rewards obtained by taking action $a \in \mathscr{A}$ from state $x \in \mathscr{X}$ at time-step $t = 0$ and acting according to a policy π

[1] A *deterministic policy* is a particular type of policy that deterministically maps states into actions.

thereafter:

$$\begin{aligned} Z^{\pi}(x,a) &:= \sum_{t=0}^{\infty} \gamma^t R(x_t, a_t) \\ x_t &\sim P(\cdot|x_{t-1}, a_{t-1}), a_t \sim \pi(\cdot|x_t), x_0 = x, a_0 = a \end{aligned} \tag{1.1}$$

A crucial quantity in RL is the *value function* Q^{π} of a policy π which is defined as the expected value of the return:

$$Q^{\pi}(x,a) := \mathbb{E}Z^{\pi}(x,a) \tag{1.2}$$

Simple algebraic manipulations on Eq. 1.2 lead to a fundamental result in RL and DP, the Bellman's equation (Bellman [1954]):

$$Q^{\pi}(x,a) = \mathbb{E}R(x,a) + \gamma \mathop{\mathbb{E}}_{P,\pi} Q^{\pi}(x',a') \tag{1.3}$$

This equation describes how expected returns at each state-action pair $(x,a) \in \mathscr{X} \times \mathscr{A}$ are related to the expected returns at possible next state-action pairs $(x',a') \in \mathscr{X} \times \mathscr{A}$ in the MDP.

Typically, most RL algorithms are devised to find a policy π that maximises the expected return, as defined in Eq. 1.2. In other words, the goal is to find a policy such that:

$$Q^{*}(x,a) := \max_{\pi} Q^{\pi}(x,a) \tag{1.4}$$

The most common method to find this policy involves the so-called *Bellman optimality equation*[2]:

$$Q^{*}(x,a) = \mathbb{E}R(x,a) + \gamma \mathbb{E}_{P} \max_{a' \in \mathscr{A}} Q^{*}(x',a') \tag{1.5}$$

Eq. 1.3 and Eq. 1.5 are the fundamental starting points of many popular RL algorithms. However, an interesting observation is that these equations are written in terms of the value function and therefore they do not capture the complexity of the entire return distribution associated with the random variable Z^{π}.

The aim of this project is to explore the emerging field of Distributional Reinforcement Learning (DRL) that is based on the study of the return distribution. As discussed in Chapter 3, the goal of DRL is indeed to study the behaviour of extended versions of Eq. 1.3 and Eq. 1.5 that are written in terms of the return, Z^{π}, instead of its expected value, Q^{π}.

[2]The Bellman optimality equation can be viewed as a special version of the Bellman's equation 1.3 written in terms of the optimal expected return Q^{*}.

1.2 Dynamic Programming

In this section, we discuss how Eq. 1.3 and Eq. 1.5 can be solved by using the methods of Dynamic Programming (DP).

The fundamental idea of DP is to transform Eq. 1.3 and Eq. 1.5 into the following recursive update rules:

$$\begin{aligned} Q_{k+1}(x,a) &= \mathbb{E}R(x,a) + \gamma \mathop{\mathbb{E}}_{P,\pi} Q_k\left(x',a'\right) \\ Q_{k+1}(x,a) &= \mathbb{E}R(x,a) + \gamma \mathbb{E}_P \max_{a' \in \mathbb{A}} Q_k\left(x',a'\right) \end{aligned} \tag{1.6}$$

The goal of these recursive relations is to provide increasingly accurate approximations of the true value functions, Q^π and Q^*, as the number of iterations grows.

An alternative way of looking at Eq. 1.6 is to introduce the so-called *Bellman operator* $\mathscr{T}^\pi$ and *Bellman optimality operator* $\mathscr{T}$:

$$\begin{aligned} \mathscr{T}^\pi Q(x,a) &:= \mathbb{E}R(x,a) + \gamma \mathop{\mathbb{E}}_{P,\pi} Q\left(x',a'\right) \\ \mathscr{T} Q(x,a) &:= \mathbb{E}R(x,a) + \gamma \mathbb{E}_P \max_{a' \in \mathscr{A}} Q\left(x',a'\right) \end{aligned} \tag{1.7}$$

It can be shown (Bertsekas and Tsitsiklis [1996]), that these two operators are *contraction mappings*. If an operator T possesses this special property, for any pair of bounded functions $K, K' : \mathscr{X} \times \mathscr{A} \to \mathbb{R}$, the following condition is fulfilled:

$$\left\|TK - TK'\right\|_\infty \leq \gamma \left\|K - K'\right\|_\infty \tag{1.8}$$

where $0 \leq \gamma < 1$. This property is particularly important because the Banach fixed point theorem ensures that a unique fixed point exists for both the Bellman operators defined in Eq. 1.7 and these fixed points will be Q^π and Q^* respectively.

As discussed in the next section, the Bellman operators play a crucial role in RL since they provide a description of the expected behaviour of popular algorithms such as SARSA and Q-Learning (Bellemare et al. [2017], Bertsekas and Tsitsiklis [1996]). In Chapter 3, we introduce the distributional versions of the operators defined in Eq. 1.7 and we study their convergence properties.

One last important observation is that DP methods, in and of themselves, are not particularly useful for the design of RL algorithms. The reason is that they rely on a complete knowledge of the environmental model which is not typically the case in any realistic scenario. The next section introduces TD learning methods that transform the ideas of DP into practical algorithms.

1.3 Temporal-Difference Methods

TD learning refers to a class of RL methods that enable DP ideas to be exploited without making the strong assumption of a complete knowledge of the environmental model. In particular, these methods rely on learning values from experience trajectories. Specifically, our approximation of the value function is recursively adjusted by means of the following update rule:

$$Q_{k+1}(x,a) \leftarrow Q_k(x,a) + \alpha \left[R(x,a) + \gamma Q_k\left(x',a'\right) - Q_k(x,a)\right] \tag{1.9}$$

where α is a constant step-size called *learning rate*. In other words, this update modifies the Q-function evaluated at a particular state-action pair (x,a) so that it gets closer to the target value, $R(x,a) + \gamma Q_k(x',a')$.

TD methods can be thought of as a combination of Monte Carlo (MC) and DP ideas. Indeed, the expectation of the Q-function w.r.t. the transition probability is replaced by a MC sample (the agent learns from experience) and the current estimate of the Q-function is used in the update rule instead of the true value, Q^π (bootstrapping (Bickel and Freedman [1981])).

Now, we introduce SARSA and Q-Learning, two popular algorithms that implement the concepts of TD learning. The rationale underlying these algorithms, in particular Q-Learning, is key for understanding some successful deep RL methods that are discussed in Chapter 2 and for the distributional algorithms that are the focus of Chapter 3.

SARSA. SARSA (Rummery and Niranjan [1994]) is an on-policy method meaning that decisions are made by following the same policy as the one that is evaluated and improved. It makes use of the update rule defined in Eq. 1.9 and takes its name from the fact that it

considers sample sequences of the form "State, Action, Reward, next State, next Action". The complete algorithm pseudocode is shown below[3]

Algorithm 1 SARSA

Initialise: $Q(x,a), \forall x \in \mathscr{X}, \forall a \in \mathscr{A}$
repeat {for each episode}
  Initialise x
  Select action a from x using ε-greedy policy derived from Q
  repeat {for each step of episode}
    perform action a, observe R, x'
    Select a' from x' using ε-greedy policy derived from Q
    $Q(x,a) \leftarrow Q(x,a) + \alpha [R(x,a) + \gamma Q(x',a') - Q(x,a)]$
    $x \leftarrow x'$; $a \leftarrow a'$
  until x is terminal
until convergence

Q-Learning. Q-learning (Watkins and Dayan [1992]) is an off-policy TD algorithm which is based on the following update rule:

$$Q_{k+1}(x,a) \leftarrow Q_k(x,a) + \alpha \left[R(x,a) + \gamma \max_{a'} Q_k(x',a') - Q_k(x,a) \right] \tag{1.10}$$

Contrarily to on-policy algorithms, off-policy methods introduce two different policies: one for exploring, the *behaviour policy* (used to generate behaviour), and one that is learned, the *target policy*. The complete Q-Learning algorithm is shown below.

[3] With ε-greedy policy we mean that the agent chooses the optimal action with probability $p_{opt} = 1 - \varepsilon$ and a random action with probability $p_{rand} = \varepsilon$ to explore the environment.

Algorithm 2 Q-Learning

Initialise: $Q(x,a), \forall x \in \mathscr{X}, \forall a \in \mathscr{A}$
repeat {for each episode}
  Initialise x
  repeat {for each step of episode}
    Select a from x using ε-greedy policy derived from Q
    perform action a, observe R, x'
    $Q(x,a) \leftarrow Q(x,a) + \alpha [R(x,a) + \gamma \max_{a'} Q(x',a') - Q(x,a)]$
    $x \leftarrow x'$; $a \leftarrow a'$
  until x is terminal
until convergence

The goal of the TD algorithms presented in this section is to produce an estimate of the value function for each separate state-action pair, thus providing a "lookup-table" representation. If the state and action spaces are large, this approach is practically unfeasible for two main reasons. First, it would be impractical due to the large memory requirements necessary to store all possible values. Second, a robust estimate of the Q-function associated with a particular state-action pair would require multiple visits to the same pair which is not necessarily guaranteed, especially in large state-action spaces.

These complications could be addressed if the Q-function evaluated at a particular state-action pair could provide some information about the Q-function at some other state-action pairs. In other words, we would need a representation of the value function that allowed generalisation across state-action pairs. This idea can be realised by introducing a parametrised version of the original Q-function, i.e. $Q(x,a;\theta) \approx Q(x,a)$, and by iteratively estimating its parameters.

A popular choice in the RL community is a linear parametrisation[4], even though nonlinear function approximators such as neural networks have gained significant interest in the last few years thanks to the success of DL in various diverse domains.
The combination of RL and DL gives rise to the field of deep RL that is the subject of Chapter 2.

[4] Indeed, using linear parametrisations makes it easier to obtain convergence proofs.

Chapter 2

Deep Reinforcement Learning

Over the last few years, DL techniques have been widely used in a large variety of fields due to their ability to infer abstract representations of high-dimensional data and improve learning performances.

DL methods have been also successfully applied in the context of RL, especially in problems characterised by high-dimensional state spaces. The class of algorithms resulting from the intersection of DL and RL methods is called Deep Reinforcement Learning.

The most popular deep RL technique is arguably the Deep Q-Network (DQN) agent (Mnih et al. [2015]). The success of this algorithm derives from its capacity of mastering a diverse range of Atari games (Bellemare et al. [2013]), receiving only visual frames and rewards as input. The surprising performances of DQN motivate the development of deep RL algorithms aimed at emulating some human decision-making capabilities.

This chapter provides an overview of some well-established deep RL techniques with particular attention to the algorithms that have been revisited under the DRL perspective and on those that we believe could be extended to the distributional framework.

The chapter is divided into two parts: the first one introduces a number of deep RL techniques in the context of *model-free RL*, i.e. when the agent does not have any knowledge about the environment, whereas the second part describes some recent applications of deep RL algorithms in the *model-based* framework, where the agent learns a dynamic model of the environment.

2.1 Model-Free Deep RL

Deep RL algorithms have been mainly incorporated into model-free frameworks where they have proved on many occasions to be very powerful and versatile tools, reaching state-of-the-art performances in a wide range of domains.

Model-free RL describes a category of algorithms that do not rely on any model of the environment and that train their agents by repeatedly interacting with it.

In this section, we analyse two of the most popular model-free techniques in the context of deep RL, namely the DQN agent and the Deep Deterministic Policy Gradient (DDPG) algorithm.

2.1.1 Deep Q-Network

The application of RL techniques to real-world decision-making problems requires to design algorithms capable of learning successful policies directly from high-dimensional natural stimuli. DQN can be considered as one of the first steps in this direction.

In Mnih et al. [2015], the authors propose to approximate the optimal value function Q^* with a deep Convolutional Neural Network (CNN) whose parameters will be referred to as θ. The choice of CNNs is motivated by their well-known capacity to efficiently process visual imagery (Krizhevsky et al. [2012]; Lee et al. [2009]).

In the following, we first briefly present the model architecture of the DQN agent and secondly we report some details of the training algorithm used to find the approximation of the optimal Q-function.

Model Architecture

The goal of DQN is to model $Q^*(x,a)$ for each $x \in \mathscr{X}$ and $a \in \mathscr{A}$. A computationally cheap method to do this consists in considering the state x (instead of a state-action pair) as the input of the network and as many nodes as the number of actions for the output layer. In this way, the Q-function of each individual action for a given input state is obtained with a single forward pass through the network. The final architecture consists of three convolutional

layers followed by a non-linear fully-connected layer and a linear fully-connected output layer with a node for each action.

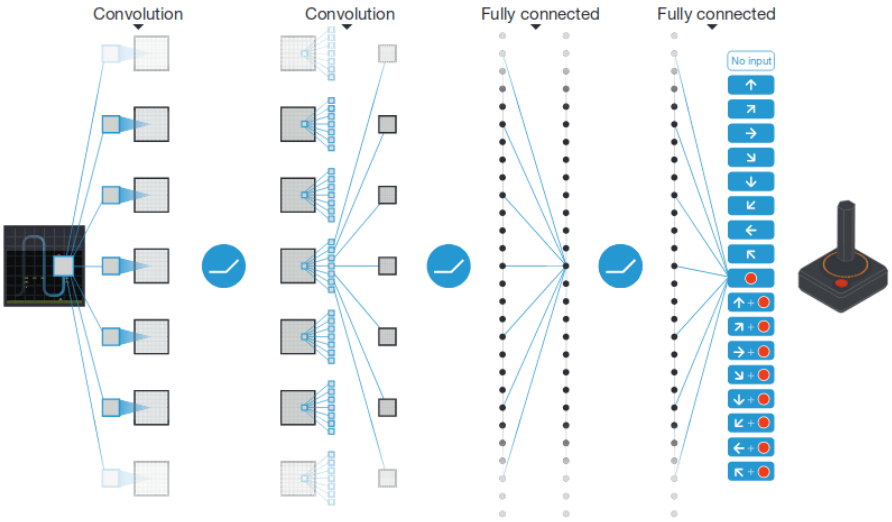


Fig. 2.1 DQN architecture: the network receives a visual frame as input and has a single output for each action available to the agent (Mnih et al. [2015]).

Training Details

Given the estimate of the optimal value function, $Q(x,a;\theta)$, provided by the neural network architecture described above, the DQN algorithm is based on the minimisation of the following sequence of loss functions, $L(\theta_i)$, one for each iteration i:

$$L(\theta_i) = \mathbb{E}_{x,a,R,x'}\left[(y - Q(x,a;\theta_i))^2\right] \tag{2.1}$$

where $y = R(x,a) + \gamma \max_{a'} Q(x',a';\theta_i^-)$ is the approximate target value and θ_i^- is a separate set of parameters from some previous iteration. The gradient of this loss function w.r.t. the network weights is given by:

$$\nabla_{\theta_i} L(\theta_i) = \mathbb{E}_{x,a,R,x'}\left[\left(R(x,a) + \gamma \max_{a'} Q\left(x',a';\theta_i^-\right) - Q(x,a;\theta_i)\right)\nabla_{\theta_i} Q(x,a;\theta_i)\right] \tag{2.2}$$

Since the computation of the full expectation in Eq. 2.2 can be computationally demanding, SGD-based techniques are used to tackle the minimisation problem above. In particular, in Mnih et al. [2015], the RMSProp optimisation technique (Tieleman and Hinton [2012]) is used.

However, using nonlinear models such as neural networks to approximate the Q-function is known to be problematic and often results in unstable or diverging algorithms (Tsitsiklis and Van Roy [1997]). DQN introduces three additional refinements to the above procedure in order to prevent these issues.

1) Experience Replay. At each time-step t, the experience tuple $e_t = (x_t, a_t, R_t, x_{t+1})$ is stored into a set $\mathscr{D}$ with a maximum capacity of N experience tuples. For the optimisation step, random mini-batches of transitions (x, a, R, x') are uniformly sampled from $\mathscr{D}$.
This approach is called *experience replay* and presents two advantages: (i) improved data-efficiency is guaranteed by the fact that each step of experience is potentially used in many weight updates and (ii) the variance of the updates is reduced by breaking the correlations between consecutive samples.

2) Target Network. Every C updates, the network Q is cloned to obtain a target network, $\hat{Q}$, that is used for generating the Q-learning targets y for the following C TD backups.
By applying this method, an update to $Q(x,a)$ does not introduce any change in $\hat{Q}(x',a')$ for all a', and hence in the target y. This method reduces the risk of oscillating or diverging policies.

3) Huber Loss. The classical squared loss is modified as follows: if the value of the error term $R(x,a) + \gamma \max_{a'} Q\left(x', a'; \theta_i^-\right) - Q\left(x, a; \theta_i\right)$ is not included into the interval $(-1,1)$, an absolute value loss is used, otherwise the squared loss is kept. This modification contributes to improving the stability of the algorithm.

Discussion

The performances of DQN on the Atari games suite (Bellemare et al. [2013]) are comparable or superior to those of professional human players. This algorithm effectively demonstrates that a deep RL approach can tackle several challenging decision-making tasks only by processing information extracted from visual inputs.

Furthermore, the DQN agent, as presented by Mnih et al. [2015], leaves room to a number of improvements that have been gradually incorporated into the original algorithm over the last few years. Some of these refinements, such as Double-DQN (Hasselt et al. [2016]) or Prioritised Experience Replay (Schaul et al. [2016]) to name a few, have led to further

performance improvements on the Atari games.

However, DQN presents also a number of shortcomings that limit its applicability to real-world tasks. Indeed, DNQ is designed for RL problems characterised by discrete action spaces, whereas in most concrete applications it is necessary to deal with a continuous range of possible actions. Furthermore, DQN is an entirely model-free algorithm and for this reason, it is prone to sample-inefficiency.

The DDPG algorithm effectively addresses the first problem, while a discussion of how to cope with sample-inefficiency is presented later in the chapter.

2.1.2 Deep Deterministic Policy Gradient

As mentioned in the last section, despite its great success, DQN can not handle problems with continuous or high-dimensional action spaces. When the action space is discrete, the policy can be iteratively updated in the usual way as follows:

$$\pi_{k+1}(x) = arg\max_{a \in \mathscr{A}} Q^{\pi_k}(x,a) \tag{2.3}$$

However, in the continuous case, the greedy update in the equation above would require a global maximisation at every time-step; this approach would be too slow and practically inapplicable.

In Lillicrap et al. [2015], a combination between an actor-critic approach (Sutton and Barto [1998]) and the usage of neural networks as function approximators is proposed to extend the success of DQN to continuous action spaces.
Specifically, the actor function, $\mu(x|\theta^\mu)$, represents the current deterministic policy, parametrised by the weights θ^μ of a neural network. The key idea is to define a performance measure J whose gradient allows us to iteratively update the parameters θ^μ of the policy μ:

$$\theta^\mu \leftarrow \theta^\mu + \alpha \nabla_{\theta^\mu} J(\theta^\mu) \tag{2.4}$$

By choosing J as follows:

$$J(\theta^\mu) = \mathbb{E}[Q^\mu(x, \mu(x|\theta^\mu))] \tag{2.5}$$

it can be shown (Silver et al. [2014]) that the gradient in Eq. 2.4 can be written as:

$$\nabla_{\theta^\mu} J(\theta^\mu) \approx \mathbb{E}_{\rho^\beta}\left[\nabla_{\theta^\mu}\left(\mu(x|\theta^\mu)\right) \nabla_a \left(Q^\mu(x,a)\right)|_{a=\mu(x|\theta^\mu)}\right] \tag{2.6}$$

where $\rho^{\beta} = \sum_{t=0}^{\infty} \gamma^t \Pr\{x_t = x | x_0, \beta\}$ is the state-visitation distribution for a given behaviour policy β. As shown in Eq. 2.6, the actor parameter update relies on the knowledge of the gradient of the Q-function. This motivates the introduction of a critic function, $Q(x,a|\theta^Q)$, parametrised by the set of weights θ^Q of an additional neural network, that provides an approximation of the value function for the current policy. The critic is updated by minimising the following loss function:

$$L(\theta^Q) = \mathbb{E}_{\rho^{\beta}}\left[\left(Q(x,a|\theta^Q) - y\right)^2\right] \tag{2.7}$$

where $y = R(x,a) + \gamma Q(x', \mu(x'|\hat{\theta}^{\mu})|\hat{\theta}^Q)$ and $(\hat{\theta}^{\mu}, \hat{\theta}^Q)$ are the parameters of the target networks associated with the policy and the critic respectively. These separate target networks, along with experience replay, are used in order to stabilise learning as in Mnih et al. [2015]. Additionally, *batch normalization* (Ioffe and Szegedy [2015]) is used to scale the input features so that their range of variation does not change significantly when different units or environments are considered. The resulting algorithm is called Deep Deterministic Policy Gradient (DDPG).

The performances of DDPG have been evaluated on a number of increasingly complex tasks simulated using MuJoCo (Todorov et al. [2012]). The algorithm provides very good results in most of these environments both in the case of low-dimensional state spaces (mechanical features of the environment, e.g. angles, coordinates, etc...) and when raw image frames are used.

DDPG effectively enlarges the range of application of deep RL techniques to continuous action spaces, paving the way to potential real-world applications, such as robotics or finance.

2.1.3 Discussion

The methods described above represent two milestones in the rapidly growing field of deep RL. Several refinements to these basic algorithms have been proposed over the last few years, leading to improved performances both in continuous and discrete action spaces (Fortunato et al. [2017]; Hasselt et al. [2016]; Schaul et al. [2016]; Wang et al. [2016]).

However, despite their excellent performances on relatively complex tasks, these algorithms are still significantly far from replicating the way humans learn and take decisions. Indeed, their main shortcoming is that the number of iterations they require to converge is typically very high. For instance, they need, on average, around tens of millions of time-steps to

learn how to play Atari games, corresponding to weeks of training in real time (Kaiser et al. [2019]). On the other hand, humans can reach very high performances on the same tasks with substantially fewer amounts of interactions with the environment (Tsividis et al. [2017]).

Thus, it is clear that a strong effort needs to be made in order to increase the sample-efficiency of deep RL algorithms. The next section discusses how model-based techniques can be used to cope with sample-inefficiency and introduces some of these approaches in the context of deep RL.

2.2 Model-Based Deep RL

Model-based algorithms differ from model-free techniques since they learn a dynamic model of the environment. This approach can drastically improve sample-efficiency, since the agent updates its approximation of the policy or of the Q-function by means of experience samples generated by the learned dynamic model, without directly interacting with the environment.

However, these approaches suffer from the so-called *model-bias*. The agent tends to be overly confident about the learned dynamic model that may be inaccurate due to limited prior knowledge about the environment (Deisenroth and Rasmussen [2011]). To address this issue, model-based approaches are often combined with model-free techniques (Nagabandi et al. [2018]).

In the context of deep RL, the combination of model-based and model-free techniques is a very active research area. The final goal would be to design algorithms that can reproduce the excellent performances of model-free techniques and be sample-efficient at the same time.

In the following, we introduce two of these hybrid approaches in the context of deep RL. These methods are among the most recent attempts at combining deep RL with model-based approaches. The first one is a straightforward extension of DQN that is further investigated in Chapter 4. The second one is arguably one of the most promising works in model-based RL and its application to the Atari games provides state-of-the-art results in terms of the trade-off between sample-efficiency and learning performances.

2.2.1 DQN with Model-Based Exploration

A possible strategy to improve sample-efficiency of model-free algorithms is to make use of better exploration techniques. In principle, having access to a more informative picture of the environment could allow the agent to design good policies more rapidly by better exploiting the amount of experience gained so far.

The simple idea proposed by Gou and Liu [2019] is based on this rationale. The DQN-agent discussed in Section 2.1.1 explores the environment by selecting a random action with probability ε . The method proposed in Gou and Liu [2019] consists of a slight modification to the DQN-agent that preserves the basic structure of the algorithm but instead uses a model-based exploration strategy.

Here again the agent acts greedily with respect to the current policy with probability $1-\varepsilon$ and explores with probability ε. However, the exploration is carried out differently in this case. The key step of the proposed approach is to extract the last M states visited by the agent, $x_{t-M+1},\ldots,x_t$, from the replay buffer and compute their mean $\boldsymbol{\mu}_M$ and covariance $\boldsymbol{\Sigma}_M$. Then, the action at time-step t is determined by the following equation:

$$a_t = \arg\min_a \mathcal{N}(D(x_t,a,\theta_D)|\boldsymbol{\mu}_M,\boldsymbol{\Sigma}_M), \tag{2.8}$$

where D is a deterministic transition function parametrised by a neural network with weights θ_D. The model-based component of this algorithm consists in modelling this transition function as accurately as possible. This additional neural network is trained in parallel to the DQN network by using the same experience tuples sampled from the replay buffer.

Intuitively, Eq. 2.8 selects the action leading to the state that is most distant from the last M visited states. This explicitly encourages the agent to explore new regions of the environment that it did not consider before then.

The key assumption of this method is that the distribution of the states visited by the agent is well approximated by a Gaussian distribution. The results of the application of this algorithm to some simple environments in the OpenAI gym (Brockman et al. [2016]) platform show that, when the Gaussian assumption is reasonably satisfied, exploration is effectively enhanced with a consequent improvement in terms of sample-efficiency. On the other hand, when the state distribution is more complicated the algorithm fails in its purpose.

This method proposes a very intuitive idea to enhance exploration and to improve sample-efficiency and is based on a little modification to the DQN agent. However, it relies on the Gaussian assumption on the state distribution and on a high level of accuracy in modelling the transition function D. In Chapter 4, we investigate a simple modification to this method that replaces the aforementioned Gaussian assumption.

2.2.2 Simulated Policy Learning

In this section, we briefly describe Simulated Policy Learning (SimPLe) (Kaiser et al. [2019]), one of the most recent and convincing model-based methods aimed at addressing the problem of sample-efficiency.

The basic idea of SimPLe is intimately related to the Dyna algorithm (Sutton [1991]). As shown in Fig. 2.2, SimPLe implements a three-step learning process: i) the agent first interacts with the environment and collects real-world experience samples; ii) these are then used to train its internal world-model of the environment; iii) this world-model is used to simulate artificial experience samples that allow a model-free algorithm to improve the current policy.

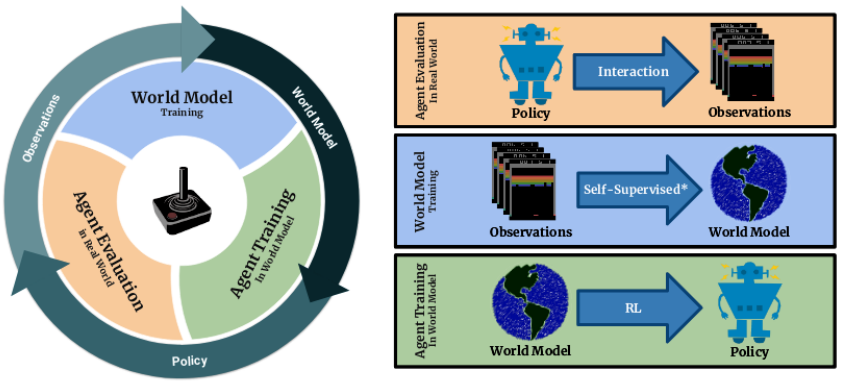


Fig. 2.2 The three-step procedure implemented by the SimPLe algorithm (Kaiser et al. [2019]).

The trickiest part of model-based algorithms is to learn an accurate model of the environment in order to generate consistent experience samples and make planning possible. SimPLe is based on a quite complex neural network model aimed at providing reliable predictions of future video frames of the Atari games. This model is constituted by two elements: a convolutional encoder-decoder module and an inference network and is inspired by ideas

from Babaeizadeh et al. [2017]; Łukasz Kaiser and Bengio [2018][1].

The model-free method responsible for the policy optimisation step of SimPLe is the Proximal Policy Optimisation (PPO) algorithm (Schulman et al. [2017]). PPO is a successful policy-gradient method that takes advantage of the implementation simplicity of vanilla policy-gradient methods (Mnih et al. [2016]) and the reliability of trust region techniques (Schulman et al. [2015]). As mentioned before, PPO is used to train the agent by means of the experience samples drawn from the simulated environment.

The results of the experiments show that SimPLe is effectively able to drastically improve sample-efficiency. In particular, in Kaiser et al. [2019], the performances of SimPLe are compared to those of Rainbow[2] (Hessel et al. [2017]), the state-of-the-art Q-learning algorithm for the Atari games. The experiments show that Rainbow needs many more steps to reach the same score reached by SimPLe after 100K iterations. For some games, the number of time-steps required by Rainbow is almost one order of magnitude larger than that needed by SimPLe.

However, the *final* scores obtained by SimPLe are significantly lower than those reached by Rainbow. Relatively poor final performances represent the major drawback of model-based approaches. Overcoming this limitation is arguably a key research direction for future work. A possible strategy to tackle this problem could be to use alternative model-free approaches than PPO.

We come back to the model-based approach described in this section at the end of Chapter 4, where we discuss some potentially fruitful research directions in the context of DRL, which is the subject of the next chapter.

[1] For more details about the neural network design choices, we refer the interested reader to the original paper (Kaiser et al. [2019]) or to the official paper's project website.

[2] Please refer to Chapter 3 for a brief description of the Rainbow agent.

Chapter 3

Distributional Reinforcement Learning

3.1 Motivation

As we saw in the previous chapters, the Bellman's equation is the fundamental starting point for most RL algorithms. This equation is written in terms of the *expected return* which therefore represents a key element for guiding the agent in its interactions with the environment.

However, the return, as defined in Eq. 1.1, can be a very complex quantity, especially when the RL problem under consideration is characterised by high-dimensional state and action spaces. In particular, the *return distribution* can be multimodal and only considering its expected value might result in losing relevant information regarding the intrinsic randomness characterising the RL task.
DRL studies how the entire return distribution can be modelled and used in practice to tackle RL tasks in a more general way.

Before going through the details of the DRL theoretical and algorithmic frameworks, we provide a visual example and one thought experiment aimed at emphasising some possible advantages resulting from the adoption of a distributional point of view on RL.

Atari Games: Space Invaders. Figure 3.1 describes a particular frame from "Space Invaders", one of the games included in the Atari 2600 suite (Bellemare et al. [2013]) and the return distribution associated with each possible action the agent can perform. In this case, the return distributions are able to capture all the different outcomes of the episode. Most importantly, they separate the low returns and the high ones associated with losing or

winning the game respectively. This aspect represents a substantial difference from classical RL, where instead only unrealisable expected values are taken into account.

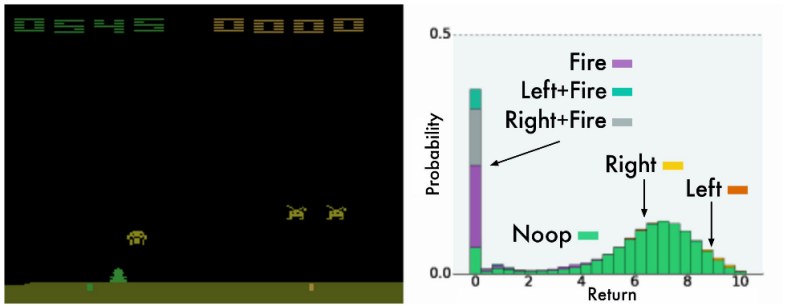


Fig. 3.1 Return distributions generated by the C51 algorithm associated with a particular frame of Space Invaders (Bellemare et al. [2017]).

Financial Transactions. A possible scenario where a distributional approach could be useful is within the context of financial transactions. Let's suppose that an agent needs to choose between two different chains of investments that are forecasted to pay him back the same expected return. A possible action-selection criterion could be to select the investment strategy corresponding with the smallest variance so as to minimise the risk. In other words, it would be reasonable to follow the strategy that leads to the same final expected return with a relatively high level of precision instead of that whose outcome is more fluctuating.
This example proposes an application in which the design of *risk-aware* policies plays a crucial role. In the final part of this chapter, we present some recent works that use DRL to cope with the problem of uncertainty and risk-estimation in RL.

3.2 An Introduction to DRL: Key Concepts and Algorithms

In this section, we introduce the main aspects of the DRL framework both from the algorithmic and the theoretical points of view.

As explained in Chapter 1, the Bellman's equation is written in terms of the expected return, $Q^\pi(x,a)$, as follows:

$$\begin{aligned} Q^\pi(x,a) &= \mathbb{E}\left[R(x,a) + \gamma Q^\pi\left(x',a'\right)\right] \\ \text{where } x' &\sim P(\cdot|x,a) \text{ and } a' \sim \pi\left(\cdot|x'\right) \end{aligned} \tag{3.1}$$

The basic idea of DRL is to consider a distributional version of Eq. 3.1 as the core of its new theoretical structure. This new equation is called *distributional Bellman's equation* and is

given by:

$$Z^{\pi}(x,a) \overset{D}{=} R(x,a) + \gamma Z^{\pi}\left(X',A'\right), \tag{3.2}$$

where the random return $Z^{\pi}(x,a)$ is implicitly defined by $Q^{\pi}(x,a) = \mathbb{E}\left[Z^{\pi}(x,a)\right]$ and R, X' and A' are random variables. The symbol $\overset{D}{=}$ indicates that Eq. 3.2 is a distributional equation which means that the random variable to its LHS is distributed in the same way as the one to the RHS.

The goal of the following sections of this chapter is to try to address two different sets of questions related to Eq. 3.2, that summarise some of the latest research directions emerged in the DRL literature:

1 Does Eq. 3.2 maintain the same convergence guarantees provided by the "classical" Bellman's equation?

2 How can we practically design an algorithm based on Eq. 3.2 whose output is a distribution over returns and how can we exploit the additional amount of information carried by this distribution?

Although these two questions are strictly related to each other, it is quite common to find the theoretical (Question 1) and the algorithmic (Question 2) approaches separated in the literature.

In the remaining part of this section, we start by introducing the theoretical background and the implementation details of two of the most successful distributional algorithms, namely *C51* and *QR-DQN*. Then in the next sections, we first summarise some recent additional mathematical findings aimed at making DRL a more principled branch of classical RL, and finally, we present an overview of some recent applications of DRL algorithms.

3.2.1 Categorical DRL: C51

The idea of a distributional approach to RL is not new and has been already investigated by a relatively large number of previous works in the RL literature. However, the first practical attempts at studying the return distribution aimed to cope with very specific tasks, such as the implementation of risk-sensitive algorithms (Morimura et al. [2010a,b]), or the estimation of the epistemic uncertainty (Dearden et al. [1998]).

In contrast, Bellemare et al. [2017] propose a new algorithm (C51) which is devised to approximate the return distribution and whose implementation is based on the distributional

version of the Bellman's equation 3.2. The ideas proposed in Bellemare et al. [2017] paved the way to a rich series of new interesting methods in DRL and, for this reason, Bellemare et al. [2017] can be considered as the first demonstration of the potential of this field.

The paper is divided into two parts: in the first part, a theoretical analysis of Eq. 3.2 and of its properties under the policy evaluation and control settings is presented, while in the second one, the C51 algorithm is introduced and the results of its application to the Atari games are discussed. The most salient aspects of Bellemare et al. [2017] are summarised in the following.

The Distributional Bellman Operator

In Section 1.1.1, we defined the Bellman operator $\mathscr{T}^{\pi}$ and the optimality operator $\mathscr{T}$ as:

$$\mathscr{T}^{\pi}Q(x,a) := \mathbb{E}R(x,a) + \gamma \mathop{\mathbb{E}}_{P,\pi} Q\left(x',a'\right) \tag{3.3}$$

$$\mathscr{T}Q(x,a) := \mathbb{E}R(x,a) + \gamma \mathbb{E}_{P} \max_{a' \in \mathscr{A}} Q\left(x',a'\right) \tag{3.4}$$

As discussed in Chapter 1, popular algorithms such as SARSA (Rummery and Niranjan [1994]) and Q-Learning (Watkins and Dayan [1992]) can be described in terms of these operators and rely on their contraction properties.

A fully distributional approach inevitably requires a characterisation of the behaviour of the *distributional operators* associated with Eq. 3.2. These operators can be straightforwardly defined as follows[1]:

$$\mathscr{T}^{\pi}Z(x,a) := R(x,a) + \gamma Z\left(X',A'\right) \tag{3.5}$$

$$\mathscr{T}Z(x,a) := R(x,a) + \gamma Z\left(X', \mathop{\text{argmax}}_{a' \in \mathscr{A}} \mathbb{E}Z\left(X',a'\right)\right) \tag{3.6}$$

for the policy evaluation and the control case, respectively. As for the "classical" Bellman operators, we need to understand if the operators introduced above are contraction mappings in order to determine their convergence properties.

Intuitively, the idea behind the Banach fixed point theorem is closely related to the concept of *distance*. If we define a metric with respect to which distances are measured, an operator T

[1] Here again we adopt the convention that capital letters are used to indicate random variables.

is a contraction mapping if its application to two distinct functions Q_0 and Q_1 generates two new functions whose relative distance (measured with respect to the initially defined metric) is smaller than that between the original ones. This rationale implies that the contraction mapping property *depends* on the particular metric under consideration.

As regards the distributional Bellman operator, it has been proved (Chung and Sobel [1987]) that $\mathscr{T}^\pi$ is not a contraction mapping relatively to the total variation distance, the Kullback-Leibler divergence and the Kolmogorov distance. On the other hand, in Bellemare et al. [2017] a new metric called *Wasserstein distance* (Bickel and Freedman [1981]) is introduced and used to demonstrate new convergence properties of the distributional operators $\mathscr{T}^\pi$ and $\mathscr{T}$. A brief description of this metric is reported in the next paragraph.

The Wasserstein Distance. Given two random variables U and V with inverse cumulative distribution functions F_U^{-1} and F_V^{-1} respectively, the p-Wasserstein distance is defined as[2][3]:

$$d_p(F_U, F_V) = \left(\int_0^1 \left| F_U^{-1}(u) - F_V^{-1}(u) \right|^p du \right)^{1/p} \tag{3.7}$$

It can be shown (Levina and Bickel [2001]) that the p-Wasserstein distance is equivalent to the so-called Earth Mover's Distance (EMD) when $p = 1$. This distance is closely related to the minimal cost necessary for transporting one distribution's mass into the other to make the two distributions identical (Dabney et al. [2018a]).
Given $Z_1, Z_2 \in \mathscr{Z}$, where $\mathscr{Z}$ is the space of value distributions with bounded moments, the *maximal form* of the Wasserstein metric is defined as follows:

$$\overline{d}_p(Z_1, Z_2) := \sup_{x,a} d_p(Z_1(x,a), Z_2(x,a)) \tag{3.8}$$

In the following, we use the Wasserstein distance to characterise the behaviour of the distributional Bellman operators defined by Eq. 3.5 and Eq. 3.6. We prefer to focus on the main results, omitting the mathematical details used to obtain them in order to maintain the exposition as fluid as possible.

Policy Evaluation. We now consider the policy evaluation setting where the goal consists in evaluating the return distribution Z^π associated with a given policy π.
Given an initial distribution $Z_0 \in \mathscr{Z}$, we are interested in the iterative process implicitly

[2] In the following, we often use the notation $d_p(U,V)$ instead of $d_p(F_U, F_V)$, for the sake of clarity.
[3] For $p = \infty, W_\infty(Y,U) = \sup_{\omega \in [0,1]} \left| F_Y^{-1}(\omega) - F_U^{-1}(\omega) \right|$

defined by the distributional Bellman operator, $Z_{k+1} := \mathscr{T}^{\pi} Z_k$. The following important result holds:

- **Theorem 3.2.1 (Bellemare et al. [2017])** *$\mathscr{T}^{\pi} : \mathscr{Z} \to \mathscr{Z}$ is a γ-contraction in $\overline{d}_p$, where $\gamma \in [0,1)$ is the discount factor.*

The above contraction property holds under the maximal form of the Wasserstein metric, $\overline{d}_p$, and this implies that all the moments of Z_k converge exponentially quickly. In particular, this is true for $\mathbb{E}[Z_k]$, as expected from the contraction property of the "classical" Bellman operator.

Control. In the control setting, the goal is to find a policy that maximises the value function. Unfortunately, in this case, we can only prove that the corresponding Bellman operator (Eq. 3.6) converges in a weak sense, to the set of nonstationary optimal value distributions.

- **Theorem 3.2.2 (Bellemare et al. [2017])** *Let $\mathscr{X}$ be measurable and suppose that $\mathscr{A}$ is finite. Then*

$$\lim_{k \to \infty} \inf_{Z^{**} \in \mathscr{Z}^{**}} d_p\left(Z_k(x,a), Z^{**}(x,a)\right) = 0 \quad \forall x, a$$

where $\mathscr{Z}^{**}$ is the set of non nonstationary optimal value distributions, i.e. the set of value distributions associated with a sequence of optimal policies.
If the optimal policy, π^*, is unique, then the iterates $Z_{k+1} \leftarrow \mathscr{T} Z_k$ converge to Z^{π^*}.

It can be easily shown that Theorem 3.2.2 implies that, in general, the operator $\mathscr{T}$ *is not* a contraction mapping. By comparing theorem 3.2.1 and 3.2.2 we can see that the distributional Bellman operators behave in a significantly different way to each other.

Finally, for the sake of completeness, it is worth mentioning that the "classical" case can be recovered from the optimality operator in the distributional case:

- **Theorem 3.2.3 (Bellemare et al. [2017])** *Given $Z_1, Z_2 \in \mathscr{Z}$, then*

$$\|\mathbb{E}\mathscr{T} Z_1 - \mathbb{E}\mathscr{T} Z_2\|_{\infty} \leq \gamma \|\mathbb{E} Z_1 - \mathbb{E} Z_2\|_{\infty}$$

and in particular $\mathbb{E} Z_k \to Q^*$ exponentially quickly.

The C51 Algorithm

In this section, we describe the C51 algorithm, firstly introduced in Bellemare et al. [2017]. C51 is based on the distributional Bellman optimality operator. However, its design details do not take into account the theoretical considerations reported in the first part of the paper. The link between theory and practice is more thoroughly investigated in the following sections.

The basic idea of C51 is to parametrise the return distribution by a finite set of N equally-spaced support points[4] included into the interval $[V_{MIN}, V_{MAX}]$. The probability associated with each support point z_i is given by a parametric model $\theta : \mathscr{X} \times \mathscr{A} \to \mathbb{R}^N$:

$$p_i(x,a) := \frac{e^{\theta_i(x,a)}}{\sum_j e^{\theta_j(x,a)}} \qquad \text{for } i = 1,...,N \tag{3.9}$$

The parametric model is chosen to be a deep neural network of the same kind of the DQN architecture[5] (Mnih et al. [2015]) described in Chapter 2. The only difference is that, in this case, the outputs are the probabilities $p_i(x,a)$ associated with each action, given a certain state as input. An illustration of the C51 network architecture is showed in Fig. 3.2.

The goal of the algorithm is to iteratively minimise a distance measure between the Bellman update $\mathscr{T}Z_\theta$ and the distribution Z_θ returned by the parametric model defined above.

However, the computation of the target distribution presents a complication: $\mathscr{T}Z_\theta$ and Z_θ have often disjoint supports. To cope with this issue, the sample Bellman update $\hat{\mathscr{T}}Z_\theta$ is firstly projected onto the support of Z_θ. This operation is summarised by the application of a projection operator Φ. Finally, the distance between the projected update and the parametric distribution is measured by the cross entropy term of the KL divergence:

$$D_{\mathrm{KL}}\left(\Phi\hat{\mathscr{T}}Z_{\tilde{\theta}}(x,a) \| Z_\theta(x,a)\right) \tag{3.10}$$

that naturally lends itself to SGD-based minimisation. Given a sample transition (x,a,R,x'), the action of the projection operator Φ on the parametric distribution Z_θ is given by the

[4] The name of the algorithm derives from the choice $N = 51$ which guaranteed the best performance on several Atari games.

[5] As in the DQN algorithm, a replay buffer and a distinct target network with a separate set of parameters $\tilde{\theta}$ are used.

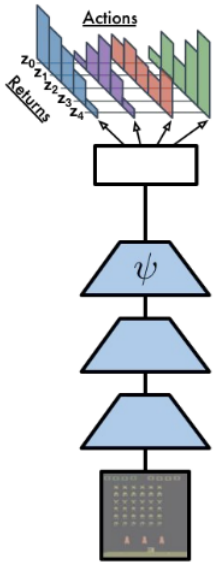


Fig. 3.2 C51 neural network architecture. The outputs are the distributions over returns associated with each action. Image extracted from Rémi Munos' presentation at ENS Organization.

following equation:

$$\left(\Phi \hat{\mathcal{T}} Z_\theta(x,a)\right)_i = \sum_{j=0}^{N-1} \left[1 - \frac{|[\hat{\mathcal{T}} z_j]_{\mathrm{V_{MIN}}}^{\mathrm{V_{MAX}}} - z_i|}{\Delta z}\right]_0^1 p_j\left(x', \pi\left(x'\right)\right) \tag{3.11}$$

where $\Delta z = \frac{V_{MAX} - V_{MIN}}{N-1}$ and $\hat{\mathcal{T}} z_j := R + \gamma z_j$. Eq. 3.11 shows how the probability associated with each atom z_j is distributed to the nearest neighbours of $\hat{\mathcal{T}} z_j$.
The complete version of the C51 algorithm is reported below.

Algorithm 3 Categorical Algorithm (C51)

Require: A transition $x_t, a_t, R_t, x_{t+1}, \gamma \in [0,1)$

$Q(x_{t+1}, a) := \sum_i z_i p_i(x_{t+1}, a)$

$a^* \leftarrow \arg\max_a Q(x_{t+1}, a)$

$m_i = 0, \quad i = 0, \ldots, N-1$

for $j \in 0, \ldots, N-1$ **do**

Initialise x

Compute the projection $\hat{\mathcal{T}} z_j$ onto the support $\{z_i\}$

$\hat{\mathcal{T}} z_j \leftarrow [R_t + \gamma z_j]_{V_{MIN}}^{V_{MAX}}$

$b_j \leftarrow (\hat{\mathcal{T}} z_j - V_{MIN})/\Delta z$

$l \leftarrow \lfloor b_j \rfloor \;\; u \leftarrow \lceil b_j \rceil$

Distribute probability of $\hat{\mathcal{T}} z_j$

$m_l \leftarrow m_l + p_j(x_{t+1}, a^*)(u - b_j)$

$m_u \leftarrow m_u + p_j(x_{t+1}, a^*)(b_j - l)$

end for

return $-\sum_i m_i \log p_i(x_t, a_t)$

Discussion. When it was published in 2017, the application of the $C51$ algorithm to the Atari games suite resulted in state-of-the-art results in the majority of the games.
Despite its surprising performances, the gap between theory and practice was significant. In particular, the projection step heuristic defined by Eq. 3.11 and the use of the KL divergence as the loss function to be minimised by SGD did not find any clear mathematical justification. These two points have been recently more accurately investigated by Bellemare et al. [2018] and Rowland et al. [2018]. The most salient findings of these works are discussed later within this chapter.
In the next section, we describe the QR-DQN algorithm that partially addresses some of the aforementioned open-problems and further improves performances.

3.2.2 Quantile Regression DRL

The KL divergence is directly related to maximum likelihood estimation and it is relatively easy to optimise. However, its major drawback is that it does not take into account how geometrically close two outcome events might be, but only their relative probability (Bellemare et al. [2018]).

On the other hand, the Wasserstein metric is sensitive to the underlying geometry between outcomes. As pointed out by Dabney et al. [2018b], this property makes the choice of the Wasserstein metric particularly suitable in those cases where the evaluation of outcomes' similarity is more or equally important than the probabilistic agreement between likelihoods.

The algorithm proposed in Bellemare et al. [2017] does not provide an end-to-end approach based on the Wasserstein metric. Indeed, the C51 algorithm makes use of the KL divergence to minimise the distance between the parametrised distribution Z_θ and the target distribution obtained by the Bellman's update projected onto the support of Z_θ, i.e. $\Phi\hat{\mathcal{T}}Z_{\tilde{\theta}}$.

In Dabney et al. [2018b], a new approach motivated by the previous considerations is proposed. The resulting algorithm, named *Quantile-Regression-DQN* (QR-DQN), has a more solid theoretical structure than C51 and further improves the experimental results obtained by Bellemare et al. [2017].

Quantile Parametrisation

QR-DQN is based on a new parametrisation of the return distribution. Specifically, if the aim of Eq. 3.9 consisted in approximating the return distribution by fixing a set of support points and modelling the corresponding probabilities, now the goal is to estimate the *quantiles* of the return distribution. The main difference is that, this time, the probabilities are fixed at N constant numbers but the support points are variable. Given a parametric model $\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^N$, a quantile distribution supported on $\{\theta_i(x,a)\}_{i=1}^N$ can be defined as:

$$Z_\theta(x,a) := \frac{1}{N}\sum_{i=1}^{N} \delta_{\theta_i(x,a)}, \tag{3.12}$$

where δ_{θ_i} indicates a Dirac delta centred at $\theta_i \in \mathbb{R}$. Following the notation of Dabney et al. [2018b], we indicate as $\mathcal{Z}_Q$ the space of quantile distributions for fixed N. The cumulative probabilities associated with the quantile distribution defined in Eq. 3.12 will be denoted by $\tau_i = \frac{i}{N}$ for $i = 1, \dots, N$.

Quantile Projection and Contraction Property. Similarly to Bellemare et al. [2017], we now define a new operator which will be referred to as Π_{W_1}, that projects an arbitrary return

distribution $Z \in \mathscr{Z}$ onto the space $\mathscr{Z}_Q$:

$$\begin{aligned}\Pi_{W_1} Z :=& \underset{Z_\theta \in \mathscr{Z}_Q}{\arg\min} W_1(Z, Z_\theta) = \\ &= \underset{Z_\theta \in \mathscr{Z}_Q}{\arg\min} \left(\sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} \left| F_Z^{-1}(\omega) - \theta_i \right| d\omega \right)\end{aligned} \tag{3.13}$$

It turns out that the set of $\theta \in \mathbb{R}$ minimising $\int_\tau^{\tau'} \left| F^{-1}(\omega) - \theta \right| d\omega$ for any $\tau, \tau' \in [0,1]$ is given by:

$$\left\{ \theta \in \mathbb{R} | F(\theta) = \left(\frac{\tau + \tau'}{2} \right) \right\} \tag{3.14}$$

The previous result shows that $W_1(Z, Z_\theta)$ is minimised by $\theta_i = F_Z^{-1}(\hat{\tau}_i)$ for $i = 1, ..., N$, where $\hat{\tau}_i = \frac{\tau_{i-1} + \tau_i}{2}$.

The reason for the particular choice of the projection operator defined in Eq. 3.13 is that its combination with the distributional Bellman operator $\mathscr{T}^\pi$ is a contraction in the ∞-Wasserstein metric, as shown by the following theorem:

- **Theorem 3.2.4 (Dabney et al. [2018b])** *Let Π_{W_1} be the quantile projection defined as above, and when applied to value distributions gives the projection for each state-value distribution. For any two value distributions $Z_1, Z_2 \in \mathscr{Z}$ for an MDP with countable state and action spaces,*

$$\bar{d}_\infty (\Pi_{W_1} \mathscr{T}^\pi Z_1, \Pi_{W_1} \mathscr{T}^\pi Z_2) \leq \gamma \bar{d}_\infty (Z_1, Z_2) \tag{3.15}$$

  where $\bar{d}_\infty$ is the maximal form of the Wasserstein metric, as defined in Eq. 3.8.

QR-DQN makes use of the projection operator Π_{W_1} in its implementation, providing an algorithm capable of effectively exploiting the benefits of the Wasserstein metric (with respect to which $\Pi_{W_1} \mathscr{T}^\pi$ is a contraction and that defines the projection operator Π_{W_1}), bridging the gap with theory. Before going through the details of QR-DQN, we need to include one last important ingredient in the theoretical formulation: the quantile regression loss.

Biased Gradients

Using the p-Wasserstein metric as a loss function with the parametrisation introduced in Eq. 3.12 leads to *biased gradient*, as shown by the following theorem:

- **Theorem 3.2.5 (Dabney et al. [2018b])** *Let Z_θ be a quantile distribution, and $\hat{Z}_m$ the empirical distribution composed of m samples from Z. Then for all $p \geq 1$, there exists*

a Z such that

$$\arg\min \mathbb{E}\left[W_p\left(\hat{Z}_m, Z_\theta\right)\right] \neq \arg\min W_p\left(Z, Z_\theta\right) \tag{3.16}$$

This result implies that this loss can not be minimised by SGD. A solution to this complication is provided by the so-called *quantile-regression loss* (Koenker [2005]) that we define in the following paragraph.

Quantile Regression Loss. Given a distribution Z and a quantile τ, the inverse cumulative distribution function $F_Z^{-1}(\tau)$ can be defined as the minimiser of the quantile regression loss:

$$\begin{gathered} \mathscr{L}_{\mathrm{QR}}^{\tau}(\theta) := \mathbb{E}_{\hat{Z}\sim Z}\left[\rho_\tau(\hat{Z}-\theta)\right], \text{ where} \\ \rho_\tau(u) = u\left(\tau - \delta_{\{u<0\}}\right), \forall u \in \mathbb{R} \end{gathered} \tag{3.17}$$

It turns out (Koenker [2005]) that the minimisation of this loss gives *unbiased sample gradients*.

We are now in the position to exploit all the theoretical arguments developed so far. We know that the operator $\Pi_{W_1}\mathscr{T}^\pi$ is a contraction and that the values of $\{\theta_1,\ldots,\theta_N\}$ that minimise $W_1(\mathscr{T}^\pi Z, Z_\theta)$ are given by $\theta_i = F^{-1}_{\mathscr{T}^\pi Z}(\hat{\tau}_i)$. Therefore, the quantile regression loss can be rewritten as follows:

$$\mathscr{L}_{\mathrm{QR}}^{\hat{\tau}}(\theta) = \sum_{i=1}^{N} \mathbb{E}_{\hat{Z}\sim\mathscr{T}^\pi Z}\left[\rho_{\hat{\tau}_i}\left(\hat{Z}-\theta_i\right)\right] \tag{3.18}$$

The minimisation of $\mathscr{L}_{\mathrm{QR}}^{\hat{\tau}}(\theta)$ yields the set of support points $\theta_i = F^{-1}_{\mathscr{T}^\pi Z}(\hat{\tau}_i)$ which in turn minimise the 1-Wasserstein distance to $\mathscr{T}^\pi Z$.

The QR-DQN Algorithm

In Dabney et al. [2018b], the theoretical results discussed above are used to propose a new control algorithm, named QR-DQN. Its main steps are summarised in the pseudocode reported below.

Algorithm 4 Quantile Regression DQN (QR-DQN)

Require: N

input: $x, a, R, x', \gamma \in [0,1)$

Compute distributional Bellman target

$Q(x',a') := \frac{1}{N}\sum_j \theta_j(x',a')$

$a^* \leftarrow \arg\max_{a'} Q(x,a')$

$\mathscr{T}\theta_j \leftarrow R + \gamma\theta_j(x',a^*), \quad \forall j$

Compute quantile regression loss

output: $\sum_{i=1}^{N} \mathbb{E}_j \left[\rho_{\hat{\tau}_i}\left(\mathscr{T}\theta_j - \theta_i(x,a)\right)\right]$

In order to minimise the quantile regression loss defined above, a very similar architecture to DQN (Mnih et al. [2015]) is used[6]. The main differences between the two implementations are the following:

- The network output is of size $|\mathscr{A}| \times N$, meaning that for for each action, N support points $\{\theta_i\}_{i=1}^{N}$ are returned. The QR-DQN network architecture is illustrated in Fig. 3.3
- The quantile regression loss[7] is considered instead of the Huber loss used in DQN.
- The Adam optimiser (Kingma and Ba [2014]) is used instead of RMSProp.

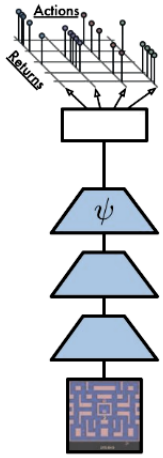


Fig. 3.3 QR-DQN neural network architecture. The outputs are the quantile distributions associated with each action. Image extracted from Rémi Munos' presentation at the ENS Organization.

[6] As in Mnih et al. [2015], a separate target network is used to compute the distributional Bellman update.

[7] In Dabney et al. [2018b], a slightly different version of the quantile regression loss, called quantile Huber loss, is used. The reason for this modification is that the quantile regression loss is not smooth at zero, generating constant gradients. We refer the interested reader to Dabney et al. [2018b] for a more detailed explanation.

Discussion. The QR-DQN algorithm described in Fig. 3.3, further improves the performances of the C51 algorithm on the Atari games suite. Contrarily to C51, the relation between theory and practice is more consistent in this case. Furthermore, the new parametrisation introduced in 3.12 does not constrain the return distribution on a finite set of support points. As suggested by Dabney et al. [2018b], this may be advantageous when the underlying range of returns varies significantly across different states.

Both algorithms described so far have been widely tested on the Atari Games, resulting in excellent performances. A comparison between the *human-normalised scores* (Hasselt et al. [2016]) obtained by C51, QR-DQN and some of the most successful deep RL algorithms is reported in Table 3.1.

	Mean	**Median**
DQN	228%	79%
DDQN	307%	118%
DUEL.	373%	151%
PRIOR.	434%	124%
PRIOR. DUEL.	592%	172%
C51	701%	178%
QR-DQN	864%	193%

Table 3.1 *Best* mean and median performances (expressed as human normalised scores) of DQN, Double DQN (DDQN) (Hasselt et al. [2016]), Dueling Architecture (DUEL) (Wang et al. [2016]), Prioritised Replay (PRIOR) (Schaul et al. [2016]), C51 and QR-DQN. These scores are obtained by training each agent for 200 million frames.

Citing the words of Bellemare et al. [2017]: "*It's already evident from our empirical results that the distributional perspective leads to better, more stable reinforcement learning*". However, several questions remain open:

- Why does the C51 algorithm provide very good performances despite the gap between theory and practice present in its formulation?
- Why does DRL work better than "classical" RL?
- Is it possible to introduce a common theoretical ground for apparently different DRL algorithms such as C51 and QR-DQN?

3.3 Towards a More Principled Formulation of DRL

In this section, we provide a summary of the most significant attempts at answering the set of questions introduced at the end of the previous paragraph. The following results have been extracted from the most recent works in the DRL literature and, as we will see, some of them leave room for further improvements and refinements.

3.3.1 C51: Bridging the Theory-Practice Gap

In this section, we provide a formal justification of the projection step introduced in Bellemare et al. [2017] which, despite being a key component of the final algorithm, does not find any theoretical motivation in the original paper.

We start by introducing a new metric, the *Cramér metric*, that combines useful properties of the KL divergence and the Wasserstein distance.

The Cramér Distance

The Cramér distance (Rizzo and Székely [2016]) between two distributions P and Q is given by:

$$l_2(P,Q) := \left(\int_{-\infty}^{\infty} |F_P(x) - F_Q(x)|^2 \, \mathrm{d}x \right)^{1/2} \tag{3.19}$$

where F_P and F_Q are the cumulative distribution functions of P and Q respectively[8]. It can be proved (Bellemare et al. [2018]) that the Cramér distance satisfies three important properties, as shown by the following theorem.

- **Theorem 3.3.1 (Bellemare et al. [2018])** *Consider two random variables X, Y, a random variable A independent of X, Y, and a real value $c > 0$. Then,*

$$l_2(A+X, A+Y) \leq l_2(X,Y) \quad (\boldsymbol{I}) \qquad l_2(cX, cY) \leq |c|^{\beta} l_2(X,Y) \quad (\boldsymbol{S}) \tag{3.20}$$

where $\beta = \frac{1}{2}$. Furthermore, the Cramér distance has unbiased sample gradients. That is, given $\mathbf{X}_m := X_1, \ldots, X_m$ drawn from a distribution P, the empirical distribution $\hat{P}_m := \frac{1}{m} \sum_{i=1}^{m} \delta_{X_i}$, and a distribution Q_θ:

$$\mathbb{E}_{\mathbf{X}_{m} \sim P} \nabla_\theta l_2^2 \left(\hat{P}_m, Q_\theta \right) = \nabla_\theta l_2^2 \left(P, Q_\theta \right) \quad (\boldsymbol{U}) \tag{3.21}$$

[8] Similarly to the Wasserstein metric, given two value distributions $Z_1, Z_2 \in \mathscr{Z}$, we can define the maximal form of the Cramér distance as $\bar{l}_2(Z_1, Z_2) := \sup_{x,a} l_2(Z_1(x,a), Z_2(x,a))$

Interestingly, Bellemare et al. [2018] showed that the p-Wasserstein metric possesses properties $\boldsymbol{I}$ and $\boldsymbol{S}$ (with $\beta = 1$) but not $\boldsymbol{U}$. As opposite, the KL divergence possesses property $\boldsymbol{U}$ but not $\boldsymbol{S}$.

As shown in Bellemare et al. [2017], $\boldsymbol{I}$ and $\boldsymbol{S}$ are crucial to prove the contraction property of the distributional Bellman operator. On the other hand, as we saw in the previous section when we defined the quantile regression loss, property $\boldsymbol{U}$ is key to design loss functions that are compatible with SGD and whose minimisation converges to the right minimum.
Therefore, in virtue of its good properties, the Cramér distance seems to be an ideal candidate for DRL algorithms.

The Projection Operator in the C51 Algorithm

The results reported in this section are mainly extracted from Rowland et al. [2018] and presented here with a slight change of notation for the sake of coherence with the previous sections.
We start by the most important result in Rowland et al. [2018] which provides a meaningful interpretation of the C51 projection operator Φ in terms of the Cramér metric (in its maximal form) defined in the previous section.

- **Theorem 3.3.2 (Rowland et al. [2018])** *The operator $\Phi\mathcal{T}^\pi$ is a $\sqrt{\gamma}$-contraction in $\bar{l}_2$, where $\gamma \in [0,1)$ is the discount factor. Further, there is a unique distribution function $Z_\mathscr{C}$ such that given any initial distribution function Z_0, we have:*

$$(\Phi\mathcal{T}^\pi)^m Z_0 \to Z_\mathscr{C} \text{ in } \bar{l}_2 \text{ as } m \to \infty \tag{3.22}$$

Although the previous theorem proves that the combination of the projection operator Φ and the Bellman operator $\mathcal{T}^\pi$ is contraction in the Cramér metric, it is still unclear how the limiting distribution $Z_\mathscr{C}$ is related to the "true" distribution Z^π. Fortunately, it can be proved (Rowland et al. [2018]) that the following result holds:

$$\bar{l}_2^2(Z_\mathscr{C}, Z^\pi) \leq \frac{1}{(1-\gamma)} \max_{1 \leq i < N} |z_{i+1} - z_i| \tag{3.23}$$

where N is the number of support points of the parametrised distribution. Interestingly, Eq. 3.23 shows that, as the density of support points in the interval $[z_1, z_N]$ increases, the true distribution can be approximated with increasingly higher accuracy.

3.3.2 Why does DRL work better than "classical" RL?

A clear theoretical explanation of the successful performances of DRL algorithms compared to standard "expectation-based" RL techniques is still missing. However, the first attempts at coping with this open-problem have provided interesting results.

In Lyle et al. [2019], an investigation of the behavioural differences between distributional and expectation-based RL algorithms is carried out. The methodology used to draw this comparison is based on coupling the experience exploited by the update rules of the two classes of algorithms. In other words, given a distributional update rule U_D and the corresponding expectation-based operator U_E, the sequence of updates in the two cases is described as follows:

$$Q_{t+1} := U_E(Q_t, \omega_t) \quad Z_{t+1} := U_D(Z_t, \omega_t) \tag{3.24}$$

where ω_t is the common sample of experience used by the two agents at time-step t.

In Lyle et al. [2019], the goal is to identify under which conditions the following relation is satisfied:

$$Z_0 \overset{\mathrm{E}}{=} Q_0 \iff Z_t \overset{\mathrm{E}}{=} Q_t \quad \forall t \in \mathbb{N}, \forall\, Z_0, Q_0 \tag{3.25}$$

where the notation $\overset{\mathrm{E}}{=}$ has the following interpretation:

$$Z \overset{\mathrm{E}}{=} Q \iff \mathbb{E}[Z(x,a)] = Q(x,a) \quad \forall (x,a) \in \mathscr{X} \times \mathscr{A} \tag{3.26}$$

In the paper, the authors use the criterion in Eq. 3.25 to establish if a standard RL algorithm and its distributional counterpart share the same behaviour.

Surprisingly, it has been shown that, when the tabular and the linear approximation settings are considered, using DRL instead of standard RL algorithms *is not beneficial* (the algorithms are equivalent in the sense specified by Eq. 3.25) and it has been empirically demonstrated that it can sometimes hinder performances. Similar negative results concerning the linear approximation setting have been highlighted in Bellemare et al. [2019].

On the other hand, when non-linear approximators such as deep neural networks are used, Eq. 3.25 does not hold, meaning that that the two classes of algorithms behave differently, as confirmed by the success of DRL over standard RL approaches.

The aforementioned recent findings seem to suggest that a distributional approach to RL is actually beneficial only within a DL framework. The reason why this is the case is still not clear and needs to be further investigated.

3.3.3 DRL: a Comprehensive Interpretation

In this section, we provide a brief summary of one of the most recent and interesting advances in the DRL theoretical formulation (Rowland et al. [2019]). The main contribution of this work is to propose a unifying framework that allows us to view apparently diverse DRL algorithms such as C51 and QR-DQN under the same perspective.

Specifically, as explained in Rowland et al. [2019], DRL algorithms can be rephrased in terms of two basic complementary concepts, namely *statistics* and *imputation strategies*, which we define in the next section.

Statistics and Imputation Strategies in DRL

A statistic is simply defined as a function $s : \mathscr{P}(\mathbb{R}) \to \mathbb{R}$, where $\mathscr{P}(\mathbb{R})$ is the space of probability distributions[9].

DRL algorithms can be generally interpreted as methods devised to recursively estimate sets of statistics. While the statistics learned in QR-DQN can be easily identified as the finite set of quantiles extracted from the target distribution by minimising the quantile regression loss, the statistics associated with the C51 algorithm are less obvious to recognise. As shown by Rowland et al. [2019], given a set of support points $z_1 < \ldots < z_K$, the statistics learned in C51 are defined as:

$$s_{z_k, z_{k+1}}(\mu) = \mathbb{E}_{Z \sim \mu}\left[h_{z_k, z_{k+1}}(Z)\right] \text{ for } k = 1, \ldots, K-1 \tag{3.27}$$

[9] Here, we use a slightly different and more imprecise notation than the original paper. In Rowland et al. [2019] the return distribution is indicated as $\eta_\pi(x,a)$, whereas here $Z_\pi(x,a)$ is used. Although this notation is incorrect since Z is used also to indicate the return random variable, we use it for the sake of coherence with the previous sections.

where $\mu := (\mathscr{T}^\pi Z)(x,a)$ is the target distribution and the function $h_{a,b} : \mathbb{R} \to \mathbb{R}$ is defined as follows:

$$h_{a,b} = \begin{cases} 1 & \text{if } x \leq a \\ \frac{x-b}{a-b} & \text{if } x \in (a,b) \\ 0 & \text{if } x \geq b \end{cases} \tag{3.28}$$

DRL algorithms are all based on a common recursive update scheme for the corresponding statistics, which is given by:

$$\hat{s}_k(x,a) \leftarrow s_k\left(\left(\mathscr{T}^\pi Z\right)(x,a)\right) \tag{3.29}$$

where $\hat{s}_k(x,a)$ denotes the current approximation of the value of the statistic $s_k(x,a)$ at a state action pair (x,a). In fact, in order to use Eq. 3.29, we would need to have access to the target distribution[10] $\mathscr{T}^\pi Z$ which in turn depends on the return distribution $Z(x',a')$ at each possible next state-action pair (x',a'). Indeed, the knowledge of the target distribution would allow us to compute the argument of the statistic function s_k and perform the update.

The next state-action pair distribution $Z(x',a')$, and hence the target distribution $\mathscr{T}^\pi Z(x,a)$ can be obtained by applying an *imputation strategy*. An imputation strategy $\Psi : \mathbb{R}^K \to \mathscr{P}(\mathbb{R})$ is a function that maps a set of statistic values into a distribution that is characterised by those statistic values.

As shown in Rowland et al. [2019], both C51 and QR-DQN use imputation strategies in their implementations. In QR-DQN, given a collection of statistic values $\sigma_{1:K} \in \mathbb{R}^K$, the imputation strategy is given by $\Psi(\sigma_{1:K}) = \frac{1}{K}\sum_{k=1}^K \delta_{\sigma_k}$. In C51, given approximate statistics $\hat{s}_{z_k,z_{k+1}}(x,a)$ for $k = 1,\ldots,K-1$, the imputation strategy is given by selecting the distribution $\sum_{k=1}^K p_k \delta_{z_k}$ such that $p_1 = \hat{s}_{z_1,z_2}(x,a)$, $p_k = \hat{s}_{z_k,z_{k+1}}(x,a) - \hat{s}_{z_{k-1},z_k}(x,a)$ for $k = 2,\ldots,K-1$, and $p_K = 1 - \sum_{k<K} p_k$.

We are now in the position to define a general three-step procedure to guide the design of new DRL algorithms and to explain the behaviour of the existing ones (Rowland et al. [2019]):

- Select the family of statistics to learn

[10] Or an approximation of it.

- Select an imputation strategy
- Apply Eq. 3.29 to perform updates

This rationale is summarised in the pseudocode reported below and is illustrated in Fig. 3.4.

Algorithm 5 Generic DRL update algorithm.

Require: Statistic estimates $\hat{s}_{1:K}(x,a)\ \forall(x,a) \in \mathscr{X} \times \mathscr{A}$
and $k = 1,...,K$, imputation strategy Ψ
Select state-action pair $(x,a) \in \mathscr{X} \times \mathscr{A}$ to update.
Inpute distribution at each possible next state-action pair:
$Z(x',a') = \Psi(\hat{s}_{1:K}(x',a')), \quad \forall(x',a') \in \mathscr{X} \times \mathscr{A}$
Update statistics at $(x,a) \in \mathscr{X} \times \mathscr{A}$:
$\hat{s}_k(x,a) \leftarrow s_k\left((\mathscr{T}^\pi Z)(x,a)\right)$

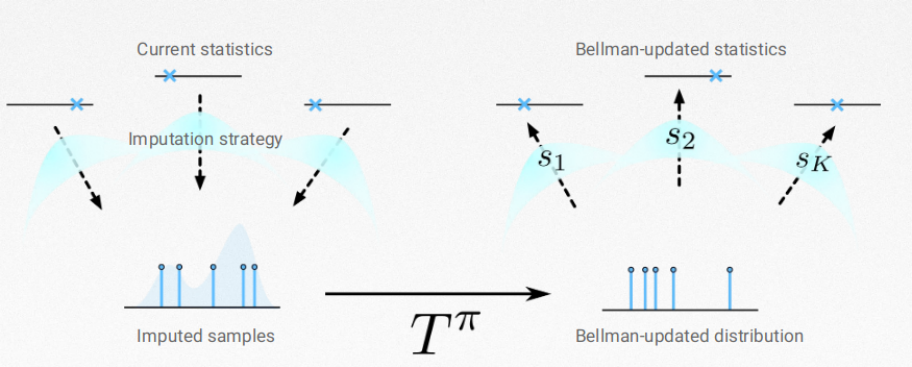


Fig. 3.4 DRL formulated in terms of statistics and imputation strategies. Image extracted from Will Dabney's presentation at ICML 2019.

The new framework described in this section is particularly interesting because it allows us to interpret all the existing DRL algorithms under the perspective of learning sets of statistics. At the same time, this new point of view provides a more principled and consistent way to design novel DRL algorithms. For instance, in Rowland et al. [2019], the authors introduce Expectile-Regression DQN (ER-DQN), a new DRL algorithm based on learning *expectiles* instead of quantiles. The first tests of ER-DQN on the Atari games seem to provide improved results, in terms of mean human-normalised performances, relative to existing DRL techniques, such as QR-DQN.

3.4 Applications and Current Trends

In this section, we build upon the concepts introduced so far to describe how distributional methods can either be effectively exploited to accomplish specific RL tasks or how they can be integrated into other algorithms to boost their performance in various RL contexts.

In particular, in the following, we analyse three scenarios in which distributional algorithms have been recently employed:

- **Uncertainty and risk estimation**: Can we exploit the richer information content provided by the return distribution to model uncertainty in RL tasks or to design risk-aware policies?
- **Performance optimisation**: Can the combination of DRL algorithms with recent technical advances in the deep RL framework produce better performances?
- **Continuous control setting**: Can DRL algorithms be efficiently applied to RL environments characterised by continuous action spaces, such as robotics tasks?

3.4.1 Uncertainty and Risk Estimation

Learning the return distribution empirically results, in and of itself, in better performances. Several interpretations of this behaviour have been proposed and some of them impute the success of DRL algorithms to their interaction with DL techniques (Lyle et al. [2019]) or to the introduction of a set of *auxiliary tasks* carrying more training signals than a scalar value $Q(s,a)$ (Francois-Lavet et al. [2018]).

Despite their empirical success, all the algorithms described in the previous sections do not directly take advantage of the estimated return distribution. Indeed, they follow the same approach as standard RL algorithms where the resulting policies are entirely based on the mean of the return distribution.

Intuitively, as mentioned in the thought experiment at the beginning of the chapter, having access to the entire return distribution could provide useful information about the risk and the uncertainty characterising the RL task under consideration.

In the following section, we first briefly discuss the two main types of uncertainty characterising a typical RL problem and then we present some examples of DRL algorithms applied to risk and uncertainty estimation.

Two Types of Uncertainty

RL algorithms are characterised by two types of uncertainty: *epistemic or parametric uncertainty* and *aleatoric or intrinsic uncertainty* (Moerland et al. [2018]). The former arises from our limited knowledge of the environment and can be reduced by collecting more data. The second derives from the inherent stochasticity characterising the environment or the policy and represents an intrinsic feature of the RL task under consideration.

As explained in Moerland et al. [2017a], estimating these two types of uncertainty can be useful for different purposes: the epistemic uncertainty can be used to guide exploration towards poorly known states, whereas the intrinsic uncertainty can be more informative for designing risk-sensitive policies, where careful planning strategies can be devised from a better knowledge of the environmental uncertainty. This last kind of uncertainty can be efficiently modelled by the distribution over returns (Moerland et al. [2017a]; Clements et al. [2019]; Dabney et al. [2018a]; Nikolov et al. [2018]).

In the following, we present three recent examples from the literature of how DRL can be used to design risk-aware policies and enhance exploration.

1) Quantile-Based Risk Estimation. In Zhang et al. [2018], a new interesting action-selection heuristic, called QUOTA, is introduced. This simple approach is entirely based on the quantile distribution, that represents the output of the QR-DQN algorithm.

The basic idea is that high (low) quantiles correspond to *optimistic* (*pessimistic*) estimates of the action value. In Zhang et al. [2018], actions are selected w.r.t. specific quantile intervals that are chosen depending on the level of optimism of the agent at a particular time-step during its interaction with the environment.

To put this idea in practice, in Zhang et al. [2018], the *option framework* (Sutton et al. [1999]) is used: besides the standard policy, a higher-level policy, Q_Ω, that establishes which quantile interval to use is learned.
More formally, at time-step t, actions are selected as follows:

$$a_t \leftarrow \begin{cases} \text{random action} & \text{w.p. } \varepsilon \\ \arg\max_{a \in \mathscr{A}} \sum_{k=(j-1)K+1}^{(j-1)K+K} \theta_k(x_t, a) & \text{w.p. } 1-\varepsilon \end{cases} \tag{3.30}$$

where $j = 1, ..., M$ indicates the particular option selected by the high-level policy and θ_k denotes the k-th quantile.

This method has been tested on both a relatively simple toy experiment and on the Atari games suite. The toy experiment consisted of two simple Markov chains whose solution required optimistic and pessimistic strategies respectively. The comparison of the quantile-based action-selection rule described above and other techniques designed by taking into account chain-specific knowledge, shows that QUOTA is able to adapt to the particular environment by following either an optimistic or pessimistic strategy automatically.
QUOTA is then tested on the Atari games, resulting in improved performances than QR-DQN in 23 games and worse results in 14.

As explained in Dabney et al. [2018a], the canonical approach to design risk-averse or risk-seeking policies is based on the introduction of a utility function that is used to distort a value distribution. The method proposed in Zhang et al. [2018] is particularly interesting because, as the authors suggest, "using the quantile-based action-selection implicitly adopts the idea of utility function.", whose role is played by the high-level policy Q_Ω in QUOTA. Compared to classical approaches where a utility function is specified in advance, in QUOTA Q_Ω is automatically learned from experience samples.

2) A Bayesian Perspective on DRL. This paragraph discusses a new exploration technique recently proposed in Clements et al. [2019]. In this work, a Bayesian approach is combined with the QR-DQN algorithm to decouple epistemic and aleatoric uncertainty. In particular, the quantile regression loss, given by:

$$\mathscr{L}(\boldsymbol{\theta}) = \mathbb{E}_{z \sim Z(x,a)} \left[\sum_{i=1}^{N} \rho_{\tau_i} \left(z - \theta_i(x,a) \right) \right], \text{ where } \quad \rho_{\tau_i}(u) = u \left(\tau_i - \delta_{\{u<0\}} \right) \tag{3.31}$$

can be viewed as the negative log-likelihood of a dataset D consisting of K samples $(z_1, ..., z_K)$ drawn form a distribution $Z(x,a)$ and N given quantile estimates[11]:

$$P(D|\boldsymbol{\theta}) = C \exp \left(-\frac{1}{K} \sum_{j=1}^{K} \sum_{i=1}^{N} \rho_{\tau_i} \left(z_j - \theta_i(x,a) \right) \right) = C \exp(-\mathscr{L}(\boldsymbol{\theta})), \tag{3.32}$$

[11] Interestingly, in the paper it is shown that the quantile loss can be related to a likelihood function based on the asymmetric Laplace distribution.

where $C = \prod_{i=1}^{N} \tau_i(1-\tau_i)$.
A Bayesian approach to uncertainty estimation requires to specify a prior $P(\boldsymbol{\theta})$ over quantiles. The product of this prior by the likelihood defined in Eq. 3.32, $P(D|\boldsymbol{\theta})P(\boldsymbol{\theta})$, is proportional to the posterior distribution over quantiles.

However, if the quantiles are approximated by a neural network, they will be parametrized by the set of weights and biases of the network, that we indicate as $\boldsymbol{\psi}$. A popular choice of the prior over the network parameters is given by a Gaussian distribution with zero mean and variance $\sigma^2_{\boldsymbol{\psi}}$. With this choice of prior, the posterior distribution can now be written as:

$$P(\boldsymbol{\theta}(\boldsymbol{\psi})|D) \propto \exp\left(-\mathscr{L}(\boldsymbol{\theta}(\boldsymbol{\psi})) - \sigma^2 \|\boldsymbol{\psi}\|^2\right) \tag{3.33}$$

where $\sigma = \sigma_{\boldsymbol{\theta}}/\sigma_{\boldsymbol{\psi}}$, and $\sigma_{\boldsymbol{\theta}}$ is a parameter regulating the relative magnitudes of quantile and network parameters. The framework introduced above allows us to decouple parametric and intrinsic uncertainty. In particular, the first is estimated by the variance of the posterior distribution, whereas the second is quantified by the variance of the return distribution.

The approach to exploration proposed in Clements et al. [2019] is mainly based on Thompson sampling (Thompson [1933]) and consists in the following steps:

- Estimate the mean $\mu_{Z,a}$ and the variance $\sigma^2_{Z,a}$ of the posterior distribution for each action a. This step is based on a modified version of the "anchored neural network" approach used in Pearce et al. [2018].
- Draw a sample $\hat{Q}_a$ from $\mathscr{N}\left(\mu_{Z,a}, \sigma^2_{Z,a}\right)$
- Select an action: $\text{argmax}_a\left[\hat{Q}_a\right]$

The technique described above has been tested on the CartPole environment[12] and a comparison with the classical ε-greedy strategy has been drawn. In particular, the exploration ability of the two strategies has been assessed by starting the game from different initial positions. The idea is that a good exploration should guarantee better generalisation performances. The result is that the Bayesian approach based on DRL described in Clements et al. [2019] generalizes better that the simple ε-greedy strategy, as shown in Fig. 3.5.

[12] For a complete description of this environment, please refer to the relative OpenAI webpage.

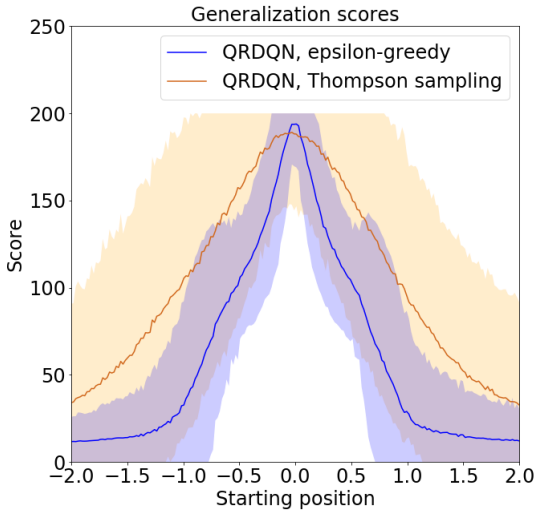


Fig. 3.5 QR-DQN with Thompson sampling generalizes better than QR-DQN with ε-greedy exploration strategy (Clements et al. [2019]).

The main shortcoming of this work is that, although the proposed method seems promising, there is not an obvious way to replicate the same generalisation experiment described before on the Atari games. In the absence of an explicit assessment of the behaviour of the algorithm in more challenging environments is not possible to demonstrate its efficacy.

3) Information Directed Exploration. Common approaches to exploration do not take into account an important aspect that plays a major role in RL: observation noise is rarely uniformly distributed and often depends on the evaluation point. In other words, in RL observation noise is *heteroscedastic*. This means that the variance of the return distribution Z is a function of states and actions.

In Nikolov et al. [2018], an interesting method that copes with heteroscedasticity in RL is introduced. This new approach combines ideas from DRL and *Information-Directed Sampling* (IDS) (Kirschner and Krause [2019]), a framework for designing exploration-exploitation strategies based on the ratio between regret and information gain.

More formally, at each time-step t, the IDS policy is defined as:

$$a_t^{\text{IDS}}(x) \in \arg\min_{a \in \mathscr{A}} \frac{\hat{\Delta}_t^{\pi}(x,a)^2}{I_t(x,a)} \tag{3.34}$$

where $I_t(x,a)$ is an arbitrary *information gain function* and $\hat{\Delta}_t^\pi(x,a)$ is a conservative estimate of the instantaneous regret[13]:

$$\Delta_t^\pi(x,a) := \mathbb{E}_P\left[\max_{a'} Q^\pi\left(x,a'\right) - Q_t^\pi(x,a)|\mathscr{F}_{t-1}\right] \tag{3.35}$$

with $\mathscr{F}_t = \{x_1, a_1, r_1, \ldots x_t, a_t, r_t\}$ being the observation history at time t. In order to explicitly compute the regret-information ratio defined in Eq. 3.34, we need to introduce a meaningful approximation, $\hat{\Delta}_t^\pi(x,a)$, of the instantaneous regret in Eq. 3.35 and we must specify a particular information gain function.

In Nikolov et al. [2018], an approximation of the instantaneous regret based on confidence intervals is proposed:

$$\hat{\Delta}_t^\pi(x,a) = \max_{a' \in \mathscr{A}} \left(\mu_t\left(x,a'\right) + \lambda_t \sigma_t\left(x,a'\right)\right) - \left(\mu_t(x,a) - \lambda_t \sigma_t(x,a)\right) \tag{3.36}$$

where λ_t is a scaling hyperparameter while μ and σ are obtained by computing the empirical mean and variance of the Q-values estimated by an ensemble of K action-value functions, following the approach of Osband et al. [2016a]:

$$\mu(x,a) = \frac{1}{K}\sum_{k=1}^{K} Q_k(x,a), \quad \sigma(x,a)^2 = \frac{1}{K}\sum_{k=1}^{K}\left(Q_k(x,a) - \mu(x,a)\right)^2 \tag{3.37}$$

Finally, in Nikolov et al. [2018] the following information gain function is proposed:

$$I_t(x,a) = \log\left(1 + \sigma_t(x,a)^2/\rho_t(x,a)^2\right) \tag{3.38}$$

where $\rho(x,a)^2$ is equal to the variance of the return distribution, i.e. $Var(Z(x,a))$. Given a state x, the information gain function defined in 3.38 is small for actions with little uncertainty in the estimated Q-values or with reward characterised by high observation noise.

From an algorithmic standpoint, the method proposed in Nikolov et al. [2018] estimates epistemic and aleatoric uncertainties by splitting the DQN architecture into $K+1$ heads after the convolutional modules: the first K heads are used to compute the mean and the variance of the Q-values (Eq. 3.37), while the remaining one outputs the estimate of $Z(x,a)$ that is in turn used to compute $\rho(x,a)^2$. In Nikolov et al. [2018], this last head is trained with C51 and

[13] This quantity can not be computed directly since it depends on the true value function, Q^π, that is not available in practice.

for this reason the resulting algorithm is called C51-IDS.

The results of the experiments on the Atari games demonstrate that C51-IDS outperforms both C51 and QR-DQN in terms of mean and median of best human-normalised scores. Furthermore, C51-IDS provides a significant performance improvement over its homoscedastic counterpart, called DQN-IDS, which simply sets $\rho(x,a)^2$ to a constant. This last observation gives further evidence of the beneficial empirical impact of a distributional approach in the context of deep RL.

The approach described in this section represents a convincing demonstration that the return distribution provided by DRL algorithms can be efficiently exploited to model uncertainty in RL problems. Moreover, this method is particularly interesting because it explicitly builds its policy (Eq. 3.34) by including information on both the parametric uncertainty (that is considered in Eq. 3.37 by averaging multiple predictions of the Q-function) and the intrinsic uncertainty (modelled by the return distribution and encoded in the term $\rho(x,a)$ used in Eq. 3.38).

3.4.2 DRL: Performance Optimization

Deep RL is a well established and growing research field. DQN (Mnih et al. [2015]) is probably the most popular deep RL algorithm and since its first publication, many independent extensions have been proposed to further enhance its performances. In Hessel et al. [2017], the most successful improvements to the DQN baseline are combined together and their relative contribution in boosting performances is measured to establish their effective importance. The resulting algorithm is called *Rainbow* and provides the current state-of-the-art results on the Atari games suite. A comparison of the performances of Rainbow and other deep RL and DRL methods is shown in Table 3.2.

Six extension have been taken into account to devise the Rainbow algorithm: Double Q-Learning (Hasselt et al. [2016]), Prioritised Replay (Schaul et al. [2016]), Dueling Networks (Wang et al. [2016]), Multi-Step Learning (Sutton [1988]), Noisy Nets (Fortunato et al. [2017]) and C51 (Bellemare et al. [2017]).

When analysing the single contribution of each of the aforementioned techniques to the final performance, it turns out the three most important methods are Prioritised Replay, Multi-Step Learning and C51. In particular, the distributional approach results in being particularly important in those games where Rainbow performed at the human level or above. This last

observation can be interpreted as a further proof of the positive impact of a distributional approach on RL.

	Mean	**Median**
DQN	228%	79%
DDQN	307%	118%
DUEL.	373%	151%
PRIOR.	434%	124%
PRIOR. DUEL.	592%	172%
*C*51	701%	178%
$QR-DQN$	864%	193%
RAINBOW	1189%	230%

Table 3.2 Updated version of Tab. 3.1, including the performances of Rainbow under the same training conditions.

3.4.3 Continuous Action Space

In this section, we discuss an example where the DRL framework is applied to the continuous control setting.
The design of efficient algorithms that can solve complicated tasks in high-dimensional state and action spaces represents a crucial challenge for modern AI. Such techniques could indeed find numerous applications in several complex real-world scenarios. In robotics, for instance, dealing with continuous action spaces is key.

The great success of DQN (Mnih et al. [2015]) lays on its capacity of implementing an efficient decision-making pipeline through a direct interpretation of real-world visual stimuli. However, deep RL algorithms such as DQN are often limited to discrete and finite action spaces.

In this section, we briefly present a relatively recent algorithm called Distributed Distributional Deterministic Policy Gradient (D4PG) (Barth-Maron et al. [2019]), that builds on the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al. [2015]) we introduced in Chapter 2 to provide state-of-the-art performances in multiple continuous control tasks such as manipulations tasks and hard obstacle-based locomotion tasks.

D4PG introduces a number of refinements to DDPG to boost its performances and, in Barth-Maron et al. [2019], an analysis of the individual contribution of each of these improvements is carried out. In particular, prioritised experience replay, N-step returns and the distributional framework (C51) are incorporated into the DDPG baseline architecture.

From the ablation of the aforementioned individual components, it is possible to assess which of them contributes most significantly to the final performances. It turns out that the combination of DDPG with C51 is beneficial in the vast majority of control tasks considered in Barth-Maron et al. [2019]. Additionally, the inclusion of the distributional framework results in being particularly important in the hardest tasks where it significantly enhances performances.

Chapter 4

Experiments and Future Directions

In the last chapter, we introduced the motivation and the logic behind DRL along with an overview of the most recent works in the field.
The goal of this chapter is to present a number of experiments aimed at supporting some of the considerations reported in Chapter 3 and discuss a number of research scenarios where we believe DRL could be fruitfully applied.

We start from an analysis of the two most popular distributional algorithms, namely C51 and QR-DQN. Specifically, we compare their performances with those of DQN on three OpenAI gym environments and we analyse their output return distributions.

Next, we propose a simple exploration method to cope with the problem of sample-inefficiency affecting model-free RL algorithms. In particular, we present an extension of the model-based approach proposed in Gou and Liu [2019], based on the Kernel Density Estimation technique.

In the last part of the chapter, we discuss three future research directions, with a particular focus on presenting our findings on the potential application of DRL to the financial field.

4.1 The Return Distribution

In this section, we present a comparison between C51, QR-DQN and DQN in terms of performances on three popular OpenAI gym environments, namely Mountain-Car, CartPole and Acrobot. Finally, we present some illustrations of the return distribution provided by C51 and QR-DQN when applied to the Cartpole and the Mountain-Car environments.

4.1.1 Performance Comparison

Figs. 4.1, 4.2, 4.3, show a comparison of the performances of DQN, C51 and QR-DQN. All the environments used for these experiments are characterised by finite-dimensional state and action spaces, as reported in Tab. 4.1.

	State-Space	**Action Space**
CartPole	4	2
Acrobot	6	3
MountainCar	2	3

Table 4.1 State and action spaces dimensions of the three environments under consideration.

Since the state is not represented by images as it was for the Atari Games, in the architectures used for these experiments we substituted the convolutional modules of the original algorithms with standard feed-forward fully-connected layers. In particular, for each experiment we used the same architecture for all the three algorithms[1].

Specifically, we used a three-layer neural network with 256 units in the hidden layer. The Adam optimiser (Kingma and Ba [2014]) was chosen to train the network and the learning rate providing the best performances was selected individually for each algorithm. A mini-batch size of 32 was used and the target network was updated every 10 episodes for all the experiments. Finally, the number of quantiles and atoms was set equal to 51 and the same ε-greedy exploration strategy was used for each algorithm.

The results in Figs. 4.1, 4.2, 4.3 have been obtained from 30 independent runs of each algorithm. Solid lines are used to indicate the mean and shaded regions represent the confidence region.

[1]The three algorithms are characterised by different final layers, as described in Chapter 2 and Chapter 3.

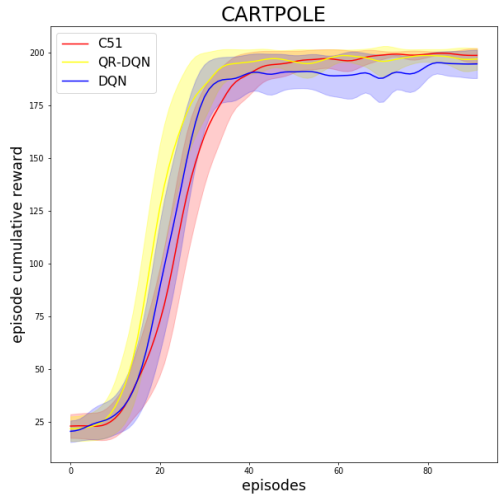
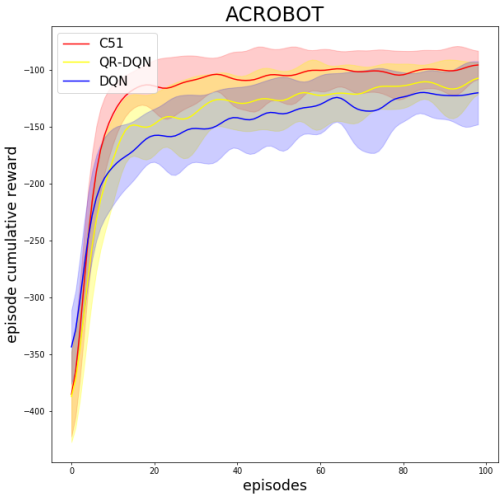


Fig. 4.1 (Left) Running Average of the episode cumulative rewards obtained by C51, QR-DQN, DQN on (Right) the CartPole environment. The learning rates used for C51, QR-DQN and DQN are $10^{-3}, 10^{-2}$ and 10^{-2} respectively. A complete description of this environment can be found here.



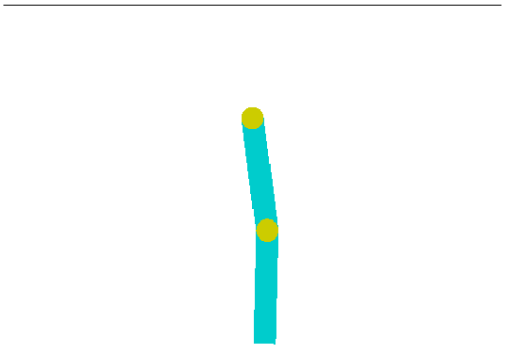
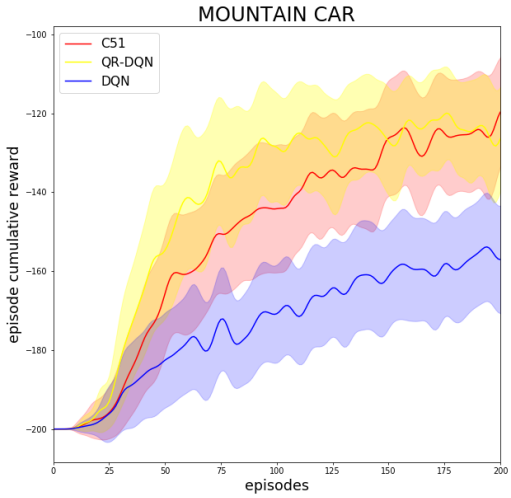


Fig. 4.2 (Left) Running Average of the episode cumulative rewards obtained by C51, QR-DQN, DQN on (Right) the AcroBot environment. The learning rates used for C51, QR-DQN and DQN are $10^{-3}, 10^{-2}$ and 10^{-3} respectively. A complete description of this environment can be found here.



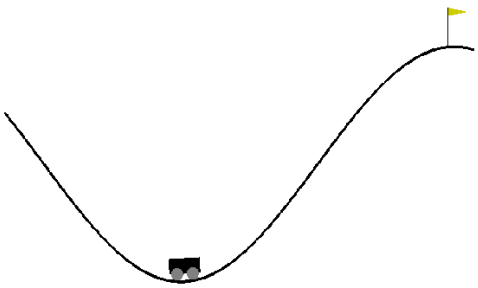


Fig. 4.3 (Left) Running Average of the episode cumulative rewards obtained by C51, QR-DQN, DQN on (Right) the MountainCar environment. The learning rates of used for C51, QR-DQN and DQN are $10^{-3}, 10^{-2}$ and 10^{-2} respectively. A complete description of this environment can be found here.

The empirical evaluation reported in this section shows that in general, the distributional algorithms provide better performances than DQN, although they converge to the same final score on CartPole and Acrobot. On MountainCar, the difference in performance is more distinct even if, also in this case , the three algorithms converge to the same final value after many iterations. These results are in agreement with the experimental findings reported in Bellemare et al. [2017]; Dabney et al. [2018b]; Lyle et al. [2019].

The reason why DRL algorithms provide better performances is still largely unaddressed by the works in the literature. The current working hypothesis in the community is that better results are induced by an *auxiliary task effect* (Jaderberg et al. [2016], Francois-Lavet et al. [2018]; Lyle et al. [2019]). Specifically, the return distribution provides a richer set of predictions than the single value function. These additional training signals lead to better performances despite not being directly necessary for the maximisation of the expected return.

We believe that the idea of including auxiliary tasks into the RL problem under consideration can be interpreted as taking into account a number of *additional constraints* that must be satisfied throughout the learning process. The change of perspective from strictly value-based to distributional algorithms can be viewed as the addition of the aforementioned types of

constraints that enrich the set of training signals available to the agent.

However, including auxiliary tasks often requires a significant amount of tuning to make sure they do not conflict with the main task (Jaderberg et al. [2016]). In DRL, this does not seem to be the case and this could be due to some special properties of distributional algorithms. A deeper comprehension of these properties could lead to the design of even better auxiliary tasks.

4.1.2 Return Distribution Visualisation

In this section, we provide a number of visualisations of the return distribution provided by C51 and QR-DQN. Fig. 4.4 illustrates some examples of the output of the algorithms at four different time-steps.

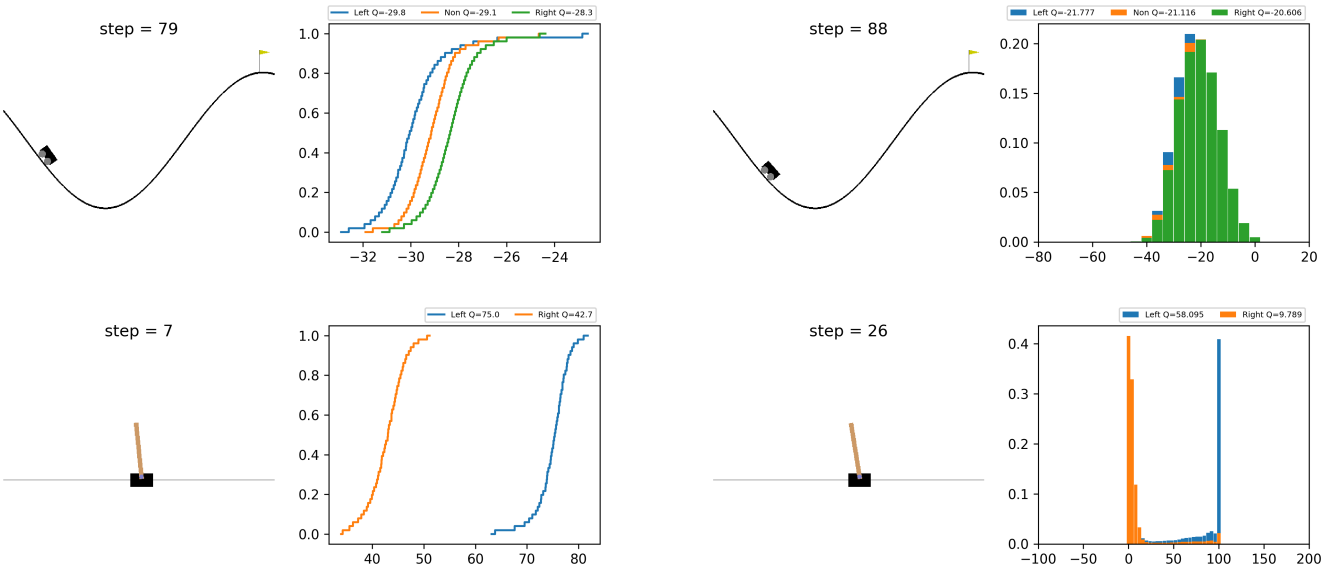


Fig. 4.4 The two different parametrisations of the return distributions associated with each action, provided by C51 and QR-DQN on the CartPole and MountainCar environments. The plots to the left show the cumulative distribution function (y-axis) associated with the return values (x-axis) output by QR-DQN. The plots to the right show the output probabilities provided by the C51 algorithm (y-axis) as a function of the return values (x-axis). Each colour corresponds to a different action.

The first and second rows in Fig. 4.4 describe the output of QR-DQN and C51 associated with two similar input states of the MountainCar and the CartPole environments, respectively. The two algorithms have similar behaviours, indeed they both select the same action (“right”

for the first row and "left" for the second) as the best one in order to achieve the largest expected return from that state.

However, the output of C51 tends to overestimate the variance of the return distribution. This is a well-known problem of C51 and has already been highlighted by the experiments on the Atari games performed in Bellemare et al. [2017] and by the theoretical considerations developed in Rowland et al. [2019].
In particular, this behaviour can be viewed as a side-effect of the action of the projection operator Φ (see Eq. 3.11), which spreads probability mass across the space of returns. Bellemare et al. [2017] interpret this effect as a consequence of "the discretization of the probability diffusion process induced by γ".
We believe that this bias introduced by the C51 algorithm could be potentially harmful in those applications that aim to design risk-aware policies or estimating risk and uncertainty.

Fig. 4.5 shows an interesting aspect of the return distribution provided by the C51 algorithm. Specifically, in the case illustrated in the left image, the car does not have enough velocity to reach the top of the hill and this is reflected by the second mode of the return distribution associated with smaller values. The shape of the return distribution in this case highlights the agent's uncertainty about the outcome of the episode.

On the other hand, in the case illustrated in the right image, the agent has enough speed to climb the hill and this results in a more peaked distribution reflecting the agent's belief of successfully terminating the episode.

This example is particularly interesting since the MountainCar environment is purely deterministic given that the state-transition function is entirely determined by the physics of the problem. The multimodality in the return distribution showed in Fig. 4.5 captures the agent's uncertainty resulting from an incomplete knowledge of the environment.

Furthermore[2], in C51, we perform the policy improvement step as though the evaluation is complete even if we have only done a small number of updates. This means that the distribution shifts towards the potentially deterministic return for a policy, but then the policy moves on ahead before it converges. This induces a non-stationary target that looks like a non-deterministic signal. As the policy converges, the distribution moves more consistently towards a single (now stationary) target and this effect disappears.

[2]We thank Will Dabney for this useful hint.

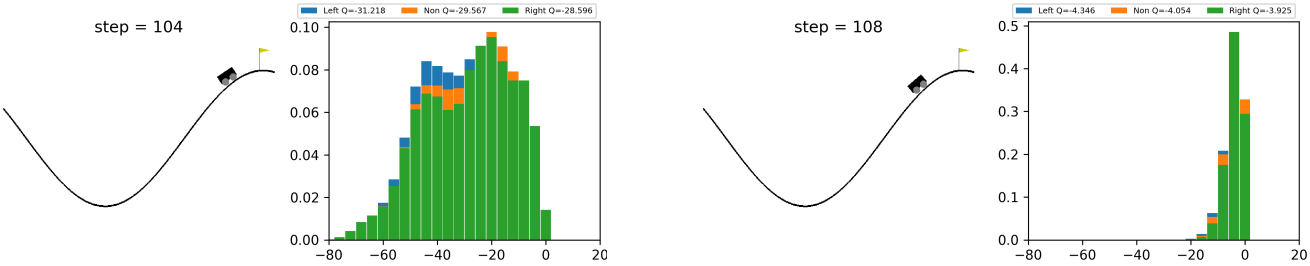


Fig. 4.5 Comparison of the return distributions associated with two different scenarios in the MountainCar environment.

4.2 Model-Based Exploration

As mentioned in Chapter 2 and Chapter 3, the major drawback of model-free RL methods and hence of all the existing distributional algorithms, lays in their sample-inefficiency, i.e. the large number of agent-environment interactions they require to converge. A possible way to improve this aspect is to implement better exploration strategies that can allow the agent to get a better knowledge of the environment by selecting actions that lead to previously unexplored states.

On the other hand, DQN, C51 and QR-DQN make use of an ε-greedy exploration strategy based on choosing a random action with probability ε, where ε is a decreasing function of the number of iterations. This method can be particularly inefficient (Osband et al. [2016b]), especially on environments characterised by sparse rewards (Gou and Liu [2019]).

In this section, we propose a variation of the model-based exploration method described in Section 2.2.1. In our experiments, we used a feed-forward neural network characterised by two 256-units hidden layers to model the environment dynamics. The input of the network is a state-action pair (x,a), while its output is given by the next-state prediction x'. The effectiveness of deep neural networks to describe the model dynamics has been previously demonstrated for example in Nagabandi et al. [2018], where the authors claim that: "multi-layer neural network models can in fact achieve excellent sample-complexity in a model-based reinforcement learning algorithm".

In order to drive exploration, we modify Eq. 2.8 by replacing the Gaussian assumption on the state distribution with the model provided by a non parametric density estimation technique called *Kernel Density Estimation*[3], KDE for short. The advantage of this method over the original one is that it is more flexible and can better adapt to the a-priori unknown shape of the state distribution.

Fig. 4.6 shows the states visited by the agent over 50 episodes of pure exploration ($\varepsilon = 1$) based on the standard ε-greedy method (left) and on the KDE model-based exploration method.

[3]For more details about this method, please refer to Appendix A

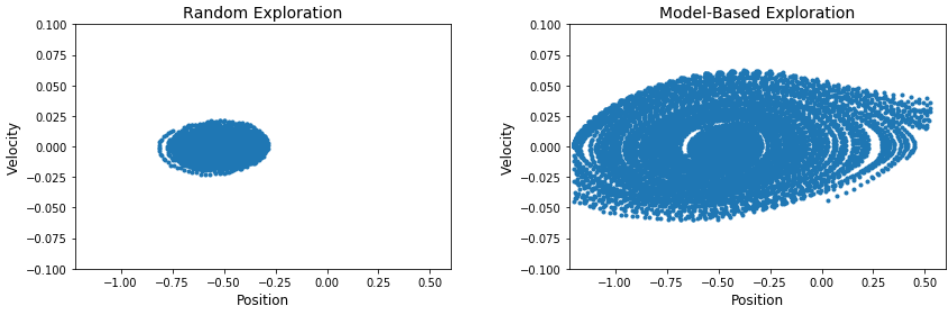


Fig. 4.6 Comparison between standard ε-greedy exploration (left) and model-based exploration (right).

As shown above, the model-based exploration strategy seems to be much more effective than the standard technique based on random actions.
Fig. 4.7 reports a comparison in terms of performances of the C51 agent with the two types of exploration strategies under consideration, showing that the exploitation of the knowledge of the environment gained with the model-based approach leads to faster convergence than the case when the standard exploration technique is used.

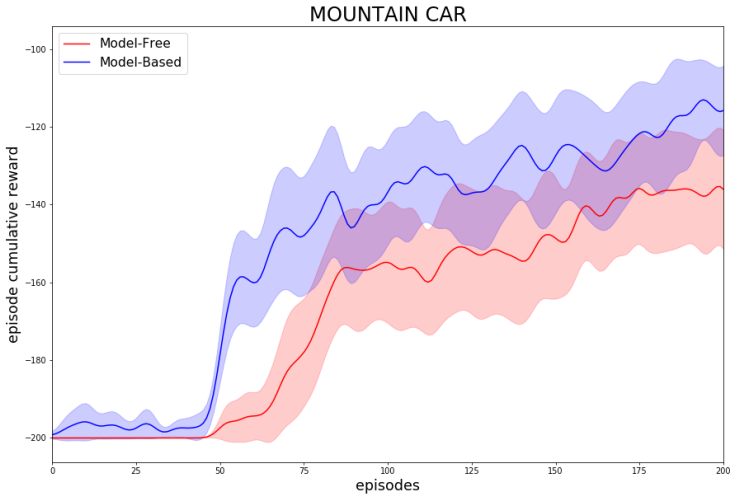


Fig. 4.7 Performance improvement of the C51 agent with model-based exploration over the C51 agent with ε-greedy exploration. These results are obtained by forcing the agent to explore for the first 50 episodes ($\varepsilon = 1$) and then fixing $\varepsilon = 0.01$.

The application of the KDE-based exploration method seems to be effective also on the Acrobot environment which is characterised by a higher-dimensional state-space than Moun-

tainCar. In particular, Fig. 4.8 shows a comparison between the regions of the state space (first two dimensions) visited by the agent after 50 episodes of pure exploration.

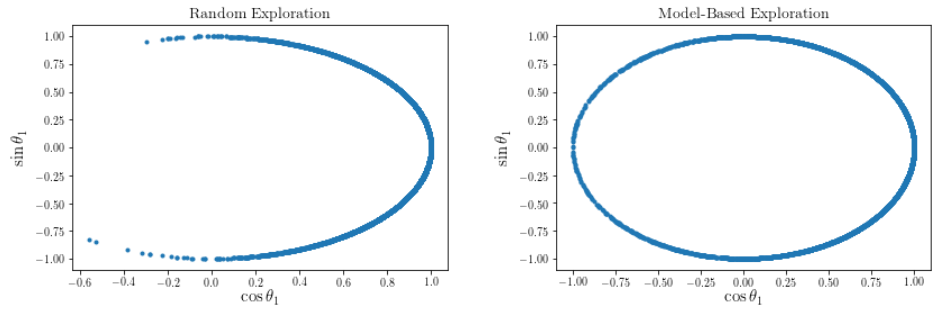


Fig. 4.8 Comparison between standard ε-greedy exploration (left) and model-based exploration (right) on the Acrobot environment. The first two dimensions reported above are the sine and cosine of the first rotational joint angle, θ_1 (measured w.r.t the vertical axis, see Fig. 4.9).

Again, the proposed model-based approach seems to provide better results in terms of exploration than the standard one. In particular, with the ε-greedy exploration strategy, the first link rarely exceeds the horizontal axis passing through the first joint (see Fig. 4.9).

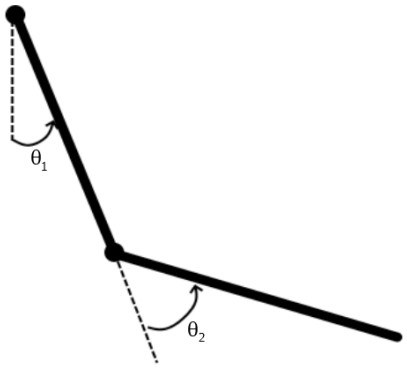


Fig. 4.9 A schematic model of the Acrobot environment (Sutton [1996]).

A possible interpretation of the better performances provided by this method could be that it represents an *active* exploration technique in contrast to the standard method based on random action that can be viewed as a *reactive* exploration approach. As explained in Shyam et al. [2019], the difference between reactive and active methods is that the former

"accidentally observe something novel and then decide to obtain more information about it", whereas the latter actively perform the action that they believe could lead to unexplored states.

Although the method proposed in this section seems to outperform the standard exploration strategy used by DQN, C51 and QR-DQN, further experiments on more challenging control environments need to be carried out to assess its real value. Furthermore, when the complexity of environments increases, it might be useful to consider different neural network architectures to model the dynamics. For example, as suggested by Nagabandi et al. [2018], instead of considering the next state, x_{t+1}, as the output of the network, a better strategy could be to model the difference, $x_{t+1} - x_t$ between two consecutive states. This could be advantageous when different actions result in very similar transitions, especially when the time interval between two steps of the MDP is small.

As a final remark, it is worth mentioning that the specific type of exploration strategy can actually depend on the characteristics of the particular environment under consideration (e.g. sparse rewards, high-dimensional state and action spaces, etc..).

.

4.3 Future Work

In this section, we propose three future research directions where we believe distributional algorithms could be successfully employed. Specifically, we focus on the combination of DRL with existing model-based approaches, the design of a new algorithm capable of potentially improving performances on the Atari games and finally, we briefly discuss the application of distributional algorithms to the financial setting.

Model-Based Approaches. All the distributional algorithms proposed so far are model-free and therefore they suffer from sample-inefficiency. Since DRL methods seem to provide very good results in terms of learning performances, a new natural research direction consists in finding a way to reduce the number of interactions with the environment they need to converge. The intersection between model-based and model-free techniques seems to be a very interesting avenue for potentially combining the sample-efficiency of the former and the learning performances of the latter.

One possible research direction is to further investigate whether model-based methods such as the one proposed in Section 4.2 can be valuable options to enhance exploration of model-free

algorithms. In practice, new experiments on more complex environments can be performed to further assess the value of the KDE-based technique used in Section 4.2 and different non-parametric density estimation techniques can be investigated. Moreover, we expect that, as the state-dimensionality grows, more advanced neural network architectures will be needed to fit the transition dynamics.

Beside the design of better exploration strategies, model-based approaches can be applied following the paradigm introduced with the Dyna algorithm (Sutton [1991]). We believe that a very interesting future research direction could be to explore how to combine DRL techniques with model-based algorithms, such as the SimPLe algorithm (Kaiser et al. [2019]) described in Section 2.2.2, so that experience samples can be simulated and efficient strategies can be planned directly from the learned model of the environment. On the other hand, a limitation of this approach lays in its potential high computational load. SimPLe makes use of a complex architecture to model the environment and its combination with DRL algorithms could be harmful from this perspective. Interestingly, the return distribution is closely related to the randomness of outcome in the model-based transition function. Therefore, DRL algorithms could actively cooperate with the model-based component to build the world-model.

Performance Optimization. DRL algorithms provide state-of-the-art performances on several challenging RL environments. In particular, as explained in Section 3.4.2, the Rainbow algorithm combines several successful deep RL techniques in order to optimise the final results on the Atari games. An important component of Rainbow is the C51 algorithm that significantly contributes to the algorithm final performances.

An interesting subject of future work could be to modify the Rainbow agent by replacing C51 with QR-DQN in light of the better results obtained by the latter compared to the first (see Tab. 3.2). This modification has the potential of further improving performances on the Atari games, leading to new state-of-the-art results.

Applications to Finance. The application of ML techniques to finance is a very active research area (Atsalakis and Valavanis [2009]; Emerson et al. [2019]; Fischer [2018]; Li [2018]). However, the attention of the scientific community is mainly focused on SL methods (Atsalakis and Valavanis [2009]), typically based on predicting financial assets or future returns by using DL algorithms (Fischer [2018]).

The application of RL, and specifically of deep RL, to trading financial assets and to the portfolio optimisation problem has witnessed an increasing interest in the last few years. As pointed out by Fischer [2018], deep RL has the potential to overcome some limitations of the SL setting that can deteriorate its final performances. In particular, deep RL combines the prediction of future asset's prices and the actual trading decision problem into a unique step, whereas classical ML approaches separate these two processes, leading to a series of detrimental effects (Fischer [2018]; Hegde et al. [2018]).

The majority of the deep RL methods applied to trading are critic-only (e.g. Huang [2018]) (value-based) and actor-only (e.g. Deng et al. [2017]) although a number of very recent works are based on actor-critic algorithms such as DDPG (Hegde et al. [2018]; Liang et al. [2018]; Xiong et al. [2018]) since these methods have the potential to combine the advantages of the first two approaches. An interesting feature of actor-critic methods in the context of finance is that they allow the agent to trade a potentially large number of assets or, more generally, to perform many diverse trading decisions. Indeed, as mentioned in Section 2.6, methods like DDPG, as opposed to value-based techniques like DQN, can be used to handle problems characterised by high-dimensional action spaces.
The first potential contribution of DRL to finance could be to replace DDPG with D4PG in light of its state-of-the-art performances on challenging control tasks (Barth-Maron et al. [2019]; Huang et al. [2019]). The application of D4PG in this context would consists in a relatively straightforward extension of the existing works based on DDPG whose results could be used as a benchmark for comparing performances.

As explained by Li [2018] and Fischer [2018], taking into account risk is crucial in portfolio construction. Many interesting methods have been introduced in the literature to cope with this important task. For instance, in Ritter [2017], it is shown that by choosing a particular reward function, it is possible to apply the Q-learning algorithm in such a way that a risk-averse trading strategy is followed.
However, most of the proposed deep RL-based methods in the literature are focused on maximising the expected value of some notion of return depending on the problem under consideration. As suggested by Huang [2018], an alternative approach could be to choose the action greedily with respect to the ratio between the expected return and its standard deviation, i.e. the so-called *Sharpe Ratio* (Sharpe [1994]). The standard deviation of the return could be directly estimated by a DRL algorithm such as QR-DQN (in order to avoid the variance overestimation problem introduced by C51). A similar idea has been proposed by Stanko [2018], where instead a risk-averse policy is learned by maximising the so-called

Conditional Value-at-Risk (CVaR), one of the most widely used parameters to measure risk in finance.
Furthermore, having access to the return distribution could be insightful for understanding if any interesting pattern emerges under certain market conditions. In principle, given the high level of stochasticity inherently present in the financial domain, we expect significantly more complex distributions compared to those obtained from the deterministic environments of the Atari games.

Chapter 5

Conclusions

In this thesis, we have described the most important features of DRL, a relatively new framework based on a more general approach than that used by classic RL. The main ingredient of this new formulation is the distributional Bellman's equation which focuses on the entire return distribution instead of its expected value. The basic idea is that the distributional Bellman's equation could potentially provide a more consistent description from a probabilistic point of view than its "classical" counterpart by taking into account all the additional constraints introduced by the higher-order moments of the return distribution.

To the best of our knowledge, this work represents the first extensive review of the most relevant achievements reached in the field of DRL in the last few years. Since the literature in this branch is quite sparse and very little connected, we believe that this project could be a valuable resource for anyone interested in working on DRL.

We strongly believe that DRL is a very promising scientific area in light of the results obtained by the existing techniques. However, as mentioned in the previous chapters, there are still many questions that need to be addressed.

From a theoretical point of view, several advances have been made in the last few years, although a mathematically principled justification of the positive impact of the distributional approach on learning performances is still lacking. The interpretation based on auxiliary tasks, although plausible and object of an active research area, is still scarcely supported by robust theoretical arguments.

As a conclusive remark, we believe that DRL naturally lends itself to applications to those fields where uncertainty and risk estimation leads to better performances. As shown in

Chapter 3, several recent works are pointing in this direction. However, the application of these methods to real-world problems is still strongly limited by their model-free nature. As discussed in Chapter 4, a hybrid approach including both model-free and model-based components could potentially bridge this gap.

Bibliography

Each article is associated with a short annotation, and citations are grouped into topical sections. Note that, a small part of the works reported below are not referenced in the text but have been included for the sake of completeness.

Bibliography I: Reinforcement Learning

Bellman, R. (1954). The theory of dynamic programming. *Bull. Amer. Math. Soc.*, pages 503–515.

> *The paper presents a pioneering presentation of the field of dynamic programming. Of particular note is the introduction where the author highlights that solving multi-stage decision processes does not necessarily require to know all the possible sequences of decisions but, under certain conditions, the decision at a certain stage can be taken relatively to only the current state of the system.*

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, 1st edition.

> *A beautiful RL textbook, characterised by a more mathematical and formal approach than other introductions to RL.*

Deisenroth, M. P. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11.

> *This paper presents PILCO, a model-based policy search method capable of achieving state-of-the-art results in terms of sample-efficiency on environments characterised by continuous state and action spaces. PILCO uses non-parametric Gaussian Processes to learn a probabilistic model of the dynamics. Despite its excellent performances, PILCO results in being impractical for high-dimensional problems due to its significant computational cost.*

Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems. Technical report.

In this paper, the SARSA algorithm is introduced. Additionally, the use of back-propagation-based neural networks as value function approximators is proposed, in contrast to classical tabular techniques.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*.

Pionering work that introduces various TD learning techniques in the context of RL.

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4).

Introduces Dyna, a very popular model-based algorithm that uses the learned dynamic model to plan future actions without directly interacting with the environment, thus improving sample-efficiency. Several modern model-based algorithms rely on this idea.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press.

The paper discusses the use of parametrised function approximators in RL. In particular, sparse-coarse-coded function approximators are introduced and applied to a number of continuous control tasks.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

One of the most popular and successful introductions to RL.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181 – 211.

In this work, the "option-framework" is introduced. In particular, the authors highlight the potential of this method in the context of model-based RL and specifically for the long-term planning problem.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*.

The paper introduces Q-Learning and provides a proof that the algorithm converges under certain conditions.

Bibliography II: Deep Reinforcement Learning

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. (2019). Distributed distributional deterministic policy gradients.

Introduces the Distributed Distributional Deep Deterministic Policy Gradient algorithm, D4PG for short. This new technique combines some of the most successful advances in the deep RL literature along with a distributional approach of the same kind as that used for the C51 algorithm. D4PG is designed for continuous control tasks and, in this context, achieves state-of-the-art performances on several challenging manipulation and locomotion tasks. Ablation studies on the various techniques used by the algorithm reveal that the distributional approach is one of the most effective.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47.

Introduces ALE (Arcade Learning Environment), a widely used software platform designed to test the performances of RL agents. ALE provides an interface with a large number of Atari games, specifically meant to be challenging for human players.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

OpenAI Gym is an evolving platform of simulated environments, specifically designed for the evaluation of RL algorithms.

C. Machado, M., G. Bellemare, M., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents (extended abstract). *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

This work reviews the most significant applications of RL algorithms to the Arcade Learning Environment (ALE) platform.

Deisenroth, M. P. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11.

This paper presents PILCO, a model-based policy search method capable of achieving state-of-the-art results in terms of sample-efficiency on environments characterised by continuous state and action spaces. PILCO uses non-parametric Gaussian Processes to learn a probabilistic model of the dynamics. Despite its excellent performances, PILCO results in being impractical for high-dimensional problems due to its significant computational cost.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2017). Noisy networks for exploration.

Introduces a new deep RL agent, named "NoisyNet", that guarantees better exploration performances than classical heuristics, such as epsilon-greedy or entropy reward. This method is mainly based on introducing a certain amount of parametric noise into the weights of the network. This modification is not computationally demanding and yields good results in terms of exploration.

Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. (2018). An introduction to deep reinforcement learning. cite arxiv:1811.12560.

Provides a complete presentation of the main motivations behind deep RL and a review of the most important algorithms developed over the last few years.

Gou, S. Z. and Liu, Y. (2019). Dqn with model-based exploration: efficient learning on environments with sparse rewards.

A simple model-based extension to the DQN agent is presented. Exploration is enhanced by selecting actions that lead to states that are as far as possible from the last visited states according to the current transition function model.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration.

This work proposes an extension of the Q-Learning algorithm to continuous domains and combines this new method with a model-based approach that implements on-policy imagination rollouts, similarly to the Dyna algorithm. This technique is tested on a number of simulated robotics tasks and it is compared with similar existing methods.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2018). Learning Latent Dynamics for Planning from Pixels. *arXiv e-prints*.

This paper introduces one of the most successful fully model-based deep RL approaches. The proposed agent, called Deep Planning Network (PlaNet), achieves final learning results that almost match those of state-of-the-art model-free algorithms (D4PG) on simulated control tasks. PlaNet learns the environment dynamic model from images and plans future actions in the latent space. This allows the agent to drastically reduce the number of interactions with the environment, resulting in improved sample-efficiency.

Hasselt, H. v., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16.

In this work, the Double DQN (DDQN) agent is introduced. This algorithm is meant to cope with the well-known problem of value overestimation that emerges from the application of standard Q-Learning-based techniques. DDQN proposes an alternative update rule to that used by Q-Learning. This modification leads to improved stability and better performances.

Huang, C. Y. (2018). Financial Trading as a Game: A Deep Reinforcement Learning Approach. *arXiv e-prints*.

Introduces a number of modifications to the recurrent DQN agent in order to make it suitable for the financial trading setting. Interestingly, the authors suggest that DRL algorithms can be used to extend the proposed work in order to design risk-sensitive policies.

Huang, S. H., Zambelli, M., Kay, J., Martins, M. F., Tassa, Y., Pilarski, P. M., and Hadsell, R. (2019). Learning gentle object manipulation with curiosity-driven deep reinforcement learning.

This recent work describes an interesting application of the D4PG algorithm. In particular, the paper focuses on the design of "gentle" policies that can enable the agent to interact with fragile objects in a safe way. The experiments are carried out both on the MuJoCo platform and on a real robot.

Irpan, A. (2018). Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html.

An interesting blog post that highlights the main shortcomings of the current existing RL algorithms.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks.

This work provides an empirical assessment of the usefulness of auxiliary tasks in RL problems. Specifically, several additional pseudo-reward signals, such as pixel changes, are incorporated in the training procedure, leading to surprising performance gains on a series of challenging three dimensional "Labyrinth" tasks.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Sepassi, R., Tucker, G., and Michalewski, H. (2019). Model-based reinforcement learning for atari.

This work provides a new deep RL algorithm that drastically improves sample-efficiency by combining model-free and model-based approaches. In particular, a new sophisticated neural network architecture is proposed to build an environmental model and the PPO algorithm is used to learn the optimal policy. The resulting algorithm provides state-of-the-art performances on the Atari games in terms of sample-efficiency.

Li, Y. (2018). Deep reinforcement learning.

This work proposes an introduction to the most important deep RL techniques introduced in the last few years. Of particular note is the section describing

existing and potential applications of these methods to the financial and to the healthcare sectors.

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

This paper proposes the Deep Deterministic Policy Gradient (DDPG) algorithm whose main contribution is to extend the DQN algorithm to the case of continuous action spaces. DDPG is mainly based on the combination between an actor-critic method and the usage of deep networks as function approximators. Of particular note for our work is the extension of this algorithm to the distributional case, i.e, the D4PG algorithm.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning.

This paper proposes a new method for training neural network-based agents that relies on asynchronous gradient descent optimisation. Specifically, multiple agents are asynchronously run on multiple instances of the same environment. The main benefits of this approach are that it allows us to collect uncorrelated data and to run experiments on standard multi-CPU machines instead of on distributed GPU-based architectures.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., 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.

The paper introduces the DQN algorithm, effectively providing one of the first demonstrations of the potential of DL in the context of RL. The DQN agent implements a decision-making pipeline directly triggered by visual inputs, drawing inspiration from the way humans interact with the environment.

Moerland, T., Broekens, J., and Jonker, C. (2017a). Efficient exploration with double uncertain value networks.

This paper is about how to describe and exploit the parametric and the intrinsic uncertainty arising in RL problems. Epistemic uncertainty is modelled by means of a Bayesian approach, while intrinsic uncertainty is described by the return distribution. However, only the case of Gaussian return distributions is considered.

Moerland, T., Broekens, J., and Jonker, C. (2017b). Learning multimodal transition dynamics for model-based reinforcement learning.

The paper focuses on the problem of learning stochastic multimodal transition functions in order to improve the efficiency of model-based RL algorithms on stochastic domains. The proposed method is based on Variational Inference (VI) and its advantages over standard baselines are highlighted by a series of experiments on simple stochastic environments.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. (2018). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*.

This paper proposes to combine model-free and model-based approaches in order to exploit the asymptotic performance guarantees of the former and the sample-efficiency of the latter. Specifically, a model-based algorithm based on relatively shallow neural networks is used to initialise a policy gradient method. This results in similar performances to model-free algorithms and improved sample-complexity.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. (2016a). Deep exploration via bootstrapped DQN. *CoRR*, abs/1602.04621.

This paper proposes the Bootstrapped DQN agent that consists in splitting the classic DQN architecture in a number of separate heads right after the convolutional layers. Each head is associated with a different target head and all heads are trained on the same data. This method was proved to be effective in estimating parametric uncertainty and hence in improving exploration.

Osband, I., Van Roy, B., Russo, D., and Wen, Z. (2017). Deep Exploration via Randomized Value Functions. *arXiv e-prints*.

This paper introduces the idea of randomised value functions to enhance exploration in RL. Besides proving the potential of this method in terms of sample-efficiency, a review of commonly used exploration techniques is carried out and their main shortcomings are highlighted.

Osband, I., Van Roy, B., and Wen, Z. (2016b). Generalization and exploration via randomized value functions. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16.

This work proposes a new algorithm, called Randomized Least-Squares Value Iteration (RLSVI) aimed at enhancing exploration and generalisation. Furthermore, it is shown that classical approaches to exploration such as ε-greedy exploration can be significantly inefficient when combined to least squares value iteration.

Pearce, T., Anastassacos, N., Zaki, M., and Neely, A. (2018). Bayesian Inference with Anchored Ensembles of Neural Networks, and Application to Exploration in Reinforcement Learning. *arXiv e-prints*.

This paper discusses an extension to the well-established approach of using ensembles of neural networks for predictive uncertainty estimation. In particular, each network in the ensemble is regularised around a set of coordinates drawn from a Gaussian prior distribution. If the networks are trained on the same dataset, they converge to a set of parameters that can be interpreted as drawn from the posterior distribution.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. *CoRR*.

This work proposes Prioritized Experience Replay (PER), an alternative version of the experience replay technique used in DQN. In that case, experience samples were uniformly drawn from the memory buffer. This paper proposes a way to sample the most important transitions with higher frequency. When

equipped with PER, DQN outperforms its original version on 41 out of 49 Atari games.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization.

This paper introduces the Trust Region Policy Optimization (TRPO) algorithm. This technique is derived from natural policy gradient methods and represents an efficient approach for the optimisation of highly nonlinear policies, such as those parametrised by neural networks.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

This paper introduces the Proximal Policy Optimization (PPO) algorithm, a variant of the TRPO algorithm that drastically improves its performances in terms of sample-efficiency. This algorithm is one of the most widely used at OpenAI because of its relatively simple implementation and its results that are comparable to the state-of-the-art ones.

Shyam, P., Jaśkowski, W., and Gomez, F. (2019). Model-based active exploration.

This paper proposes a new algorithm, named MAX (Model-based Active eXploration) that aims to provide better exploration performances. This is achieved by designing policies based on a measure of state-novelty obtained by a Bayesian approach which estimates the level of disagreement among an ensemble of forward models. The proposed work focuses on pure exploration and MAX is not integrated with any policy learning algorithm.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14.

This paper studies the deterministic policy gradient and provides the important proof that this gradient is related to the expected gradient of the action-value function. This work provides the basis for successful algorithms such as DDPG and D4PG.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

This paper presents MuJoCo (Multi-Joint dynamics with Contact), a software platform including several simulated continuous control environments, particularly suitable for the evaluation of RL algorithm performances.

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*.

This paper provides a theoretical investigation of the properties of TD learning combined with linear and nonlinear function approximation methods. In particular, this work highlights a number of complications, such as divergence issues, arising from the usage of nonlinear approximators.

Tsividis, P., Pouncy, T., Xu, J. L., Tenenbaum, J. B., and Gershman, S. J. (2017). Human learning in atari. In *AAAI Spring Symposia*.

This paper provides an evaluation of human performances on the Atari games, showing that humans are often able to learn much more quickly than the most advanced deep RL algorithms.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1995–2003.

Introduces an extension to the DQN algorithm based on the separate estimation of the value function and the advantage function. This modification in the original DQN architecture leads to "better generalisation across actions without imposing any change to the underlying reinforcement learning algorithm." This techniques results in significant performance improvements over DQN.

Bibliography III: Distributional Reinforcement Learning

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., and Lillicrap, T. (2019). Distributed distributional deterministic policy gradients.

Introduces the Distributed Distributional Deep Deterministic Policy Gradient algorithm, D4PG for short. This new technique combines some of the most successful advances in the deep RL literature along with a distributional approach of the same kind as that used for the C51 algorithm. D4PG is designed for continuous control tasks and, in this context, achieves state-of-the-art performances on several challenging manipulation and locomotion tasks. Ablation studies on the various techniques used by the algorithm reveal that the distributional approach is one of the most effective.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. *arXiv e-prints*.

Introduces the DRL framework. It first investigates the theoretical properties of the distributional Bellman operator in the policy evaluation and control settings and then introduces C51, an algorithm based on a categorical parametrisation of the return distribution.

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2018). The cramer distance as a solution to biased wasserstein gradients. *CoRR*, abs/1705.10743.

The paper compares the properties of two widely used metrics, the KL-divergence and the Wasserstein distance, and introduces the Cramér distance that combines some advantageous properties of the first two metrics.

Bellemare, M. G., Roux, N. L., Castro, P. S., and Moitra, S. (2019). Distributional reinforcement learning with linear function approximation. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2203–2211.

Proposes a new DRL algorithm entirely based on the minimisation of the Cramér metric, motivated by the good properties of this distance measure compared to the KL divergence. Positive theoretical convergence results are reported when the linear approximation framework and the policy evaluation setting are considered. However, the proposed algorithm yields worse results than C51.

Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. (2018). Dopamine: A Research Framework for Deep Reinforcement Learning.

Dopamine is an open-source deep RL framework that provides several implementations of successful deep RL algorithms. In particular, the Rainbow algorithm is included. Dopamine is an ongoing project and is constantly updated with new tools and new algorithms extracted from the literature.

Chung, K.-J. and Sobel, M. J. (1987). Discounted mdp's: Distribution functions and exponential utility maximization. *SIAM J. Control Optim.*, 25(1).

This work reports a number of theoretical results regarding the distributional Bellman's equation. In particular, it provides a mathematical proof that the Bellman operator is not a contraction in total variation distance.

Clements, W. R., Robaglia, B.-M., Delft, B. V., Slaoui, R. B., and Toth, S. (2019). Estimating risk and uncertainty in deep reinforcement learning.

The paper provides a way to decouple epistemic and aleatoric uncertainty by using DRL. Particularly interesting is the Bayesian interpretation of QR-DQN that allows the estimation of epistemic uncertainty. The presented experiments are not particularly insightful, but the proposed ideas could provide room for further improvements.

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018a). Implicit quantile networks for distributional reinforcement learning. In *ICML*.

This paper provides an extension to the QR-DQN algorithm by modelling the full quantile function instead of just a discrete set of quantiles. This results in improved performances close to those of Rainbow, the state-of-the-art algorithm on the Atari games. Additionally, the paper discusses a number of risk-sensitive policies based on the return distribution.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018b). Distributional reinforcement learning with quantile regression.

Introduces a different way to parametrise the return distribution compared to the original C51 algorithm. The new approach is based on the minimisation of the Wasserstein distance by means of the the quantile regression method.

Dearden, R., Friedman, N., and Russell, S. (1998). Bayesian q-learning. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98.

This paper proposes a Bayesian extension to the original Q-Learning algorithm based on modelling probability distributions over Q-values. The authors suggest that this method can be used to model the parametric uncertainty of the agent about the return distribution, resulting in better exploration. Several ideas from this work have been re-elaborated under the perspective of DRL.

Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. (2018). An introduction to deep reinforcement learning. cite arxiv:1811.12560.

Provides a complete presentation of the main motivations behind deep RL and a review of the most important algorithms developed over the last few years.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning.

The paper introduces Rainbow, the algorithm providing the current state-of-the-art results on the Atari games. It incorporates the most successful advances proposed in the deep RL community over the last few years. Specifically, Rainbow integrates the distributional architecture of the C51 algorithm that provides a key contribution to achieve its final performances.

Hu, W. and Hu, J. (2019). Distributional reinforcement learning with quantum neural networks. *Intelligent Control and Automation*, 10:63–78.

The paper investigates the application of QR-DQN with a quantum neural network and compares the performances of the resulting algorithm with those of quantum Q-learning. The experiments on a simple grid-world environment demonstrate that the distributional approach is beneficial for the final results in terms of exploration efficiency.

Huang, S. H., Zambelli, M., Kay, J., Martins, M. F., Tassa, Y., Pilarski, P. M., and Hadsell, R. (2019). Learning gentle object manipulation with curiosity-driven deep reinforcement learning.

This recent work describes an interesting application of the D4PG algorithm. In particular, the paper focuses on the design of "gentle" policies that can enable the agent to interact with fragile objects in a safe way. The experiments are carried out both on the MuJoCo platform and on a real robot.

Kirschner, J. and Krause, A. (2019). Information Directed Sampling and Bandits with Heteroscedastic Noise. *arXiv e-prints*.

In this paper, a new class of methods based on the so-called regret-information ratio is introduced to cope with heteroscedastic noise in stochastic bandit problems.

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

This paper proposes the Deep Deterministic Policy Gradient (DDPG) algorithm whose main contribution is to extend the DQN algorithm to the case of continuous action spaces. DDPG is mainly based on the combination between an actor-critic method and the usage of deep networks as function approximators. Of particular note for our work is the extension of this algorithm to the distributional case, i.e, the D4PG algorithm.

Lyle, C., Castro, P. S., and Bellemare, M. G. (2019). A comparative analysis of expected and distributional reinforcement learning.

This work provides a comparison between "expectation-based" and DRL algorithms. The main result of the paper is to prove the existence of a relation between the good performances of DRL algorithms and the DL framework.

Martin, J. D., Lyskawinski, M., Li, X., and Englot, B. (2019). Stochastically dominant distributional reinforcement learning.

Introduces the idea of stochastic dominance to propose a new approach to design risk-aware policies. Moreover, a new algorithm, named Dominant Particle Agent (DPA) and based on Wasserstein gradient flows, is proposed and its ability of handling the trade-off between performance and safety are tested against some simple baselines.

Mavrin, B., Zhang, S., Yao, H., Kong, L., Wu, K., and Yu, Y. (2019). Distributional reinforcement learning for efficient exploration.

This paper proposes a new approach to model intrinsic and parametric uncertainty from the return distribution provided by the QR-DQN algorithm. The proposed method outperforms QR-DQN on several Atari games. However, several implementation choices are based on heuristics not adequately justified.

Moerland, T., Broekens, J., and Jonker, C. (2017). Efficient exploration with double uncertain value networks.

This paper is about how to describe and exploit the parametric and the intrinsic uncertainty arising in RL problems. Epistemic uncertainty is modelled by means of a Bayesian approach, while intrinsic uncertainty is described by the return distribution. However, only the case of Gaussian return distributions is considered.

Moerland, T., Broekens, J., and Jonker, C. (2018). The potential of the return distribution for exploration in rl.

This work studies the performances of a number of distributional algorithms on deterministic environments. In this setting, the return distribution is entirely induced by the stochastic policy and can be used to cope with the exploration-exploitation trade-off.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010a). Nonparametric return distribution approximation for reinforcement learning. In *ICML*.

One of the most significant attempts at investigating the properties of the return distribution before C51. The paper proposes a more general version of the standard Bellman's equation describing the return distribution. This equation is then solved by using a non-parametric approach that extends standard TD learning. However, the final purpose of this work is limited to the design risk-sensitive policies.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010b). Parametric return density estimation for reinforcement learning. In *UAI*.

This work has been published in parallel to the paper reported above by the same authors. A distributional approach focused on the design of risk-aware policies is proposed. The distributional Bellman equation is solved by a parametric approach based on the natural gradient method. Interestingly, the authors propose the use of quantiles to analyse the return distribution, anticipating the idea of QR-DQN.

Nikolov, N., Kirschner, J., Berkenkamp, F., and Krause, A. (2018). Information-directed exploration for deep reinforcement learning. *CoRR*, abs/1812.07544.

Introduces a new interesting method to cope with the exploration-exploitation trade-off based on DRL and the concept of Information-Directed-Sampling. Specifically, this new approach introduces a new policy exploiting information on both parametric and aleatoric uncertainty resulting in excellent performances on the Atari games. One of the most convincing works that uses DRL to improve exploration.

Rasmussen, C. E. and Kuss, M. (2003). Gaussian processes in reinforcement learning. In *NIPS*.

In the paper, both the transition probability and the value function are modelled by Gaussian Processes (GP). The Gaussian Process representation of the value function naturally yields a richer description of the value. However, the paper only considers the expectation of the return distribution provided by the GP. This approach leads to a closed form solution of the Bellman's equation obtained by solving a set of m simultaneous linear equations in the value function calculated in a set of m support points.

Rowland, M., Bellemare, M., Dabney, W., Munos, R., and Teh, Y. W. (2018). An analysis of categorical distributional reinforcement learning. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 29–37.

Proposes a theoretical justification based on the Cramér metric of the projection operator used in the C51 algorithm. Furthermore, it introduces a convergence proof for Categorical Q-Learning based on an alternative update rule to the KL divergence step in C51.

Rowland, M., Dadashi, R., Kumar, S., Munos, R., Bellemare, M. G., and Dabney, W. (2019). Statistics and samples in distributional reinforcement learning. *ArXiv*.

This work sheds new light on the field of DRL by introducing a unifying framework capable of describing apparently different distributional algorithms such as C51 and QR-DQN. Moreover, the paper proposes a new algorithm based on learning a new set of statistics called expectiles. The performances of this new technique are then compared to those of other existing DRL algorithms.

Tang, Y. and Agrawal, S. (2018). Exploration by distributional reinforcement learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2710–2716. International Joint Conferences on Artificial Intelligence Organization.

This paper combines a Bayesian approach with the C51 algorithm in order to handle the exploration-exploitation trade-off. The proposed method provides a unifying framework for a number of previous deep RL techniques aimed at enhancing exploration.

Zhang, S., Mavrin, B., Kong, L., Liu, B., and Yao, H. (2018). Quota: The quantile option architecture for reinforcement learning.

The proposed method, named QUOTA, is designed to exploit the quantile distribution provided by the QR-DQN algorithm in order to guide exploration. This interesting method combines the option framework and DRL to generate a policy capable of taking pessimistic or optimistic decisions depending on the particular time-step and state configuration.

Bibliography IV: Miscellany

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *CoRR*, abs/1701.07875.

As the title suggests, the paper extends classical Generative Adversarial Networks (GANs) by introducing a new metric based on the Earth Mover's Distance. The resulting algorithm, named Wasserstein GAN is able to cope with several issues affecting classical GANs, such as learning stability or mode collapse.

Atsalakis, G. S. and Valavanis, K. P. (2009). Surveying stock market forecasting techniques – part ii: Soft computing methods. *Expert Systems with Applications*, 36(3, Part 2):5932 – 5941.

This work presents a survey of over 100 published papers that make use of neural networks in order to forecast future returns of trading assets.

Babaeizadeh, M., Finn, C., Erhan, D., Campbell, R. H., and Levine, S. (2017). Stochastic variational video prediction.

This paper proposes a method entirely based on Variational Inference (VI) aimed at forecasting future frames in natural videos. The reported experiments compare this approach with the existing ones, demonstrating performance improvements.

Bickel, P. J. and Freedman, D. A. (1981). Some asymptotic theory for the bootstrap. *Ann. Statist.*, (6):1196–1217.

This paper provides a mathematical analysis of some properties of the "bootstrap" method. For our purposes, the paper is interesting since it defines the Wasserstein metric (named Mallows metric in the paper).

Billingsley, P. (1986). *Probability and Measure*. John Wiley and Sons, second edition.

A classic probability textbook. Each concept is explained in great detail and the mathematical level is higher compared to the average of other probability textbooks.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*.

This paper proposes a method that models the trading problem as a two-step procedure. First, a deep neural network is used to provide a high-level representation of the current market condition by means of a 20-dimensional feature vector. Second, a recurrent neural network implements the actual trading

decision process by using the state representation provided by the deep neural network model.

Emerson, S., Kennedy, R., O'Shea, L., and O'Brien, J. (2019). Trends and applications of machine learning in quantitative finance. In *8th International Conference on Economics and Finance Research (ICEFR 2019)*.

A very recent survey of the main papers studying the application of ML techniques to finance.

Fischer, T. G. (2018). Reinforcement learning in financial markets - a survey.

A highly recommended resource to explore the application of RL techniques to finance. This survey provides a detailed description of the most popular critic-only, actor-only and actor-critic RL algorithms applied to trading financial assets.

Gao, R. and Kleywegt, A. J. (2016). Distributionally robust stochastic optimization with wasserstein distance.

The paper introduces Distributionally Robust Stochastic Optimization (DRSO) and combines this method with the Wasserstein distance. The Wasserstein metric results in being more advantageous than other popular choices of distances between distributions.

Hegde, S., Kumar, V., and Singh, A. (2018). Risk aware portfolio construction using deep deterministic policy gradients. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*.

This work applies a recurrent version of the DDPG algorithm to the portfolio construction setting. The original neural networks of both actor and critic are modified to include LSTM layers. The paper shows that DDPG is able to handle the problem of risk-aware portfolio optimisation. Experiments are carried out on a portfolio of 20 stocks.

Huang, C. Y. (2018). Financial Trading as a Game: A Deep Reinforcement Learning Approach. *arXiv e-prints*.

Introduces a number of modifications to the recurrent DQN agent in order to make it suitable for the financial trading setting. Interestingly, the authors suggest that DRL algorithms can be used to extend the proposed work in order to design risk-sensitive policies.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

This popular paper introduces Batch Normalisation, a method to cope with the problem of "internal covariance shift", i.e the constant change of the input distribution of each layer due to the variation of the parameters of the previous layer. This technique makes the learning process faster and introduces an useful regularisation effect to contrast overfitting.

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

This popular paper presents Adam, a widely used variant of SGD based on adaptive approximations of lower-order moments.

Koenker, R. (2005). *Quantile Regression*. Econometric Society Monographs. Cambridge University Press.

This monography presents a complete theoretical investigation of the subject of quantile regression along with a discussion of a variety of applications of this technique ranging from finance to biology.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

This paper represents a milestone for the application of CNNs to image recognition problems. The proposed CNN architecture, named AlexNet, won the Imagenet Challenge (ILSVRC) in 2012 with a top 5 test error of 15.4%.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA. ACM.

This paper introduces the "convolutional deep belief network", a scalable generative model capable of successfully learning internal hierarchical representations of natural images in a UL fashion.

Levina, E. and Bickel, P. (2001). The earth mover's distance is the mallows distance: some insights from statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*.

Reports a formal proof of the equivalence between Earth Mover's distance and the Mallows (Wasserstein) distance.

Li, Y. (2018). Deep reinforcement learning.

This work proposes an introduction to the most important deep RL techniques introduced in the last few years. Of particular note is the section describing existing and potential applications of these methods to the financial and to the healthcare sectors.

Liang, Z., Chen, H., Zhu, J., Jiang, K., and Li, Y. (2018). Adversarial Deep Reinforcement Learning in Portfolio Management.

This work compares the performances of three relatively recent reinforcement learning algorithms, namely DDPG, PPO and PG (Policy Gradient), to the portfolio management problem. Experiments are performed on datasets of China stock market. Authors provide open-source code.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Very good ML introductory textbook.

Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods.

A comprehensive review of various applications of probabilistic inference to artificial intelligence. The theory of Markov Chains is thoroughly presented and several Markov Chain Monte Carlo algorithms are described.

Ritter, G. (2017). Machine learning for trading.

Well written paper that shows an example of the application of the Q-learning algorithm to a simulated market environment. This work is interesting because it provides an extensive description of how to cast the problem of designing risk-averse financial strategies into a RL framework.

Rizzo, M. L. and Székely, G. J. (2016). Energy distance. *WIREs Comput. Stat.*, 8.

Provides a description of the properties of the Cramér distance. In the paper, the name "energy distance" is used because of the analogy of this metric with Newton's gravitational potential energy.

Rosasco, L. (2018). Introductory machine learning notes.

Notes used by Prof. Lorenzo Rosasco in his course, Introduction to Machine Learning, at the University of Genoa.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*.

This pioneering work introduces the Earth Mover's Distance in the context of image retrieval. In particular, it proposes a method for comparing images that detects perceptual similarities more efficiently than other baseline methods.

Sharpe, W. F. (1994). The sharpe ratio. 21(1):49–58.

First introduction of the popular Sharpe Ratio, a commonly used parameter among investors that is meant to quantify the ratio between expected return and the risk associated with an investment.

Stanko, S. (2018). Risk-averse distributional reinforcement learning: a cvar optimization approach.

Master's thesis at the Czech Technical University describing a new class of RL agents designed to maximize the so-called Conditional Value at Risk instead of the expected return.

Thompson, W. R. (1933). On The Likelihood That One Unknown Probability Exceeds Another In View Of The Evidence Of Two Samples. *Biometrika*.

Popular introduction to the Thompson sampling technique.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

These slides are extracted from a Coursera class taught by Geoffrey Hinton. The proposed algorithm, RMSProp, is an extension of SGD where the learning rate is adapted throughout learning for every single parameter.

Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H., and Walid, A. (2018). Practical deep reinforcement learning approach for stock trading.

This paper explores the application of the DDPG algorithm to the maximisation of investment return by the optimisation of stock trading strategy. DDPG is shown to outperform two traditional baseline trading strategies. Experiments are performed by training the DDPG agent on a vector of daily stock prices consisting of 30 stocks data.

Łukasz Kaiser and Bengio, S. (2018). Discrete autoencoders for sequence models.

This work proposes to extend sequence models, such as RNNs, with autoencoders in order to extract meaningful internal representations, for instance in the context of language modelling and machine translation.

Appendix A

Kernel Density Estimation

Kernel Density Estimation[1] is a classical approach to provide an estimate p_n of the true underlying probability distribution p describing a set of n independent observations $x_1, ...x_n$, where $x_i \in \mathbb{R}^D$.

The basic idea of KDE is summarised by the following equation:

$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}^D K\left(\frac{\|x - x_i\|}{h}\right), \tag{A.1}$$

where K is called *kernel*, whereas h is a scale parameter called *bandwidth*. In one-dimensional problems ($D = 1$) a kernel can be defined as a function $K : \mathbb{R} \to \mathbb{R}$, possessing the following properties:

$$\int K(x)dx = 1, \quad \int xK(x)dx = 0, \quad \int x^2K(x)dx < \infty \tag{A.2}$$

One of the most popular choices is the Gaussian kernel, which is defined as follows:

$$K(x) = \frac{1}{\sqrt{2\pi}}e^{x^2/2} \tag{A.3}$$

Common ways to define higher-dimensional kernels are given by:

$$\prod_{j=1}^{D} K\left(x^j\right) \quad \text{or} \quad K(\|x\|) \tag{A.4}$$

The choice of the scale parameter h is important to obtain a good estimate of the true distribution. One possible way to determine an appropriate value for this quantity is to use

[1] This brief summary of KDE is extracted from Rosasco [2018].

cross-validation of the training data. In our experiments in Chapter 4, we use the *Scipy* implementation of KDE that utilises a Gaussian kernel and implements automatic bandwidth determination.