Università degli Studi di Napoli Federico II
Ph.D. Program in
Information Technology and Electrical Engineering
XXXV Cycle

Thesis for the Degree of Doctor of Philosophy

Reinforcement Learning for Control

by
Francesco De Lellis

Advisor: Prof. Mario di Bernardo

Co-advisor: Prof. Giovanni Russo

Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

If a machine is expected to be infallible,
it cannot also be intelligent.
- Alan Turing, 1947

Reinforcement Learning for Control

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

Francesco De Lellis

March 2023

Approved as to style and content by

Prof. Mario di Bernardo, Advisor

Prof. Giovanni Russo, Co-advisor

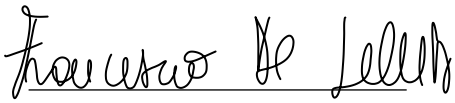
Università degli Studi di Napoli Federico II
Ph.D. Program in Information Technology and Electrical Engineering
XXXV cycle - Chairman: Prof. Stefano Russo

http://itee.dieti.unina.it

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, March 9, 2023



Francesco De Lellis

Abstract

In this thesis, we go through the current state of the art reinforcement learning for control applications. We analyze the pros and cons of the methods provided in the literature. Then we establish a common framework to describe both reinforcement learning and control problems and we present four benchmark problems to analyze and compare reinforcement learning algorithms. Also, we propose a novel solution, the minimal performance Q-learning, capable of searching and guaranteeing a solution that meets a desired level of performance in terms of steady-state error and settling time. Moreover, we also present the control tutored reinforcement learning, an architecture where a feedback controller derived from an approximate model of the environment assists the learning process to enhance its data efficiency. We apply this idea to the classical Q-learning, in the form of a deterministic Control-Tutored Q-Learning (CTQL), that defines the reward function so that a Boolean condition can be used to determine when the control tutor policy is adopted. We also introduce a probabilistic CTQL (pCTQL) that is instead based on executing calls to the tutor with a certain probability during learning. Moreover, we also develop a control tutored deep reinforcement learning the CT-DQN. All the strategies proposed in this thesis are thoroughly analyzed and compared with the literature via the set of metrics to evaluate learning and control performances. Eventually, we discuss how control techniques can be applied to face the global COVID-19 pandemic.

Keywords: Control Theory, Reinforcement Learning, Optimization, Deep Learning

Sintesi in lingua italiana

Nella sua tesi, Francesco De Lellis affronta la tematica della sintesi e validazione di nuovi algoritmi di controllo basati su Reinforcement Learning (RL). In particolare, egli propone nel suo elaborato un framework innovativo per la progettazione di leggi di controllo basate su Reinforcement Learning che si propone di risolvere il problema dei lunghi tempi di apprendimento di algoritmi RL guidando l'apprendimento attraverso l'uso di controllori sintetizzati su modelli anche non sufficientemente accurati del sistema. Questo metodo, detto Control-Tutored Reinforcement Learning (CTRL), viene poi validato attraverso una attenta campagna di simulazioni numeriche basate su Python per risolvere alcuni problemi di benchmark scelti dalla libreria di OpenAI gym. La tesi mostra come il CTRL riesca a migliorare le performance di base del controllore basato su modello parziale addestrandolo tramite RL in tempi ridotti rispetto ad approcci di apprendimento privi del controllore tutor. Nella tesi si propone, inoltre, una metodologia analitica di progettazione dello schema di controllo basato su RL e per la scelta della funzione di reward in modo da poter garantire livelli minimi di performance di controllo al termine dell'addestramento. La tesi analizza in dettaglio sia da un punto di vista analitico che teorico le proprietà della metodologia proposta verificandone l'efficacia e l'applicabilità alla sintesi di controllori MIMO e SISO in un ventaglio di applicazioni rappresentative. Infine, la tesi riporta lo sviluppo e la validazione su dati di un modello a rete di diffusione dell'epidemia da COVID-19 svolto durante la pandemia.

Parole chiave: Teoria del Controllo, Apprendimento per Rinforzo, Ottimizzazione, Apprendimento Profondo.

Contents

Acknowledgements

I deeply thank Mario di Bernardo, who wisely guided me during my scientific research. He successfully pushed me towards my goals sharing his precious years of experience making him the best mentor I could ask for. I also thank Giovanni Russo and Mirco Musolesi for their valuable collaboration during the development of this research. Their point of view and knowledge has been a key element for the progress of this research. They showed me different sides of the process of making scientific research which I will always be grateful for. Also, I would like to thank Marco Coraggio who supported and guided me during some of the toughest times I encountered as a researcher. To them goes my unconditional respect and admiration for their genuine efforts toward the development of a better scientific community. Last but not least, I am profoundly grateful to my family and all the loving people that supported me during these beautiful years.

List of Acronyms

The following acronyms are used throughout the thesis.

ML	Machine Learning
RL	Reinforcement Learning
MBRL	Model Based Reinforcement Learning
DRL	Deep Reinforcement Learning
TD	Temporal Difference
QL	Q-Learning
SARSA	State Action Reward State Action
DQN	Deep Q-Networks
CT	Control Tutor
CTRL	Control Tutored Reinforcement Learning
CTQL	Control Tutored Q-Learning
pCTQL	probabilistic Control Tutored Q-Learning
CT-DQN	Control Tutored Deep Q-Networks

mpQL	Minimal Performance Q-Learning
DDPG	Deep Deterministic Policy Gradient
A2C	Advantage Actor Critic
A3C	Asyncronous Advantage Actor Critic
NAF	Normalized Advantage Function
TRPO	Trust Region Policy Optimization
ME-TRPO	Model Ensable Trust Region Policy Optimization
PPO	Proximal Policy Optimization

List of Figures

List of Tables

Chapter 1

Introduction

Modern day scientific developments generated a set of techniques, methodologies, and know-how for model-based control used in many different applications. In most cases, such applications are not amenable to empirical models or derivations based on first principles. For this reason, the scientific community has embraced more frequently data-driven approaches. These new methods have been applied to solve a variety of problems in a diverse range of complex systems applications, such as nuclear fusion [17], the brain [52], epidemiology [81], finance [41], synthetic biology [92], social networks [20], power grids [75], and robotics [84]. Such systems manifest themselves as heavily nonlinear, multi-scale, and high-dimensional; however, in most cases, scientists can extrapolate dominant underlying patterns characterized and modeled for correct monitoring and control. The rise of data-driven modern mathematical methods is fueled by the unprecedented availability of data as well as computational resources and it showed to be capable of achieving unprecedented, sometimes *superhuman*, performance in a number of tasks [94, 59, 82, 43, 26]. In the literature there exists several example of data-driven approaches such as the autotuning techniques for PID controller which range from Ziegler and Nichols methods [101] to genetic algorithms [42].

However, a remarkable example of a data-driven approach is Reinforcement (RL). It is one of the main paradigms of Machine Learning (ML) and its key feature is to create artificial agents capable of solving a given control task without, in principle, assuming any knowledge about the underlying

dynamics. This type of learning process is rooted in animal psychology. With the *Law of Effect*, in 1898, Edward Thorndike advanced the idea that animals, as well as humans, use a trial-error process to learn and understand their physical environment [91]. This idea was opposed to the previous, state of the art, work by George Romanes, who, instead, supported the idea that animals and humans think things through when dealing with a new environment or situation and a trial-error process is not involved to learn from such situations [76]. These findings posed the grounds for modern day data-driven methods and, more in general, artificial intelligence, and we are now able to train artificial agents using a trial-error process via reinforcement learning. Specifically, we are able to train control systems by giving positive or negative reinforcements every time they interact with the environment, eventually achieving the objective that such reinforcements represent.

Although partially replaced, the ideas of George Romanes can be easily associated with Classic Control Theory. Given a mathematical description of the system of interest, a controller can drive the behavior of such a system; in the same way, a human can deeply think about its action provided that she/he holds sufficient knowledge about the environment. There are more analogies between control theory and reinforcement learning. A notable example is the Markov Decision Process formalism, introduced in 1957 by Richard Bellman [6], which jumpstarted the development of reinforcement learning algorithms that we now know today.

In this thesis, I try to answer the question of whether modern day reinforcement learning can benefit from control theory, and vice versa, whether control theory can benefit from reinforcement learning. In particular, I show how merging a controller with a reinforcement learning algorithm can be crucial in the learning process to build better reinforcement learning algorithms. Also, I show how learning processes can be leveraged to enhance performance, stability, and robustness of a given control law. Also, I show how learning processes in combination with modeling and control can deal with complex tasks.

Differently from other attempts in the literature, I show how trial and error approach can be embedded with modeling and planning in the sense of control theory and coexist in one algorithm. Furthermore, I also show how control theoretic arguments can be leveraged to certify the perfor-

mance of learning algorithms.

Lastly, I discuss how control can be applied to real world problems like COVID-19 to synthesize effective non-pharmaceutical strategies. I propose a compartmental model of Italy as a network of 20 interconnected regions and propose a decentralized control strategy.

1.1 Outline of the Thesis

The rest of this thesis is organized as follows:

1. **State of the art:** Here the ground foundation of RL is showcased with a list of algorithms. We go through them and analyze the pros and cons of the main solution used in the literature;

2. **Testbed problems:** We provide a mathematical formulation for bot reinforcement learning and control. Also, we define a set of metrics to perform quantitative analysis. Eventually, we introduce a few control problems to test the algorithm proposed.

3. **Guarantee of performance via reward shaping:** We provide a design pipeline for reinforcement learning to design artificial agents capable of guaranteeing a minimum level of performance.

4. **Control tutored reinforcement learning:** In this section we formalize the RL problem and propose the Control Tutored Reinforcement Learning (CTRL). This idea deals with the creation of an extra tutoring unit able to improve existing RL algorithms. Such a tutor figure is designed in the form of control laws based on a rough estimate of the environment dynamics. The main objective of such solutions is to improve data efficiency and learning times which many state of the art algorithm struggle with;

5. **COVID-19, modeling and control:** We model the COVID-19 Italian epidemic as a network of interconnected region. We analyze the network and develop a controller to limit the epidemic spread while reducing economic costs.

6. **Conclusions:** Final remarks are discussed and a proposal for the next steps of this research is highlighted.

Chapter 2

State of the Art

In this chapter, we review the current state of the art techniques in reinforcement learning. We also go through the key works in the literature about deep reinforcement learning and mixed strategies which make use of control techniques as well as reinforcement learning.

2.1 Reinforcement learning

Reinforcement Learning (RL) [87, 8] is increasingly used to learn control policies from real world or computer generated data in a number of different applications [44, 28, 13]. Despite the advantage of the possibility of learning effective control policies without the need for a mathematical model, one of its key drawbacks is the requirement of performing typically long training sessions to explore and learn a sub-optimal control policy for the dynamics of interest.

In particular, in RL, the optimal policy is found by combining explorative and greedy actions, thus accepting possible failures in the training phase during which the agents are learning an optimal policy. Unfortunately, in control applications like autonomous driving, multi-joint robotics, and complex systems, long training phases are often unacceptable and failures while learning might lead to unsafe situations [97, 11, 34].

Moreover, these applications are often characterized by continuous state space and using RL as is, requires a dense discretization of the system state space or a function approximation that is also subject to the

learning process [87]. This limitation generated a set of algorithms called Deep Reinforcement Learning (DRL) to solve the issue of discrete spaces that we will discuss later [59, 50].

In the current literature, several different algorithms exist which could be classified in different ways (e.g. the one proposed in [22, 29]). We propose a classification as proposed in Fig. 2.1 in which a key feature to discriminate between different RL algorithms is the use of a direct mathematical description also subject to learning. In particular, we refer to an algorithm that does not assume anything about the system dynamics as *model-free* and the ones that leverage a mathematical model (also subject to learning) as *model-based*.

Another important factor to take into account is the specific task the RL algorithm is trying to learn. As shown in Fig. 2.1, there exists in the literature algorithms which try to learn the relationship between the objective function and the state space (or the couple action-state space) that we refer to as *value-based* [87, 22, 98]. Moreover, some algorithms fix the structure of the control policy and try to learn the optimal value of their parameters which we call *policy-based* [87, 22, 80].

Finally, a set of algorithms that tries to combine both the learning process of value-based and policy-based giving birth to *actor-critic* methods [50, 57, 35]. As a matter of fact, the critic is in principle the value of the current policy while the actor is, in general, a policy-based algorithm. The two entities coexist and in particular, the actor tries to find the optimal values of the parameters based also on the feedback of the critic.

2.1.1 Model-free vs model-based

Model-free algorithms do not use an explicit mathematical model to learn, however, they use different functions (subject to learning) to actively learn control solutions. A popular *value-based* algorithm of this kind is *Q-learning* [98, 66]. This algorithm uses a tabular representation of the state-action value function, often referred to as the Q-function (for "quality"; a tool to learn the relationship between the cost function and the couple action-space). In its classical version, Q-learning is typically tabular and often refers to discrete spaces [87, 98]. In this algorithm an artificial agents aims at learning the Q-function during its exploration of the state space. The data collected is then used to update the approximation of the

future objective contained in the Q-function via an iterative update rule [98]. Q-learning leverages the stochastic Robbins-Monro approximation to prove its convergence to the optimal Q-values with probability 1 [8, 74]. An interesting yet important property of Q-learning is the capability of learning optimal values of the Q-values with respect to a greedy policy which always selects the action according to the maximum Q-value. This happens regardless of the policy used for exploration and makes this algorithm an *off-policy* method. The implication of such property is that we can freely formulate exploration policies to learn the Q-values of a fixed policy (the greedy policy) which optimizes a given objective in the form of a sum of rewards. We further discuss how we can define exploration policies for Q-learning and leverage the *off-policy* property in Chapters 4 and 5. Another algorithm which tries to approximate the objective via the Q-function is SARSA which is the *on-policy* counterpart of the Q-learning. As a matter of fact, the objective of SARSA is to learn the Q-values associated to the exploration policy which is used for exploration but also as its target [87].

Policy-based RL, aim at defining a policy and then tuning its parameters according to the experience gathered so far [80, 79, 35]. An example of this is REINFORCE. This algorithm maximizes a given objective function by searching the parameters of its policy using a gradient ascent algorithm [87]. Given their nature, *policy-based* algorithms are mostly DRL algorithms that use Neural Networks [39] as policy structure; we will discuss these algorithms in greater detail later in this chapter.

As anticipated, the learning process may be completely model-free or not. In particular, there exists a number of *model-based* approaches (MBRL) that aim at learning the model of the environment [18, 85, 47]. More in general, *model-based* RL techniques have been developed to learn the system dynamics; the model is then used to perform simulated rollouts, which generate new synthetic data for learning. This technique was first introduced in [86] and used for further different implementations of the model inside the RL architecture.

An example of a very effective MBRL algorithm is PILCO [18]. This algorithm fits data gathered via the interaction with the dynamics via Gaussian processes. Therefore, PILCO is capable of drastically reducing learning times, however, it exhibits poor performances when used to deal

with large state spaces and exhibits instability between different random seeds [95].

MBRL techniques tend to be effective, however, they may introduce biases in the learning process as part of the data used is not generated by the actual dynamics.

2.2 Deep reinforcement learning

Recent developments of nonlinear approximators, such as Neural Networks (NN) [39, 32], opened the possibility to extend the application of RL algorithms to continuous spaces. The main goal of Deep Reinforcement Learning [22, 58] is to leverage the scalability of Neural Networks in continuous spaces [32].

In [59] the Deep Q-Networks (DQN) algorithm is proposed. DQN, a *value-based* algorithm, uses Neural Networks to iteratively approximate the Q-function; it is among the most popular implementations of DRL and can be used also for continuous state spaces. Differently from tabular *Q-learning* [98], there are currently no guarantees of convergence towards the optimal policy for DQN [25], although its effectiveness is supported by strong empirical evidence [59, 58].

In [79] a *policy-based* method is proposed and named as Trust Region Policy Optimization (TRPO). This algorithm is used to optimize large nonlinear functions (such as neural networks) which are used as policy. Further improvements were made with the Proximal Policy Optimization (PPO) which simplifies the TRPO algorithm without degrading its performance [80].

Another notable example present in the literature is the approach presented in [50]. In this work, an *actor critic* architecture is proposed, the Deep Deterministic Policy Gradient Algorithm (DDPG), capable of acting in a continuous action space. DDPG uses two NNs to learn the Q-function and the policy and can be seen as an extension of DQN for continuous action spaces. A comprehensive analysis in terms of learning performance and sensitivity to parameter tuning of DRL algorithms has been made in [23].

Control algorithms-based on DRL have shown impressive performance in different application fields, including the control of plasma in nuclear

fusion [17] and that of microbial cultures in bioreactors [92]. However, a crucial challenge for these algorithms is that, depending on the state space and action space dimensions, they typically require extensive training and the results are very sensitive with respect to parameter variations such as reward scaling factors, neural network hyperparameters, learning rate and discount factor [23].

2.3 Advances in reinforcement learning for control

RL and DRL cannot ensure data efficiency, learning stability, and scalability to large scale and multi-agent systems [22, 23]. State-of-the-art algorithms require large data sets that need to be acquired via multiple interactions with the environment. This results in different data efficiency characteristics that usually require long training times.

Several attempts exist in the literature to solve this issue, mainly using MBRL [95]. One of the first to be proposed as such is the DYNA-Q framework [86]. The PILCO algorithm, modeling the environment using Gaussian processes, is also an MBRL algorithm that is particularly effective with low dimensional state spaces [18]. Moreover, there are also attempts in extending existing *model-free* algorithms to leverage information from a learned model like the DQN with model-based acceleration [33] and the ME-TRPO [45].

However, DRL leverages some other techniques to improve data efficiency like the replay buffer which has been applied in DRL algorithms such as DQN and DDPG [4, 59].

RL algorithms often express high sensitivity to parameter variation exhibiting poor stability of the learning process and requiring longer training times depending on the values selected by the programmer [23]. An example of such behavior can be seen in DDPG and TRPO algorithms [50, 79, 23]. Eventually, RL algorithms tend to have problems in finding a policy that solves the control problem when the state space dimensionality increases as, for example, in multi-agent reinforcement learning [10, 88].

To further address these issues related to RL algorithms, new approaches are being developed to leverage control theoretic methods. In [53, 73] a formalization of the RL problem is proposed as an optimization problem. Also, in recent years, classical control theoretical tools and RL

have been intertwined in a number of ways. For example, in [72], Model Predictive Control (MPC) was used in state-space regions where a model of the dynamics is available, while tabular Q-learning was used in the other regions.

Instead in [100], an RL algorithm is used to vary the parameters of the model and the objective function used by an MPC. In [33], a variant of the Q-learning algorithm (normalized advantage functions) is discussed; the authors show that their solution is able to accelerate the learning process by using local linear models fitted iteratively with exploration data.

Other solutions combining control strategies with RL include those in [1], where a policy gradient algorithm is adopted which uses preexisting knowledge of the system dynamics in the form of an approximate Markov decision process; and in [48], where a *reference action governor* is used to enforce safety constraints (in the sense of restricting the state space to admissible regions). In so doing, the action is decided via an optimization problem that penalizes deviations from the action suggested by an RL strategy, making these approaches a valuable solution to achieve safe RL.

Despite the many remarkable successes of current state-of-the-art RL [61], two key problems remain to be solved: (i) potentially *long* learning times and (ii) the lack of convergence or performance guarantees (important for example in safety-critical applications) during learning [7, 68]. To address these issues, in this thesis, we formulate an alternative approach, the Control-Tutored Reinforcement Learning (CTRL) algorithm which relies on the introduction of a *beneficial* bias, based on the use of control laws, in the exploration process to speed up learning, which we discuss in Chapter 5. Furthermore, to move our first steps towards a reliable set of algorithms capable of guaranteeing a minimum level of performance, we also present a principle design of a reinforcement learning solution in Chapter 4.

2.4 Summary

State of the art reinforcement learning offers a wide variety of algorithms to solve complex control tasks. The recent rise of Deep Learning in

combination with Reinforcement Learning boosted the performance that such algorithms can gain. However, they require intensive training hence long learning times and a huge amount of data. Moreover, artificial agents are not capable of guaranteeing final performances due to high parameters sensitivity.

In what follows, we use control theoretic arguments to develop artificial agents capable of autodetecting and guaranteeing a minimum level of performance required on a given task. Moreover, we develop an original framework for RL, the control tutored reinforcement learning. This class of algorithms is capable of leveraging partial information about the system dynamics, in the form of control laws, to drive the exploration process and reduce learning times.

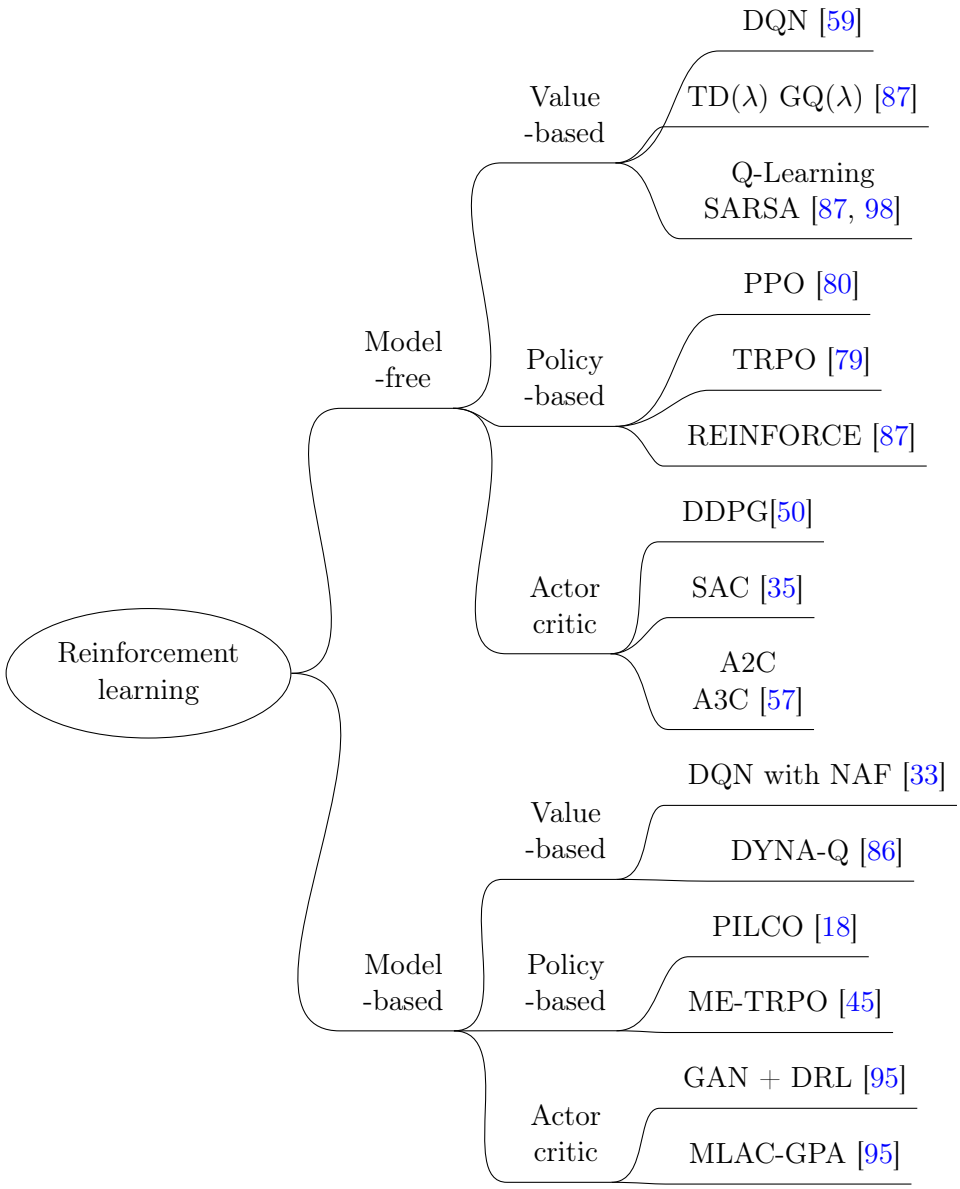


Figure 2.1. Taxonomy of reinforcement learning algorithms. We classify the algorithm-based on the *model-free* and *model-based* property (see Sec. 2.1) as well as the type of learning architecture (*value-based*, *policy-based*, *actor-critic*).

Chapter 3

Testbed problems

In this chapter, we state the control problem using an optimization framework. Also, we introduce and describe the programming tools used for numerical validation (e.g. the OpenAI Gym suite [65], which provides a set of environments to test Reinforcement Learning algorithms). Finally, we introduce two sets of metrics used to benchmark performance and control performance, respectively.

3.1 Problem statement

3.1.1 Notation

Sets are denoted by calligraphic capital characters, random variables are denoted by capital letters e.g. X, and their realization by e.g. x. The probability density (mass) function of a continuous (discrete) random variable X is denoted by $p(x)$ and we use the notation $x \sim p(x)$ to denote the sampling of a random variable from its probability function.

For both continuous and discrete random variables, we often consider the situation where the support of $p(x)$ is compact; $\text{rand}(\mathscr{A})$ denotes the uniform distribution over the set $\mathscr{A}$. The expectation of a function, say $h(\cdot)$, of X is defined as $\mathbb{E}_p[h(X)] \coloneqq \int h(X)p(x)\mathrm{d}x$, when continuous; if X and as $\mathbb{E}_p[h(X)] \coloneqq \sum h(x)p(x)$ if X is discrete. In both cases, the integral/sum is taken on the support of $p(x)$, and we might omit p in $\mathbb{E}_p$ when there is no ambiguity. We denote by $\|\cdot\|$ the Euclidean norm.

3.1.2 Problem formulation

We consider a discrete time dynamical system affected by noise, of the form

$$X_{k+1} = f_k(X_k, U_k, W_k), \quad x_0 = \tilde{x}_0, \tag{3.1}$$

where $k \in \mathbb{N}_{\geq 0}$ is discrete time, $X_k \in \mathscr{X}$ is the state of the system at time k, with $\mathscr{X}$ being the state space, $\tilde{x}_0 \in \mathscr{X}$ is the initial condition, $U_k \in \mathscr{U}$ is the control input (or action) and $\mathscr{U}$ is the set of feasible inputs. Also, W_k is a random variable representing noise and $f_k : \mathscr{X} \times \mathscr{U} \times \mathscr{W} \to \mathscr{X}$ is the system's dynamics.

Following e.g. [53, 73], given this set-up, we consider the problem of learning a plan of actions $\pi_1, \ldots, \pi_{N-1}$ to solve the following finite-horizon optimization problem:

$$\begin{aligned}
\max_{\pi_1,\ldots,\pi_{N-1}} \quad & \mathbb{E}[J^{\bar{\pi}}], && \text{(3.2a)}\\
\text{s.t.} \quad & X_{k+1} = f_k(X_k, U_k, W_k), \quad k \in \{1, \ldots, N-1\}, && \text{(3.2b)}\\
& U_k = \pi_k(X_k), \quad k \in \{0, \ldots, N-1\}, && \text{(3.2c)}\\
& x_0 \text{ given}, && \text{(3.2d)}
\end{aligned}$$

where the time horizon is between 0 and N. In (3.2) the cost is set as the expectation of the *objective function*

$$J^{\bar{\pi}} = r_N(X_N) + \sum_{k=1}^{N} \gamma^k r_k(X_k, X_{k-1}, U_{k-1}), \tag{3.3}$$

with $r_k : \mathscr{X} \times \mathscr{X} \times \mathscr{U} \to \mathbb{R}$ and $r_N : \mathscr{X} \to \mathbb{R}$ being the *reward* received by the agent at each k and $\gamma \in (0, 1]$ being the discount factor.

In some of the results proposed, we also consider the infinite horizon problem defined as in (3.2) but considering $N \to \infty$.

3.2 Learning and control metrics

In all scenarios we consider, each study is repeated in S independent *sessions*, each composed of E *episodes*, which are simulations lasting N time steps.

Depending on the algorithm considered, the learning parameters are carried over from one episode to the next, and re-initialized at each session. An episode can end earlier if a (scenario-specific) *terminal condition* is met, and we denote by J_e^π the cumulative reward in episode e and obtained via (3.3).

As usual, maximizing J^π amounts to fulfilling some problem-specific *goal*: we define the *goal condition* c_g as a Boolean variable that is true if and only if the goal is achieved in an episode. Next, we define three metrics to assess learning performance.

Definition 3.2.1 (Learning metrics)**.** *To evaluate the learning performance we use the following metrics:*

- *The* average cumulative reward *is*

$$J_\mathrm{avg}^\pi \coloneqq \frac{1}{E}\sum_{e=1}^{E} J_e^\pi. \tag{3.4}$$

- *The* terminal episode E_t *is the smallest episode such that* c_g *is true for all* $e \in \{E_\mathrm{t}-10,\dots,E_\mathrm{t}\}$.

- *The* average cumulative reward after terminal episode *is*

$$J_\mathrm{avg,t}^\pi \coloneqq \frac{1}{E_\mathrm{t}}\sum_{e=E_\mathrm{t}}^{E} J_e^\pi. \tag{3.5}$$

J_avg^π is often used in RL as a measure of performance during the training[23, 95]; E_t is used to assess the effective duration of the learning phase, and consequently, data efficiency; $J_\mathrm{avg,t}^\pi$ quantifies the quality of the controller, once the learning phase is completed.

Next, we define three metrics inspired by those commonly used in control theory, to assess transient and steady-state performance. Moreover, when the goal is (or entails) achieving some *goal state* $x^* \in \mathscr{X}$ (or region containing x^*), we say the goal is a *regulation problem*. considering that the *greedy policy* is defined as follows:

$$\pi_\mathrm{g}(x) = \arg\max_{u\in\mathscr{U}} Q(x,u) \tag{3.6}$$

where $Q : \mathscr{X} \times \mathscr{U} \to \mathbb{R}$ is the state-action value function approximated by a learning algorithm [98].

Definition 3.2.2 (Control metrics)**.** *To evaluate the control performance we use the following metrics:*

- *The* cumulative reward *(see § 3.1)* obtained following π_{g} *is* $J^{\pi_{\mathrm{g}}}$*;*
- *in an episode, the* settling time k_{s} *is a time instant such that the goal is achieved or a related task is completed (defined uniquely in each scenario, when possible);*
- *in regulation problems, the* steady state error *is*

$$e_{\mathrm{s}} \coloneqq \frac{1}{N - k_{\mathrm{s}} + 1} \sum_{k=k_{\mathrm{s}}}^{N} \|x - x^*\| \,. \tag{3.7}$$

3.3 OpenAI Gym

Gym is an Application Programming Interface (API) developed by OpenAI and widely used in the literature to test reinforcement learning algorithms [23, 95]; it can be used as a python library to access a collection of reference environments. In particular, from the Gym library, we use the Pendulum, Lunar Lander, and Car Racing environments; further information about the full Gym suite is provided in [65].

We selected these environments for the following reasons. The Pendulum is a classical nonlinear benchmark problem in control theory. Lunar Lander represents a harder control problem with multiple inputs and outputs (MIMO) in which certain regions of the state space must be avoided. Finally, Car Racing was selected as it is a tracking problem where the state is observable as a matrix of pixels, rather than measured physical quantities. Also, in this environment, the state-action space is larger, which typically makes it more difficult the process of learning a policy.

3.3.1 Pendulum

In this environment, a pendulum is modeled as a rigid rod of length $l = 1$ m, with a homogeneous distribution of mass $m = 1$ kg; its moment of inertia is $I = ml^2/3$ and it is affected by gravity, with acceleration g. A snapshot of the output of this simulator is shown in Fig. 3.1.

We let $x_k := [x_{1,k} \;\; x_{2,k}]^\mathsf{T}$, where $x_{1,k}$ and $x_{2,k}$ are the angular position and angular velocity of the pendulum, respectively; $x_{1,k} = 0$ corresponds to the unstable vertical position. The control input u_k is a torque applied to the pendulum.

In this environment, the objective is to stabilize the pendulum in its upward position, $x^* = [0 \;\; 0]^\mathsf{T}$. For this reason, we say that the goal condition is met if $\|x_k\| < 0.6, \quad \forall k \in [N - \hat{N}, N]$. By doing so, we say that the goal condition is met if the rod stayed in a neighborhood of the unstable vertical position in the last $\hat{N}$ steps of an episode.

To train the artificial agent we consider two reward functions, the former:

$$r^{\mathrm{a}}(x_k, x_{k-1}, u_{k-1}) = d(x_{k-1}) - d(x_k) + \rho(x_k), \tag{3.8}$$

where $d(x) := x_1^2 + 0.1x_2^2$, and ρ is a prize function as in (5.6) that is

Figure 3.1. Pendulum environment simulation [63]. The red bar represents the rod, while the arrow represents the torque

going to be discussed further discussed. While the latter is defined as the standard Gym reward:

$$r^{\mathrm{g}}(x_k, x_{k-1}, u_{k-1}) = -x_{1,k}^2 - 0.1x_{2,k}^2 - 0.001u_{k-1}^2. \tag{3.9}$$

Whenever we have to, we apply a discretization of the action and state spaces as follows. The spaces for states and control variables are bounded, so that $x_k \in [-\pi, \pi] \times [-8, 8]$, and $u_k \in [-2, 2]$. Both spaces are discretized non-uniformly, employing a finer discretization close to the origin of the state space and for small control actions. We verified that this allows us to select values more precisely when close to the regulation point, reducing regulation error and control energy; on the other hand, a coarser discretization far from the regulation point yields a shorter learning time.

Concerning $x_{1,k}$, the interval $\left[-\pi, -\frac{\pi}{9}\right]$ is discretized into 8 equally spaced values, $\left(-\frac{\pi}{9}, -\frac{\pi}{36}\right]$ into 7 values, and $\left(-\frac{\pi}{36}, 0\right]$ into 5 values; $[0, \pi]$ is discretized in an analogous fashion. Concerning $x_{2,k}$, $[-8, -1]$ is discretized into 10 values, and $(-1, 0]$ into 9 values (analogously for $[0, 8]$).

Concerning u_k, $[-2, -0.2]$ is discretized into 9 values, and $(-0.2, 0]$ into 4 values (analogously for $[0, 2]$.

3.3.2 Lunar lander

In a 2-D space, a spaceship subject to gravity, in the absence of friction, must use its thrusters to land with small velocity on a landing pad.

The states are the coordinates and orientation of the lander, the corresponding velocities, and two Boolean variables to determine the contact of the two legs with the ground. The lander has three thrusters, on the left, on the right, and on the bottom (main) of the spacecraft. The possible (four) control inputs are the following: use only the left thruster, only the right one, the main one, or no activation of any thruster.

The position of the landing pad and the initial position and orientation of the lander are fixed, while the initial linear speed is random, as well as the terrain topography aside from the landing pad. The spacecraft *lands correctly* if it impacts on the pad with its legs at a moderate velocity, while it *crashes* if its body touches the ground or lands with a velocity that is too high. Further detail can be found in [64].

The agent obtains a high reward for landing correctly, a largely negative one for crashing, and a small negative one for consuming fuel. Following [64], we set the goal condition as achieving $J^\pi \geq 200$ (i.e., c_g true if $J^\pi \geq 200$).

It is worth noting that this might also be seen as a regulation problem, with the objective of reaching the center of the pad (x^*), in the origin of the reference frame. Thus, we define the settling time k_s as the instant when the spacecraft lands correctly if it happens. An episode ends immediately if the lander lands correctly or if it crashes.

3.3.3 Car racing

Car Racing considers a 2-D space where a car must complete a random track as fast as possible.

The state is composed of the pixel matrices of three consecutive image frames. The actions are the possible combinations of "steer left/right", "accelerate", and "brake", all by a fixed amount.

The agent is rewarded positively each time it visits a new bit of road, and receives a small negative reward when time steps pass. Further details are reported in Section 3.3.3 and in [62].

In this problem, rather than defining a specific goal that can or cannot

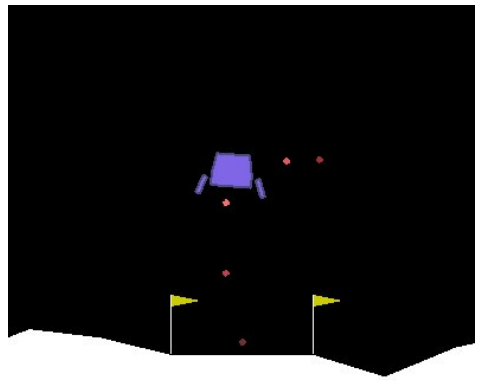


Figure 3.2. Lunar Lander environment simulation [64]. A spacecraft is shown in purple while the red dots represent fuel expelled. Moreover, the region of a safe landing is delimited by the two yellow flags.

be satisfied in an episode, we deem it more natural to consider the task just as that of maximizing the reward; therefore, the only metric we consider is the cumulative rewards J^{π}_{avg} and $J^{\pi_{\text{greedy}}}_{\text{avg}}$.

Eventually, an episode can end earlier if the car gets too far from the track or visits 95% of the track.

3.4 The herding problem

We consider the problem of controlling one or more mobile agents (the herders) that have the task of influencing the motion of a group of autonomous agents (the targets) in the plane. Under the assumption that the herders only possess very limited knowledge of the dynamics of the targets, this problem is used as an application example for the $CTQL$ approach.

Given N mobile target agents in the plane, described by:

$$\dot{x}^i_t(t) = f^i(x^i_t, t) \qquad \forall i = 1, ..., N \tag{3.10}$$

where $x^i_t \in \mathscr{R}^2$ is the position of the i-th target position, $f^i : \mathscr{R}^{2\times 1} \to \mathscr{R}^2$ is

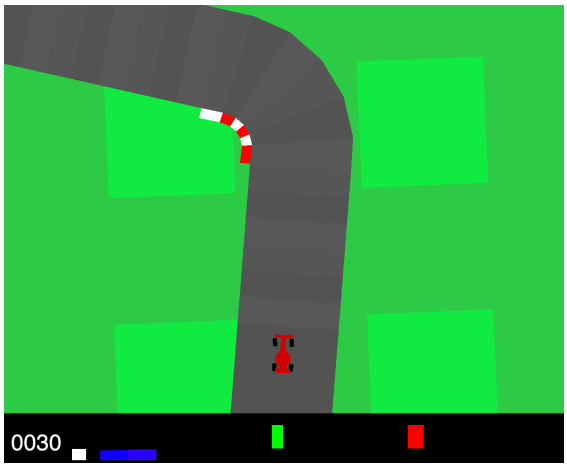


Figure 3.3. Car Racing environment simulation [62]. The figure shows a red car running on a randomly generated circuit. On the bottom, the following indicators are shown: step counter, true speed in white, ABS sensors in blue, steering wheel position in green, and gyroscope in red.

the i-th target dynamics, and given M herder agents in the plane, described by:

$$\dot{x}_h^j(t) = u^j(t) \qquad \forall j = 1, ..., M \tag{3.11}$$

where $x_h^j \in \mathscr{R}^2$ is the position of the j-th herder position, u^j is the control input of the j-th herder agent, the control objective is to design the input vector $u = [u^1, ..., u^M]$ able to force targets to reach and remain in the circular goal region $G \coloneqq \{x \in \mathbb{R}^2 : \|x - x_g\| < \rho_g\}$ of center $x_g = 0$ and radius $\rho_g = 5$m. That is, to achieve the goal:

$$\lim_{t \to \infty} \|x_t^i(t) - x_g\| < \rho_g \qquad \forall i = 1, ..., N \tag{3.12}$$

Note that the goal region G is a region of stability for the targets' dynamics and that the design of the control inputs u^j will need to take into account the coupling between the targets and the herders. As done in [49] and [70] we consider a purely kinematic model for the herder dynamics assuming, for the sake of simplicity, that their inertia can be neglected.

We assume the dynamics of the target agents f^i to be composed of

two terms; a deterministic term f_1^i, related to the interaction with the herder, and a stochastic term f_2^i that models some uncertainty in the target motion.

More specifically, following the arguments of [69], we assume the i-th target is repelled by the j-th herder through the term modeled using the artificial potentials [40] as follows:

$$f_1^i = \mu \frac{x_t^i - x_h^j}{\|x_t^i - x_h^j\|^3} U(x_t^i, x_h^j, \rho_t) \tag{3.13}$$

where $\mu = 1$ is a constant gain modeling the intensity of the coupling with the herder and $U(x_t, x_h, \rho_t)$ is an interaction function defined as:

$$U(x_t, x_h, \rho_t) = \begin{cases} 1, & \|x_t - x_h\| < \rho_t \\ 0, & \text{otherwise} \end{cases} \tag{3.14}$$

that ensures that the coupling between the target and herder agents is active only if their relative position is smaller than some target's influence radius $\rho_t = 3\text{m}$.

The target's own random dynamics is modeled as:

$$f_2^i(t) = \beta(t) e^{\jmath\theta(t)} \tag{3.15}$$

where $\beta(t)$ and $\theta(t)$ are updated every Δt seconds from the uniform distributions $\mathscr{U}(0, \beta_{max})$ and $\mathscr{U}(0, 2\pi)$, respectively.

To make the dynamics more realistic, the maximum speed of the target agent has been upper bounded as follows:

$$\dot{x}_t = \begin{cases} f_1 + f_2, & \|f_1 + f_2\| < v_{t,max} \\ v_{t,max} e^{\jmath \angle(f_1 + f_2)}, & \text{otherwise} \end{cases} \tag{3.16}$$

with $v_{t,max} = 9\text{ms}^{-1}$ being the maximum admissible speed of the target agents.

The herders are assumed to be able to adjust their velocities almost instantaneously, as done for example in [3]. Their speed, as for the targets, is saturated to a maximum fixed value, $v_{h,max} = 14\text{ms}^{-1}$ so that their

dynamics can be written as:

$$\dot{x}_h^j = \begin{cases} u^j, & \|u^j\| < v_{h,max} \\ v_{h,max} e^{\angle u^j}, & \text{otherwise} \end{cases} \tag{3.17}$$

The state space $\mathscr{X}$ is defined, for each herder, in terms of the possible discretized values of:

1. the relative distance between the herder and the target chased by it,
2. the angular position of the herder,
3. the speed of the chased target.

These quantities are defined in the region, between herders and targets, in which the latter is influenced by the presence of the former. In particular, we will assume that this is a circle of fixed radius ρ_{er} around the target, which is only an estimate of the actual influence region ρ_t in their dynamics, and that the learning process will be active only if such a region exists.

The discretization is obtained by reducing the sampling the range $[0, \hat{\rho}_\tau]$ of relative distances with stepsize $T_{m,d} = \frac{\hat{\rho}_\tau}{10}$m and the range of angles [0,2π] with $T_{a,d} = \frac{2\pi}{10}$rad. The angular position of the herder was discretized in the range $[0, \frac{\pi}{2}]$ with stepsize $T_{a,h} = \frac{\pi}{10}$rad. The target speed was discretized in the range $[0, v_h^{\max}]$ with stepsize $T_{m,v_\tau} = \frac{v_h^{\max}}{50}\text{ms}^{-1}$. , and the range of angles [0,2π] with $T_{a,v_\tau} = \frac{2\pi}{20}$rad

The action space $\mathscr{U}$ is chosen to be the set of possible values of the input vector u^j to the herder dynamics given by (3.11), whose norm is bounded by the herder maximum speed $v_{h,max}$. The action space consisted of possible herder velocities discretized in the range [0,$v_h^{\max}$] with stepsize $T_{m,v_h} = \frac{v_h^{\max}}{10}\text{ms}^{-1}$ and possible angular orientation in the range [0, 2π] with $T_{a,v_h} = \frac{2\pi}{20}$rad.

With this choice, each herder decides its next action by evaluating how close to the goal the chased target is (R_1), how close itself is to the chased target (R_2), and if the herder is inside the goal region after the action is taken (R_3).

Using x_t (x_h) to indicate the position of a generic target (herder) agent at a fixed time instant and x'_t (x'_h) to indicate the position of the same

agent at the following time instant and assuming the goal region centered in the origin, we propose to choose as reward function the sum of three contributions:

$$R = k_1 R_1 + k_2 R_2 + k_3 R_3 \tag{3.18}$$

where

$$R_1 = (|x_t| - |x'_t|) \tag{3.19}$$

$$R_2 = \sigma(\bar{k}(|x'_t| - \rho_g))(|x_t - x_h| - |x'_t - x'_h|) \tag{3.20}$$

$$R_3 = \sigma(\bar{k}(\rho_g - |x'_h|)). \tag{3.21}$$

with k_1, k_2, k_3 and $\bar{k}$ being positive constant gains. Here $\sigma(\cdot)$ is the sigmoid function defined as:

$$\sigma(z) = \frac{1}{(1 - e^{-z})} \tag{3.22}$$

that decays exponentially as a function of the argument so that term R_1 dominates as R_2 and R_3 decay as a function of their arguments.

3.5 Summary

We adopt the optimal control formalism to formulate the reinforcement learning problem as done in Section 3.1. By doing so we can differentiate between the learning process and the design elements (e.g. reward, control policy). Moreover, Section 3.2 we define a set of learning metrics to evaluate data efficiency of RL algorithm. Also, we define a set of control metrics to evaluate the final control performance of the artificial agents after the training phase has ended. Finally, we introduce a set of testbed problems that we use to validate and compare the RL algorithms proposed. In particular, we select four different control problems to evaluate control performances in different contexts.

In what follows, we will define and test the minimal performance Q-learning algorithm on the problem of stabilizing an inverted pendulum as posed in Section 3.3.1. We show how the algorithm is capable of autoregulate exploration to stop the learning process autonomously and guarantee a minimum level of performance on the task (in terms of steady-state error

and settling time). Also, will use the metric of Section 3.2 to make a quantitative comparison of control tutored strategies with respect to classical RL. For the sake of the state space dimension, we will test the tabular strategy only on the inverted pendulum and the herding problem. We will also test deep architectures on the problem of landing a spacecraft as posed in Section 3.3.2 and train tutored and untutored algorithms directly from pixels generated by the car racing simulator introduced in Section 5.4.4. We will show how the control tutored reinforcement learning algorithms proposed are capable of shrinking learning times and improving final control performances using only control laws generated by partial modeling of the system dynamics.

Chapter 4

Guarantee of performance via reward shaping

In what follows, for the case of regulation problems, we propose a reward design strategy to induce recognizable properties in the objective function (3.3) By doing so, we can synthesize feedback control policies aware of minimal requirements in performance encoded in the objective function. To give an intuition, our goal is to design a reward function such that the optimization process will find a solution with "good enough" final control performances providing stability certification for the final learned solution.

4.1 Control performance requirements

Let $x^* \in \mathscr{X}$ be a *goal state*. Then, we assume the state space $\mathscr{X}$ is measurable, and let $d : \mathscr{X} \to \mathbb{R}_{\geq 0}$ be a distance function of a point in $\mathscr{X}$ with respect to x^*. Let $\theta \in \mathbb{R}_{>0}$, and define the *goal region* $\mathscr{G} := \{x \in \mathscr{X} \mid d(x) < \theta\}$. We denote by *trajectory* a sequence of states $\xi = \{x_k\}_{k\in[0,+\infty)} \in x^\infty$.

Remark 4.1.1. *Any $\xi \in \mathscr{T}$ is simply a sequence of states. In general, given some $\xi \in \mathscr{T}$ with first state $\tilde{x}_0$, there is no guarantee that there exists a sequence of control inputs $\{u_k\}_{k\in[0,\infty)}$ such that a system with dynamics (3.1), starting in x_0, will have a trajectory that is equal to ξ.*

Moreover, given any trajectory ξ, let us define $k_{\text{exit}}(\xi) \in \mathbb{N}_{>0}$ as the smallest time instant such that $x_{k_{\text{exit}}(\xi)-1} \in \mathscr{G}$ and $x_{k_{\text{exit}}(\xi)} \notin \mathscr{G}$; if this condition never occurs for the trajectory ξ, we say that $k_{\text{exit}}(\xi) = \infty$. Let us select two design parameters $k_{\text{s}}, k_{\text{p}} \in \mathbb{N}_{>0}$; the former has the meaning of a settling time, whereas the latter is a guaranteed time of presence in the goal region.

Given such parameters, we define the set of trajectories that satisfy the minimum control performance requirements.

Definition 4.1.2 (Reference trajectories)**.** *The set of reference trajectories which meet the control performance requirements is denoted by $\mathscr{T} \subseteq \mathscr{X}^{\infty}$ and any of its trajectories, say $\xi = \{x_k\}_{k\in[0,+\infty)} \in \mathscr{T}$, fulfills the following conditions:*

1. *the agent is in $\mathscr{G}$ before k_{s}, i.e.,*

$$\exists k \leq k_{\text{s}} : x_k \in \mathscr{G}. \tag{4.1}$$

2. *the agent does not leave $\mathscr{G}$ before k_{p}, i.e.,*

$$k_{\text{exit}}(\xi) > k_{\text{p}}. \tag{4.2}$$

In this chapter, the Definition 4.1.2 represent a set of sub-optimal trajectories which meet a minimum level of performance required for the task at hand.

4.2 Infinite horizon problems

In this section, given the problem formulation (3.2) with Assumption 4.2.1, we seek to find conditions on the cost function $J(\xi)$ associated with a generic trajectory ξ that ensure that $\xi \in \mathscr{T}$ (c.f. Definition 4.1.2). Moreover, we consider the case where the discount factor $\gamma \in (0, 1)$ is not equal to the unit since, for practical applications, undiscounted objective are cause of instability for many RL algorithm, Q-learning included [8].

Assumption 4.2.1 (Reward structure)**.** *We assume the reward has the following structure:*

$$r_k(x_k, x_{k-1}, u_{k-1}) = r_k^{\text{b}}(x_k, x_{k-1}, u_{k-1}) + c(x_k, x_{k-1}), \tag{4.3}$$

where

- $r_k^{\mathrm{b}} : \mathscr{X} \times \mathscr{X} \times \mathscr{U} \to \mathbb{R}$ *is a bounded reward term, i.e., with real scalars* $U_{\mathrm{out}}, U_{\mathrm{in}}, L_{\mathrm{in}}$ *such that*

$$\sup_{k\in\mathbb{N}_{\geq 0},\ x_k\in\mathscr{X}\setminus\mathscr{G},\ x_{k-1}\in\mathscr{X},\ u_k\in\mathscr{U}} r_k^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) = U_{\mathrm{out}}, \tag{4.4a}$$

$$\sup_{k\in\mathbb{N}_{\geq 0},\ x_k\in\mathscr{G},\ x_{k-1}\in\mathscr{X},\ u_k\in\mathscr{U}} r_k^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) = U_{\mathrm{in}}, \tag{4.4b}$$

$$\inf_{k\in\mathbb{N}_{\geq 0},\ x_k\in\mathscr{G},\ x_{k-1}\in\mathscr{X},\ u_k\in\mathscr{U}} r_k^{\mathrm{b}}(x_k, x_{k-1}, u_{k-1}) = L_{\mathrm{in}}. \tag{4.4c}$$

- $c : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ *is a* correction term *given by*

$$c(x_k, x_{k-1}) = \begin{cases} c_{\mathrm{in}}, & \text{if } x_k \in \mathscr{G}, \\ c_{\mathrm{exit}}, & \text{if } x_{k-1} \in \mathscr{G} \text{ and } x_k \notin \mathscr{G}, \\ 0, & \text{otherwise}, \end{cases} \tag{4.5}$$

with $c_{\mathrm{in}} > 0$ *and* $c_{\mathrm{exit}} < 0$. *Moreover, it holds that*

$$c_{\mathrm{in}} \geq U_{\mathrm{out}} - L_{\mathrm{in}}. \tag{4.6}$$

Note that Assumption 4.2.1 is not particularly restrictive in general. Indeed, if in some problem we want to use a preexisting reward, this can be done provided that the reward is bounded (see (4.4)); then the additional correction term c can be added to the preexisting reward.

Proposition 4.2.2 (Large cost function implies trajectory as in Definition 4.1.2)**.** *Let Assumption 4.2.1 hold. Let* $\sigma \in \mathbb{R}$ *such that*

$$\sigma \geq \frac{U_{\mathrm{out}}}{1-\gamma}. \tag{4.7}$$

Assume that

$$c_{\mathrm{in}} \leq -U_{\mathrm{in}} - U_{\mathrm{out}} \frac{1-\gamma^{k_{\mathrm{s}}}}{\gamma^{k_{\mathrm{s}}}} + \sigma \frac{1-\gamma}{\gamma^{k_{\mathrm{s}}}}, \tag{4.8}$$

$$c_{\text{exit}} \leq -\frac{1}{\gamma^{k_{\text{p}}-1}}\left[(U_{\text{in}}+c_{\text{in}})\frac{1-\gamma^{k_{\text{p}}+1}+\gamma^{k_{\text{p}}}}{1-\gamma}+\sigma\right], \tag{4.9}$$

Then, given a trajectory $\xi = \{x_k\}_{k\in[0,+\infty)}$*, it holds that*

$$J^{\pi}(\xi) > \sigma \quad \Rightarrow \quad \xi \in \mathscr{T}. \tag{4.10}$$

Proof. We will show that $\xi \notin \mathscr{T} \;\Rightarrow\; J^{\pi}(\xi) \leq \sigma$. Then, the thesis is obtained by noting that given two logical propositions A and B, $A \Rightarrow B$ is equivalent to $\neg B \Rightarrow \neg A$.

Consider the case that $\xi \notin \mathscr{T}$; this can happen for three reasons:

1. the agent never finds itself in the goal region $\mathscr{G}$;

2. the agent is in $\mathscr{G}$ for the first time at a time later than k_{s};

3. the agent exits from $\mathscr{G}$ at an instant $k_{\text{exit}}(\xi) \leq k_{\text{p}}$.

We now consider the three cases one by one and show that in all of them, it must hold that $J^{\pi}(\xi) \leq \sigma$.

Case 1. Formally, in this case, we have that $\forall k \in [0,\infty), x_k \notin \mathscr{G}$. Therefore, from (4.5), we have $c(x_k, x_{k-1}) = 0$ for all k. From (3.3), exploiting in order (4.3), (4.4a), and (4.7), we have

$$\begin{aligned} J^{\pi}(\xi) &\coloneqq \sum_{k=1}^{+\infty} \gamma^{k-1} r_k(x_k, x_{k-1}, u_{k-1}) = \\ &= \sum_{k=1}^{+\infty} \gamma^{k-1} r_k^{\text{b}}(x_k, x_{k-1}, u_{k-1}) \leq \\ &\leq U_{\text{out}} \sum_{k=1}^{+\infty} \gamma^{k-1} = \frac{U_{\text{out}}}{1-\gamma} \leq \sigma, \end{aligned} \tag{4.11}$$

Case 2. In this case, defining $k_{\text{enter}} \coloneqq (\min k \text{ s.t. } x_k \in \mathscr{G})$, we have that $k_{\text{enter}} > k_{\text{s}}$. For the sake of simplicity and without loss of generality, assume that the agent never leaves the region $\mathscr{G}$ after k_{enter} (i.e. $x_k \in$

$\mathscr{G}, \forall k \geq k_{\text{enter}}$).[1] From (3.3), (4.3), and (4.5), we have

$$\begin{aligned} J^{\pi}(\xi) &:= \sum_{k=1}^{+\infty} \gamma^{k-1} r_k(x_k, x_{k-1}, u_{k-1}) \\ &= \sum_{k=1}^{k_{\text{enter}}-1} \gamma^{k-1} r_k^{\text{b}}(x_k, x_{k-1}, u_{k-1}) + \\ &\quad + \sum_{k=k_{\text{enter}}}^{+\infty} \gamma^{k-1} [r_k^{\text{b}}(x_k, x_{k-1}, u_{k-1}) + c_{\text{in}}]. \end{aligned} \tag{4.12}$$

Exploiting (4.6), and recalling that $k_{\text{enter}} > k_{\text{s}}$ from (4.12), we have [2]

$$\begin{aligned} J^{\pi}(\xi) \leq & \sum_{k=1}^{k_{\text{s}}} \gamma^{k-1} r_k^{\text{b}}(x_k, x_{k-1}, u_{k-1}) + \\ & + \sum_{k=k_{\text{s}}+1}^{+\infty} \gamma^{k-1} [r_k^{\text{b}}(x_k, x_{k-1}, u_{k-1}) + c_{\text{in}}]; \end{aligned} \tag{4.13}$$

Then, from (4.13) and exploiting (4.4), we have

$$\begin{aligned} J^{\pi}(\xi) &\leq U_{\text{out}} \sum_{k=1}^{k_{\text{s}}} \gamma^{k-1} + (U_{\text{in}} + c_{\text{in}}) \sum_{k_{\text{s}}+1}^{\infty} \gamma^{k-1} \\ &= U_{\text{out}} \sum_{k=0}^{k_{\text{s}}-1} \gamma^{k} + (U_{\text{in}} + c_{\text{in}}) \gamma^{k_{\text{s}}} \sum_{k=0}^{+\infty} \gamma^{k} \\ &= U_{\text{out}} \frac{1-\gamma^{k_{\text{s}}}}{1-\gamma} + (U_{\text{in}} + c_{\text{in}}) \frac{\gamma^{k_{\text{s}}}}{1-\gamma} \leq \sigma \end{aligned} \tag{4.14}$$

Exploiting (4.8), it is immediate to see that $J^{\pi} \leq \sigma$.

Case 3. By definition of k_{exit} (see first paragraph of Section 4.1) , we

[1] This assumption does not make us lose generality because we are interested in upper bounding $J^{\pi}(\xi)$ with σ and the assumption makes $J^{\pi}(\xi)$ the largest possible, because the smallest reward obtainable inside $\mathscr{G}$ (i.e. $c_{\text{in}} + L_{\text{in}}$) is at least equal to the largest reward obtainable outside $\mathscr{G}$ (4.6).

[2] On the right hand side we would have obtained if the agent entered $\mathscr{G}$ at time $k_{\text{s}}+1$

have that $x_{k_{\text{exit}(\xi)-1}} \in \mathscr{G}$ and $x_{k_{\text{exit}(\xi)}} \notin \mathscr{G}$. From (4.6), the largest $J^\pi(\xi)$ is obtained if ξ is such that the agent has been in $\mathscr{G}$ since $k = 1$, leaves $\mathscr{G}$ at $k_{\text{exit}}(\xi) = k_\text{p}$ and enters again at time $k_\text{p} + 1$. Therefore, without loss of generality, we assume the above is the case. Then, we have the inequality

$$\begin{aligned} J^\pi(\xi) \leq & \sum_{k=1}^{k_\text{p}-1} \gamma^{k-1}[r_k^\text{b}(x_k, x_{k-1}, u_{k-1}) + c_\text{in}] + \gamma^{k_\text{p}-1} c_\text{exit} + \\ & + \sum_{k=k_\text{p}+1}^{\infty} \gamma^{k-1}[r_k^\text{b}(x_k, x_{k-1}, u_{k-1}) + c_\text{in}] \leq \\ & \leq (U_\text{in} + c_\text{in}) \left[\sum_{k=0}^{k_\text{p}-2} \gamma^k + \gamma^{k_\text{p}} \sum_{k=0}^{\infty} \gamma^k \right] + \gamma^{k_\text{p}-1} c_\text{exit} = \\ & = (U_\text{in} + c_\text{in}) \frac{1 - \gamma^{k_\text{p}+1} + \gamma^{k_\text{p}}}{1 - \gamma} + \gamma^{k_\text{p}-1} c_\text{exit}. \end{aligned} \tag{4.15}$$

Exploiting hypothesis (4.9), we immediately verify that $J^\pi \leq \sigma$. □

Note that Proposition 4.2.2 does not guarantee that there exist a $\xi \in \mathscr{T}$ such that $J^\pi(\xi) > \sigma$. This is outlined in the next Proposition, which states that c_in cannot be selected too low.

Corollary 4.2.3. *Let assumption 4.2.1 hold. If*

$$c_\text{in} \leq \sigma(1 - \gamma) - U_\text{in}, \tag{4.16}$$

then, $\nexists \xi \in \mathscr{T}$ such that $J^\pi(\xi) > \sigma$.

Proof. From (4.6), the theoretical largest possible value of $J^\pi(\xi)$ is given as follows (see also (3.3)):

$$J^\pi(\xi) \leq (U_\text{in} + c_\text{in}) \frac{1}{1 - \gamma} \tag{4.17}$$

then since (4.16) holds, we obtain $J^\pi(\xi) \leq \sigma, \forall \xi$, proving the thesis. □

4.3 Minimal performance Q-learning

Proposition 4.2.2 and Corollary 4.2.3 are particularly relevant since they allow the detection of good trajectories by just looking at the cumulative objective J^π. The key idea is to use the Q-learning algorithm to approximate J^π and build an artificial agent capable of auto-detecting whether the learned solution meets the performance requirements. To do so, we recall the *greedy policy* introduced in (3.6), given a state-action value function Q, as follows:

$$\pi_{\mathrm{g}}^{Q}(x) = \arg\max_{u \in \mathscr{U}} Q(x, u). \tag{4.18}$$

Also, we refer to a ϵ-greedy policy as a probabilistic switching policy which introduces a chance of taking a random action with probability $\epsilon \in (0, 1)$ or an action according to π_g^Q.

Definition 4.3.1 (Greedy policy generated trajectories)**.** *Given a state $x \in \mathscr{X}$, the greedy policy generated trajectories $\phi^Q(x) \in \mathscr{X}^\infty$ are defined as an infinite set of states generated by starting from x and applying policy π_{g}^Q, according to dynamics (3.1).*

Moreover, considering the set of trajectories defined in Definition 4.3.1, we state the following result

Theorem 4.3.2. *Consider a state $x_k \in \mathscr{X}$ at time k, and let $x_{k+1} = f_k(x_k, \pi_{\mathrm{g}}^Q(x_k), w_k)$. Assume that*

1. *process noise is absent, i.e., $w_k = 0, \forall k \in [0, +\infty)$;*
2. *the assumptions of Proposition 4.2.2 hold;*
3. *it holds that*

$$\operatorname{sign}\left(\max_{u \in \mathscr{U}} Q(x_k, u) - \sigma\right) = \operatorname{sign}\left(J^{\pi_{\mathrm{g}}^Q}(\phi^Q(x_{k+1})) - \sigma\right). \tag{4.19}$$

Then, if $\max_{u \in \mathscr{U}} Q(x_k, u) > \sigma$, the following hold:

a. *By following policy π_{g}^Q, the agent will find itself in $\mathscr{G}$ at least once before $k_{\mathrm{s}} + 1$ time instants have passed, i.e., $\exists k' \in [k+1, k+1+k_{\mathrm{s}}]$: $x'_k \in \mathscr{G}$.*

b. By following policy π_g^Q, the agent will not leave $\mathscr{G}$ before $k_p + 1$ time instants have passed, i.e., $\nexists k_{exit} \in [k+2, k+1+k_p] : x_{k_{exit}-1} \in \mathscr{G}, x_{k_{exit}} \notin \mathscr{G}$.

Proof. First, note that, since π_g^Q is deterministic and we assumed the absence of process noise (hypothesis (1)), $\phi^Q(x_{k+1})$ is deterministic, and so is $J^{\pi_g^Q}(\phi^Q(x_{k+1}))$. Exploiting the hypothesis (3), the fact that $\max_{u \in \mathscr{U}} Q(x_k, u) > \sigma$ implies that $J^{\pi_g^Q}(\phi^Q(x_{k+1})) > \sigma$.

Thus it is immediate to apply Proposition 4.2.2, and obtain the thesis (thanks to hypothesis (2)). □

Remark 4.3.3. *Normally, given some policy π, through policy evaluation [87], it is possible to compute iteratively a state-value function Q so that $Q(x_k, \pi(x_k))$ approximates $J^\pi(f(x_k, \pi(x_k), w_k))$. Therefore, if the policy is selected as π_g^Q (see (4.18)), and Q is well approximating $J^{\pi_g^Q}$ in the sense described above, then we should have*

$$Q(x_k, \pi_g^Q) = \max_{u \in \mathscr{U}} Q(x_k, u) \approx J^{\pi_g^Q}(\phi^Q(x_{k+1})) \tag{4.20}$$

which satisfies to hypothesis (4.19).

We now proceed to develop an algorithm capable of autodetecting a minimum level of performance by leveraging results from Theorem 4.3.2. The key idea is to use the approximation of J^π contained in the Q-table to decide whether the agent should keep exploring the state action space or just use the greedy policy. In Algorithm 1, we have a switching policy that uses the current approximation of J^π to decide when exploration should be used. In particular, the algorithm compares Q-values with the threshold σ (see (4.7)) to decide when the artificial agent should end the training and halt exploration.

Note that it is possible that the approximation of J^π contained in the Q-value is overestimated, in the sense that $Q > \sigma$, but $J \not> \sigma$. In this case, the agent will use the greedy policy even if it should be exploring because $\phi \notin \mathscr{T}$ (see Definition 4.1.2). However, in [24] the authors proved that, if Q is overestimating the objective, the Q- learning algorithm still converges using the greedy policy. In our case, this mechanism will make the Q-

Algorithm 1: minimal performance Q-learning

```
Initialize Q(x, u) arbitrarily;
Set reward function according to Assumption 4.2.1;
Select c_in, c_exit, σ according to Proposition 4.2.2 ;
Set e = 0, k = 0;
while e < E do
    Initialize x_0 ;
    while k < N do
        if arg max_u Q(x_k, u) ≤ σ then
            choose action u_k using an exploration policy;
        else
            choose action u_k using the greedy policy;
        end
        Take action u_k and observe next state x_{k+1} and reward r_k;
        Q(x_k, u_k) ←
        Q(x_k, u_k) + α[r_k + γ arg max_u Q(x_{k+1}, u) − Q(x_k, u_k)];
        k + +;
    end
    e + +;
end
```

values converge during the process even when Q-values are overestimating J^{π}.

4.3.1 Numerical results

We test our algorithm on the problem of stabilizing the inverted pendulum described in Section 3.3 by running $S = 10$ training sets of $E = 1000$ episodes of simulations made by $N = 1500$ steps.

To apply our algorithm we perform a state and action discretization as presented in Section 3.3.1

Assuming reward (3.9), we derive the quantities defined in (4.4) as:

- $U_{\text{out}} = -0.1 \cdot \bar{d}^2$,

- $U_{\text{in}} = 0$,

- $L_{\text{out}} = -\pi^2 - 0.1 \cdot 8^2 - 0.001 \cdot 2^2$,

where $\bar{d}$ a desired distance from the goal point (defined as $d(x) = \|x - x^*\|$) at steady state , with $x^* = [\pi, 0]$, that we use as steady state error.

To train the agent, We also set $S = 5$, $E = 1000$, $N = 1500$, $k_p = 1000$ and the desired settling time as $k_s = 500$ steps. Selecting an arbitrary threshold $\sigma = 1000$ we derive the correction terms in (4.5) according to Proposition 4.2.2, obtaining:

1. $c_{\text{in}} = -U_{\text{in}} - U_{\text{out}} \frac{1-\gamma^{k_{\text{s}}}}{\gamma^{k_{\text{s}}}} + \sigma \frac{1-\gamma}{\gamma^{k_{\text{s}}}}$,

2. $c_{\text{exit}} = -\frac{1}{\gamma^{k_{\text{p}}-1}} \left[(U_{\text{in}} + c_{\text{in}}) \frac{1-\gamma^{k_{\text{p}}+1}+\gamma^{k_{\text{p}}}}{1-\gamma} + \sigma \right]$,

with $\gamma = 0.99$, also, When the agent is allowed to explore, it uses an ϵ-greedy policy with $\epsilon = 0.05$. We also set the learning rate $\alpha = 0.8$.

Simulation results depicted in Fig. 4.1, show a decay in exploration calculated as a percentage over the total step of an episode.

As each of the sessions reaches its end, we observe that the artificial agent is no longer exploring. Moreover, after training, we test the final policy of each of the independent sessions obtaining the state trajectory in terms of distance from the origin depicted in Fig. 4.2. In this graph, we can appreciate how the agents have learned solutions capable of guaranteeing the control specifications, in terms of steady-state error and settling time.

As a matter of fact, *all the agents* are capable of guaranteeing a steady state error less than the goal distance $\bar{d}$ in less than $k_s = 500$ steps.

4.4 Summary

In Section 4.1, we define the set of trajectories of a system dynamics which fulfill minimal performance requirements. Then, in Section 4.2 we establish an analytical criterion to link properties in the RL objective function with such a set of trajectories in deterministic scenarios. In Section 4.3 particular, we leverage such analytical results to formulate the minimal performance Q-learning. In this algorithm, the learning agent can

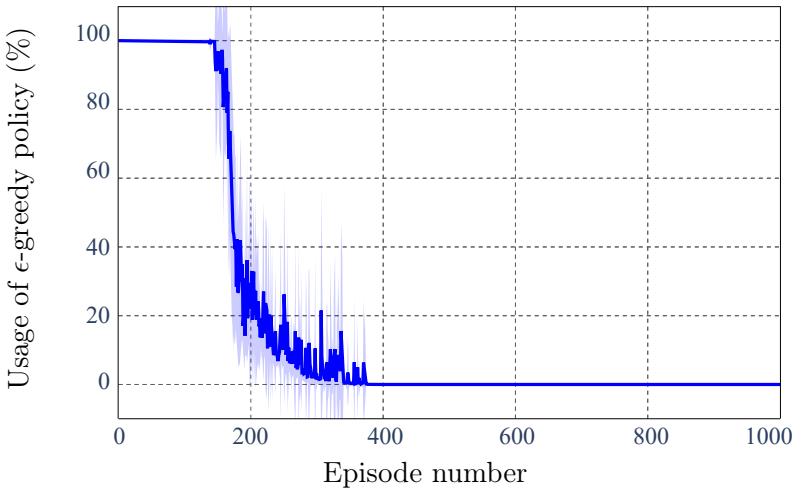


Figure 4.1. Percentage of steps the ϵ-greedy policy was used in each episode by the minimal performance Q-learning algorithm. The solid curves represent the mean of the results of $S = 10$ training sessions; shaded areas correspond to the mean plus or minus the standard deviation.

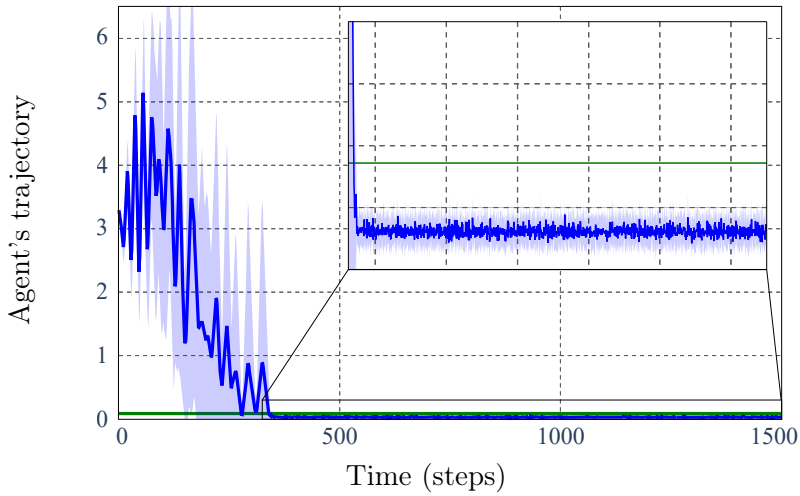


Figure 4.2. Average (solid line) and two times the standard deviation (shaded area) of the trajectory obtained by the $S = 10$ trained controllers. The results are expressed in terms of distance from the goal regulation point as L_2 norm.

evaluate whether its control policy will generate trajectories, which fulfill minimal performance requirements (e.g. steady-state error and settling time) by evaluating the approximation of the cost function in the form of Q-values.

Finally, in Section 4.3.1 we tested our algorithm on the problem of stabilizing an inverted pendulum (see Section 3.3.1) showing that our algorithm can autoregulate exploration and converge to a solution that fulfills the performance requirement of the set of trajectories defined in Section 4.1. The work presented in this chapter represents an original contribution towards the solution of the *exploration-exploitation dilemma* [87]. Indeed, minimal performance Q-learning showed to be capable of autonomously determining, in an adaptive way, when and how the artificial agent should be exploring the state space.

Chapter 5

Control Tutored Reinforcement Learning

Considering the problem introduced in Section 3.1, we observe that in many RL scenarios, even if the system dynamics f (see (3.1)) is not perfectly known, some partial knowledge about the plant (from e.g. first-principles) might be available.

We propose that this limited information can be exploited to design a feedback control law (or control tutor) that can be used to assist and drive the learning process towards the solution of a control problem of interest, reducing the learning times and improving the control performance.

In particular, the control tutor can be invoked under certain circumstances during the learning stage to suggest actions that the agent can take as an alternative to those computed using a more traditional approach, e.g., obtained by using a tabular learning strategy.

To proceed with our approach, we start by assuming that we have an estimate of f, say $\hat{f} : \mathscr{X} \times \mathscr{U} \to \mathscr{X}$, so that the dynamics of system (3.1) is rewritten as $f(x,u,w) = \hat{f}(x,u) + \delta(x,u,w), \forall x \in \mathscr{X}, \forall u \in \mathscr{U}, \forall w \in \mathscr{W}$, where $\delta : \mathscr{X} \times \mathscr{U} \times \mathscr{W} \to \mathscr{X}$ describes the effect of unknown terms in the dynamics and/or of noise on the system's dynamics. We term the policy based on the use of a control law synthesized by considering only $\hat{f}$ as the *control tutor policy* and denote it by $\pi^{\mathrm{c}} : \mathscr{X} \to \mathscr{U}$.

The architecture of the *Control Tutored Reinforcement Learning* (CTRL) strategy [16] is schematically shown in Figure 5.1. Such a scheme high-

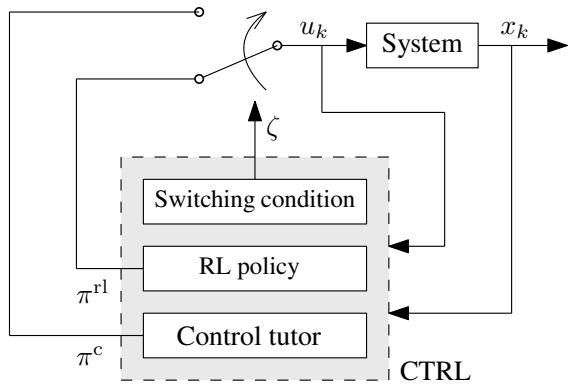


Figure 5.1. Schematic of the Control-Tutored Reinforcement Learning (CTRL) framework.

lights the presence of a switching condition ζ that orchestrates, at each k, the use of either a policy coming from an RL algorithm or the *control tutor* policy.

Therefore, the result is a switching policy used during the learning and defined as follows:

$$\pi(x) = \begin{cases} \pi^{\mathrm{rl}}(x), & \text{if } \zeta \text{ is true,} \\ \pi^{\mathrm{c}}(x), & \text{otherwise,} \end{cases} \tag{5.1}$$

where ζ is a Boolean function (that might depend on time, previous states, etc.) and π^{rl} is the policy of an RL algorithm.

To select the *control tutor* policy π^{c} in (5.1), we assume to have a feedback controller $g : \mathscr{X} \to \mathscr{U}$, designed with limited information. Then, letting $\epsilon^{\mathrm{c}} \in (0, 1)$, we select

$$\pi^{\mathrm{c}}(x) = \begin{cases} g(x), & \text{with probability } 1 - \epsilon^{\mathrm{c}}, \quad (5.2a) \\ u \sim \mathrm{rand}(\mathscr{U}), & \text{with probability } \epsilon^{\mathrm{c}}. \quad (5.2b) \end{cases}$$

In this policy, ϵ^{c} represents the probability of taking a random action with the *control tutor policy*. Moreover, g represents a control law synthesized on the dynamics available $\hat{f}$ and it may not be defined over the same action space $\mathscr{U}$ (for example, as a consequence of the discretization of the control actions).

On the other hand, the reinforcement learning policy π^{rl} in (5.1) is specific to the RL algorithm in use. For example, considering a Q-learning, we can adopt an ϵ-greedy Q-learning solution, i.e.,

$$\pi^{\mathrm{rl}}(x_k) = \begin{cases} \arg\max_{u \in \mathscr{U}} Q_k(x_k, u), & \text{with probability } 1 - \epsilon^{\mathrm{rl}}, \quad (5.3a) \\ u \sim \mathrm{rand}(\mathscr{U}), & \text{with probability } \epsilon^{\mathrm{rl}}, \quad (5.3b) \end{cases}$$

where $\epsilon^{\mathrm{rl}} \in (0, 1)$ and $Q_k : \mathscr{X} \times \mathscr{U} \to \mathbb{R}$ is the state-action value function [87, 8] at time k.

The remaining term to be defined in (5.1) is ζ. In what follows, we present two alternative choices for ζ that result in two different algorithms. However, at time k, once an action is selected from either π^{rl} or π^{c}, the corresponding reward is obtained and used to perform the learning update.

For example, in Q-learning, we proceed to update the Q-table according to the law

$$Q_{k+1}(x_k, u_k) = (1 - \alpha) Q_k(x_k, u_k) + \alpha [r(x_{k+1}, x_k, u_k) + \quad (5.4)$$

$$+ \gamma \max_{u \in \mathscr{U}} Q_k(x_{k+1}, u)], \quad (5.5)$$

where $\alpha \in (0, 1]$ is the *learning rate* and $\gamma \in (0, 1]$ is the *discount factor*.

5.1 Control tutored Q-learning

This first solution based on the CTRL framework is the *Control-Tutored Q-learning* (CTQL), [16]. This algorithm is used to solve regulation problems and uses a reward with a specific structure.

In particular, letting $x^* \in \mathscr{X}$ be a *goal state*, $\theta \in \mathbb{R}_{>0}$, and $\bar{\rho} \in \mathbb{R}_{>0}$, we define the *prize function*

$$\rho(x) = \begin{cases} \bar{\rho}, & \text{if } \|x - x^*\| < \theta, \\ 0, & \text{otherwise.} \end{cases} \quad (5.6)$$

Then, letting $d : \mathscr{X} \to \mathbb{R}_{\geq 0}$ be some distance of the argument with respect to x^*, the reward r_k in (3.3) is given as

$$r(X_k, X_{k-1}, U_{k-1}) = d(X_{k-1}) - d(X_k) + \rho(X_k), \quad k = 1, \ldots, N-1, \quad (5.7)$$

with $r_N(X_N) = 0$. Note that, in (5.7), U_{k-1} does not directly affect the reward but its effect is propagated through the system dynamics.

Furthermore, the term $d(X_{k-1}) - d(X_k)$ is positive when at time k the agent gets closer to the goal state x^*, and vice versa. The prize term $\rho(X_k)$ gives a strong positive reinforcement when a small distance with respect to the goal state is achieved.

The switching criterion ζ in (5.1) depends on the current state x_k, where

$$\zeta \text{ is } \begin{cases} \text{true}, & \text{if } \max\limits_{u \in \mathscr{U}} Q_k(x_k, u) > 0, \\ \text{false}, & \text{otherwise.} \end{cases} \tag{5.8}$$

Additionally, $\forall x \in \mathscr{X}, \forall u \in \mathscr{U}$, we initialize $Q_0(x, u) = 0$. Thus, in the first phase of learning, when limited information about the environment is available, the control tutor policy π^{c} drives the learning process. Then, gradually, as the values of the Q-table are updated using (5.4), the reinforcement learning policy π^{rl} is preferred.

5.2 Probabilistic control tutored Q-learning

Although we found the CTQL to have better performance with respect to the classical Q-learning in certain scenarios (see Section 5.4.2), the reward formulated as in (5.7) does not satisfy the hypotheses used in the classical proof of convergence used for the Q-learning (see, e.g., [8]), as it is not either non-negative or non-positive. Moreover, we verified that the CTQL might fail when the reward function is not selected following the structure given in (5.7). This might depend on the possibility that the reward in (5.7) shapes a Q-table where state-action pairs that eventually lead to the goal state (following policy π^{rl}) have positive values of Q.

Therefore, we propose next a simpler probabilistic-based choice for the Boolean condition ζ in (5.1). We name the resulting algorithm as *probabilistic Control Tutored Q-learning* (pCTQL); differently from CTQL (cf. (5.7)), we do not use any specific structure for the reward function to derive the switching condition between the two policies.

In particular, letting $\beta \in [0,1]$,

$$\zeta \text{ is } \begin{cases} \text{true}, & \text{with probability } \beta, \\ \text{false}, & \text{otherwise}, \end{cases} \tag{5.9}$$

by combining (5.1), (5.3) and (5.9), we have that action (5.3a) is taken with probability $\beta(1-\epsilon^{\text{rl}})$, action (5.2a) is taken with probability $\omega := (1-\beta)(1-\epsilon^{\text{c}})$ and the random action with probability $\beta\epsilon^{\text{rl}} + (1-\beta)\epsilon^{\text{c}}$.

Eventually, we obtain the pCTQL policy defined as

$$\pi(x) = \begin{cases} \arg\max_{u \in \mathscr{U}} Q_k(x,u), & \text{with probability } \beta(1-\epsilon^{\text{rl}}), \\ g(x), & \text{with probability } \omega := (1-\beta)(1-\epsilon^{\text{c}}), \\ u \sim \text{rand}(\mathscr{U}), & \text{otherwise}, \end{cases} \tag{5.10}$$

Note that it is also possible to introduce a dependency of the probability β on the current state, time, or other quantities.

5.3 Control tutored deep Q-networks

In the Control-Tutored Deep Q-Networks (CT-DQN), we leverage a Deep Q-Network (DQN) algorithm which uses Deep Neural Networks to iteratively approximate the function Q; it is among the most popular implementations of DRL and can be used also for continuous state spaces $\mathscr{X}$ [59].

Differently from tabular *Q-learning* [98], there are currently no guarantees of convergence towards the optimal policy for DQN [25], although its effectiveness is supported by strong empirical evidence [59, 58]. This lack of theoretical guarantees are related to the introduction of a neural approximator which needs to be tuned. In CT-DQN, during the learning phase, we allow the agent to collect samples according to a tutored policy defined as in (5.10). This minimal change in DQN allows us to run other learning mechanisms (eg. target networks, experience replay [59]) in combination with a tutored exploration process to improve data efficiency and shows the simplicity of our approach to couple control-tutor policies with state of the art RL algorithms.

Next, we proceed to validate all the algorithms presented on some benchmark problems provided in [65]. In particular, we test the CT-DQN for the Pendulum, Lunar Lander and Car Racing problems to show different aspects of our approach related to the complexity of the different scenarios considered.

5.4 Numerical simulations

We start by evaluating the performance of the CTQL and pCTQL algorithms introduced in Sections 5.1 and 5.2 to solve the herding problem of Section 3.4 and the stabilization of an inverted pendulum as defined in Section 3.3.1.

Moreover, to test the CT-DQN of Section 5.3, we also consider the problem of landing a spacecraft described in Section 3.3.2 and the problem of driving a car on a given track described in Section 3.3.3.

5.4.1 Herding scenario

For this scenario, we apply the tabular approaches QL, CTQL, and pCTQL.

Control tutor design

The design of the tutoring control law requires some model of the expected dynamics of the targets. We assume that herders only possess a very rough estimate of the target's true dynamics.

Therefore, in determining their inputs the herders assume the following target dynamics:

$$\dot{x}_t = \delta(x_t - x_h)U(x_t, x_h, \rho_{er}) \tag{5.11}$$

where $\delta = 1$ stands for the intensity of the coupling with the herder, $U(\cdot)$ is the step function defined in (3.14), with $\rho_{er} = 1\text{m}$ being the expected influence radius between targets and herders which is different from the true influence radius ρ_t in (3.14).

Considering the following Lyapunov candidate function:

$$V = \frac{1}{2} x_t^T x_t \tag{5.12}$$

assuming the target's dynamics as in (5.11), when herders and targets interact so that $\|x_t - x_h\| < \rho_{er}$ and choosing $\alpha = 1$ w.l.o.g., the derivative of the Lyapunov function is:

$$\dot{V} = x_t^T \dot{x}_t = x_t^T (x_t - x_h) \tag{5.13}$$

Hence, choosing the control law:

$$v(x_t, \dot{x}_t) = k\dot{x}_t, \qquad k > 1 \tag{5.14}$$

from (3.17) we have:

$$x_h = kx_t \tag{5.15}$$

which assures the derivative $\dot{V}$ in (5.13) is definite negative. That is, with this choice of the herder dynamics, the relative distance between the herder and the goal center would decay to zero if (5.11) were a good model of the target dynamics.

Numerical results

For the herding problem, we test the CTQL algorithm and compared its performance with Q-learning. We discretize the state and action spaces as described in Section 3.4

Some very early stage learning performance is depicted in Figure 5.2. As expected, when applied to control the "true" target, the herder driven by (5.14) fails to achieve the desired goal.

Moreover, Figure 5.3 shows that after a number of episodes $E = 5000$, the herder is still far from the control goal.

Instead, we now show that, by using (5.14) as a tutoring control law, the $CTQL$ algorithm can instead solve the problem that neither Q-learning nor control could solve by themselves. As a matter of fact, considering the case of one herder interacting with one target, the use of a $CTQL$ approach shows successful behavior from the very first attempt as shown in Figure 5.4.

The performance of the $CTQL$ can be further improved via additional learning trials. For example, after just $E = 200$ episodes, the herder considerably improves its behavior as shown in Figure 5.5 attaining convergence in less than 10s compared with the almost 68s of the very first trial.

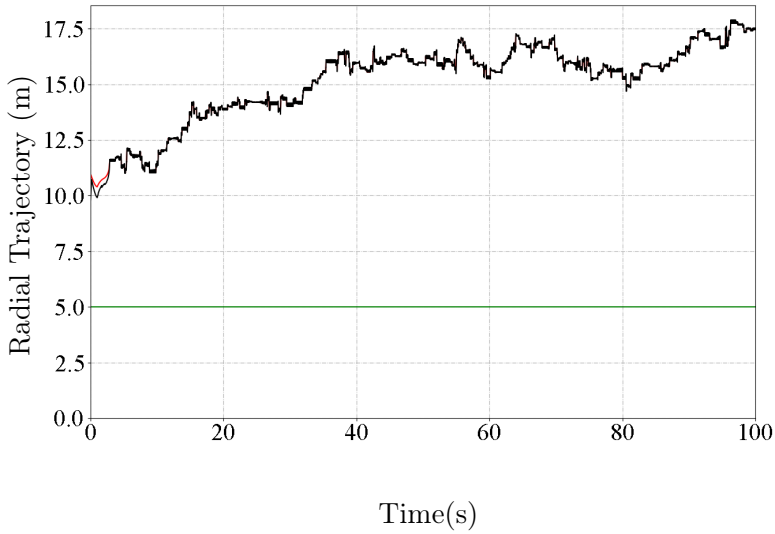


Figure 5.2. The graph shows the radial coordinate of the herder (black line) and the target (red line). The green line represents the circular goal region amplitude. The herder and the target interact in a really close range due to the control law formulation. By itself, the tutor control law is not effective since the herder is not able to push the target inside the goal region.

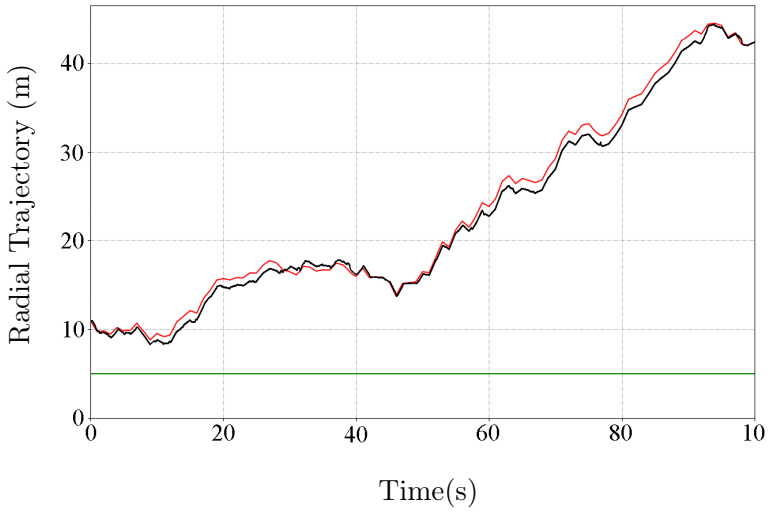


Figure 5.3. The graph shows the radial coordinate of the herder (black line) and the target (red line). The green line represents the circular goal region amplitude. The agent is using a tabular Q-learning which requires lots of episodes to reach convergence.

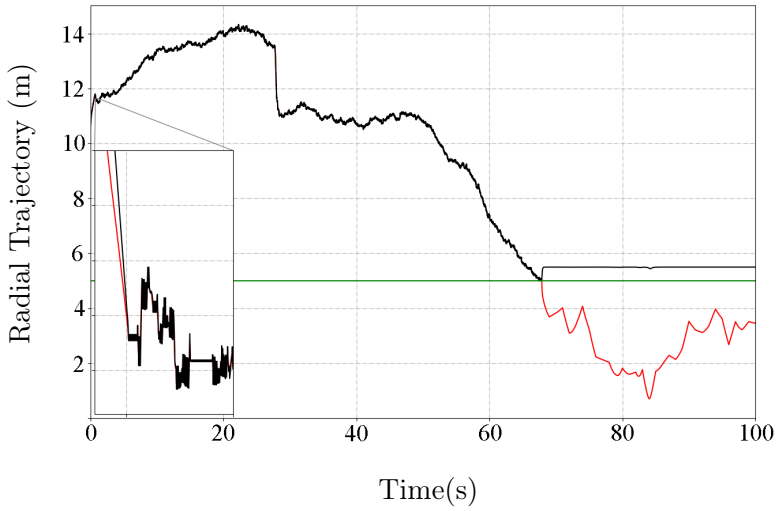


Figure 5.4. The graph shows the radial coordinate of the herder (black line) and the target (red line). The green line represents the circular goal region amplitude. The herder and the target interact at a really close range. As can be seen, the information provided by the control law is able to enhance the learning process and the herder reaches the control goal from the very first attempt.

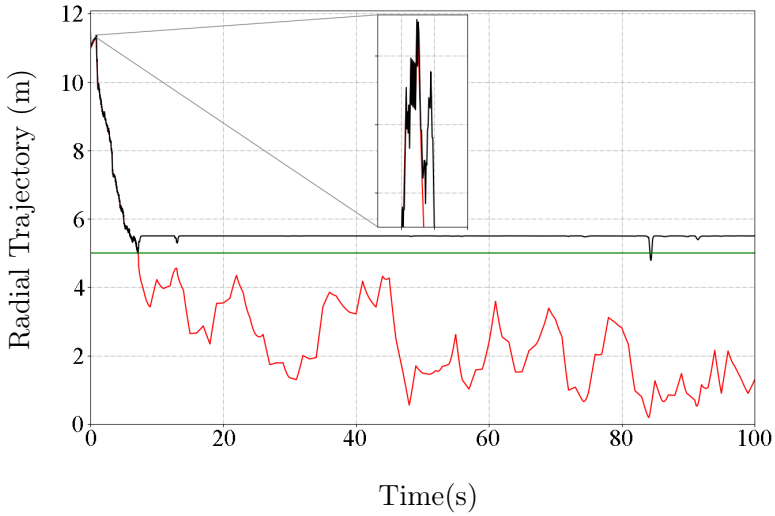


Figure 5.5. The graph shows the radial coordinate of the herder (black line) and the target (red line). The green line represents the circular goal region amplitude. The herder and the target interact at a really close range. After a number of $E = 200$ episodes, the herder considerably improves its performance and takes less time to push the target inside the goal region.

5.4.2 Inverted pendulum scenario

For this scenario, we apply the tabular approaches QL, CTQL, and pCTQL and the deep approaches DQN and CT-DQN. For the parameters of DRL the architectures see Appendix A.1. Numerical experiments are conducted considering $S = 10$ sets of training sessions with the same initial condition made of $E = 10000$ episodes of $N = 400$ steps when we run tabular approaches and of $S = 3$ sets of training sessions with the same initial condition made of $E = 100$ episodes of $N = 400$ steps when we run deep approaches.

Control tutor design

Here we propose the design of feedback control laws based on limited information about the environment considered. We derive such control laws for each of the environments considered in Chap. 3.

Assume we know the linearized dynamics of the pendulum, approximating f in (3.1) close to the upward equilibrium position x^*, namely $\hat{f}(x_k, v_k) = Ax_k + Bv_k$, where $A = \begin{bmatrix} 0 & 1+T \\ 3Tg/2l & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 0 \\ T/I \end{bmatrix}$, with $T = 0.05$ s being the sampling time, $l = 1$ m being the rod length and $I = ml^2/3$ being the moment of inertia of the rod. From this model, using a pole placement technique, we synthesize the linear feedback controller $v_k = -[5.83 \;\; 1.83]x_k$, which can stabilize x^* only locally.

Then $g(x)$ in (5.2) is obtained by projecting $v_k \in \mathbb{R}$ in $\mathscr{U}$ (which is discrete) as follows:

$$g(x) = \arg\min_{u \in \mathscr{U}} \|v_k - u\| \tag{5.16}$$

In a nutshell, whenever it is entitled to, the control-tutor policy selects the action closest to the one generated by the control v_k from the set $\mathscr{U}$. Note that this controller, if used on its own, is unable to swing up the pendulum from its downward asymptotically stable position.

Numerical results

Tabular Approaches. When we consider tabular approaches we use a discretization of states and actions as described in Sec. 3.3.1. We compare Q-learning, CTQL, and pCTQL with different values of ω, when using

reward (3.8) on the inverted pendulum problem described in Sec. 3.3.1 with parameters $\bar{\rho} = 5$ and $\theta = 0.05$.

The results obtained are reported in Figure 5.6 in terms of the cumulative reward per episode J_e^π and the frequency with which the control tutor is used. As shown, the objective J_e^π obtained by control tutored strategies increases faster with respect to the classical Q-learning. Regarding the CTQL algorithm, Fig. 5.7 shows how the frequency of action taken using a control tutor decreases as the episodes increase until reaching a point where it is not used anymore. This ultimately leads to an artificial agent capable of leveraging control tutors to learn faster without introducing a bias in the final learned controller.

Also, a quantitative comparison via the learning metrics is reported in Tab. 5.1 and via the control metrics is reported in Tab. 5.3. For the sake of clarity, in Figures, 5.6 and 5.7 the results of the pCTQL were only plotted for $\omega = 0.01$, as we found that value to give the best performance overall. From Tab. 5.1, comparing CTQL and pCTQL to Q-learning, we observe that E_t—a measure of data efficiency—is smaller (by a statistically significant margin) for the CTQL and for the pCTQL with $\omega = 0.05$; however, the presence of a constant bias from the control tutor in the pCTQL worsens the overall performances which shows a decreasing trend of J^π_avg and $J^\pi_\text{avg,t}$ as ω increases.

We also compared the performance of Q-learning and pCTQL when using reward (3.9); from Tab. 5.2 we see that pCTQL with $\omega = 0.01$ is comparable to Q-learning in terms of learning time (E_t), yet obtains a larger average reward (J^π_avg) and average reward after terminal episode ($J^\pi_\text{avg,t}$), confirming the effectiveness of a control tutor-based architecture, even when the reward has a structure different from (5.7).

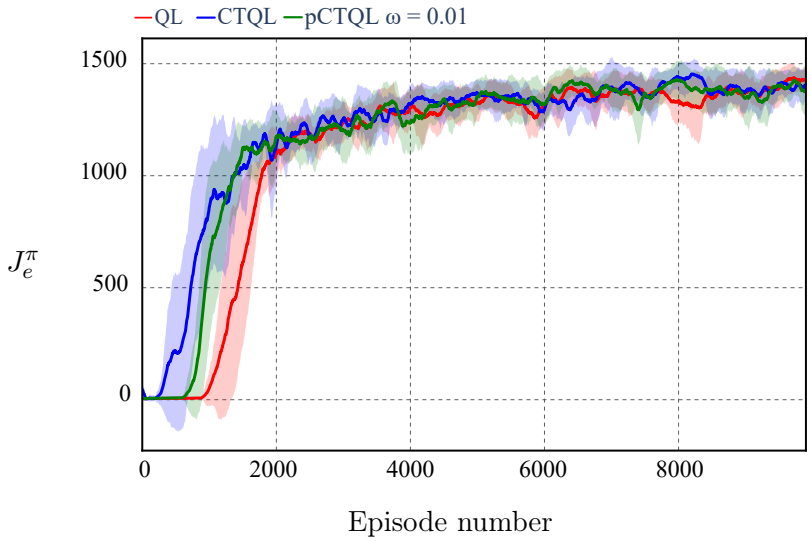


Figure 5.6. Cumulative reward per episode J_e^π, obtained with reward (3.8). The solid curves are the mean of the results of $S = 10$ sessions; for readability, the curves are averaged with a moving average of 100 samples (taken on the right); shaded areas correspond to the means plus or minus the standard deviations.

Algorithm	E_t	J^π_avg	$J^\pi_\mathrm{avg,t}$
	Inverted pendulum		
QL	2726 ± 742	1110 ± 39	1332 ± 45
CTQL	$\mathbf{2028 \pm 241}$	$\mathbf{1195 \pm 58}$	1336 ± 38
pCTQL ($\omega = 0.001$)	2670 ± 562	$\mathbf{1157 \pm 45}$	1358 ± 40
pCTQL ($\omega = 0.005$)	2439 ± 823	$\mathbf{1182 \pm 30}$	1372 ± 45
pCTQL ($\omega = 0.01$)	2106 ± 507	$\mathbf{1164 \pm 57}$	1327 ± 51
pCTQL ($\omega = 0.05$)	$\mathbf{1907 \pm 493}$	1112 ± 16	$\mathbf{1234 \pm 15}$
pCTQL ($\omega = 0.1$)	2952 ± 739	$\mathbf{992 \pm 26}$	$\mathbf{1152 \pm 24}$

Table 5.1. Learning metrics (Def. 3.2.1) for the inverted pendulum scenario, with reward (3.8) and nominal conditions. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of QL are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

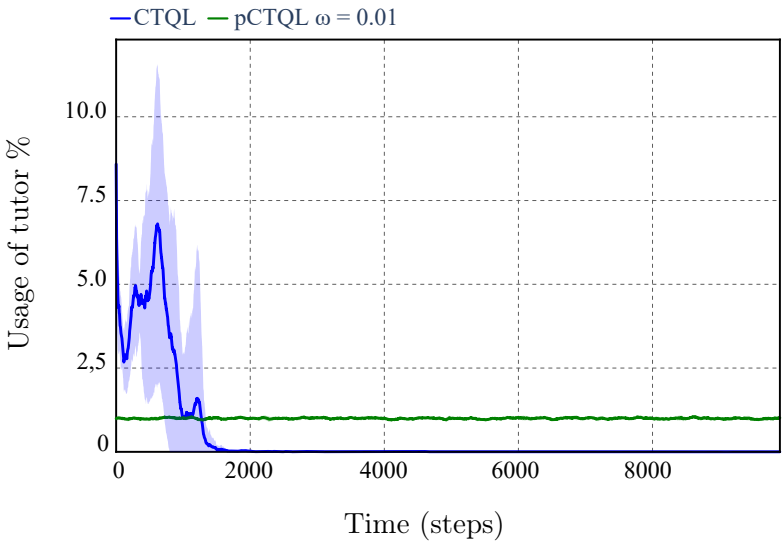


Figure 5.7. Percentage of steps the control-tutor policy π^{c} was used in each episode, with reward (3.8). The solid curves are the mean of the results of $S = 10$ sessions; for readability, the curves are averaged with a moving average of 100 samples (taken on the right); shaded areas correspond to the means plus or minus the standard deviations.

Algorithm	E_{t}	J^{π}_{avg}	$J^{\pi}_{\mathrm{avg,t}}$
	Inverted pendulum		
QL	3207 ± 767	-1045 ± 18	-734 ± 25
pCTQL ($\omega = 0.001$)	3188 ± 692	-1035 ± 11	-723 ± 36
pCTQL ($\omega = 0.005$)	3711 ± 835	-1009 ± 11	-688 ± 29
pCTQL ($\omega = 0.01$)	3684 ± 539	$\mathbf{-1010 \pm 11}$	$\mathbf{-692 \pm 16}$
pCTQL ($\omega = 0.05$)	3684 ± 539	$\mathbf{-1010 \pm 11}$	$\mathbf{-692 \pm 16}$
pCTQL ($\omega = 0.1$)	$\mathbf{4552 \pm 1003}$	$\mathbf{-1028 \pm 11}$	$\mathbf{-777 \pm 23}$

Table 5.2. Learning metrics (Def. 3.2.1) for the inverted pendulum scenario, with reward (3.9) and nominal conditions. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of QL are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

Once the training process has finished, we compare QL, CTQL, and pCTQL algorithms in terms of their control performance at the end of the learning stage, using the metrics given in Definition 3.2.2.

The results, using both rewards (3.8) and (3.9) are reported in Tables 5.3 and 5.4 where we show that, as it is desirable, the differences in settling time (k_g) of pCTQL and CTQL with respect to Q-learning are not statistically significant.

However, when using reward (3.8) we observed that the CTQL achieves the best (lowest) steady-state error, whereas when using reward (3.9) the smallest error is given by the pCTQL with $\omega = 0.01$.

Algorithm	k_{g}	$e_{\mathrm{g}}/x_{\max}$
Inverted pendulum		
QL	112 ± 20	2.6 ± 1
CTQL	118 ± 19	$\mathbf{1.5 \pm 0.6}$
pCTQL ($\omega = 0.001$)	125 ± 37	2.4 ± 1.4
pCTQL ($\omega = 0.005$)	120 ± 25	2.5 ± 1.5
pCTQL ($\omega = 0.01$)	110 ± 25	$\mathbf{3.8 \pm 1}$
pCTQL ($\omega = 0.05$)	111 ± 41	1.9 ± 0.8
pCTQL ($\omega = 0.1$)	105 ± 21	2.5 ± 0.8

Table 5.3. Control metrics (Def. 3.2.2) for the inverted pendulum scenario, with reward (3.8) and nominal conditions. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of QL are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

Deep approaches. Considering again the inverted pendulum problem, we test DQN and CT-DQN considering only reward (3.9).

Fig. 5.8 shows that CT-DQN (with different values of the switching probability ω) and DQN have comparable performance during the learning phase.

Indeed, in Tab. 5.5, a Welch's t-test reveals no statistically significant difference between the two algorithms.

In Tab. 5.6, we report the control metrics assessed after a training process of 50 episodes (larger than E_{t} for all cases, meaning learning is

Algorithm	k_g	e_g/x_{max}
Inverted pendulum		
QL	137 ± 78	1.5 ± 0.6
pCTQL ($\omega = 0.001$)	126 ± 40	$\mathbf{0.9 \pm 0.3}$
pCTQL ($\omega = 0.005$)	150 ± 76	1.4 ± 1.2
pCTQL ($\omega = 0.01$)	114 ± 33	$\mathbf{0.8 \pm 0.3}$
pCTQL ($\omega = 0.05$)	107 ± 20	1.2 ± 0.6
pCTQL ($\omega = 0.1$)	134 ± 27	1.2 ± 0.5

Table 5.4. Control metrics (Def. 3.2.2) for the inverted pendulum scenario, with reward (3.9) and nominal conditions. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of QL are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

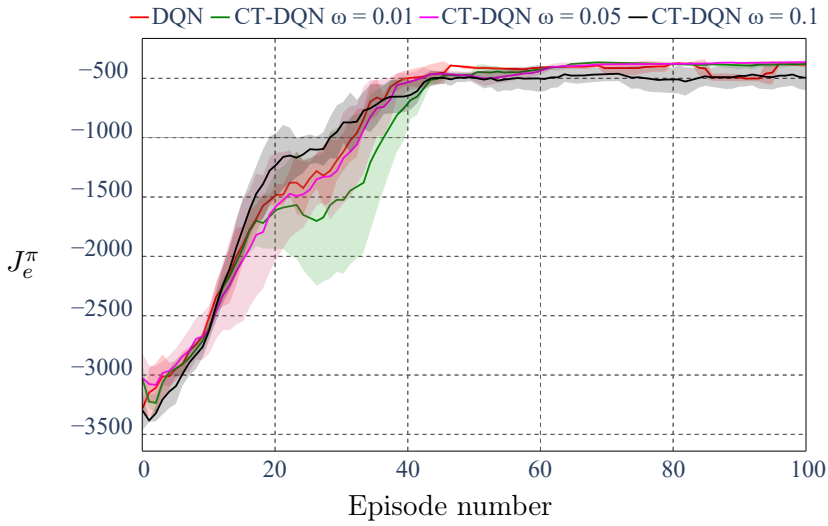


Figure 5.8. Cumulative reward per episode J_e^π for the inverted pendulum problem. The reward curves were averaged with a moving window of 10 samples taken on the left. Then mean (solid curves) and standard deviations (shaded areas) are taken across sessions.

Algorithm	E_{t}	J^{π}_{avg}	$J^{\pi}_{\mathrm{avg,t}}$
	Inverted pendulum		
DQN	38 ± 3	-837.8 ± 90.6	-422.8 ± 16.9
CT-DQN ($\omega = 0.01$)	40 ± 4	-906 ± 110.4	-397.3 ± 8.9
CT-DQN ($\omega = 0.05$)	39 ± 4	-856.5 ± 111.9	-403.1 ± 12.5
CT-DQN ($\omega = 0.1$)	38 ± 6	-858.9 ± 43.3	-507.1 ± 68.6

Table 5.5. Learning metrics (Def. 3.2.1) for the inverted pendulum scenario using DQN and CT-DQN. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of DQN are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

considered complete), and observe similar control performance, without statistically significant differences. Hence, in this scenario, under all metrics considered, CT-DQN and DQN have comparable performance.

We believe this happens because the state and action spaces are small, and DQN is already able to learn quickly, not needing additional aid from the tutor. However, this fact shows that the performance of our control tutor, at worse, does not disrupt the performance of the underlying RL algorithm, in this case, the DQN.

Algorithm	k_{s}	e_{s}	$J^{\pi_{\mathrm{g}}}$
	Inverted pendulum		
DQN	66 ± 3	0.11 ± 0.03	-366.6 ± 9.1
CT-DQN ($\omega = 0.01$)	75 ± 17	0.15 ± 0.03	-407.5 ± 65.3
CT-DQN ($\omega = 0.05$)	70 ± 3	0.16 ± 0.04	-370.3 ± 8.9
CT-DQN ($\omega = 0.1$)	66 ± 0.4	0.15 ± 0.04	-370.2 ± 1.2

Table 5.6. Control metrics (Def. 3.2.2) for the inverted pendulum scenario using DQN and CT-DQN. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of DQN are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

5.4.3 Lunar lander scenario

In this scenario, we test only deep approaches DQN and CT-DQN as the state space is too big to consider a discretized one. For the parameters of DRL the architectures see Appendix A.1. Numerical experiments are conducted considering $S = 3$ sets of training sessions with the same initial condition made of $E = 1000$ episodes of $N = 1000$ steps.

Control tutor design

In order to design the tutor, we assume the knowledge of simplified dynamics of the center of mass of the lander, neglecting gravity (as its magnitude might be unknown).

Namely, we approximate f in (3.2) with the reduced order model $\hat{f}(\chi_k, v_k) = A\chi_k + Bv_k$, where $\chi_k \in \mathbb{R}^4$ is the vector containing position and velocity on the x-axis followed by position and velocity on the y-axis (in this given order); $v_k \in \mathbb{R}^2$ are the x- and y- components of the force applied by a hypothetical swiveling thruster. Noting that $\chi = 0$ corresponds to the center of the landing pad, we exploit the state-feedback control law defined as $v_k = -K\chi_k$ to stabilize asymptotically the origin, where $K \in \mathbb{R}^{2\times 4}$.

The matrices of the reduced order model are defined as follows:

$$A = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.17}$$

$$B = \begin{bmatrix} 0 & 0 \\ T/m & 0 \\ 0 & 0 \\ 0 & T/m \end{bmatrix}, \tag{5.18}$$

$$K = \begin{bmatrix} 470 & 474.7 & 0 & 0 \\ 0 & 0 & 470 & 474.7 \end{bmatrix}, \tag{5.19}$$

where $T = 0.02$ is a sampling time and $m = 10$ kg is the mass of the lander. To obtain the control tutor's input $g(x_k)$ in (5.2) from $v_k(\chi_k)$, we proceed as follows.

If $v_y > 0$ and $|v_y| \geq |v_x|$ (the tutor mainly suggests moving upwards), we use the thruster on the bottom; if $|v_x| > |v_y|$ and $|v_x| > 0$ (the tutor mainly suggests moving right), we use the thruster on the left; if $|v_x| > |v_y|$ and $|v_x| < 0$ (the tutor mainly suggests moving left), we use the thruster on the right; in the other cases, we use no thruster.

Note that this control tutor, by itself, is unable to make the spacecraft land correctly as it has access only to a very limited amount of information on the system dynamics. However, it often helps the lander in getting closer to the landing pad and decreasing landing velocity, as portrayed in Fig. 5.9.

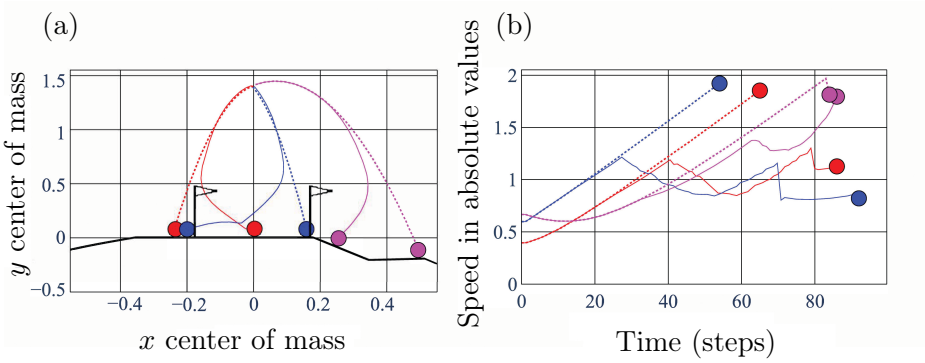


Figure 5.9. (a) Trajectories of the lander obtained using only the control tutor ($u_k = g(x_k)$; solid line) and no control input at all (dotted line). (b) Absolute value of the lander's linear speed obtained using only the control tutor (solid line) and no control input at all (dotted line). Different colors correspond to different initial linear velocities (while keeping the same ground).

Numerical results

Regarding the Lunar Lander problem, Fig. 5.10 shows that CT-DQN improves the learning performance with respect to DQN by reducing learning times. As a matter of fact, the control tutored strategies tested are capable of reaching a higher level in the objective J^π faster with respect to the untutored DQN.

Notably, as reported in Tab. 5.7, CT-DQN with $\omega = 0.05$ requires about half as many episodes as DQN to consistently achieve the goal (see

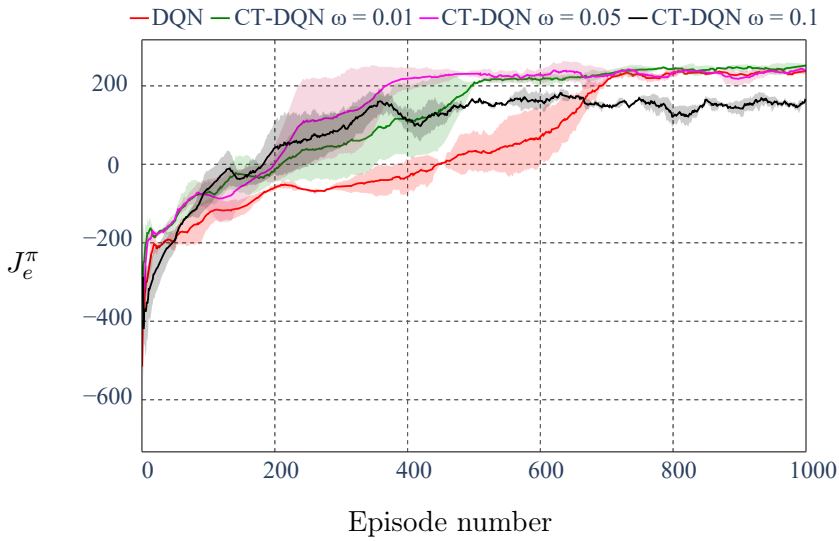


Figure 5.10. Cumulative reward per episode J_e^π for the lunar lander problem. The reward curves were averaged with a moving window of 100 samples taken on the left. Then mean (solid curves) and standard deviations (shaded areas) are calculated across sessions.

E_t). Also, the average cumulative reward across all episodes J^{π}_{avg} of CT-DQN is more than twice that of DQN, indicating a shorter learning time.

Then, after both algorithms reach their terminal episode E_t, they exhibit comparable average cumulative reward $J^{\pi}_{avg,t}$. Moreover, Tab. 5.7 shows that, for the case of $\omega = 0.1$ the artificial agent is not capable of reaching the terminal condition E_t. This happens since the frequency of calls to the control tutor policy is too high and results in an unstable learning process.

Algorithm	E_t	J^{π}_{avg}	$J^{\pi}_{avg,t}$
	Lunar lander		
DQN	665 ± 28	60.1 ± 1.9	231.6 ± 5.2
CT-DQN ($\omega = 0.01$)	$\mathbf{456 \pm 29}$	$\mathbf{135.5 \pm 25.4}$	232.5 ± 0.9
CT-DQN ($\omega = 0.05$)	$\mathbf{324 \pm 90}$	$\mathbf{155.7 \pm 25.9}$	230 ± 9.4
CT-DQN ($\omega = 0.1$)	N.A.	$\mathbf{100 \pm 13.6}$	N.A.

Table 5.7. Learning metrics (Def. 3.2.1) for the scenarios considered. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of DQN are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

In Tab. 5.8 we compare the control strategies obtained from CT-DQN and DQN. To do so, we use the *greedy policy* (as defined in π_g (3.6)) obtained by halting the training at 500 episodes (after 500 episodes, CT-DQN already converged, as its $E_t < 500$, while DQN has not, as its $E_t > 500$).

At this stage of training, the DQN agent could not learn how to land yet but kept hovering over the landing pad, wasting fuel. This is captured by the negative cumulative reward J^{π_g}, coupled with a low steady state error e_s, and the settling time k_s being not available in Tab. 5.8.

On the other hand, the CT-DQN agent has already learned how to land, even with different values of ω (introduced in Sec. 5.3), displaying a positive J^{π_g}, a finite k_s, and a low e_s.

Overall, we note that, as the tutor is synthesized with only a partial model of the system dynamics, performance might start to degrade when the tutor is used *too* often. As evidence, see the asymptotic value of the

Algorithm	k_s	e_s	J^{π_g}
	Lunar lander		
DQN	N.A.	0.18 ± 0.11	-82.5 ± 10.22
CT-DQN ($\omega = 0.01$)	$\mathbf{431 \pm 55}$	0.09 ± 0.01	$\mathbf{180.2 \pm 11.7}$
CT-DQN ($\omega = 0.05$)	$\mathbf{311 \pm 58}$	0.16 ± 0.07	$\mathbf{189.8 \pm 24.1}$
CT-DQN ($\omega = 0.1$)	$\mathbf{529 \pm 82}$	0.14 ± 0.09	$\mathbf{156.7 \pm 28.4}$

Table 5.8. Control metrics (Def. 3.2.2) for the scenarios considered using DQN and pCTDQN. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of DQN are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

reward curves of CT-DQN with $\omega = 0.1$ in Figs. 5.8 and 5.10.

5.4.4 Car racing scenario

In this scenario, we test only deep approaches DQN and CT-DQN as the state space is too big to consider a discretized one. For the parameters of DRL the architectures see Appendix A.1. Numerical experiments are conducted considering $S = 2$ sets of training sessions with the same initial condition made of $E = 500$ episodes of $N = 1000$ steps.

Control tutor design

The tutor regulates acceleration and steering separately.

Steering is regulated as follows. First, we note that the car is still in each frame, is oriented upwards, and has its center of mass at position $p_c = [x_c \quad y_c]^\mathsf{T} = [0 \quad 0]^\mathsf{T}$ (in pixels) (Fig. 5.11). Moreover, we detect the margins of the road by processing each image frame with a Roberts operator [15].

Next, we consider a point in front of the car, with position $p_h = [x_h \quad y_h]^\mathsf{T} = [x_c \quad y_c + l_p]^\mathsf{T}$, with $l_p \in \mathbb{R}_{>0}$ (see Fig. 5.11). We also consider two horizontal lines at $y_h + \Delta y$ and $y_h - \Delta y$, where $\Delta y \in \mathbb{R}> 0$. Normally, these lines will intersect the margins of the road in four points (see again Fig. 5.11), and we define v_{road} as the vector between the intersection points

on the side of the road closer to p_{h}.

Let $\theta := \angle(v_{\mathrm{car}} - v_{\mathrm{road}})$ be the angle of the road with respect to the car. Then, to align the car with the road, if $\theta < 0$ (resp. $\theta > 0$), the tutor suggests steering left (resp. right). However, if all the intersection points are on one side with respect to x_{c}, or if less than four intersection points are found, it is inferred that the car is off the road, and v_{road} is defined as the vector from p_{c} to the closest intersection point, instead (see Fig. 5.11.(b)). [1]

To regulate the speed s, first, we detect s by measuring an indicator bar printed on the image frame. Then, setting some thresholds $\eta^{\mathrm{acc}}_{\mathrm{speed}} = , \eta^{\mathrm{acc}}_{\mathrm{angle}}, \eta^{\mathrm{brk}}_{\mathrm{angle}} \in \mathbb{R}_{>0}$, the tutor suggests to accelerate if $s < \eta^{\mathrm{acc}}_{\mathrm{speed}}$ and $|\theta| < \eta^{\mathrm{acc}}_{\mathrm{angle}}$; conversely, it suggests to brake if $|\theta| > \eta^{\mathrm{brk}}_{\mathrm{angle}}$.

In numerical simulations, we use the following representative values $l_{\mathrm{p}} = 10$ pixels, $\Delta y = 2$ pixels, $\eta^{\mathrm{acc}}_{\mathrm{angle}} = 15°$, $\eta^{\mathrm{brk}}_{\mathrm{angle}} = 50°$, and $\eta^{\mathrm{acc}}_{\mathrm{speed}}$ as 40% of the maximum possible speed.

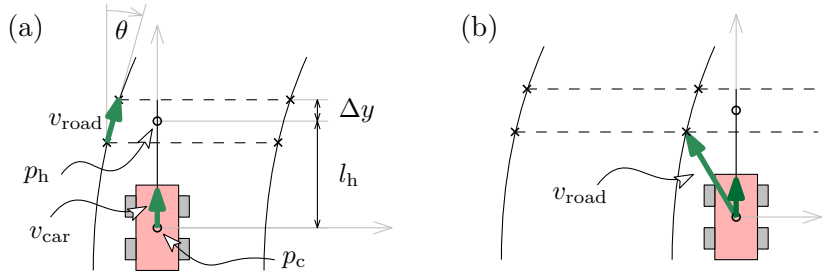


Figure 5.11. Quantities used by the control tutor, when the car is on the road (a) and off the road (b).

[1] More complicated situations might exist, e.g., where less than four intersection points are found, but the car is on the road; however, these are typically infrequent. We also do not aim to build the best possible tutor, but a simple one that is able to demonstrate the potential of the approach. Tutors that are able to provide the learning process with more accurate suggestions will lead to better performance. In a sense, a simple tutor can be considered a baseline over which improvements are possible.

Numerical results

We start by recalling that we consider the task just as that of maximizing the reward, see Sec. 3.3.3.

Results obtained considering the Car Racing scenario, show generally faster learning for CT-DQN in terms of growth of the cumulative reward as shown in Fig. 5.12. This is also confirmed in Tab. 5.9 by the larger value of J^{π}_{avg} for CT-DQN.

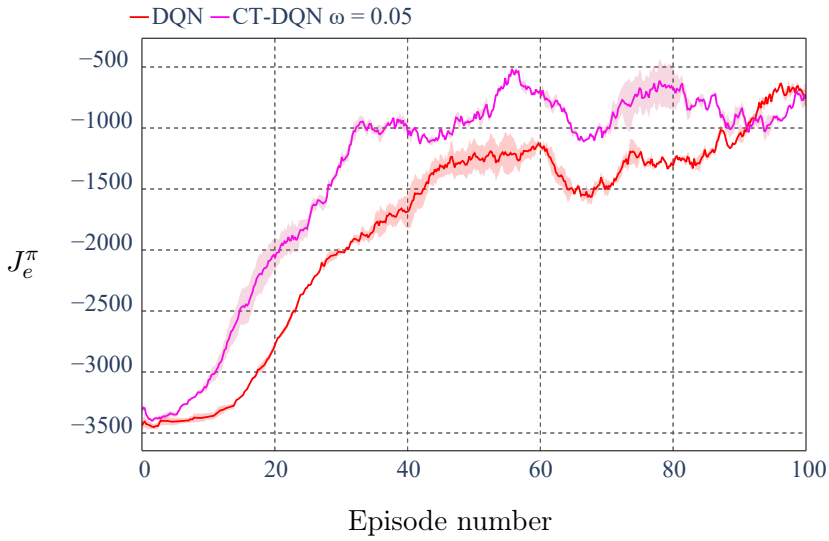


Figure 5.12. Cumulative reward per episode J^{π}_{e} of DQN and CT-DQN. The curves were obtained using a moving average of 50 samples (taken on the left).

To test control performance, we select the greedy policies obtained by CT-DQN after training for 250 episodes, on 30 tracks generated randomly (the same tracks for both algorithms). We find significantly higher rewards (see $J^{\pi_{\text{g}}}$ in Tab. 5.10) for CT-DQN, showing, also for this other scenario, the benefit of using the control tutor.

Algorithm	E_{t}	J^{π}_{avg}	$J^{\pi}_{\mathrm{avg,t}}$
Car racing			
DQN	-	363.7 ± 13.7	-
CT-DQN ($\omega = 0.05$)	-	$\mathbf{451.7 \pm 0.4}$	-

Table 5.9. Learning metrics (Def. 3.2.1) for the car racing scenario. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of DQN are in bold (according to Welch's t-test with p-value less than 0.05 [99]).

Algorithm	k_{s}	e_{s}	$J^{\pi_{\mathrm{g}}}$
Car racing			
DQN	-	-	549.2 ± 290
CT-DQN ($\omega = 0.05$)	-	-	$\mathbf{728 \pm 294.5}$

Table 5.10. Control metrics (Def. 3.2.2) for the car racing scenario using DQN and CTDQN. Means and standard deviations across sessions are reported, when $S > 1$. Values that are statistically significantly different from those of DQN are in bold (according to Welch's t-test with p-value less than 0.05[99]).

5.5 Summary

In this chapter, we introduced the control tutored reinforcement learning, a framework to develop reinforcement learning agents capable of exploring the state space using a control tutor policy. Such policy is generated from a control law based on a very simple model of the system to control. In Section 5.1 we derive the formulation of the CTQL algorithm capable of adjusting the percentage of tutor usage based on the Q-values learned by the agent itself. However, CTQL requires the reward function to be of a specific form that could be prohibitive for some applications. To solve the reward issue, in Section 5.2, we propose a probabilistic alternative to CTQL that we name pCTQL. This algorithm introduces a fixed probability of using actions suggested by the control tutor and does not assume anything about the reward function. Following the same reasoning, in Section 5.3, we also apply to the CTRL idea to more recent deep approaches introducing the CT-DQN.

Eventually, in Section 5.4, we test our algorithms on the testbed problems of Chapter 3 and compare their performance, in both learning and control performances, with their untutored counterparts. Our numerical results suggest that CTRL can reduce learning times by leveraging control laws that on their own are not sufficient to solve the tasks considered. Also, given their off-policy nature, the CTRL algorithms proposed show to be capable of faster convergence without degrading the final control performance.

Chapter 6

COVID-19, modeling and control

During the year of my Ph.D., the COVID-19 pandemic hit the world's population, hence the scientific community shifted its attention to collaborating and finding solutions also via non-pharmaceutical intervention. As part of my research, I worked on the development of a mathematical model of the COVID-19 pandemic and different control-based non-pharmaceutical strategies to limit the effects of the pandemic in terms of deaths and economic impact on the Italian nation.

6.1 Introduction

Regionalism is an integral part of the Italian constitution. Each of Italy's twenty administrative regions is independent of Health and oversees its own share of the Italian National Health Service. The regional presidents and their councils can independently take their own actions, strengthening or, at times, weakening national containment rules. Previous studies have modeled the spread of the epidemics and its evolution in the country at the national level [31, 38, 60, 67], and some have looked at the effects of different types of containment and mitigation strategies [12, 9, 90, 89, 71]. Limited work [5, 27, 30, 37, 46, 54, 77, 78, 93, 96] has taken into account the spatial dynamics of the epidemic but, to the best of our knowledge, no previous paper in the literature has explicitly taken

into consideration the pseudo-federalist nature of the Italian Republic and its strong regional heterogeneity when it comes to health matters, hospital capacity, economic costs of a lockdown and the presence of inter-regional people's flows.

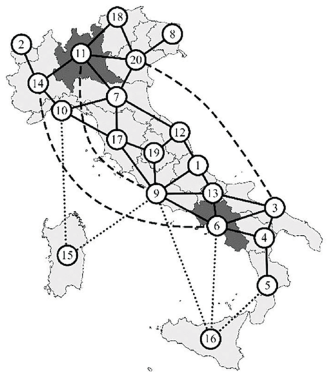
The goal is to identify if and when measures taken by the Italian government had an effect at both the national, but most importantly, at the regional level. Also, we want to uncover the effects of the epidemic spread of regional heterogeneity and inter-regional flows of people and use control theoretic tools to propose and assess differentiated interventions at the regional level to reopen the country and avoid future recurrent epidemic outbreaks.

As aggregate models of the COVID-19 epidemic cannot capture these effects, to carry out our study we derived and parameterized from real data a network model of the epidemics in the country (see Figure 6.1), where each of the 20 regions is a node and links model both proximity flows and long-distance transportation routes (ferries, train, planes). The model is first shown to possess the right level of granularity and complexity to capture the crucial elements needed to correctly predict and reproduce the available data. Then, it is used to design and test differentiated feedback interventions at the regional level to alleviate the epidemic impact. Using the model and an ad hoc algorithm to parameterize it from real data, we evaluate the effectiveness of the national lockdown strategy implemented so far by the Italian government providing evidence of its efficacy across regions. Also, we show that inter-regional fluxes must be carefully controlled as they can have dramatic effects on recurrent epidemic waves. Finally, we convincingly show that regional feedback interventions, where each of the twenty regions strengthens or weakens local mitigating actions (social distancing, inflow/outflow control) as a function of the saturation of their hospital capacity, can be beneficial in mitigating possible outbreaks and in avoiding recurrent epidemic waves while reducing the costs of a nationwide lockdown.

6.2 COVID-19: modeling

As a regional model of the COVID-19 epidemic spread we use the compartmental model shown in Figure 6.2 which we found from data analysis

(a)



(b)

Index	Region	Index	Region
1	Abruzzo	11	Lombardy
2	Aosta Valley	12	Marche
3	Apulia	13	Molise
4	Basilicata	14	Piedmont
5	Calabria	15	Sardinia
6	Campania	16	Sicily
7	Emilia	17	Tuscany
8	Friuli - Venezia Giulia	18	Trentino - Alto Adige
9	Lazio	19	Umbria
10	Liguria	20	Veneto

Figure 6.1. Schematic diagram of the network model structure and representative regional parameters. (a). Representative graph of the network model structure used in the paper. Only a subset of all links is shown for the sake of clarity (see [19] for the complete graphs). Solid lines represent proximity links, dashed lines long distance transportation routes (air, train, road), and dotted lines show major ferry routes between insular regions and the Italian mainland. (b). Table of the Italian region names and their positions in the graph.

and identification trials to be the simplest model structure able to capture the real [19]. The full model equations describe the dynamics of suscepti-

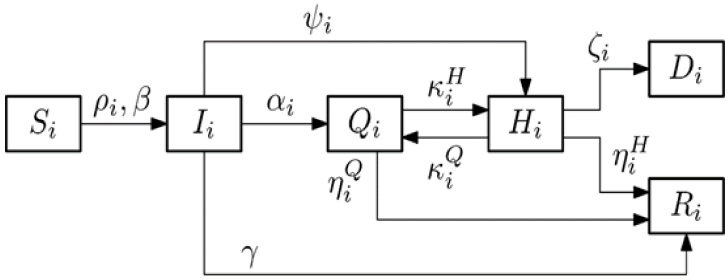


Figure 6.2. Regional compartmental model structure adopted in our study. Schematic structure of model described by equations (6.1). Compartments describe the dynamics of susceptible (S_i), infected (I_i), quarantined (Q_i), hospitalized (H_i), recovered (R_i), and deceased (D_i).

ble (S_i), infected (I_i), quarantined (Q_i), hospitalized (H_i), recovered (R_i) and deceased (D_i) and are given as:

$$S(t+1)_i = -\rho_i \beta \frac{S(t)_i I(t)_i}{N_i} \tag{6.1}$$

$$I(t+1)_i = \rho_i \beta \frac{S(t)_i I(t)_i}{N_i} - \alpha_i I(t)_i - \psi_i I(t)_i - \gamma I(t)_i \tag{6.2}$$

$$Q(t+1)_i = \alpha_i I(t)_i - \kappa_i^H Q(t)_i - \eta_i^Q Q(t)_i + \kappa_i^Q H(t)_i \tag{6.3}$$

$$H(t+1)_i = \kappa_i^H Q(t)_i + \psi_i I(t)_i - \eta_i^H H(t)_i - \zeta_i H(t)_i - \kappa_i^Q H(t)_i \tag{6.4}$$

$$D(t+1)_i = \zeta_i H(t)_i \tag{6.5}$$

$$R(t+1)_i = \gamma I(t)_i + \eta_i^Q Q(t)_i + \eta_i^H H(t)_i \tag{6.6}$$

where β is the infection rate which will be assumed to be the same for all regions since COVID-19 is transmitted from person to person and there

is no parasite vector or evidence of environmental parameters significantly altering its infection rate, $\rho_i \in [0,1]$ is a parameter modeling the effects of social distancing measures in the i-th region, α_i is the rate of infected that are detected and quarantined, ψ_i is the rate of infected that needs to be hospitalized, γ is the recovery rate of the infected that is assumed to be equal for all regions, η_i^Q is the rate of quarantined who recover, η_i^K is the fraction of hospitalized who recover, κ_i^Q is the rate of hospitalized that is transferred to home isolation, κ_i^H is the rate of quarantined who need to be hospitalized, and ζ_i is the mortality rate that was shown from data analysis to be a function of the ratio between H_i and the maximum number, say T_i^H, of patients that can be treated in ICU at the hospitals in i-th region. N_i is the actual population in the i-th region, i.e. the resident population without those removed because quarantined, hospitalized, deceased, or recovered.

Extending previous approaches for modeling Dengue fever in Brazil [83], we obtain the national network model of the COVID-19 epidemic in Italy as a network of twenty regions (see Figure 6.1) interconnected by links modeling commuter flows and major transportation routes among them. The dynamics of the disease in each region $i \in \{1,\dots,M\}$ is captured by a SIQHDR model, describing the discrete-time evolution of the number of susceptible (S_i), (undetected) infected (I_i), quarantined (Q_i), hospitalized (H_i), deceased (D_i), and recovered (R_i) as a function of time $t \in \mathbb{N}_{\geq 0}$ (in days). We also define $S \coloneqq [S_1 \ \cdots \ S_M]^\mathsf{T}$, $I \coloneqq [I_1 \ \cdots \ I_M]^\mathsf{T}$, $Q \coloneqq [Q_1 \ \cdots \ Q_M]^\mathsf{T}$, $H \coloneqq [H_1 \ \cdots \ H_M]^\mathsf{T}$, $D \coloneqq [D_1 \ \cdots \ D_M]^\mathsf{T}$, $R \coloneqq [R_1 \ \cdots \ R_M]^\mathsf{T}$, and let $x \coloneqq [S^\mathsf{T} \ I^\mathsf{T} \ Q^\mathsf{T} \ H^\mathsf{T} \ D^\mathsf{T} \ R^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{6M}_{\geq 0}$. Moreover, for each pair (i,j) of regions, $\phi_{ij}(t) \in [0,1]$ describes the fraction of people in the region i that commute and interact with people in the region j, returning to their region of origin at the end of each day t (see also [83]).

The resulting network model is

$$S_i(t+1) = S_i(t) - \beta S_i(t) \sum_{j=1}^{M} \frac{\rho_j(t)\phi_{ij}(t)}{N_j^{\mathrm{p}}(t)} \sum_{k=1}^{M} \phi_{kj}(t) I_k(t), \tag{6.7a}$$

$$I_i(t+1) = I_i(t) + \beta S_i(t) \sum_{j=1}^{M} \frac{\rho_j(t)\phi_{ij}(t)}{N_j^{\mathrm{p}}(t)} \sum_{k=1}^{M} \phi_{kj}(t) I_k(t) - (\gamma + \alpha_i(t) + \psi_i) I_i(t), \tag{6.7b}$$

$$Q_i(t+1) = Q_i(t) + \alpha_i(t) I_i(t) - (\kappa_i^H + \eta_i^Q) Q_i(t) + \kappa_i^Q H_i(t), \tag{6.7c}$$

$$H_i(t+1) = H_i(t) + \kappa_i^H Q_i(t) + \psi_i I_i(t) - \left(\eta_i^H + \kappa_i^Q + \zeta\left(H_i(t)\right)\right) H_i(t), \tag{6.7d}$$

$$D_i(t+1) = D_i(t) + \zeta\left(H_i(t)\right) H_i(t), \tag{6.7e}$$

$$R_i(t+1) = R_i(t) + \gamma I_i(t) + \eta_i^Q Q_i(t) + \eta_i^H H_i(t), \tag{6.7f}$$

where

$$N_i^{\mathrm{p}}(t) = \sum_{k=1}^{M} \phi_{ki}(t) \left(S_k(t) + I_k(t) + R_k(t)\right), \tag{6.8}$$

and the initial conditions are given at time $t_0 \in \mathbb{N}_{\geq 0}$ as $\mathbf{x}(t_0) = \mathbf{x}^0 \in \mathbb{R}^{6M}_{\geq 0}$.

In (6.7) the infection rate β and the recovery rate γ are taken to be equal for all the regions, as they have not been shown to strongly depend on regional factors. The other parameters vary among different regions to capture the heterogeneity of containment strategies, different regional healthcare systems, etc. In particular, note that

$$\gamma + \alpha_i(t) + \psi_i \leq 1, \quad \forall i \in \{1, \ldots, M\}, \ \forall t \in \mathbb{N}_{\geq 0}, \tag{6.9}$$

as in (6.7b) the number of people leaving compartment I_i can never be greater than I_i. Moreover, as done in [19], we also estimate the number of patients requiring treatment in intensive care as 10% of those who are hospitalized. Thus, letting T_i^H be the maximum number of beds in the ICUs of the region i [56, 55], following [19] we express the mortality rate as

$$\zeta(H_i(t)) = \zeta^0 + \zeta^{\mathrm{b}} \min\left\{\frac{0.1 H_i(t)}{T_i^H}, 1\right\}. \tag{6.10}$$

Finally, $N_i^{\mathrm{p}}(t)$ is the population in the region i that is free-to-move at time t, whereas N_i is the total population in the region i. All parameters' values are identified from data reported in [21] and are contained in Table 6.1. See [19] for more details about the identification procedure.

Table 6.1. Model parameter values; for a description of all parameters see [19]. The values of T_i^H are reported from [2]

Region	ρ_i	η_i^Q	η_i^H	α_i	ψ_i	κ_i^H	κ_i^Q
Abruzzo	0,321	0,010	0,000	0,025	0,049	0,000	0,087
Aosta	0,122	0,010	0,260	0,062	0,000	0,079	0,000
Apulia	0,590	0,010	0,000	0,028	0,047	0,100	0,100
Basilicata	0,177	0,010	0,060	0,037	0,021	0,025	0,025
Calabria	0,272	0,015	0,000	0,042	0,059	0,005	0,079
Campania	0,467	0,018	0,000	0,014	0,064	0,000	0,100
Emilia	0,400	0,029	0,000	0,059	0,062	0,000	0,045
Friuli	0,202	0,028	0,049	0,044	0,007	0,004	0,000
Lazio	0,483	0,015	0,012	0,029	0,076	0,055	0,100
Liguria	0,398	0,037	0,010	0,012	0,092	0,000	0,100
Lombardy	0,308	0,022	0,017	0,017	0,059	0,000	0,048
Marche	0,133	0,010	0,007	0,000	0,057	0,002	0,068
Molise	0,217	0,013	0,000	0,06	0,018	0,000	0,043
Piedmont	0,363	0,022	0,014	0,021	0,071	0,000	0,100
Sardinia	0,216	0,013	0,038	0,066	0,017	0,015	0,063
Sicliy	0,293	0,015	0,000	0,017	0,068	0,012	0,100
Trentino	0,226	0,029	0,035	0,081	0,006	0,002	0,000
Tuscany	0,353	0,012	0,000	0,046	0,062	0,000	0,093
Umbria	0,134	0,010	0,141	0,089	0,000	0,052	0,000
Veneto	0,336	0,031	0,000	0,054	0,048	0,002	0,100

6.3 COVID-19: control via non-pharmaceutical interventions

A crucial open problem is to support decision makers in determining what form of interventions might be beneficial to avoid the onset of future outbreaks while mitigating the cost of Draconian interventions at the national level. To this aim, we compared the effects of national measures (e.g., general lockdown) against those of a regional feedback strategy where social distancing measures are put in place or relaxed independently by each region according to the ratio between hospitalized individuals and the total capacity of the health system in that region.

6.3.1 Implementation and design of national and regional feedback intervention strategies

We model the implementation of regional social distancing strategies by capturing their effects as a variation of the social distancing parameters, ρ_i in (6.7), in each region. Specifically, we assume each region follows the feedback control rule:

$$\rho_i = \begin{cases} \underline{\rho}_i, & \text{if} \quad \dfrac{0.1H_i}{T_i^H} \geq 0.20 \\ \bar{\rho}_i, & \text{if} \quad \dfrac{0.1H_i}{T_i^H} \leq 0.05 \end{cases} \tag{6.11}$$

where $\underline{\rho}_i$ is set equal to the minimum estimated value in that region during the national lockdown (see [19]) and ρ_i increased as a worst case to $min(1, 3\underline{\rho}_i)$ so as to simulate the effect of relaxing the lock-down measures in each region.

Also, when a region is shut down, we assume all fluxes in and out of that region are reduced to 70% of their original values to better simulate the actual reduction in people's movement observed during the lockdown in Italy (for further details see [19]).

National lockdown measures are modeled by setting all ρ_i simultaneously to $\underline{\rho}_i$ in all regions and reducing all fluxes by 70% while the national reopening of all regions by setting all ρ_i simultaneously to $\bar{\rho}_i$ and restoring interregional flows to their pre-lockdown level.

We assume each region implements a stricter lockdown when such a ratio becomes greater or equal to 20% and relaxes the social distancing rules when it is below 5% (see [19] for further details). Figure 6.3 confirms the effectiveness of such a local strategy, where we see that a differentiated strategy among the regions is more effective than a national lockdown in avoiding future waves of the disease (Figure 6.4) but, most importantly, also in guaranteeing that no region exceeds its own hospitals' capacity. Moreover, intermittent regional measures yield lower economic costs for the country, as regional economies can be restarted and remain open for a much longer time (Table 6.2).

6.4 Summary

In this chapter, we show the effort made during the COVID-19 pandemic to effectively synthesize non-pharmaceutical interventions. In Section 6.2 we proposed an original formulation of the COVID-19 pandemic compartmental model. We consider the case of Italy by considering a network of 20 independent regions that exchange people with each other. Furthermore, in Section 6.3 we propose a bang-bang strategy to control the epidemic based on the level of hospitalization level of each region. Eventually, based on such information, the nodes of the network considered can limit flows with the neighbor nodes as well as adjust its level of lockdown measure.

With this study, we effectively demonstrated, that, with the same initial conditions, decentralized approaches can lower deaths and economic costs with respect to national measures imposed on all regions. Finally, We show these results via numerical simulation on multiple sets of parameters to obtain reliable statistics (see Appendix A.2).

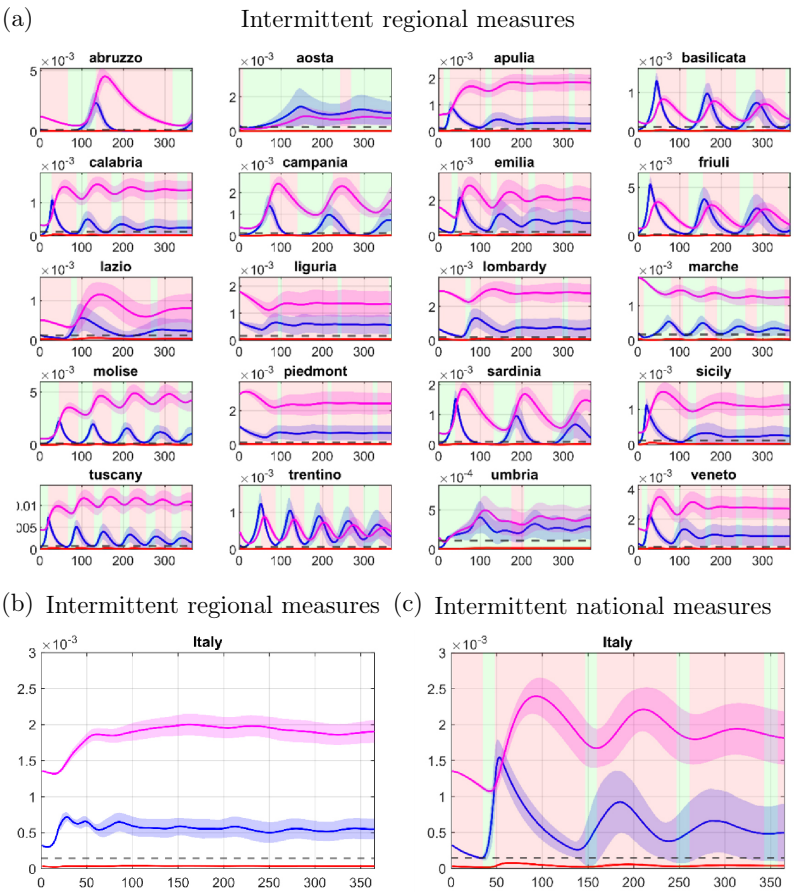


Figure 6.3. Intermittent regional measures. (a). Each of the 20 panels shows the evolution in each region of the fraction in the population of infected (blue), quarantined (magenta), and hospitalized requiring ICUs (red) averaged over 10,000 simulations with parameters sampled using a Latin Hypercube technique (see Appendix A.2 and [19]). Shaded bands correspond to twice the standard deviation. Dashed black lines represent line the fraction of the population that can be treated in ICU (T_i^H/N_i) (b). National evolution of the fraction in the population of infected (blue), quarantined (magenta), and hospitalized requiring ICUs (red) was obtained by summing those in each of the 20 regions adopting intermittent regional measures. (c). National evolution when an intermittent national lockdown is enforced with all regions shutting down when the total number of occupied ICU beds at the national level exceeds 20%.

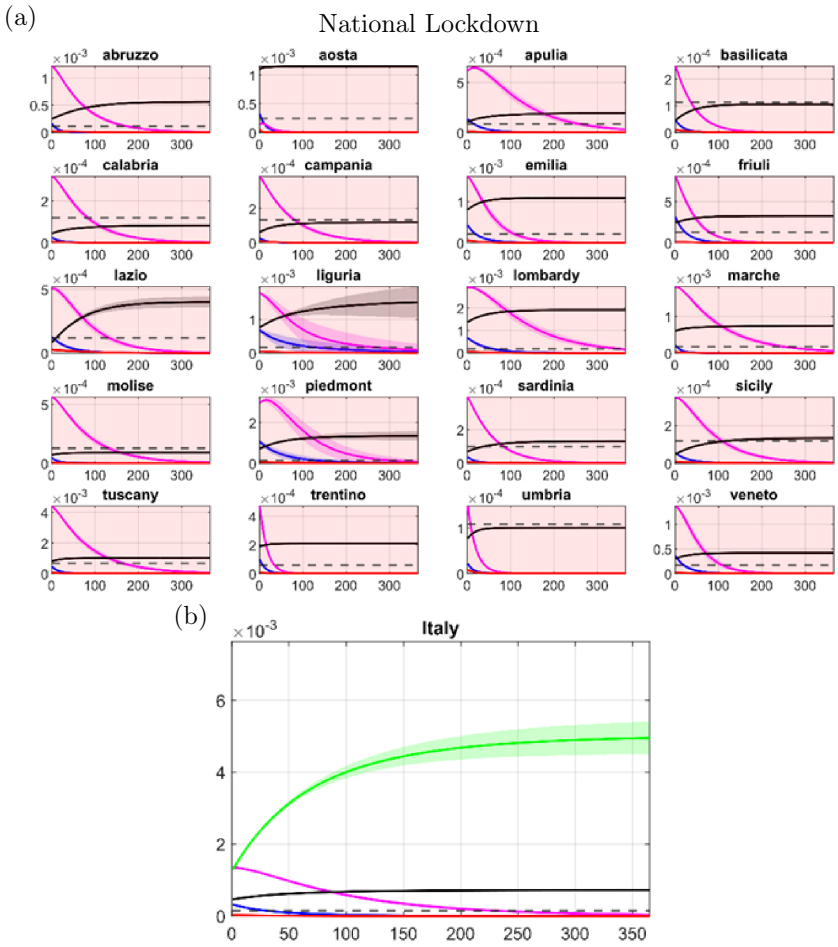


Figure 6.4. National lockdown. (a). Regional and (b). national dynamics in the case where no region relaxes its containment measures, while all regions restore the interregional fluxes to their pre-lockdown level. Blue, magenta, red, green, and black solid lines correspond to the fraction in the population of infected, quarantined, hospitalized, recovered, and deceased averaged over 10,000 simulations with parameters sampled using a Latin Hypercube technique (see Appendix A.2) around their nominal values set as those estimated in the last time window for each region (see [19]). Shaded bands correspond to twice the standard deviation. The black dashed line identifies the fraction of the population that can be treated in ICU (T_i^H/N_i). The regions identified with a red label are those where the total hospital capacity is saturated.

Table 6.2. Comparison of the simulated scenarios. Metrics (calculated according to [19, 14]) to evaluate the impact over 1 year are reported showing the effectiveness of the intermittent regional measures in avoiding any saturation of the regional health systems while mitigating the impact of the epidemic. Average values are shown ± standard deviation calculated from 10,000 repetitions with parameter values sampled using a Latin Hypercube (see Appendix A.2).

Simulation	Total cases	Total deaths	Maximum hospitalized	Days over hospital's capacity	Regions over hospital's capacity	Economic cost [M€]
	x	x				
Intermittent regional measures (Fig. 6.3.a,b)	$2,165,229 \pm 83,806$	$154,878 \pm 3,008$	$2,927 \pm 183$	0 ± 0	0	$470,735 \pm 6,353$
Intermittent national measures (Fig. 6.3.c)	$2,197,076 \pm 189,948$	$176,210 \pm 8,264$	$4,794 \pm 309$	0 ± 0	3	$532,802 \pm 12,474$
National lockdown (Fig. 6.4)	$345,552 \pm 29,841$	$43,705 \pm 1,715$	$1,915 \pm 0$	0 ± 0	0	$610,480 \pm 0$

Chapter 7

Conclusions and Future Work

In Chapter 2, I reviewed the state of the art in reinforcement learning for control applications. In particular, I found out that current control solutions provided by reinforcement learning algorithms come with long learning times, big sets of data, and no final performance guarantees.

To develop our algorithms, in Chapter 3, I introduced a problem formulation in the sense of optimal control for reinforcement learning problems establishing a link between the two disciplines. I also defined a set of metrics to quantify and compare our algorithms with the ones provided by the literature. Moreover, I introduced a set of benchmark problems to analyze and test the algorithms proposed in this thesis.

A set of analytical results have been proposed in Chapter 4 to drive a pipeline for defining reinforcement learning problems capable of automatically detecting when its performance has reached a minimum desired level. In numerical simulations, described in Section 4.3.1, I showed how the minimal performance Q-learning can autodetect and deliver a solution that fulfills some desired properties (e.g. steady-state error and settling time). Further steps in this direction would be to apply this procedure to different architectures rather than Q-learning. Eventually, we would like to also embed in the reward function safety so that the reinforcement learning agent can freely learn and autodetect unsafe behavior.

In Chapter 5, I presented a deterministic and a probabilistic control tutored Q-learning strategy for tabular approaches and a control tutored deep Q-networks, that integrate a feedback control law synthesized on

a partial model of the plant within a Q-learning framework to render the learning process faster and improve the performance of the learned policies in achieving a control goal of interest.

I compared the control-tutored strategies with a classical Q-learning and deep Q-networks approach using the inverted pendulum, lunar lander, and car racing benchmark problems from OpenAI Gym as representative control problems. Also, a solution to the single agent herding problem is proposed. From numerical simulations of Section 5.4, I found that when compared to Q-learning, CTQL requires fewer data samples and has a larger average reward, while pCTQL yields higher rewards with a comparable number of data samples; moreover, both CTQL and pCTQL yield lower regulation errors when certain reward functions are used. Moreover, regarding CT-DQN, I have shown that the addition of control tutors always proved to be non-pejorative (with the inverted pendulum) or significantly beneficial (with the lunar lander and racing car) in terms of shorter learning time. As a matter of fact, I observed that CT-DQN is able to obtain better policies with respect to classical DQN in the same number of episodes. Moreover, the better the tutor is at solving a problem (according to case-specific metrics), the larger the improvement tends to be. Our numerical results show that both from a learning and a control viewpoint using a control-tutored learning approach might be beneficial.

Further work will be focused on the formal analysis of the design of the tutor mechanism for Deep Reinforcement Learning, including the quantification of information and definition of bounds (e.g., regret bounds). Also, we wish to uncover and formally characterize the relationships among the specific choice of the reward function, the performance of the algorithms, and the approximate system dynamics needed to synthesize the control tutor. Furthermore, we wish to emphasize that embedding a control tutor in the loop could be used to render more efficient learning strategies other than Q-learning. This will also be the subject of future investigation.

Finally, an event like the COVID-19 pandemic raised many issues in current governmental organizations and society. However, it has been an opportunity to develop and test state of the art control solutions. In Chapter 6, I analyzed how the decentralization of governmental policy can be beneficial in the administration of an emergency like the COVID-19 pandemic. Moreover, as the next steps, I will keep working on the

definition of better data-driven approaches ready to use in new emergency situations that may appear in the future.

Appendix A

Appendix

A.1 Parameters of deep architectures

During training, we use a target neural network [59] which is updated at the end of every episode. We also introduce a replay buffer with size N_b, which is used to randomly sample 64 data-points to update the network parameters at every step. Moreover, we set learning rate $\alpha = 0.001$ and the discount factor $\gamma = 0.99$ [87].

With respect to the inverted pendulum of Section 5.4.2, for the neural networks in DQN, we use 2 hidden layers with rectifier linear unit activation functions (ReLu), with 128 and 64 nodes, respectively. We set $N_b = 1,000,000$ and $\epsilon^{rl} = 0.02$.

As far as the lunar lander in Section 5.4.3 is concerned, for the neural networks in DQN, we used 2 hidden layers of 128 nodes with ReLu. Moreover, we set $N_b = 1,000,000$, $\alpha = 0.0001$, $\gamma = 0.99$, and $\epsilon^{rl} = 0.1$.

Finally for the car racing of Section 5.4.4, we use convolutional neural networks. The input has dimension $94 \times 94 \times 3$. A first hidden layer convolves 6 filters of 7×7 with stride 3 with the input image, with ReLu. A second hidden layer convolves 12 filters of 4×4 with stride 1, with ReLu. A third hidden layer is present, with 216 nodes and ReLu. The output layer is a fully-connected linear layer with a single output for each possible action. Finally, we set $N_b = 5,000$, $\alpha = 0.001$, $\gamma = 0.9999$, and $\epsilon^{rl} = 0.1$.

A.2 Data fitting and sensitivity analysis in COVID-19 experiments

All computational analyses and the fitting of data were performed using MATLAB and its optimization toolbox. To account for the inherent uncertainty associated to the COVID-19 epidemic, each result reported in the manuscript is the output of 10000 numerical simulations, where we varied the values of the model parameters using the Latin Hypercube sampling method[36, 51]. Specifically, the regional parameters $\alpha_i, \psi_i, \kappa_i^Q, \kappa_i^H, \eta_i^Q, \eta_i^H$ of the model (6.7), and the estimated initial condition at May 3rd 2020 $I_(f, i)$ were varied considering a maximum variation of $\pm 20\%$ from their nominal values (see [19]).

Bibliography

[1] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *International Conference on Machine Learning (ICML'06)*, pages 1–8, 2006.

[2] Agenzia Nazionale per i Servizi Sanitari Regionali. Ricoverati e posti letto in area non critica e terapia intensiva, 2021.

[3] Giacomo Albi, Mattia Bongini, Emiliano Cristiani, and Dante Kalise. Invisible control of self-organizing agents leaving unknown environments. *SIAM Journal on Applied Mathematics*, 76(4):1683–1710, 2016.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[5] Julien Arino and Pauline van den Driessche. A multi-city epidemic model. *Mathematical Population Studies*, 10(3):175–193, 2003.

[6] Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[7] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems (NIPS'17)*, volume 30. Curran Associates, Inc., 2017.

[8] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic Programming*. Athena Scientific, 1996.

[9] Michelangelo Bin, Peter Cheung, Emanuele Crisostomi, Pietro Ferraro, Connor Myant, Thomas Parisini, and Robert Shorten. On fast multi-shot

epidemic interventions for post lock-down mitigation: Implications for simple covid-19 models. *arXiv preprint arXiv:2003.09930*, 1125, 2020.

[10] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[11] Lucian Buşoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018.

[12] Francesco Casella. Can the covid-19 epidemic be controlled on the basis of daily test reports? *IEEE Control Systems Letters*, 5(3):1079–1084, 2020.

[13] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. AAAI Press, 2019.

[14] Marco Coraggio, Shihao Xie, Francesco De Lellis, Giovanni Russo, and Mario Di Bernardo. Intermittent non-pharmaceutical strategies to mitigate the covid-19 epidemic in a network model of italy via constrained optimization. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3538–3543. IEEE, 2021.

[15] Larry S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4(3):248–270, 1975.

[16] Francesco De Lellis, Giovanni Russo, and Mario Di Bernardo. Tutoring reinforcement learning via feedback control. In *European Control Conference (ECC'21)*, pages 580–585, 2021.

[17] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[18] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. *International Conference on Machine Learning (ICML'11)*, pages 465–472, 2011.

[19] Fabio Della Rossa, Davide Salzano, Anna Di Meglio, Francesco De Lellis, Marco Coraggio, Carmela Calabrese, Agostino Guarino, Ricardo Cardona-Rivera, Pietro De Lellis, Davide Liuzza, et al. A network model of italy shows that intermittent regional strategies can alleviate the covid-19 epidemic. *Nature communications*, 11(1):5106, 2020.

[20] Shuiguang Deng, Longtao Huang, Guandong Xu, Xindong Wu, and Zhaohui Wu. On deep learning for trust-aware recommendations in social networks. *IEEE transactions on neural networks and learning systems*, 28(5):1164–1177, 2016.

[21] Dipartimento della Protezione Civile. Dipartimento della protezione civile–presidenza del consiglio dei ministri official COVID-19 data repository, 2020.

[22] Hao Dong, Zihan Ding, Shanghang Zhang, and Chang. *Deep Reinforcement Learning*. Springer, 2020.

[23] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. *International Conference on Machine Learning (ICML'16)*, pages 1329–1338, 2016.

[24] Eyal Even-dar and Yishay Mansour. Convergence of optimistic and incremental q-learning. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[25] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control (L4DC'20)*, volume 120 of *Proceedings of Machine Learning Research*, pages 486–489, 2020.

[26] Qirui Fan, Gai Zhou, Tao Gui, Chao Lu, and Alan Pak Tao Lau. Advancing theoretical understanding and practical performance of signal processing for nonlinear optical communications through machine learning. *Nature Communications*, 11(1):3694, 2020.

[27] Ayalvadi Ganesh, Laurent Massoulié, and Don Towsley. The effect of network topology on the spread of epidemics. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1455–1466. IEEE, 2005.

[28] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 16(1):1437–1480, 2015.

[29] Émiland Garrabé and Giovanni Russo. Probabilistic design of optimal sequential decision-making algorithms in learning and control. *Annual Reviews in Control*, 54:81–102, 2022.

[30] Marino Gatto, Enrico Bertuzzo, Lorenzo Mari, Stefano Miccoli, Luca Carraro, Renato Casagrandi, and Andrea Rinaldo. Spread and dynamics of

the covid-19 epidemic in italy: Effects of emergency containment measures. *Proceedings of the National Academy of Sciences*, 117(19):10484–10491, 2020.

[31] Giulia Giordano, Franco Blanchini, Raffaele Bruno, Patrizio Colaneri, Alessandro Di Filippo, Angela Di Matteo, and Marta Colaneri. Modelling the covid-19 epidemic and implementation of population-wide interventions in italy. *Nature medicine*, 26(6):855–860, 2020.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[33] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning (ICML'16)*, pages 2829–2838, 2016.

[34] Vijaykumar Gullapalli. *Reinforcement learning and its application to control*. PhD thesis, University of Massachusetts at Amherst, 1992.

[35] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML'18)*, pages 1861–1870. PMLR, 2018.

[36] Jon C Helton and Freddie Joe Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, 81(1):23–69, 2003.

[37] Herbert W Hethcote. An immunization model for a heterogeneous population. *Theoretical population biology*, 14(3):338–349, 1978.

[38] Herbert W Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

[39] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[40] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed autonomous robotic systems 5*, pages 299–308. Springer, 2002.

[41] Ronen Israel, Bryan T Kelly, and Tobias J Moskowitz. Can machines' learn'finance? *Journal of Investment Management*, 2020.

[42] AH Jones and PB De Moura Oliveira. Genetic auto-tuning of pid controllers. 1995.

[43] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[44] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[45] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations (ICLR'18)*, 2018.

[46] Ana Lajmanovich and James A Yorke. A deterministic model for gonorrhea in a nonhomogeneous population. *Mathematical Biosciences*, 28(3-4):221–236, 1976.

[47] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning (ICML'13)*, pages 1–9. PMLR, 2013.

[48] Yutong Li, Nan Li, H Eric Tseng, Anouck Girard, Dimitar Filev, and Ilya Kolmanovsky. Safe reinforcement learning using robust action governor. In *Learning for Dynamics and Control (L4DC'21)*, pages 1093–1104. PMLR, 2021.

[49] Ryan A Licitra, Zachary I Bell, Emily A Doucette, and Warren E Dixon. Single agent indirect herding of multiple targets: A switched adaptive control approach. *IEEE Control Systems Letters*, 2:127–132, 2017.

[50] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971v6*, 2019.

[51] Wei-Liem Loh. On latin hypercube sampling. *The Annals of Statistics*, 24(5):2058–2080, 1996.

[52] Adam H Marblestone, Greg Wayne, and Konrad P Kording. Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience*, page 94, 2016.

[53] Nikolai Matni, Alexandre Proutiere, Anders Rantzer, and Stephen Tu. From self-tuning regulators to reinforcement learning and back again. In *Conference on Decision and Control (CDC'19)*, pages 3724–3740, 2019.

[54] Wenjun Mei, Shadi Mohagheghi, Sandro Zampieri, and Francesco Bullo. On the dynamics of deterministic epidemic propagation over networks. *Annual Reviews in Control*, 44:116–128, 2017.

[55] Ministero della Salute. Linee di indirizzo organizzative per il potenziamento della rete ospedaliera per emergenza COVID-19, 2020.

[56] Ministero della Salute. Posti letto per regione e disciplina, 2020.

[57] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML'16)*, pages 1928–1937, 2016.

[58] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

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

[60] Anna Mummert and Olusegun M Otunuga. Parameter identification for a stochastic seirs epidemic model: case study influenza. *Journal of mathematical biology*, 79:705–729, 2019.

[61] Rui Nian, Jinfeng Liu, and Biao Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, 2020.

[62] OpenAI. *OpenAI Gym Car Racing Online Documentation*, 2022.

[63] OpenAI. *OpenAI Gym Inverted Pendulum Online Documentation*, 2022.

[64] OpenAI. *OpenAI Gym Lunar Lander Online Documentation*, 2022.

[65] OpenAI. *OpenAI Gym online documentation*, 2022.

[66] Jing Peng and Ronald J Williams. Incremental multi-step q-learning. *Machine Learning*, pages 226–232, 1994.

[67] Liangrong Peng, Wuyue Yang, Dongyan Zhang, Changjing Zhuge, and Liu Hong. Epidemic analysis of covid-19 in china by dynamical modeling. *arXiv preprint arXiv:2002.06563*, 2020.

[68] Mark Pfeiffer, Samarth Shukla, Matteo Turchetta, Cesar Cadena, Andreas Krause, Roland Siegwart, and Juan Nieto. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3(4):4423–4430, 2018.

[69] Alyssa Pierson and Mac Schwager. Bio-inspired non-cooperative multi-robot herding. In *International Conference on Robotics and Automation (ICRA'15)*, pages 1843–1849, 2015.

[70] Alyssa Pierson and Mac Schwager. Controlling noncooperative herds with robotic herders. *IEEE Transactions on Robotics*, 34:517–525, 2017.

[71] Eduardo Ramírez-Llanos and Sonia Martínez. A distributed dynamics for virus-spread control. *Automatica*, 76:41–48, 2017.

[72] Meghana Rathi, Pietro Ferraro, and Giovanni Russo. Driving reinforcement learning with models. In *Intelligent Systems and Applications (ISWA'21)*, pages 70–85, 2021.

[73] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):253–279, 2019.

[74] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[75] Roberto Rocchetta, Luca Bellani, Michele Compare, Enrico Zio, and Edoardo Patelli. A reinforcement learning framework for optimal operation and maintenance of power grids. *Applied energy*, 241:291–301, 2019.

[76] George John Romanes. *Animal intelligence*, volume 44. D. Appleton, 1883.

[77] Lisa Sattenspiel and Klaus Dietz. A structured epidemic model incorporating geographic mobility among regions. *Mathematical biosciences*, 128(1-2):71–91, 1995.

[78] Samuel V Scarpino and Giovanni Petri. On the predictability of infectious disease outbreaks. *Nature communications*, 10(1):898, 2019.

[79] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. pages 1889–1897. PMLR, 2015.

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

[81] Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. Deep learning applications for covid-19. *Journal of big Data*, 8(1):1–54, 2021.

[82] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[83] Lucas M Stolerman, Daniel Coombs, and Stefanella Boatto. Sir-network model and its application to dengue fever. *SIAM Journal on Applied Mathematics*, 75(6):2581–2609, 2015.

[84] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International journal of robotics research*, 37(4-5):405–420, 2018.

[85] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings*, pages 216–224. Morgan Kaufmann, San Francisco (CA), 1990.

[86] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

[87] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.

[88] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning (ICML'93)*, pages 330–337, 1993.

[89] Imperial College COVID-19 Response Team. Report 13: Estimating the number of infections and the impact of non-pharmaceutical interventions on covid-19 in 11 european countries, 2020.

[90] Imperial College COVID-19 Response Team. Report 9: Impact of non-pharmaceutical interventions (npis) to reduce covid-19 mortality and healthcare demand, 2020.

[91] Edward L Thorndike. Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, 2(4):i, 1898.

[92] Neythen J Treloar, Alex JH Fedorec, Brian Ingalls, and Chris P Barnes. Deep reinforcement learning for the control of microbial co-cultures in bioreactors. *PLoS Computational Biology*, 16(4):e1007783, 2020.

[93] Piet Van Mieghem, Jasmina Omic, and Robert Kooij. Virus spread in networks. *IEEE/ACM Transactions On Networking*, 17(1):1–14, 2008.

[94] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[95] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv, arXiv:1907.02057*, 2019.

[96] Wendi Wang and Xiao-Qiang Zhao. An epidemic model in a patchy environment. *Mathematical biosciences*, 190(1):97–112, 2004.

[97] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

[98] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[99] Bernard L Welch. The generalization of 'student's'problem when several different population varlances are involved. *Biometrika*, 34(1-2):28–35, 1947.

[100] Mario Zanon and Sébastien Gros. Safe reinforcement learning using robust MPC. *IEEE Transactions on Automatic Control*, 66:3638–3652, 2021.

[101] John G Ziegler and Nathaniel B Nichols. Optimum settings for automatic controllers. *Transactions of the American society of mechanical engineers*, 64(8):759–765, 1942.

Author's Publications

The scientific results obtained during my Ph.D. period, and discussed in this thesis, have been disseminated in the following publications:

- F. De Lellis, M. Coraggio, G. Russo, M. Musolesi, M. di Bernardo "Control-Tutored Reinforcement Learning: Towards the Integration of Data-Driven and Model Based Control", *Learning for Dynamics and Control Conference (L4DC)*, Proceedings of Machine Learning Research (PMLR), pp. 1048-1059, 2022.
- M. Coraggio*, S. Xie*, F. De Lellis, G. Russo, M. di Bernardo, "Intermittent non-pharmaceutical strategies to mitigate the COVID-19 epidemic in a network model of Italy via constrained optimization", *Conference on Decision and Control (CDC)*, pp. 3538-3543, 2021.
- F. De Lellis, F. Auletta, G. Russo, P. De Lellis, M. di Bernardo, "An Application of Control-Tutored Reinforcement Learning to the Herding Problem", *International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, 2021.
- F. De Lellis, G. Russo, M. di Bernardo, "Tutoring Reinforcement Learning via Feedback Control", *European Control Conference (ECC)*, pp. 580-585, 2021.
- F. Della Rossa*, D. Salzano*, A. Di Meglio*, F. De Lellis*, M. Coraggio, C. Calabrese, A. Guarino, R. Cardona-Rivera, P. De Lellis, D. Liuzza, F. Lo Iudice, G. Russo, M. di Bernardo, "A network model of Italy shows that intermittent regional strategies can alleviate the COVID-19 epidemic", *Nature Communications*, 11, 5106, 2020.

*Authors who contributed equally to the development of the research