Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Doctoral Programme In Information Technology

Challenges and Opportunities in Multi-Agent Reinforcement Learning

Doctoral Dissertation of:
Giorgia Ramponi

Supervisor:
Prof. Marcello Restelli

Tutor:
Prof. Barbara Pernici

The Chair of the Doctoral Program:
Prof. Barbara Pernici

2021 – XXXIII

To my grandfather Adriano.

Acknowledgements

First of all, I would like to thank my thesis reviewers, Professor Lillian Ratliff and Professor Olivier Pietquin. It was a great honor to have you as reviewers. I am really grateful to you for your valuable suggestions, which helped me improve this thesis and for the time you spent reading it.

Il finale. Il finale di un lungo percorso, faticoso, nuovo, emozionante, stressante, divertente, appassionante. Penso che la passione sia quello che ci spinge ad intraprendere questa via, anche perché senza di essa le lunghe giornate, nottate, weekend, spesi a studiare, scrivere paper, derivare teoremi, riflettere, cancellare, e ricominciare, sarebbero impossibili da superare.

Ma questo percorso non sarebbe possibile affrontarlo senza le persone giuste.

Parto con ringraziare la prima persona che mi ha fatto scoprire il mistero e il fascino della ricerca, la professoressa Gaia Maselli. Grazie per aver convinto una ragazzina del secondo anno di triennale ad appassionarsi a questo tipo di vita.

Un grazie speciale va, senza dubbio, al mio advisor, Marcello Restelli, che mi ha accolto tra i suoi dottorandi, scommettendo su di me. Grazie perché hai trovato del tempo da dedicarmi quando già avevi tante persone da seguire. Grazie soprattutto perché, in questo anno e mezzo, penso di aver imparato molto, partendo da zero, e senza di te non sarebbe mai stato possibile.

Un grazie va sicuramente ai professori Marco Brambilla, Stefano Ceri e Florian Daniel, che mi hanno accompagnato in questa prima parte di

dottorato. Grazie Marco per avermi lasciata andare quando ho capito che la mia passione mi portava su altri temi. Grazie Stefano per i consigli, i caffé per capire cosa mi rendesse infelice, l'interesse che metti per tutti i tuoi studenti e la dedizione che dedichi al tuo lavoro. Un grazie a Florian (spero che in qualche modo tu lo possa sentire) per le birre a Lambrate, i pomeriggi spesi con me e Marco Di Giovanni a parlare di cosa significhi essere un ricercatore e per averci insegnato che il dottorato, qualunque argomento avessimo scelto, ci avrebbe aiutato a maturare.

Grazie ai miei colleghi che in questi anni hanno riempito gran parte delle mie giornate. Grazie a Marco, mio compagno di ufficio per i primi due anni, e penso di poter dire anche amico. All'inizio non ci andavamo molto a genio, abbiamo discusso spesso, ma poi ci siamo stati accanto, nelle varie school, conferenze e viaggi. Le chiacchiere sono state infinite, tra un caffé e l'altro. Incredibilmente ho conosciuto una persona molto più ansiosa di me, ma alla fine siamo riusciti a terminare questo percorso. Grazie Gaia per essere stata un'amica oltre che una collega. Dal momento imbarazzante in cui ci hanno presentate, ho subito capito che in te avrei trovato una persona speciale. Grazie per le cene, le chiacchiere, e tutti i momenti che abbiamo passato insieme. Grazie anche a tutti gli altri del gruppo GECO, Andrea, Anna, Michele, Luca, Eirini, Pietro, Arif, con cui ho passato pranzi indimenticabili. Grazie a Sara, che e' una delle persone più buone e gentili mai conosciute, che in poco tempo è diventata un'amica. Sono sicura che queste qualità ti faranno fare grandi cose.

Grazia all'altra mia famiglia del Politecnico, l'Airlab. In primis devo ringraziare i miei compagni di ufficio Alberto, Andrea e Matteo e scusarmi per aver imposto la mia presenza. Grazie per le pause dal lavoro, le discussioni, per aver esaudito i miei desideri dei bollitori e delle piantine (che purtroppo non hanno superato questi mesi). Ho imparato molto da voi. Grazie a Giulia, Andrea, Alessandro e Alberto per le serate e i momenti di svago che purtroppo non sono potuti essere tanti per colpa della pandemia. E' bello trovare persone così divertenti e allo stesso tempo brillanti. Grazie a Mirco per i confronti su argomenti di ricerca e non, le conversazioni con te sono sempre stimolanti. Grazie a Nico e Giuseppe per aver condiviso con me l'esperienza di Lille, ho scoperto due persone speciali. Un grazie poi a tutti gli altri, Simone, Francesco, Luca, Alessio, Matteo, Lorenzo, Pierre, Mattia, Alessandro, Marco, Amarildo. Amarildo, grazie anche per essere stato coautore di uno dei lavori di questa tesi.

Grazie alle altre persone che questo percorso mi ha fatto incontrare. Chi ho conosciuto alle conferenze o alle school. Indimenticabile MLSS, dove ho

incontrato persone incredibili, che condividevano la mia stessa curiosità; un grazie anche alla famiglia fantastica della ragazza di Jose Carlos (che non ho conosciuto ma che spero un giorno di conoscere) per avermi fatto vivere la vera Spagna e mangiare la vera Paella.

Grazie alle persone con cui ho condiviso il mio periodo a Boston, in assoluto le personalità più stravaganti che io abbia mai incontrato, in particolare Nicola, Elena, Beatrice, Rudi, Jamir e Roberto. Non ho dimenticato Luna, la persona che mi ha fatto subito sentire a casa in una città a me sconosciuta, e con cui ci siamo sentite immediatamente affini. I momenti che abbiamo vissuto sono indimenticabili, da feste su navi della Marina Militare a gite in barca, da giornate al mare sponsorizzate Harvard a pagaiate sul Charles River. La cosa però più importante è l'amicizia che abbiamo costruito e che continua ancora.

Un grazie speciale va a Matteo Pirotta, che tra uno scherzo e l'altro, è diventato mio amico. Sei una persona indubbiamente intelligente, ma oltre questo anche buona, divertente e sorprendente. I nostri aperitivi online hanno reso piu "normale" l'ultimo periodo covid a Milano.

Grazie alla mia portiera Stella, che oltre ad essere la portiera migliore che si possa avere, sempre pronta a dare una mano, mi è stata particolarmente vicina durante il covid. Un grazie a tutte le persone, Alessandro, Giulia, Andrea, Maurizio, le farmaciste, Valentino che mi hanno aiutato in quel periodo difficile.

Un grazie agli amici. Questa scelta di vita, che ti porta a lasciare casa, ad avere poco tempo libero, inevitabilmente toglie tempo agli affetti di sempre e alcuni purtroppo si perdono per strada. Grazie però agli amici che invece sono rimasti e che hanno data luce a questi anni. Grazie Bea per essere stata presente sempre, come se fossimo vicine. La tua amicizia è un dono importante perché oltre le risate, le giornate al mare, gli aperitivi, sei una persona che quando sceglie di esserti amica lo fa dando tanto. Grazie Flavi perché Milano non sarebbe stata la stessa senza di te. Ci siamo conosciute possiamo dire durante questi anni e con te non mi sono mai sentita sola. Grazie perché ci sei sempre stata, senza mai dirmi di no. Abbiamo avuto tante serate assurde e speso ore e ore a chiacchierare. Grazie Giuli, perché senza di te mi sarei sentita, soprattutto all'inizio, sperduta a Milano. Grazie perche, anche se ogni tanto sei sfuggente, nei momenti difficili ci sei sempre stata, dalle delusioni d'amore, a momenti no di questi tre anni. Grazie Marti, per le varie volte che mi sei venuta a trovare a Milano, per la tua semplicità e sotto sotto dolcezza. Grazie Lulla, anche se io non ti chiamo mai così, perché, anche se la distanza si è fatta sentire nel nostro rapporto, so che la nostra amicizia supererà anche questi momenti. Sei un'amica e una persona incredibile. Grazie Luca (G.) perché, anche se lontani, la nostra amicizia ha continuato ad essere una parte importante della mia vita. Grazie Luca (R.), mio fratello non di sangue ma di vita, per quando ti ospitavo e non mi facevi dormire perché volevi sempre parlare e parlare, per avermi fatto uscire e respirare un po' di vita quando ero chiusa nel mio lavoro, per le volte che mi hai portato in questura e perché se mi serve qualcosa non mi dici di no. Un grazie anche a Pietro per aver passato con me il primo anno di dottorato, tra treni Roma-Milano, e a cui, anche se ora ci siamo allontanati, non posso non voler bene. Grazie a Soeren, il mio fratello acquisito potrei dire. Grazie per essere per Vale la persona che sei e per i pomeriggi passati a chiacchierare, a giocare a giochi da tavola e per darmi voi due, ogni giorno, un esempio dell'amore che tutti vorremmo. Grazie anche a tutti gli altri amici, Seme, Vitto, Bene, Carla, tutti gli amici di Golfo Sereno, e qualche nome che forse ora non avrò scritto, ma che non per questo non è nel mio cuore.

Un grazie speciale ad Andrea che è stato il faro in un momento difficile e delicato della mia vita. La tua bontà e gentilezza sono doti difficili da trovare. Il tuo supporto è stato per me il più grande aiuto e i momenti insieme la parte bella di questo viaggio. Il dottorato mi ha per certi versi tolto tanto ma sicuramente mi ha regalato la possibilità di incontrarti e già questo basterebbe. Già sai tutto, spero che il nostro cammino continui così.

Infine grazie alla mia famiglia perché senza di loro niente di questo sarebbe stato possibile. Grazie a mia madre, per essere sempre disponibile, buona, per la tua intelligenza e perché il nostro rapporto penso sia maturato tanto in questi anni lontane. Grazie a mio padre, per le discussioni, confronti, per la sua perspicacia, per aver imparato insieme ad ascoltare le diverse opinioni; vorrò sempre sapere la tua. Non c'è giorno in cui, se viviamo in due posti diversi, io non senta la vostra mancanza. Grazie a mia sorella, la mia migliore amica, forse la persona da cui è stato più difficile allontanarmi. Sai quanto vorrei che vivessimo nella stessa città, quanto vorrei passare più tempo con te e quanto sei stata importante per non farmi arrendere in questi tortuosi anni. Grazie ai miei nonni che non ci sono più, ma che sento che mi seguono nel mio tragitto. Grazie a mia nonna che è la persona al mondo che più ha intuito per i miei sentimenti e che vorrei avere più vicina.

Finisce così questa lunga lista di ringraziamenti, probabilmente ce ne sarebbero molti altri da fare.

Giorgia
Milano, 16 Maggio 2021

Abstract

Reinforcement Learning (RL) is a Machine Learning area that studies sequential decision-making problems, where a learning agent interacts with an unknown environment in order to maximize its rewards. In recent years, RL methods have made substantial progress in solving real-world problems. However, the most successful applications, such as beating the world champion player of Go, solving robotic control problems, managing the power consumption of households, and achieving promising results in autonomous driving, involve more than one agent and can be cast in the Multi-Agent Reinforcement Learning (MARL) setting. However, although the MARL setting is an important research area of practical interest, this framework is still poorly understood from a theoretical point of view. In general, the presence of many agents makes the learning problem more complex, and in many situations, single RL algorithms cannot be applied.

In this thesis, we take a step toward solving this problem, providing theoretically sound algorithms for this setting. We analyze the challenges and opportunities that a multi-agent environment creates in the RL framework, providing new approaches in three RL sub-problems while also showing how they are interconnected. The contributions of the thesis are theoretical, algorithmic, and experimental. We take inspiration from practical problems, we design new algorithms with desirable theoretical properties to solve them, and we show their performances in benchmarks domains and on real-world data.

The thesis is divided into four parts. In the first part, we provide the background and preliminaries necessary to follow the rest of the thesis. We start by introducing the RL problem and classical algorithms to solve it. Then, we introduce the Inverse RL problem, i.e., the problem of recovering reward functions from an expert's demonstrations. Finally, we provide the necessary background on game theory and MARL.

In the second part, we analyze how the presence of multiple agents affects the Inverse Reinforcement Learning problem and objective. We provide two novel algorithms: the first considers how to recover and cluster the intentions of (i.e., the rewards optimized by) a set of agents given demonstrations of near-optimal behavior; the second aims at inferring the reward function optimized by an agent while observing its actual learning process. The experimental evaluation is conducted on synthetic problems and two real-world problems.

We then show the importance of learning (or knowing) the other agent's intention to construct efficient algorithms. In particular, in the third part, we study online learning in the MARL scenario, showing how the presence of other agents can increase the hardness of the problem while proposing statistically efficient algorithms. We design two algorithms to solve the Configurable MDP problem, a setting where an external entity can partially control the transition model. Then, we analyze the statistical limits of general-sum stochastic games when we control only one agent, providing a new lower bound and a near-optimal algorithm.

Finally, in the fourth part, we study MARL from an optimization viewpoint while showing the difficulties that arise from multiple function optimization problems. Then, we present a new algorithm for this scenario, providing convergence results and extensively evaluating it against SotA baselines.

Contents

I Introduction and Preliminaries **5**

1 Introduction **7**

1.1 Overview 9
1.2 Contributions 10

2 Reinforcement Learning **15**

2.1 Markov Decision Processes 16
2.1.1 Returns and Episodes 17
2.1.2 Policies and Value functions 18
2.1.3 Bellman operators 19
2.1.4 Characteristics of Reinforcement Learning algorithms 20
2.2 Exact Algorithms 21
2.2.1 Policy evaluation 22
2.2.2 Policy iteration 22
2.2.3 Value Iteration 24
2.3 Value-based algorithms 24
2.3.1 SARSA 25
2.3.2 Q-learning 26
2.4 Policy Gradient Approaches 26
2.4.1 REINFORCE 28
2.4.2 G(PO)MDP 28
2.4.3 Policy Gradient Theorem 29
2.5 Actor-critic algorithms 30

3 Inverse Reinforcement Learning **31**
3.1 Problem statement 32
3.2 Model-based Inverse Reinforcement Learning 33
3.2.1 Feature Expectation Matching 33
3.2.2 Maximum Entropy Inverse Reinforcement Learning . 34
3.2.3 Maximum Likelihood Inverse Reinforcement Learning 35
3.3 Model-free Inverse Reinforcement Learning 36
3.3.1 Relative Entropy Inverse Reinforcement Learning . . 36
3.3.2 Cascaded Supervised Learning Approach to Inverse Reinforcement Learning 37
3.3.3 Gradient Inverse Reinforcement Learning 37

4 Multi-agent Reinforcement Learning **39**
4.1 Normal-form Games 40
4.2 Stochastic Games 42
4.3 Cooperative stochastic games 44
4.4 Competitive stochastic games 45
4.5 General-sum stochastic games 47

II Inverse Reinforcement Learning in Multi-Agent Systems **51**

5 Inverse Reinforcement Learning for multi-agent systems **55**
5.1 Preliminaries 57
5.2 Related work 59
5.2.1 Inverse Reinforcement Learning about Multiple Intentions 59
5.2.2 Inverse Reinforcement Learning from a Learner . . . 60

6 Inverse Reinforcement Learning about Multiple Intentions **63**
6.1 Problem statement 64
6.2 Σ-Gradient Inverse Reinforcement Learning 65
6.3 Theoretical analysis of Σ-GIRL 68
6.3.1 Approximation of Σ as in Corollary 6.2.1 69
6.3.2 Correctness of the recovered weights 71
6.4 Multiple-Intention Σ-GIRL 73
6.4.1 Computational Complexity Analysis 76
6.5 Discussion on the related work 76
6.6 Experiments 77
6.6.1 Single-IRL experiments 77
6.6.2 Multiple-intentions experiments 80

6.6.3 Twitter experiment 83

7 Inverse Reinforcement Learning from a Learner **85**
7.1 Problem statement 86
7.2 Learning from a learner following the gradient 87
7.2.1 Exact gradient 88
7.2.2 Approximate gradient 90
7.3 Learning from improvement trajectories 92
7.3.1 Recovering learning rates and reward weights 93
7.3.2 Theoretical result 94
7.4 Discussion on the related works 98
7.5 Experiments 98
7.5.1 Gridworld 99
7.5.2 MuJoCo environments 102
7.5.3 Autonomous driving scenario 103

III Online Learning in Multi-Agent Reinforcement Learning **105**

8 Online Learning in Stochastic Games **109**
8.1 Problem setting 110
8.2 Related works 112

9 Non-Cooperative Configurable Markov Decision Processess **115**
9.1 Configurable MDPs 117
9.2 Non-Cooperative Configurable MDPs 118
9.3 Problem Formulation 119
9.4 Action-feedback Optimistic Configuration Learning 121
9.4.1 Regret Guarantees 122
9.5 Reward-feedback Optimistic Configuration Learning 126
9.5.1 Regret Guarantees 128
9.6 Comparison between the two algorithms 134
9.7 Discussion on the related works 135
9.8 Experiments 136
9.8.1 Configurable Gridworld 136
9.8.2 Configurable Market 138
9.8.3 Student-Teacher 139

10 Online Learning in General-sum Turn-based Stochastic Games **141**
10.1 Problem statement 142
10.2 Lower bound on the regret 143

10.2.1 Discussion on the lower bound 148
10.3 TSG Optimistic Policies Value Iteration 149
10.3.1 Regret Guarantees 150
10.4 Discussion 154

IV Policy Optimization in Multi-Agent Reinforcement Learning 155

11 Continuous games and gradient-based approaches 159
11.1 Continuous Games 160
11.1.1 Helmhotz game decomposition 161
11.1.2 Desired convergence points 163
11.2 Related work 164
11.3 MARL and policy-gradient algorithms 167

12 Newton Optimization On Helmhotz Decomposition 173
12.1 Newton method for non-convex functions 174
12.2 Newton for Games 175
12.2.1 Newton's method for Potential games 175
12.2.2 Newton's method for Hamiltonian games 175
12.2.3 Newton's method for General games 177
12.2.4 Discussion on desired convergence points 180
12.3 Experiments 181
12.3.1 Matrix games 181
12.3.2 Continuous gridworlds 184
12.3.3 Generative Adversarial Network 184
12.3.4 Computational time 185

13 Conclusions and Future works 197
13.1 Inverse Reinforcement Learning in Multi-Agent Systems . . 198
13.1.1 Future directions 198
13.2 Online Learning in Stochastic Games 199
13.2.1 Future directions 200
13.3 Optimization methods for Multi-Agent Reinforcement Learning 200
13.3.1 Future directions 201

Bibliography 203

A Learning in Continuous Games: a (brief) introduction **233**
A.1 Continuous Games 233
A.2 Classical learning dynamics 234

B Supporting lemmas for Chapter 6 **237**

C Supporting lemmas and additional results for Chapter 7 **243**
C.1 Matrix perturbation and linear least square problems 243
C.2 Additional results 245

Mathematical notation

I have tried to use the same notation during all the thesis to improve the readability, at the expense, sometimes, of using a different notation from that in the literature. However, in these cases, I have specified the differences between the literature. I hope that the notation, in general, would be clear and understandable.

Vectors are indicated with lower case bold letters; matrices with upper case letters. We indicate in calligraphic functions and spaces. The operator T indicates the transpose of a matrix, $[v_1, \dots, v_n]$ indicates a row vector, instead $[v_1, \dots, v_n]^T$ a column vector. An element of a matrix is indicated with $v_{i,j}$ where i is the row and j the column. We indicate with $[N]$ the set of integers between 1 and N. The $\mathbb{1}\{\text{predicate}\}$ operator indicates the random variable that is equal to 1 if the predicate is true and 0 if the predicate is false. I report below a table with the notation used in the thesis.

$\mathcal{S}$	State space
$\mathcal{A}$	Action space
$\mathcal{P}$	Transition probability function
H	Horizon
γ	Discount factor
γ_i	Discount factor of agent i
μ	Initial state distribution
$\mathcal{R}$	Agent's reward function
$\mathcal{R}_i$	Agent i's reward function
π	Policy
$\pi_{\boldsymbol{\theta}}$	Policy parametrized by $\boldsymbol{\theta}$
$V_h^\pi(s)$	Value function of policy π in state s
$Q_h^\pi(s,a)$	Action-value function of policy π in state s and action a
$V^\star$	Optimal value function in state s

$Q^\star$	Optimal action-value function in state s and action a
$J(\pi)$	Expected discounted return of policy π
$J_i^{\pi_i,\pi_{-i}}$	Expected discounted return of joint policies π_i, π_{-i} for agent i
T^π	Bellman operator of policy π
$T^\star$	Optimal bellman operator
$\boldsymbol{\psi}(\pi)$	Expected discounted feature expectation of policy π
$\boldsymbol{\psi}(\boldsymbol{\theta})$	Expected discounted feature expectation of policy $\pi_{\boldsymbol{\theta}}$
$\phi(s,a)$	Features' vector of state s and action a
$\boldsymbol{\omega}$	Reward's weights
$\mathcal{R}_{\boldsymbol{\omega}}(s,a)$	Linear reward with weights $\boldsymbol{\omega}$
$J(\boldsymbol{\theta},\boldsymbol{\omega})$	Expected discounted return of policy $\pi_{\boldsymbol{\theta}}$ and reward $\mathcal{R}_{\boldsymbol{\omega}}$
$\mathrm{vec}(A)$	Vectorization of matrix A
$\sigma_{\min}(A)$	Minimum singular value of matrix A
Part III	
br	Best response function
$Q_{i,h}^{\pi_1,\pi_2}(s,a)$	Action-value for agent i with policy π_1 and pi_2 at time step h in state s and action a
$V_{i,h}^{\pi_1,\pi_2}(s)$	Value function for agent i with policy π_1 and pi_2 at time step h in state s
$V_{i,h}^{\pi_1,\pi_2}$	Agent i expected return with policy π_1 and pi_2 at time step h
$\mathcal{R}_o$	Agent's reward function (in Chapter 9)
$\mathcal{R}_c$	Configurator's reward function
$\mathfrak{P}$	Configuration space
$Q_{c,h}^{\pi,\mathcal{P}}(s,a)$	Configurator's Q-value with policy π and configuration $\mathcal{P}$
$Q_{o,h}^{\pi,\mathcal{P}}(s,a)$	Agent's Q-value with policy π and configuration $\mathcal{P}$
$V_{c,h}^{\pi,\mathcal{P}}(s)$	Configurator's value function with policy π and configuration $\mathcal{P}$
$V_{o,h}^{\pi,\mathcal{P}}(s)$	Agent's value function with policy π and configuration p
$V_c^{\pi,p}$	Configurator's expected return with policy π and configuration $\mathcal{P}$
$V_o^{\pi,\mathcal{P}}$	Agent's expected return with policy π and configuration $\mathcal{P}$
$\pi_i = \pi^*_{\mathcal{P}_i}$	Agent's best response to configuration $\mathcal{P}_i$
$V^i = V_c^{\pi^*_{\mathcal{P}_i},\mathcal{P}_i}$	Configurator's expected return with the agent's best response policy $\pi^*_{\mathcal{P}_i}$ to configuration $\mathcal{P}_i$
$V^* = V_c^{\pi^*_{\mathcal{P}_{i^*}},\mathcal{P}_{i^*}}$	Configurator's expected return with the agent's best response policy $\pi^*_{\mathcal{P}_i}$ to the best configuration $\mathcal{P}_{i^*}$
$\widetilde{V}_k^i$	Optimistic configurator's expected return for configuration $\mathcal{P}_i$ at episode k
$\widetilde{\pi}_{i,k}$	Estimated agent's best response policy for configuration $\mathcal{P}_i$ at episode k
$\Delta_i = V^* - V^i$	Suboptimality gap of the configuration $\mathcal{P}_i$ (Chapter 9)
K	Number of episodes
N_i	Number of times the configuration $\mathcal{P}_i$ is played (Chapter 9)
$N_k(s)$	Number of visits of state s before episode k (Chapter 9)
$N_{k,h}^i(s)$	Number of visits of state s at step h before episode k with configuration $\mathcal{P}_i$ (Chapter 9)
$\underline{\mathcal{R}}_{o,k}(s)$	Lower confidence value for the agent's reward Chapter 9)
$\overline{\mathcal{R}}_{o,k}(s)$	Upper confidence value for the agent's reward Chapter 9)
$\widehat{\mathcal{R}}_{o,k}(s)$	Sample mean of observed rewards Chapter 9)
$\underline{Q}_{o,k,h}^i(s,a)$	Lower confidence value of the agent's Q-function with configuration $\mathcal{P}_i$ Chapter 9)
$\overline{Q}_{o,k,h}^i(s,a)$	Upper confidence value of the agent's Q-function with configuration $\mathcal{P}_i$ Chapter 9)

$\mathcal{A}^i_{k,h}(s)$	Set of agent's plausible actions in state s at step h up to episode k Chapter 9)
$d^i_h(s)$	Visitation probability the state s at step h with configuration p_i under the agent's best response policy π_i Chapter 9)
$\widetilde{d}^i_h(s)$	Visitation probability the state s at step h with configuration p_i under the estimated agent's best response policy $\widetilde{\pi}_{i,k}$ Chapter 9)
Regret$^{\mathfrak{A}}(K)$	Regret of Algorithm $\mathfrak{A}$ after K episodes
$KL(\cdot,\cdot)$	KL-divergence between two probability laws
$kl(\cdot,\cdot)$	binary KL-divergence
$\mathcal{A}^i_{k,h}(s)$	Set of agent 2's possible actions in state s at step h up to episode k and playing policy π_1 (Chapter 10)
$N_k(s,a)$	Number of times the state-action pair s,a is visited up to episode k (Chapter 10)
$N^i_k(s)$	Number of times the state s is visited up to episode k playing the policy π^i_1 (Chapter 10)
d	Minimum probability of visit every reachable state (Chapter 10)
Part IV	
$\xi(\boldsymbol{\theta})$	Simultaneous gradient
$\mathcal{J}(\boldsymbol{\theta})$	Game Jacobian
C_i	Expected discounted cost function for agent i
$S(\boldsymbol{\theta})$	Symmetric component of the game Jacobian
$A(\boldsymbol{\theta})$	Antisymmetric component of the game Jacobian
H^{-1}_m	PT-inverse of matrix H with parameter m

Part I

Introduction and Preliminaries

CHAPTER 1

Introduction

"This was simply the idea of a learning system that wants something, that adapts its behavior in order to maximize a special signal from its environment."

R. S. Sutton & A. G. Barto, Reinforcement Learning: An Introduction

"The importance of the social phenomena, the wealth and the multiplicity of their manifestations, and the complexity of their structure, are at least equal to those in physics. It is therefore to be expected, that mathematical discoveries of a statue comparable to that of calculus will be needed. [. . .] We believe that it is necessary to know as much as possible about the behavior of the individual and about its forms of exchange."

O. Morgenstern & J. von Neumann, Theory of Games and Economic Behavior

We can say that this thesis started from these two quotes. Both authors are observers: they watched what was happening in the world around them to be able to model it mathematically. The first observed how humans learn to perform a task. As they said, the idea is "simple": humans adapt their behavior to maximize a signal from the environment. Imagine a child who has to learn to ride a bicycle. The child starts by sitting on the seat, and nothing happens. Then she puts her feet on the pedals but rides too slowly, causing the bicycle to lose its balance, and she falls. She thus learned from her experience that she must pedal faster to avoid falling again. This concept was mathematically modeled by *Reinforcement Learning* [Sutton et al., 1998]. The second authors note that we are “social beings”, i.e., we act in a social system in which multiple entities interact with each other. Therefore, the actions of each entity can not only result in an individual income, but also change the income of other entities. If we decide to buy a stock on the stock exchange, the result of our action will affect not only us but also the entire stock market. It is easy to imagine that these interactions can be very complex, and we can hardly understand how our decisions can affect the world around us. One of the sciences that mathematically model these interactions is called *Game Theory* [Morgenstern and Von Neumann, 1953].

From these considerations, we can conclude that in order to create a system that is capable of acting autonomously, we must study how to build an autonomous learning agent (*Reinforcement Learning*) and model how this is influenced by the other entities that surround it (*Game Theory*). *Multi-agent Reinforcement Learning* (MARL) [Buşoniu et al., 2010, Zhang et al., 2019a] is a bridge between these two worlds. The MARL framework studies the problem of learning by interacting with an unknown system, considering that it is composed of more than one entity.

In this thesis, we analyze the *challenges* and the *opportunities* that arise from switching from single-agent RL to MARL. A first opportunity is the possibility to model real-world situations (e.g., autonomous driving [Shalev-Shwartz et al., 2016], robotic control problems [Lillicrap et al., 2015], or networking systems [Yu et al., 2018]) in a better way. The first challenge is that the environment becomes non-stationary [Hernandez-Leal et al., 2019]: the outcome that we receive from an action also depends on the other agents' actions. So it is clear that it could be fundamental to learn the interests of the other agents to learn how to act. Moreover, our goal is no longer clear. Are we sure that we only want to maximize our interest? For example, our aim could be to optimize the total gain the system makes. Or we might be interested in only converging to a solution that is "good" for everyone (an equilibrium [Nash et al., 1950, Von Stackelberg, 2010]). But what is this good solution? Plus, what should we do if we cannot agree with the other agents?

In the same way, we can say that we are analyzing the difficulties of moving from game theory, where we have perfect knowledge of the interactions between agents, to MARL where we have to learn them by interacting with the environment. Good algorithms must also take into account that there could be some uncontrollable agents that pursue different objectives.

We try to solve some of these challenges. We analyze the MARL framework from different but related points of view. We start by studying the Inverse Reinforcement Learning framework (IRL) in the multi-agent setting. IRL is a paradigm to recover an RL agent's intention, and this capability, as we have mentioned above, could be essential in a multi-agent system, where the interests of the other entities condition our decisions. Then, we consider the learning problem in Multi-Agent RL from two different but complementary perspectives: online learning and optimization. These two extensively studied ways to solve the single-agent RL framework acquire new challenges in the multi-agent setting.

1.1 Overview

The thesis is organized into four parts; except for the first part, each one takes into account a different perspective of the problem of learning in a multi-agent environment, as we have described in the previous section.

The first part is dedicated to the introduction and the preliminaries necessary to follow the rest of the thesis. In Chapter 2 we report an overview on RL: we provide the background on Markov Decision Processes [Puterman, 2014] and RL [Sutton et al., 1998], describing the most popular settings and algorithms. Chapter 3 introduces the IRL problem [Osa et al., 2018], presenting the most relevant approaches to solve it. Then, in Chapter 4 we present the necessary background for MARL [Buşoniu et al., 2010, Zhang et al., 2019a], starting from the Game theory concepts necessary for this framework, then formalizing the Stochastic games [Shapley, 1953], and finally proposing some algorithms to solve this setting. If the reader is already familiar with RL, IRL and MARL, we suggest skipping these chapters and go directly to Part 2.

In the second part, we analyze the problem of IRL in a multi-agent environment. The multi-agent setting creates new challenges and opportunities for the IRL setting. In fact, in a multi-agent environment, we can have observed, for example, more than one expert. Moreover, in a multi-agent context, we can use the IRL algorithms to learn our opponent's reward function. In Chapter 5, we introduce the IRL problem in a multi-agent domain, describing the new settings that arise. Then in Chapter 6, we propose an algorithm to deal with IRL about Multiple Intentions, i.e., the problem of recovering the reward functions from a set of experts. In Chapter 7 we introduce a new algorithm to deal with the problem of inferring the intentions from a learning agent, i.e., an agent that is not an expert but that is learning a task. This setting is of particular interest in multi-agent RL, where most of the algorithms use the other agent's reward functions.

In the third part, we address the problem of online learning in Stochastic Games. Online learning is interested in measuring the quality of the whole learning process. In Stochastic Games, there are additional challenges compared with MDPs, since we have to learn the other agents' behaviors to design our policy. In Chapter 8, we introduce the online learning problem, and we revise the literature. In Chapter 9 we propose an algorithm to deal with the online learning problem in the Configurable Markov Decision Process, which we cast to a particular multi-agent problem where we can control only the configurator. In Chapter 10, we introduce a new lower bound on the online learning problem in Stochastic Games, proposing an algorithm that nearly-matches this lower bound.

The fourth part of this thesis is devoted to the optimization viewpoint of learning in Stochastic Games. We start by introducing, in Chapter 11, a more general problem, learning in Continuous Games. Then, we show how to deal with Continuous Stochastic Games, proposing a way to approximate the gradient and the hessian of the game's expected discounted returns. In Chapter 12 we introduce a new optimization algorithm for Continuous Games, proving its quadratic convergence rate in two classes of games and linear convergence in general games. Finally, we test the algorithm in many simulated domains, comparing it with many state-of-the-art baselines.

1.2 Contributions

In this thesis, we study opportunities and challenges that can arise from learning in a multi-agent environment. The rest of this section summarizes the thesis's contributions, where most of the work is published or under review. In all these works, the author of this thesis contributed to the design of the algorithms, their theoretical analysis, the empirical evaluation, and the realization of the manuscript.

Inverse Reinforcement Learning about Multiple Intentions The first contribution is about the problem of Inverse Reinforcement Learning about Multiple Intentions [Babes et al., 2011], i.e., the problem of estimating the unknown reward functions optimized by a group of experts that demonstrate optimal behaviors and cluster the experts by the recovered rewards. There are not many works that addressed this problem; moreover, these ones either require access to a model of the environment [Babes et al., 2011, Choi and Kim, 2012] or repeatedly compute the hypothesized rewards' optimal policies [Almingol and Montesano, 2015, Rajasekaran et al., 2017]. These requirements are rarely met in real-world applications, in which interacting with the environment can be expensive or even dangerous. We first propose a new IRL algorithm that cast the single IRL problem as a constrained likelihood maximization, and then we use this formulation to cluster agents based on the likelihood of the assignment. In this way, we can efficiently solve, without interactions with the environment, both the IRL and the clustering problem. The algorithm is tested on a real Twitter dataset, clustering the users by their intentions, i.e., what posts to retweet. These results are reported in Chapter 6. We also propose an extension of this work (listed below), where we tested the algorithm in other real-world applications: in recovering the intentions of controlling the Lake Como dam and learning the reward functions of a set of people who use an autonomous driving simulator. This extension is not reported in the thesis.

Ramponi, G., Likmeta, A., Metelli, A. M., Tirinzoni, A., & Restelli, M. (2020, June). Truly Batch Model-Free Inverse Reinforcement Learning about Multiple Intentions. In International Conference on Artificial Intelligence and Statistics (pp. 2359-2369). PMLR.

Likmeta, A., Metelli, A.M., Ramponi, G., Tirinzoni, A., Giuliani M. & Restelli. (2021, March) Dealing with multiple experts and non-stationarity in inverse reinforcement learning: an application to real-life problems. Machine Learning Journal.

Inverse Reinforcement Learning from a Learner In a multi-agent setting, to achieve the convergence to a Nash Equilibrium, in most situations, it is useful to know the reward function of the other agents. However, in IRL, we have to wait for the agent's convergence to its optimal policy before applying IRL algorithms. We propose a new algorithm to infer a learning agent's reward function [Jacq et al., 2019]. Our approach is based on the assumption that the observed agent is updating her policy parameters along the gradient direction. For the proposed algorithm, we provide theoretical insights into our algorithms' performance. Then we evaluate the approach in simulated domains and on an autonomous driving scenario. We reported the results in Chapter 7.

Ramponi, G., Drappo, G., & Restelli, M. (2020). Inverse Reinforcement Learning from a Gradient-based Learner. In Proceedings of the 33rd International Conference on Neural Information Processing Systems (pp. 2458–2468).

Online Learning in a Non-cooperative Configurable Markov Decision Process In Chapter 9 we analyse the online learning problem in Configurable Markov

Decision Processes [Metelli et al., 2018, Metelli et al., 2019b, Metelli et al., 2019a]. In the Configurable Markov Decision Processes, there are two entities, an RL agent and a configurator, that can modify some environment parameters. We introduce the Non-Cooperative Configurable Markov Decision Process, a framework that allows two (possibly different) reward functions for the configurator and the agent. This setting generalizes the Configurable Markov Decision Process. We consider the problem in which we can control only the configurator, and we have to find the best among a finite set of possible configurations. For this problem, we propose two learning algorithms to minimize the configurator's expected regret, which exploits the problem's structure.

Ramponi, G., Metelli, A. M., Concetti, A., & Restelli, M. (2021). Online Learning in Non-Cooperative Configurable Markov Decision Process. In Reinforcement Learning in Games workshop at Association for the Advancement of Artificial Intelligence.

Online Learning in General-sum Turn-based Stochastic Games The theoretical problem of learning in Stochastic Games [Shapley, 1953] has been divided into two different setting: *online* and *offline* settings. In the *online* setting [Littman, 1994, Bowling and Veloso, 2002, Brafman and Tennenholtz, 2002, Conitzer and Sandholm, 2007], we have the control of only one agent, which has to maximize its own rewards in a multi-agent environment. Instead, in the *offline* setting [Szepesvári and Littman, 1996, Lagoudakis and Parr, 2002, Perolat et al., 2015] we have control over all the agents. While the *offline* framework has received considerable attention in the last two years [Bai et al., 2020, Bai and Jin, 2020, Zhang et al., 2020b], it fails at modeling many use-cases of practical interest. For example, in many robot control problems, the agents interact with humans, which are non-controllable agents; another example is card/video-games, where it is unrealistic that the opponent will use the same learning algorithm. On the other hand, while the online setting is more suitable to model previous examples, it remains less studied. In Chapter 10 we analyze the online problem in general-sum Stochastic games, providing new insights on the performance limits and proposing a new algorithm for this setting. These contributions are original for this thesis.

Policy Optimization for Continuous (Stochastic) Games In Chapter 12 we propose a new policy-based algorithm for MARL. In MARL, the standard policy gradient algorithms fail due to the non-stationarity of the setting and the different interests of each agent [Mertikopoulos et al., 2018b, Mazumdar et al., 2020a]. In fact, algorithms must consider the complex dynamics of these systems to guarantee rapid convergence towards a (local) Nash equilibrium. We start the chapter reviewing the main concept of continuous games [Ratliff et al., 2013] and we start by casting the MARL problem in this setting. We propose a Newton-like algorithm for MARL problems. This method ensures quadratic convergence in two classes of games: (exact) Potential and Hamiltonian Games. Furthermore, we show that our algorithm is attracted to symmetric stable fixed points (a subset of stable fixed points) in general games and repelled by strict saddle ones.

Ramponi, G., & Restelli, M. (2021). Newton Optimization on Helmholtz Decomposition for Continuous Games. In Proceedings of the AAAI Conference on Artificial Intelligence.

CHAPTER 2

Reinforcement Learning

Within the research on Machine Learning, Reinforcement Learning (RL) proposes a method for learning through tests: the agent performs an action and receives feedback from the environment that corresponds to the "goodness" of its decision. Do you remember anything? When we have to learn a new task, we start taking “random” actions and learn from our mistakes. RL, taking inspiration by the human learning process, is a paradigm to learn in a sequential decision-making setting to optimize some reward signal. The RL problem involves a learning *agent* (or learner) which interacts with an *environment* during a sequence of discrete-time steps. This interaction is described by three components: the *state*, the *action* and the *reward* (see Figure 2.1). The *state* describes the actual configuration of the environment perceived by the agent, which can be a subset of the environment state’s characteristics. The *action* consists of the decision taken by the RL agent. The environment responds to every performed action with a state change and a *reward*. The *reward* is a numeric feedback of the agent’s performances. In RL, the interactions are formally described by a Markov Decision Process (MDP) [Bellman, 1957, Puterman, 2014]. In this chapter, we revise the basic concepts of RL taking inspiration from textbooks on this topic [Sutton et al., 1998, Puterman, 2014, Szepesvári, 2010]. The goal is not to provide

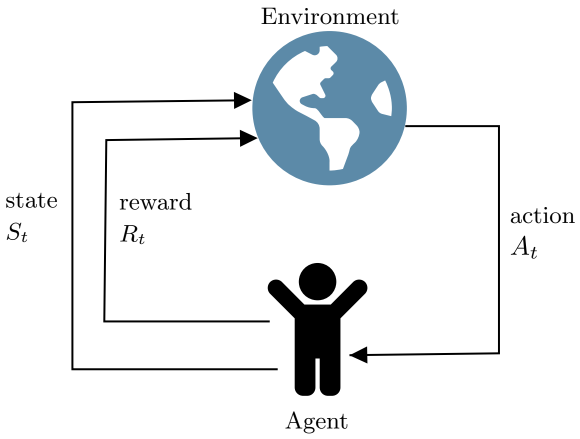


Figure 2.1: *The agent–environment interaction in a Markov decision process.*

an exhaustive review of the literature, but to discuss the fundamental ideas relevant to this thesis.

2.1 Markov Decision Processes

Markov Decision Process (MDP) [Puterman, 2014] is the canonical framework of RL to describe the interaction between agent and environment. In this section, we introduce the formal definition of the MDPs. Then we present the basic concepts of this mathematical formulation: policies, returns, value functions, action-value functions, and Bellman operators.

Definition 2.1.1 (Markov Decision Process)**.** *A (discounted) Markov Decision Process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu, H)$ specified by:*

- *A state space $\mathcal{S}$, which may be finite or infinite*[1].
- *An action space $\mathcal{A}$, which may be finite or infinite*[1].
- *A transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where with $\Delta(\mathcal{S})$ we indicate the space of probability distributions over $\mathcal{S}$. With $\mathcal{P}(s'|s, a)$ we indicate the probability of transitioning to state s' from state s taking action a.*
- *A reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. With $\mathcal{R}(s, a)$ we describe the immediate reward obtained in state s taking action a.*

[1]In this thesis, for mathematical convenience, we assume that the space is *measurable*.

- *A discount factor $\gamma \in [0, 1]$.*
- *An initial state distribution $\mu \in \Delta(\mathcal{S})$ which describes the probability of starting from any state.*
- *The horizon of the problem H that can be infinite or a positive integer.*

.

An MDP can be *finite* or *continuous*, depending if the state and the action spaces are finite or infinite. If $H = \infty$ we say that the MDP is infinite-horizon, instead if $H < \infty$ we say that the MDP is finite-horizon. Some MDPs, called *episodic*, have some special states called *terminal* or *absorbing* states; when the agent ends up in these states it cannot escape and the reward is 0. As introduced in the previous section, the interaction between the agent and the environment is partially controlled by the agent and partially by the environment. In fact, at each discrete time step h the agent takes an action a_h, and the state s_h transits to the state s_{h+1} based on the transition probability function $\mathcal{P}(\cdot|s_h, a_h)$. The agent at every interaction receives also a reward $\mathcal{R}(s_h, a_h)$ which depends on the current state s_h and the performed action a_h. In an MDP, the probabilities given by $\mathcal{P}$ completely characterize the environment dynamics. This property is called *Markov* property [Puterman, 2014].

2.1.1 Returns and Episodes

The reward function formalizes the agent's goal or intention; informally, the agent aims to maximize cumulative sum of the received rewards. Given a sequence of state-action pairs, we can evaluate the *discounted return*, which determines the importance of future rewards:

$$G_H = \sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t).$$

This quantity is a random variable depending on the agent's decisions and the transition probability. It is to notice that the discounted return is always finite in finite-horizon and also in infinite-horizon MDPs if $\gamma < 1$ [Sutton et al., 1998]. For example, if the reward is 1 in all states,

$$G_H = \sum_{h=0}^{\infty} \gamma^h \mathcal{R}(s_h, a_h) = \frac{1}{1-\gamma}.$$

We denote an episode or trajectory with $\tau = \tau_{0:H} = \{s_0, a_0, \ldots, s_{H-1}, a_{H-1}, s_H) \in \mathcal{T}$ where $\mathcal{T}$ is the space of all the trajectories associated to an

MDP. A trajectory describes the sequence of states and actions that the agent experiences in a certain simulation.

2.1.2 Policies and Value functions

The goal of an RL agent is to find a *policy* that maximizes the discounted return. A *policy*[2] is a set of decision rules for each time-steps $\pi : \{\pi_1, \dots, \pi_H\}$. Each decision rule π_h is a mapping from states to probability distributions over actions. In other words, if the agent is following the policy π at time h, $\pi_h(a|s_h)$ is the probability to select the action a in state s_h. If all decision rules in a policy π are equivalent we say that the policy is *stationary*. A policy is *deterministic* if for every state $s \in \mathcal{S}$, at every time step $h \in [0, H]$ the associated distribution $\pi_h(\cdot|s)$ is deterministic. In this case, with some abuse of notation, we write $\pi_h(s)$.

The value function of a state s under a policy π is the expected return when starting from a state s and following the policy π. We define formally the value functions as:

Definition 2.1.2 (Value function)**.** *Given a state $s \in \mathcal{S}$, a policy $\pi \in \Pi$, and a timestep $h \in [0, H)$ we define the value function $V^\pi_h(s)$ as:*

$$V^\pi_h(s) = \mathbb{E}\left[G^h_H|s_0 = s\right] = \mathbb{E}\left[\sum_{h'=h}^{H-1} \gamma^{h'}\mathcal{R}(s_{h'}, a_{h'})|s_h = s\right], \qquad (2.1)$$

where the expectation is taken under $a_h \sim \pi_h(\cdot|s_h)$ and $s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)$.

The value function describes the expected total amount of reward that the agent receives if it follows the policy π.

Similarly the value of taking an action a in state s at time h under a policy π defines the expected return of taking the action a in state s at time h following the policy π.

Definition 2.1.3 (Action-value function)**.** *Given a state $s \in \mathcal{S}$, an action $a \in \mathcal{A}$ and a policy $\pi \in \Pi$, at time h, we define the action-value function $Q^\pi_h(s, a)$, also called Q-function, as:*

$$Q^\pi_h(s, a) = \mathbb{E}\left[G^h_H|s_h = s, a_h = s\right] = \mathbb{E}\left[\sum_{h'=h}^{H-1} \gamma^{h'}\mathcal{R}(s_{h'}, a_{h'})|s_h = s, a_h = a\right], \qquad (2.2)$$

where the expectation is taken under $a_{h'} \sim \pi_h(\cdot|s_{h'})$ and $s_{h'+1} \sim \mathcal{P}(\cdot|s_{h'}, a_{h'})$.

[2]In this thesis we consider only Markov policies.

When $h = 0$ we will write $V^\pi(s)$ and $Q^\pi(s, a)$. Solving an RL task means finding a policy that maximizes the acquired reward over all the run. We say that a policy π is better or equal than a policy π' if the expected return of π is greater than the expected return of π' for all the states, i.e. $\forall s \in \mathcal{S}\ V^\pi(s) \geq V^{\pi'}(s)$. If a policy is greater or equal to all the other policies, this policy is called *optimal policy*. In every MDP there always exists at least one policy that is *optimal*:

$$\forall s \in \mathcal{S} : V^{\pi^\star}(s) = \sup_{\pi \in \Pi} V^\pi(s).$$

From this concept we can define the optimal value function and action-value function for every state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$:

$$V^\star(s) = \sup_{\pi \in \Pi} V^\pi(s),$$
$$Q^\star(s, a) = \sup_{\pi \in \Pi} Q^\pi(s, a).$$

To solve the episodic RL problem the agent seeks to find a policy that maximizes the expected discounted return under the initial state distribution.

Definition 2.1.4 (Expected discounted return)**.** *We define the expected discounted return of a policy π as:*

$$J(\pi) := \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t \mathcal{R}(s_h, a_h)\right] = \mathop{\mathbb{E}}_{s_0 \sim \mu}\left[V_\pi(s_0)\right],$$

where the expectation is taken with respect to $s_0 \sim \mu$, $s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)$, $a_h \sim \pi(\cdot|s_h)$.

So the the expected discounted return of optimal policy is:

$$J(\pi^\star) = \sup_{\pi \in \Pi} J(\pi).$$

2.1.3 Bellman operators

For infinite-horizon MDPs where $\gamma \in [0, 1)$ we define the following Bellman equations [Bellman, 1957]:

Definition 2.1.5 (Bellman equations)**.** *Let π be a stationary policy. Then $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$, $V^\pi(s)$ and $Q^\pi(s, a)$ satisfy the following Bellman equa-*

tions:

$$V^\pi(s) = \int_{a \in \mathcal{A}} \pi(a|s) Q^\pi(s,a) da,$$
$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \int_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) V^\pi(s') ds.$$

From these equations we define the Bellman operators:

Definition 2.1.6 (Bellman operators)**.** *Let π be a stationary policy. Then $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$ we define the following Bellman operators T^π for the value function and action-value function:*

$$(T^\pi V)(s) := \int_{a \in \mathcal{A}} \pi(a|s) \left(r(s,a) + \gamma \int_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) V^\pi(s') ds' \right) da,$$
$$(T^\pi Q)(s,a) := \mathcal{R}(s,a) + \gamma \int_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \int_{a' \in \mathcal{A}} Q^\pi(s',a') da' ds'.$$

From these definitions it easy to see that the discounted value functions satisfy the following consistency linear system:

$$(T^\pi V^\pi) = V^\pi.$$

In these class of MDPs an optimal policy also exists, and, moreover, a *deterministic* stationary policy always exists [Sutton et al., 1998]. The optimal value functions are the fixed points of the optimal Bellman operator:

Definition 2.1.7 (Bellman operators)**.** *We define $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$ the optimal Bellman operators $T^\star$ for the value function and action-value function:*

$$(T^\star V)(s) := \sup_{a \in \mathcal{A}} r(s,a) + \gamma \int_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) V(s') ds',$$
$$(T^\star Q)(s,a) := \mathcal{R}(s,a) + \gamma \int_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \sup_{a' \in \mathcal{A}} Q(s',a') ds'.$$

An important property of the Bellman operators is that they are γ-contracting with respect to the l_∞ norm [Puterman, 2014], if $\gamma \in [0,1)$:

$$\|T^\pi V - T^\pi V'\|_\infty \leq \gamma \|V - V'\|_\infty,$$
$$\|T^\star V - T^\star V'\|_\infty \leq \gamma \|V - V'\|_\infty.$$

2.1.4 Characteristics of Reinforcement Learning algorithms

Model-based vs model-free RL algorithms can be divided into *model-based* and *model-free* methods. Model-based algorithms learn or require the model of the environment and use techniques as dynamic programming. Instead, model-free approaches do not need an explicit formulation of the model and rely on learning from interactions with the environment.

On-policy vs off-policy The optimal solution of an MDP can be learned using two paradigms: *on-policy* and *off-policy* learning. The on-policy methods evaluate or improve the policy that is used to generate the data. On the other hand, off-policy algorithms evaluate or improve a different policy from the one used to interact with the environment.

Exploration-exploitation dilemma To learn an optimal policy on-line, so while interacting with the environment, an agent has to follow a policy that *sufficiently explores* state-action pairs, but that *exploits* the acquired knowledge of the environment to control the returns. This is known in literature as the exploration-exploitation dilemma.

Common policies We describe common policies used in RL problems. The first policy is the ϵ-greedy policy. The ϵ-greedy policy balances between exploration and exploitation. The policy at every time-step and for every state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ is equal to:

$$\pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} & \text{if } a = \arg\max_{a' \in \mathcal{A}} Q(s, a') \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}.$$

The Boltzmann policy instead uses the Boltzmann distribution with the action-value functions. So at every time-step h and for every state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$:

$$\pi(a|s) = \frac{\exp(\beta Q(s, a))}{\sum_{a' \in \mathcal{A}} \exp(\beta Q(s, a'))},$$

where β is a tunable parameter. The two parameters ϵ and β influences the exploration of the policy: lower is ϵ (or higher is β) more the policy is deterministic, and so more it exploits the best action; on the other hand, higher is ϵ (less is β more the policy is decide uniformly between the actions, so more it explores.

2.2 Exact Algorithms

This section provides some algorithms to exactly solve an MDP when we have full information, including the transition model $\mathcal{P}$ and the reward

Algorithm 1 Policy evaluation

```
Input: $\pi$ (the policy to be evaluated), $\epsilon$ (a small threshold determining the accuracy)
$\Delta = \infty$
$V_\pi(s) = 0 \quad \forall s \in \mathcal{S}$
while $\Delta > \epsilon$ do
  for each $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)V(s'))$
    $\Delta \leftarrow \min(\Delta, |V(s) - v|)$
  end for
end while
Output: value-function $V$
```

function. The methods that we expose are based on Dynamic Programming (DP) [Bertsekas et al., 1995, Powell, 2007]). In this section, we assume that the state and action spaces are finite, but a common way to find approximate solutions when the spaces are continuous is to discretize the state and action spaces and then apply finite-state algorithms.

2.2.1 Policy evaluation

The first problem that we solve is evaluating the performance of a policy. This task is also called *policy evaluation* or *policy prediction*. Since the environment dynamics are known, we can repetitively use the Bellman equations to get an increasingly accurate approximation of the value function. The initial approximation is chosen arbitrarily, and then we apply the Bellman operator to the current approximation of the value function (see Algorithm 1).

2.2.2 Policy iteration

With the policy evaluation we can estimate how "good" is a policy with respect to another one. Starting from a policy π we seek to find a policy π' that increases the performances of π. In order to do this we use the *policy improvement theorem* [Sutton et al., 1998] which states that, given two policies π and π', such that $\forall s \in \mathcal{S}$:

$$Q^\pi(s, \pi'(s)) \geq V^\pi(s),$$

then $\forall s \in \mathcal{S}$:

$$V^{\pi'}(s) \geq V^\pi(s).$$

The idea is that if following the policy π' we do not take worse decisions than following π, then π' is globally no worse than π. A way to construct

Algorithm 2 Policy iteration

```
Input: $\epsilon$ (a small threshold determining the accuracy)
$\Delta = \infty$
$V_\pi(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A} \quad \forall s \in \mathcal{S}$
while not policy-stable do
  Policy Evaluation
  while $\Delta > \epsilon$ do
    for each $s \in \mathcal{S}$ do
      $v \leftarrow V(s)$
      $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)V(s')$
      $\Delta \leftarrow \min(\Delta, |V(s) - v|)$
    end for
  end while
  Policy Iteration
  policy-stable $\leftarrow$ true
  for each $s \in \mathcal{S}$ do
    old-action $\leftarrow \pi(s)$
    $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)V_\pi(s')$
    If old-action $\neq \pi(s)$, then policy-stable $\leftarrow$ false
  end for
end while
Output: policy $\pi$
```

these improvements is to *greedy* update the current policy π selecting the best action at every state according to the current values of the action-value functions. In other words we create the *greedy* policy π' of π:

$$\begin{aligned}\pi'(s) &= \arg\max_{a \in \mathcal{A}} Q^\pi(s,a) \\ &= \arg\max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) V^\pi(s') \right).\end{aligned}$$

We can see that we are applying the Bellman optimal operator to the value function in order to increase the performance of the current policy. So this greedy operation generates a sequence of monotonically policy improvements:

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \cdots \rightarrow \pi^\star \rightarrow V^{\pi^\star},$$

where we alternate between a policy evaluation and a policy improvement. This process is called *policy iteration*, and we give a complete pseudo-code in Algorithm 2.

Algorithm 3 Value iteration

```
Input: $\epsilon$ (a small threshold determining the accuracy)
$\Delta = \infty$
$V_\pi(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A} \quad \forall s \in \mathcal{S}$
while $\Delta > \epsilon$ do
  for each $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)V(s')$
    $\Delta \leftarrow \min(\Delta, |V(s) - v|)$
  end for
end while
Output: a deterministic policy $\pi$ such that:
$\pi(s) = \arg\max_{a \in \mathcal{A}} \mathcal{R}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)V(s')$
```

2.2.3 Value Iteration

The problem of policy iteration is that every iteration involves a policy evaluation step. This quite expensive step is unnecessary since the policy evaluation can be truncated without effecting the policy iteration algorithm (see [Sutton et al., 1998]). The idea of Value Iteration is to stop the policy evaluation after one step. The Value Iteration algorithm combines the policy iteration and policy evaluation step:

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)V_\pi(s'),$$

for all $s \in \mathcal{S}$. In other words, the algorithm applies repetitively the Bellman optimal operator until convergence. A pseudo-code of Value Iteration is reported in Algorithm 3.

2.3 Value-based algorithms

In most RL tasks, we do not have access to the transition model $\mathcal{P}$ and the reward function $\mathcal{R}$, but we have to learn them from experience. Temporal Differences (TD) algorithms [Sutton et al., 1998] execute the current policy π and update at every step the estimation of the value function. The main idea of TD is to update the value function of a state with the estimated value of the next states and its reward. So, if we are in state s_t at time t, the TD update will be:

$$V(s_h) = V(s_h) + \alpha(\mathcal{R}(s_h, a_h) + \gamma V(s_{h+1}) - V(s_h)),$$

where α can be interpreted as a learning rate. The value inside the brackets is called Temporal Difference error, and can be regarded as how the current

Algorithm 4 SARSA

```
Input: $\alpha$ (step size), $\epsilon > 0$, $N$ (number of episodes), $H$ (episode's length)
$Q(s,a) \in \mathbb{R}\ \forall s \in \mathcal{S}, a \in \mathcal{A}$
for each episode $i \in [N]$ do
    Initialize $s \sim \mu$
    Choose action $a$ using policy derived from $Q(s)$ as $\epsilon$-greedy
    for each step $h \in [H]$ do
        Take action $a$, observe reward $\mathcal{R}(s,a)$ and next state $s'$
        Choose action $a'$ using policy derived from $Q(s')$ as $\epsilon$-greedy
        $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(\mathcal{R}(s,a) + \gamma Q(s',a'))$
        $s \leftarrow s'$, $a \leftarrow a'$
    end for
end for
Output: a deterministic policy $\pi$ such that:
$\pi(s) = \arg\max_{a\in\mathcal{A}} \mathcal{R}(s,a) + \sum_{s'\in\mathcal{S}} \mathcal{P}(s'|s,a)V(s')$
```

value function well estimates the real value function. In this section we review two well-known TD approaches: an on-policy TD algorithm (SARSA) and an off-policy TD algorihm (Q-learning).

2.3.1 SARSA

SARSA [Rummery and Niranjan, 1994] is an on-policy Temporal Difference algorithm that uses the Temporal Difference error to update the value function. The main difficulty is to estimate an action-value function instead of a value function:

$$Q(s_h, a_h) = Q(s_h, a_h) + \alpha(\mathcal{R}(s_h, a_h) + \gamma Q(s_{h+1}, a_{h+1}) - Q(s_h, a_h)),$$

so at every time step h the algorithm updates the action-value function of the current state $Q(s_h, a_h)$ with the received reward $\mathcal{R}(s_h, a_h)$ and the next action-value function $Q(s_{h+1}, a_{h+1})$. Since the rule uses the quintuple of events (s_h, a_h, $\mathcal{R}_h$, s_{h+1}, a_{h+1}) the name of the algorithm is Sarsa. The convergence properties of SARSA depend on the policies used. If the policy guarantees sufficient exploration such as ϵ-greedy policies or Boltzmann, the algorithm will converge to a deterministic one also optimal [Singh et al., 2000]. The SARSA asymptotic convergence was also studied under function approximation in [Perkins and Precup, 2003, Melo et al., 2008]. The pseudocode of SARSA update is presented in Algorithm 4.

Algorithm 5 Q-learning

```
Input: $\alpha$ (step size), $\epsilon > 0$, $N$ (number of episodes), $H$ (episode's length)
$Q(s, a) \in \mathbb{R} \, \forall s \in \mathcal{S}, a \in \mathcal{A}$
for each episode $i \in [N]$ do
    Initialize $s \sim \mu$
    for each step $h \in [H]$ do
        Take action $a$ using policy derived from $Q(s)$ as $\epsilon$-greedy, observe reward $\mathcal{R}(s, a)$
        and next state $s'$
        $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(\mathcal{R}(s, a) + \gamma \max_{a \in \mathcal{A}} Q(s', a))$
        $s \leftarrow s'$
    end for
end for
Output: a deterministic policy $\pi$ such that:
$\pi(s) = \arg\max_{a \in \mathcal{A}} \mathcal{R}(s, a) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a)V(s')$
```

2.3.2 Q-learning

The Q-learning algorithm [Watkins and Dayan, 1992] is one of the most popular algorithm in RL. As said before it is an off-policy TD algorithm and its update rule is defined as:

$$Q(s_h, a_h) = Q(s_h, a_h) + \alpha(\mathcal{R}(s_h, a_h) + \gamma \max_{a \in \mathcal{A}} Q(s_{h+1}, a) - Q(s_h, a_h)).$$

The update is very similar to the SARSA ones. The main difference is that Q-learning at each step approximates the Q^* with the maximum of the current Q-value, independently of the actual policy used. The success of this algorithm is due to its simplicity and the great convergence results [Watkins and Dayan, 1992, Tsitsiklis, 1994, Kearns and Singh, 1999, Szepesvári et al., 1997]. Recently in [Jin et al., 2018] the authors analysed the regret of this algorithm with some refining exploration techniques proving that it nearly-matches the optimal regret. A pseudocode of the classical Q-learning approach is given in Algorithm 5.

2.4 Policy Gradient Approaches

Policy gradient approaches [Deisenroth et al., 2013, Peters and Schaal, 2008b, Williams, 1992, Kakade, 2001] take a different perspective on the RL problem: we move from action-value based algorithms to methods which learn only a parametrized policy. With *parametrized policy* we intend that the agent's policy is a function $\pi_{\boldsymbol{\theta}}$ parametrized by a set of parameter $\boldsymbol{\theta} \in \mathbb{R}^d$, which determines the probability to select an action, without looking at the action-value estimates. These methods have some advantages and disadvantages over classical RL algorithms. The possibility to parametrize the policy permits to deal with high-dimensional problems [Peters and Schaal, 2008b] such as humanoid robots. Moreover, using an opportune parametrization, the policy can converge to a deterministic one, or can converge to any probability distribution. Another advantage is that the action's probabilities change in a smoother way, permitting to have stronger (and easier to prove) convergence guarantees. Finally, policy search methods permit to incorporate previous knowledge in the learning process [Deisenroth et al., 2013]. In this thesis, we present the episodic version of these algorithms (following the presentation in [Peters and Schaal, 2008b, Sutton et al., 1998]) and we refer to [Baxter and Bartlett, 2001, Deisenroth et al., 2013] for the infinite-horizon case.

In policy gradient methods, the policy can be parametrized in any way as long as it is continuously differentiable (one or two times depending on the algorithm) with respect to its parameters. We present now the common parametrization for discrete and continuous actions/states spaces. For discrete MDPs a natural parametrization is the Boltzmann (or soft-max) one:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{e^{f(a,s,\boldsymbol{\theta})}}{\sum_{a' \in \mathcal{A}} e^{f(a,s,\boldsymbol{\theta})}}.$$

The function f can be any arbitrary: computed by a Neural Network or linear in the state-action features ($f(s,a,\boldsymbol{\theta}) = \phi(s,a)^T\boldsymbol{\theta}$ where $\phi(s,a) \in \mathbb{R}^d$). For continuous MDP a natural parametrization is the Gaussian one:

$$\pi_{\boldsymbol{\theta}}(a|s) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{1}{2}\left(\frac{a - \mu_{\boldsymbol{\theta}}(s)}{\sigma}\right)^2 \right\},$$

where $\mu_{\boldsymbol{\theta}}(s)$ is the state-dependent mean and σ^2 is the variance.

The idea of policy search methods is to optimize the expected return using optimization methods, as for example stochastic gradient ascent:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{2.3}$$

where α is the learning rate. In order to evaluate the $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ different methods were proposed. In this thesis we report the REINFORCE, G(PO)MDP and Policy Gradient theorem [3]. The main skeleton of these approaches is presented in 6.

[3]We continue to use H instead of T to indicate the episode's lenght to be consistent in the thesis

Algorithm 6 Policy Gradient algorithm

```
Input: a gradient evaluation function $f$, a learning rate $\alpha$
Initialize $\boldsymbol{\theta}$ randomly
for $i = 1, 2, \dots$ do
    Perform a trial and obtain $D = \{\tau_1, \dots, \tau_k\}$
    $\widehat{g} \leftarrow f(\boldsymbol{\theta}, D)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \widehat{g}$
end for
```

2.4.1 REINFORCE

The REINFORCE algorithm was proposed by [Williams, 1992]. The REINFORCE policy gradient is given by:

$$\nabla_{\boldsymbol{\theta}} J^R(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau}\left[\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_h|s_h)\left(\mathcal{R}(\tau) - b\right)\right],$$

where b is a baselines and $\mathcal{R}(\tau) = \sum_{h=0}^{H-1} \gamma^h \mathcal{R}(s_h, a_h)$. The baseline aims at minimizing the variance of the gradient estimator, so it has to satisfy the following condition:

$$\nabla_b \text{Var}\left[\nabla_{\boldsymbol{\theta}} J^R_{\boldsymbol{\theta}}\right] = \nabla_b \mathop{\mathbb{E}}_{\tau}\left[(\nabla_{\boldsymbol{\theta}} J^R_{\boldsymbol{\theta}})^2\right] = 0.$$

So the optimal baseline, for each dimension d of the policy parameters $\boldsymbol{\theta}$, is equal to:

$$b^R_d = \frac{\mathbb{E}_\tau\left[\left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_d} \log \pi_{\boldsymbol{\theta}_d}(a_h|s_h)\right)^2 \mathcal{R}(\tau)\right]}{\mathbb{E}_\tau\left[\left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_d} \log \pi_{\boldsymbol{\theta}_d}(a_h|s_h)\right)^2\right]}.$$

2.4.2 G(PO)MDP

The G(PO)MDP algorithm [Baxter and Bartlett, 2001] considers that the rewards from the past do not depend on the future actions. So if we consider the reward at step h, we can ignore all the gradients of the future actions. From this idea we can derive the G(PO)MDP approximation of the gradient:

$$\nabla_{\boldsymbol{\theta}} J^R_{\boldsymbol{\theta}} = \mathop{\mathbb{E}}_{\tau}\left[\sum_{h=0}^{H-1}\sum_{j=0}^{h} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_j|s_j)\left(\mathcal{R}(s_h, a_h) - b_h\right)\right].$$

In this case also the baseline depends on the timestep. As we have done in the previous section for the REINFORCE algorithm, we can easily derive a baseline which minimizes the variance of the G(PO)MDP estimator:

$$b_{d,h}^{G} = \frac{\mathbb{E}_{\tau}\left[\left(\sum_{j=0}^{h} \nabla_{\boldsymbol{\theta}_d} \log \pi_{\boldsymbol{\theta}_d}(a_h|s_h)\right)^2 \mathcal{R}(s_h, a_h)\right]}{\mathbb{E}_{\tau}\left[\left(\sum_{j=0}^{h} \nabla_{\boldsymbol{\theta}_d} \log \pi_{\boldsymbol{\theta}_d}(a_h|s_h)\right)^2\right]}.$$

2.4.3 Policy Gradient Theorem

The Policy Gradient Theorem (PGT) algorithm is based on the Policy Gradient Theorem [Sutton et al., 1999], which, instead of using the reward $\mathcal{R}(s_h, a_h)$ at time step h, uses the expected value of the reward at this step. This value is given by the Q-value $Q_h^{\pi_{\boldsymbol{\theta}}}(s_h, a_h)$ at time h. The idea similarly to G(PO)MDP is that the rewards are not correlated to the future decisions. The Policy Gradient Theorem states that:

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}^{R} = \mathop{\mathbb{E}}_{\tau}\left[\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_h|s_h) \left(\sum_{j=h}^{H-1} \mathcal{R}(a_h, s_h)\right)\right].$$

This algorithm is equivalent to the G(PO)MDP version, .i.e., the two ones lead to the same gradient estimation [Deisenroth et al., 2013].

Remarks regarding policy gradient approaches A known problem of policy gradient estimators is the high variance, which is known to be a source of slow convergence. The REINFORCE algorithm is the one that suffers the most from that, in fact one of the purpose of constructing the G(PO)MDP estimator is of removing some unnecessary terms to reduce the variance. On the other hand, REINFORCE (without baseline) can be efficiently implemented using automatic differentiation tools; in fact it can be written as the gradient of a dot product between a vector of differentiable functions (the sums of log-policies) and a vector of constants (the sums of returns).

The baselines that we have introduced in the previous section aim at reducing the variance. However, the baselines cannot be computed exactly but they need to be estimated from data and, to keep the gradient estimate unbiased, it is necessary to use separate data to estimate it. However, in practice it is used a single batch of data, with the hope that the bias will have a negligible effect on the estimation error, compared to the variance reduction of the baseline.

Other more recent works investigate regularization effects on the policy gradient, to find the optimal trade-off between reducing the variance and introducing bias [Jaderberg et al., 2016, Namkoong and Duchi, 2017, Kartal et al., 2019, Lin et al., 2019b]. In RL this regularization usually corresponds to an additional entropy term, and it is used also to encourage the policy exploration. In [Zhao et al., 2016] and [Papini et al., 2018] the authors add a regularization term using the variance of the policy gradient, taking inspiration from SVRG [Johnson and Zhang, 2013]. This term provide more consistent policy improvements at the expense of reduced performance. Recently in [Flet-Berliac et al., 2021] the authors propose an actor-critic algorithm providing a new training objective for the critic based on the residual variance. However, the problem of finding the best trade-off between reducing the variance and introducing a bias is an open research question.

2.5 Actor-critic algorithms

The actor-critic algorithms are composed by: the *actor*, i.e., it is a parametrized policy $\pi_{\boldsymbol{\theta}}$ and a *critic*, i.e., it is a parametrized action-value function $Q_{\boldsymbol{\omega}}$ or value function $V_{\boldsymbol{\omega}}$ that evaluates the current policy. Actor-critic algorithms are the core components of Deep RL, where the actor and the critic component are approximated by neural networks. We revise the reader to recent surveys [Li, 2017, Arulkumaran et al., 2017, François-Lavet et al., 2018].

The idea behind actor-critic algorithms is that if the function approximation of the action-value function is "compatible" with the one used for the policy (where with compatible we intend that $\nabla_{\boldsymbol{\omega}} Q_{\boldsymbol{\omega}}(s,a) = \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(a|s)$), and $\boldsymbol{\omega}$ is a stationary point of the corresponding objective, then we can use $Q_{\boldsymbol{\omega}}$ instead of $Q_{\pi_{\boldsymbol{\theta}}}$ to evaluate the policy gradient [Sutton et al., 2000, Konda and Tsitsiklis, 2000]. These results where used in [Peters and Schaal, 2008a] combined with natural policy gradient. In [Schulman et al., 2015] the authors propose a trust-region optimization method (TRPO), and similarly in [Schulman et al., 2017b] it was introduced proximal policy optimization (PPO). The two algorithms achieved very effective results in learning high-dimensional tasks. In [Silver et al., 2014] the authors derive a deterministic version of the policy gradient, and it was extended to deep neural networks in [Lillicrap et al., 2015]. The advantage actor-critic (A3C) was proposed in [Mnih et al., 2015] where the authors use asynchronous actor learners and the policy is updated using the policy gradient theorem. The authors of [Haarnoja et al., 2018] use maximum entropy RL to derive soft actor-critic (SAC) algorithm, and the algorithm achieve state-of-the-art results in many continuous control tasks.

CHAPTER 3

Inverse Reinforcement Learning

In the previous chapter, we have introduced RL as a framework to learn how to perform a task described by a reward function. However, for many tasks, we already have experts (for example, humans) who know how to accomplish the same task. Moreover, in some cases it is extremely difficult to design a suitable reward function. The Imitation Learning paradigm aims to exploit the expert information to clone the experts' behavior or formalize the expert's intentions. The Imitation Learning framework is divided into two main subareas: Behavioral Cloning (BC) [Bain and Sammut, 1995] and Inverse Reinforcement Learning (IRL) [Russell, 1998]. Behavioral Cloning, as the name says, aims to clone the expert's behavior in order to use it as a policy. IRL, instead, can recover the expert's reward function to understand its intentions and use this reward function to learn an optimal policy in any environment, even different from the one in which the expert acts. In this thesis, we focus our attention on the IRL framework. For more information about BC, we suggest the reader to look at the following survey [Osa et al., 2018].

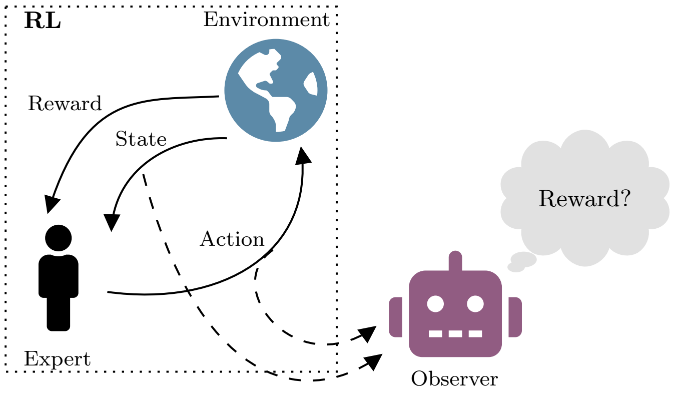


Figure 3.1: *The expert-observer interaction in the Inverse Reinforcement Learning framework.*

3.1 Problem statement

As said before, the IRL framework is composed of two agents: an expert who shows how to perform a task and an observer who watches the expert's demonstrations and learns from them the reward function. This setting is described in Figure 3.1. The framework used in IRL problems is the Markov Decision Process without Rewards (MDP$\backslash\mathcal{R}$). An MDP$\backslash\mathcal{R}$ is defined as a tuple $\mathcal{M}\backslash\mathcal{R} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mu, H)$ which is equal to an MDP but without having the notion of the reward function. [Russell, 1998] defines the objective of the IRL problem as:

Determine the reward function that the expert is optimizing.

The observer practically receives a dataset of expert's demonstrations, or it has access to the observer's policy. Then the observer has to recover a reward function, which makes the expert's behavior optimal. One of the major difficulties of the IRL problem is that the objective function is ill-posed [Ng et al., 2000] since a policy can be optimal for many reward functions. To obtain the solution many different reward choice functions were proposed based on different reasonable principles including feature-based matching [Abbeel and Ng, 2004], maximum margin planning [Ratliff et al., 2006], maximum entropy [Ziebart et al., 2008, Ziebart et al., 2010], Generative Adversarial learning [Ho and Ermon, 2016], Bayesian framework [Ramachandran and Amir, 2007], boosting methods [Ratliff et al., 2007], and Gaussian Processes [Levine et al., 2011]. The majority of the IRL algorithms require an iterative learning process that alternates between recovering the reward function and evaluating the reward function in the environment, i.e., computing the optimal policy for the recovered reward function. However, in the literature, some algorithms do not require this interaction with the environment and use only the given set of expert trajectories.

As in standard RL, the IRL algorithms can be divided in model-based approaches and model-free ones. The first approaches need to estimate a model of the environment to solve the IRL problem or to have access to the MDP. The second ones, instead, recover the reward functions using only the expert's demonstrations. For this part we follow the presentation of the survey [Osa et al., 2018].

3.2 Model-based Inverse Reinforcement Learning

The IRL algorithms of this section leverage on a previous knowledge of the system dynamics. In this section we describe three model-based algorithms for IRL: Feature Expectation Matching [Abbeel and Ng, 2004], Maximum Entropy IRL [Ziebart et al., 2008], Maximum Likelihood IRL [Babes et al., 2011].

3.2.1 Feature Expectation Matching

In [Abbeel and Ng, 2004] the authors propose an algorithm which solves the IRL problem matching the feature expectation of the expert's policy. We start by assuming that the reward function is linear in the state-action features, i.e. given a state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ the reward is given by:

$$\mathcal{R}(s, a) = \boldsymbol{\omega}^T \boldsymbol{\phi}(s, a),$$

where $\boldsymbol{\phi}(s, a) \in \mathbb{R}^d$ is a known feature vector of the state-action pair and $\boldsymbol{\omega} \in \mathbb{R}^d$ is an unknown weight vector. From this, we define the expected return, given a policy π, as:

$$\begin{aligned} J(\pi) &= \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h \mathcal{R}(s_h, a_h)\right] \\ &= \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h \boldsymbol{\omega}^T \boldsymbol{\phi}(s_h, a_h)\right] \\ &= \boldsymbol{\omega}^T \, \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h \boldsymbol{\phi}(s_h, a_h)\right], \end{aligned}$$

where the expectation is taken with respect to $s_0 \sim \mu$, $s_{h+1} \sim \mathcal{P}(\cdot|s_t, a_t)$, $a_h \sim \pi(\cdot|s_h)$. We define the right-hand side as the *feature expectation* $\boldsymbol{\psi}(\pi)$:

$$\boldsymbol{\psi}(\pi) = \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h \phi(s_h, a_h)\right].$$

From this equations we can write the expected return as:

$$J(\pi) = \boldsymbol{\omega}^T \psi(\pi).$$

The idea of [Abbeel and Ng, 2004] is to learn the reward weights which induce an optimal policy that matches the feature expectations of the expert's one. The algorithm alternates between computing the weights that minimize the distance between the actual feature expectations and the expert's feature expectation, and update the feature expectations evaluating the optimal policy for the current weights. One advantage of feature-expectation matching is that the resulting policy performs provably close to the expert's one.

However, this algorithm is still ambiguous since many policies can match the expert's feature expectations. To solve this issue, [Ratliff et al., 2006] proposed the Maximum Margin Planning algorithm, which finds the reward weights that maximize the difference between the optimal policy and the others. Another idea is to maximize over the policy distributions using the maximum entropy principle [Ziebart et al., 2008]. We explain this algorithm in the next section.

3.2.2 Maximum Entropy Inverse Reinforcement Learning

In the previous section, we described an approach that finds a reward function to match the expert's feature expectations. However, many policies have this characteristic. [Ziebart et al., 2008] propose an algorithm to solve this problem by finding the distribution that maximizes the entropy among all the distributions that match the feature expectations. This algorithm, based on the Maximum Entropy principle [Jaynes, 1957], learns a policy that maximizes the entropy:

$$\begin{aligned}
&\max_{\pi} \mathcal{H}(\pi) = \max_{\pi} \sum_{\tau} \pi(\tau) \log\left(\frac{1}{\pi(\tau)}\right) \\
&\text{s.t.} \psi(\pi) = \psi(\pi^E) \\
&\quad \sum_{\tau} \pi(\tau) = 1, \forall \tau, \pi(\tau) \geq 0,
\end{aligned}$$

where $\pi(\tau) = \prod_{h=0}^{H-1} \pi(a_h|s_h)$ and π^E is the expert policy. Ziebart et al. proved that the distribution that maximizes the entropy and satisfies the constraints follows:

$$\pi(\tau) \propto \exp\left(\mathcal{R}(\tau)\right),$$

where $\mathcal{R}(\tau) = \boldsymbol{\omega}^T \sum_{h=0}^{H-1} \boldsymbol{\phi}(s_h, a_h)$. From this equation we can express the probability to see a trajectory in terms of the reward weights:

$$p(\tau|\boldsymbol{\omega}) = \frac{\exp\left(\boldsymbol{\omega}^T \phi(\tau)\right)}{Z(\boldsymbol{\omega})},$$

where $Z(\boldsymbol{\omega}) = \sum_{\tau'} \exp\left(\boldsymbol{\omega}^T \phi(\tau)\right)$ is the partition function and $p(\tau|\boldsymbol{\omega})$ is the probability of see a trajectory τ given the reward weights $\boldsymbol{\omega}$. This equation holds only for deterministic environments, while for stochastic environments we have to add also the transition probabilities information:

$$p(\tau|\boldsymbol{\omega}) = \frac{\exp\left(\boldsymbol{\omega}^T \phi(\tau)\right)}{Z(\boldsymbol{\omega})} \mu(s_0) \prod_{h=1}^{H-1} P(s_h|s_{h-1}, a_{h-1}).$$

From this, we can rewrite the optimization problem as:

$$\arg\max_{\boldsymbol{\omega}} \sum_{\tau \in D} \log p(\tau|\boldsymbol{\omega}),$$

where D is the expert trajectories dataset. This is a convex optimization problem, which can be simply solved using gradient ascent optimization.

3.2.3 Maximum Likelihood Inverse Reinforcement Learning

The Maximum Likelihood IRL algorithm was proposed in [Babes et al., 2011]. The idea behind this algorithm is to find the reward weights which maximize the likelihood of the given dataset of expert demonstrations. The authors define the policy as a Boltzmann policy:

$$\pi_{\boldsymbol{\omega}}(a|s) = \frac{\exp\left(\beta Q(s, a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\beta Q(s, a')\right)}.$$

Then they state that,

$$Q_{\boldsymbol{\omega}}(s, a) = \boldsymbol{\omega}^T \phi(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a) \sum_{a' \in \mathcal{A}} \frac{\exp\left(\beta Q(s', a')\right)}{\sum_{a'' \in \mathcal{A}} \exp\left(\beta Q(s', a'')\right)} Q(s', a'),$$

and they optimize the log-likelihood of the given trajectories D:

$$\max_{\boldsymbol{\omega}} L(D|\boldsymbol{\omega}) = \sum_{\tau \in D} \sum_{s,a \in \tau} \log \pi_{\boldsymbol{\omega}}(a|s).$$

The algorithm alternates between computing the current Q-values and a gradient-ascent step to optimize the likelihood.

3.3 Model-free Inverse Reinforcement Learning

In many real applications the exact dynamics are often unknown, and it could be not possible to interact with the environment. The algorithms that we present in this section solve these issues without requiring the transition probabilities of the underlying MDP.

3.3.1 Relative Entropy Inverse Reinforcement Learning

The approach of Relative Entropy Inverse Reinforcement Learning (REIRL) [Boularias et al., 2011] takes inspiration by Relative Entropy Policy Search [Peters et al., 2010] and Maximum Entropy IRL [Ziebart et al., 2008]. The algorithm receives as input a set of trajectories D and minimizes the relative entropy between a prior trajectory distribution p_0 and the trajectory distribution p_L induced by the learned policy π_L. Since the transition model is unknown the authors estimate the trajectory distribution using Importance Sampling techniques. Similarly to Maximum Entropy IRL the optimization problem can be formalized as:

$$\begin{aligned}
&\min_{p_L} \sum_{\tau \in D} p_L(\tau) \log\left(\frac{p_L(\tau)}{p_0(\tau)}\right) \\
&\text{s.t } \|\boldsymbol{\psi}(\pi^L) = \boldsymbol{\psi}(\pi^E)\|_1 \leq \epsilon \\
&\quad \sum_{\tau} p_L(\tau) = 1, \forall \tau, p_L(\tau) > 0,
\end{aligned}$$

where ϵ is evaluated using a Hoeffding bound. From this formulation we can derive the Lagrangian of this problem as:

$$\begin{aligned}
\sum_{\tau \in D} p_L(\tau) \log\left(\frac{p_L(\tau)}{p_0(\tau)}\right) - \omega^T \left(\sum_{\tau \in D} p(\tau)\phi(\tau) - \psi(\pi^E)\right) \\
- \sum_{i=1}^{d} |\boldsymbol{\omega}_i|\epsilon_i + \eta\left(\sum_{\tau \in D} p(\tau) - 1\right).
\end{aligned}$$

Then the authors evaluate the dual problem and solve it using sub-gradient methods and importance sampling.

3.3.2 Cascaded Supervised Learning Approach to Inverse Reinforcement Learning

Cascaded Supervised Inverse Reinforcement Learning [Klein et al., 2013, CSI] starts with an initial classification step that defines a *score function* followed by a regression step which provides the reward function. The first step of the algorithm is a classification that associated a score function $q^C \in \mathcal{S} \times \mathcal{A}$ and a decision rule $\pi^C : \mathcal{S} \to \mathcal{A}$ for every action $a \in \mathcal{A}$ and state $s \in \mathcal{S}$. The classification can be seen as a method to infer the action-value function of the RL problem. This step is performed using a *score function-based multi-class classifier* (SFMC), as for example a Multi-class Support Vector Machine. From these two functions we can derive a reward $\mathcal{R}^C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

$$\mathcal{R}^C(s,a) = q^C(s,a) - \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) q^C(s', \pi^C(s')). \tag{3.1}$$

However, since the transition model is unknown and we need to approximate it. It is assumed to have another dataset D^R of transitions, performed by an arbitrary policy. From this dataset we can acquire the information about the dynamics of the system. From this dataset we construct another dataset D' where, for every $(s, a, s') \in D^R$, we add the triplet:

$$(s, a, \tilde{r}) \qquad \tilde{r} = q^C(s,a) - \gamma q(s', \pi^C(s')).$$

Then a regressor is used to fed the points in the dataset D' to derive the final reward function $\widetilde{\mathcal{R}}$.

In the paper it is also provided a theoretical analysis in which the authors show that the demonstrated policy is near-optimal with respect to the recovered reward function.

3.3.3 Gradient Inverse Reinforcement Learning

In [Pirotta and Restelli, 2016] the authors present the Gradient Inverse Reinforcement Learning algorithm. This is, as far as we know, the first IRL algorithm based on the policy gradient. The authors consider policy parametrized by a parameter $\boldsymbol{\theta}$ as in section 2.4. We start by defining the policy gradient [Deisenroth et al., 2013] written as its decomposition under a linear reward model $\mathcal{R}_{\boldsymbol{\omega}}(s,a) = \boldsymbol{\omega}^T \boldsymbol{\phi}(s,a)$:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}) = \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t \mathcal{R}_{\boldsymbol{\omega}}(s_t, a_t) \sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_l|s_l)\right] = \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega},$$

where $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) = (\nabla_{\boldsymbol{\theta}}\psi_1(\boldsymbol{\theta})| \dots |\nabla_{\boldsymbol{\theta}}\psi_q(\boldsymbol{\theta})) \in \mathbb{R}^{d\times q}$ is the Jacobian matrix. When the expert's policy $\pi^E_{\boldsymbol{\theta}} \in \Pi_\Theta$ optimizes the reward function $\mathcal{R}_{\boldsymbol{\omega}^E}$, $\boldsymbol{\theta}^E$ is a *stationary point* of the expected return $J(\boldsymbol{\theta}, \boldsymbol{\omega}^E)$. So, the gradient of $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^E, \boldsymbol{\omega}^E) = \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)\boldsymbol{\omega}^E$ must vanish by the first-order necessary conditions for optimality [Nocedal and Wright, 2006]. In other words, the weight $\boldsymbol{\omega}^E$, associated to the reward function optimized by the expert, belongs to the *null space* of the Jacobian $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)$.

GIRL [Pirotta and Restelli, 2016] leverages on this observation to recover the expert's weight vector $\boldsymbol{\omega}^E$. In practice we do not have access to the true Jacobian matrix $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)$, but to a finite-sample estimate $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)$, obtained starting from the trajectories in $D = \{\tau_1, ..., \tau_n\}$. As we presented in previous chapter 2, an unbiased sample-based estimate of $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ can be obtained with the standard policy gradient estimators, such as REINFORCE [Williams, 1992] or G(PO)MDP [Baxter and Bartlett, 2001]. However, this estimation might result of full rank due to estimation errors, preventing the search of the corresponding null space. For this reason, GIRL, instead of looking for the null space of $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)$, seeks for the direction of minimum growth by minimizing the L^p-norm of the gradient estimate, leading to the optimization problem:

$$\min_{\substack{\boldsymbol{\omega}\in\mathbb{R}^q_+ \\ \|\boldsymbol{\omega}\|_1=1}} \left\| \widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}^E)\boldsymbol{\omega} \right\|_p^p . \tag{3.2}$$

This objective has the desirable property of being convex for any choice of $p \geq 1$.

CHAPTER 4

Multi-agent Reinforcement Learning

In the previous chapter we have considered the basic RL setting, where one agent is learning a task in an environment without the presence of other agents. In reality, the majority of the real-world problems are multi-agent, i.e., multiple agents act in the same context. We can mention for example successful results on the application of RL algorithms: beating the world champion player of Go [Silver et al., 2017], solving robotic control problems [Lillicrap et al., 2015], managing the power consumption of households [Chung et al., 2020], and achieving promising advancements in autonomous driving [Shalev-Shwartz et al., 2016]. These tasks can be modeled as Multi-Agent Reinforcement Learning (MARL) problems, a framework that characterizes all the decision-making problems with more than one agent. The agents can cooperate, compete, and optimize their own expected return. This chapter introduces the preliminaries necessary to understand the MARL framework, following the lines of [Zhang et al., 2019a, Buşoniu et al., 2010]. In the beginning, we introduce the Normal Form Games [Myerson, 2013], then we formalize the Stochastic Games [Shapley, 1953], and we expose the equilibrium concepts. Then we introduce the primary differentiation in the MARL framework: cooperative, competitive, and general-sum games. Finally, we introduce some approaches to solve

		Player 2	
		head	*tail*
Player 1	*head*	$1, -1$	$-1, 1$
	tail	$-1, 1$	$1, -1$

Table 4.1: *Reward (payoff) function of the Matching Pennies game.*

these problems. As for RL, this chapter is not intended to give a full review of the MARL literature but only to have the necessary preliminaries for this thesis. In Chapter 8 and 11 we provide more details on the related work for the three algorithms that we presented in this thesis. We remind the reader of recent surveys on this topic [Zhang et al., 2019a, Buşoniu et al., 2010, Nguyen et al., 2020, OroojlooyJadid and Hajinezhad, 2019, Da Silva and Costa, 2019, Hernandez-Leal et al., 2019, Papoudakis et al., 2019] to have a complete overview of the MARL works. Moreover in appendix A it is reported a brief introduction of learning in games, and in particular of learning in continuous games.

4.1 Normal-form Games

A Normal-form game [Myerson, 2013, NFG] is the most straightforward game that we can consider. A Normal-form game is formally defined as follows.

Definition 4.1.1 (Normal-form game)**.** *A N-player Normal-form game is a tuple $\mathcal{NFG} = (N, \mathcal{A}, \mathcal{R})$ where:*

- *N is the number of agents in the game,*
- *$\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ where $\mathcal{A}_i$ is the finite set of actions of agent i,*
- *$\mathcal{R} = (\mathcal{R}_1, \dots, \mathcal{R}_N)$, where $\mathcal{R}_i : \mathcal{A} \to \mathbb{R}$ is a real-valued reward function (payoff function) for the agent i.*

If the action space is finite, the game is finite, otherwise if the action space is continuous the game is continuous. An example of a Normal-form game is reported in Table 4.1. The game in the table is the Matching Pennies game. The rows are the possible actions for the first agent, and the columns correspond to the second player's action. Each player's reward is described in the cell: the left for the first player and the right for the second one. As for the single-agent RL, we can define a deterministic policy (or pure strategy) if the agent selects with probability 1 only one action. Instead, the policy is stochastic (or mixed) if it specifies a probability distribution over the actions. We define the "gain" of an agent i as the expected return:

$$J_i^{\pi_i,\pi_{-i}} = \mathbb{E}[\mathcal{R}_i],$$

where the expectation is taken under the policies π_i and the joint policies $\pi_{-i} = (\pi_1, \dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_N)$, i.e., the joint policy of all the players except i. We can define the goodness of a policy by introducing the concept of *best response*.

Definition 4.1.2 (Best response). *The agent i is playing its best response to the opponents' policies (or strategy profile) π_{-i}, if its policy π_i is such that $J_i^{\pi_i,\pi_{-i}} \geq J_i^{\pi'_i,\pi_{-i}}$ for every $\pi'_i \in \Delta(\mathcal{A}_i)$.*

From this definition, we introduce the first equilibrium concept: the Nash Equilibrium [Nash et al., 1950].

Definition 4.1.3 (Nash equilibrium). *A joint strategy $\pi = (\pi_1, \dots, \pi_N)$ is a Nash equilibrium if, for all agents $i \in [N]$ [1], π_i is the best response to π_{-i}.*

It was proved that every Normal-form game has a Nash Equilibrium [Nash et al., 1950]. However, finding a Nash Equilibrium is a PPAD complete problem already in a three-player game [Papadimitriou, 1992, Daskalakis et al., 2009]. Another problem with this concept is that there could be more than one Nash Equilibrium in the game, and it is not easy to decide what equilibrium is the *best* among the others.

Another equilibrium concept is the Stackelberg Equilibrium. The Stackelberg paradigm was initially introduced to model a particular economic situation where there is a *leader* that moves first and a *follower* that reacts to the leader movement [Von Stackelberg, 2010]. More formally, in a Stackelberg game, an agent, called the leader, takes an action and then the other agents, the followers, respond to the leader's decision. The Stackelberg game can model situations in which the actions are asynchronous, and some agents decide their actions by looking at the previous agent's decision. In this thesis, we consider only the case in which there is one leader and one follower [2]. With π_{LD}, we indicate the leader policy, and with π_{FL} the follower policy.

Definition 4.1.4 (Follower best response). *A follower best response is a set of policies BR$(\pi_{LD}) = \arg\max_{\pi_{FL} \in \Pi_{FL}} J_{FL}^{\pi_{LD},\pi_{FL}}$. The follower response function is a function $\zeta : \Pi_{LD} \to \Pi_{FL}$ such that $\zeta(\pi_{LD}) \in$ BR(π_{LD}) for every leader's policy $\pi_{LD} \in \Pi_{LD}$.*

[1] With $[N]$ we indicate all the integers between 1 and N.

[2] The extension to the multiple-follower setting is easy to be inferred from our definitions.

The Stackelberg equilibrium is then defined as follows.

Definition 4.1.5 (Stackelberg equilibrium). *For a Stackelberg game with one leader and one follower, given a follower response function ζ, a ζ-Stackelberg equilibrium is a policy π^{LD} such that:*

$$\pi_{LD} = \arg\max_{\pi_{LD} \in \Pi_{LD}} J_{LD}^{\pi_{LD}, \zeta(\pi_{LD})}$$

A joint policy (π_{LD}, π_{FL}) is a Stackleberg equilibrium if there exists ζ such that π_{LD} is a ζ-Stackelberg equilibrium and $\zeta(\pi_{LD}) = \pi_{FL}$.

There are also other important solution concepts in literature, such as Correlated equilibrium, Coarse-correlated equilibrium [Aumann, 1987], Bayesian equilibrium [Fudenberg and Tirole, 1991]. Since these solution concepts were not used in this thesis, we do not report their definitions.

4.2 Stochastic Games

In this section, we extend the notion of the Normal-form game and the Markov Decision Process to Stochastic Games. Stochastic Games (or Markov Games) [Shapley, 1953, SG] are a generalization of an MDP to the multi-agent setting and an extension of the Normal-form games to the scenario where there is more than one state.

Definition 4.2.1 (Stochastic Games). *A Stochastic Game (or Markov game) is a tuple $\mathcal{SG} = (N, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu, H)$ specified by:*

- *N agents,*
- *A state space $\mathcal{S}$, which may be finite or infinite,*
- *An action space $\mathcal{A} = (\mathcal{A}_1, \dots, \mathcal{A}_N)$, where $\mathcal{A}_i$ are the actions of the i-th agent,*
- *A transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where with $\mathcal{P}(s'|s, \boldsymbol{a})$ we indicate the probability of transitioning to state s' from state s taking actions $\boldsymbol{a} = (a_1, \dots, a_N)$,*
- *A set of reward functions $\mathcal{R} = (\mathcal{R}_1, \dots, \mathcal{R}_N)$ where $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for the agent i,*
- *A set of discount factors $\gamma = (\gamma_1, \dots, \gamma_N)$, where $\gamma_i \in [0, 1]$ for all $i \in [N]$,*

- *An initial state distribution $\mu \in \Delta(\mathcal{S})$ which describes the probability of starting from any state.*
- *The horizon H of the problem that can be infinite or a finite integer.*

As for MDPs, given a state s under a joint policy π we can define the expected return starting from s.

Definition 4.2.2 (Value function)**.** *Given a state $s \in \mathcal{S}$ and a joint policy $\pi = (\pi_1, \dots, \pi_N)$, we define the value function $V_{\pi,i}(s)$ for the agent $i \in [N]$ as:*

$$V^{\pi}_{i,h}(s) = \mathbb{E}\left[\sum_{h'=h}^{H-1} \gamma_i^{h'} \mathcal{R}_i(s_{h'}, \boldsymbol{a}_{h'}) | s_h = s\right], \tag{4.1}$$

where $\boldsymbol{a}_h = (a^1_h, \dots, a^N_h)$ and the expectation is taken under $\boldsymbol{a}_h \sim \pi(\cdot|s_h)$ and $s_{h+1} \sim \mathcal{P}(\cdot|s_h, \boldsymbol{a}_h)$.

Then we can define the action-value function as follows.

Definition 4.2.3 (Action-value function)**.** *Given a state $s \in \mathcal{S}$ and a joint policy $\pi = (\pi_1, \dots, \pi_N)$, at time h, we define the action-value function $Q^h_{i,\pi}(s)$, also called Q-function, for the agent i, as:*

$$Q^{\pi}_{i,h}(s, a) = \mathbb{E}\left[\sum_{h'=h}^{H-1} \gamma^{h'} \mathcal{R}_i(s_{h'}, \boldsymbol{a}_{h'}) | s_h = s, \boldsymbol{a}_h = \boldsymbol{a}\right], \tag{4.2}$$

where $\boldsymbol{a}_h = (a^1_h, \dots, a^N_h)$ and the expectation is taken under $\boldsymbol{a}_h \sim \pi(\cdot|s_h)$ and $s_{h+1} \sim \mathcal{P}(\cdot|s_h, \boldsymbol{a}_h)$.

As we have done for the single-agent RL, we define the expected discounted return for the agent i as:

$$J^{\pi}_i = \mathbb{E}_{s \sim \mu}[V^{\pi}_i(s)]. \tag{4.3}$$

We can easily extend the concepts of Nash Equilibrium and Stackelberg equilibrium to Stochastic games (see Section 4.1).

A special case of Stochastic games are the Turn-based Stochastic games [Shapley, 1953, TSG]. The main difference between Stochastic Games and Turn-based Stochastic games is the interaction between the agents. For example, in a 2-agent Turn-based Stochastic game, the state space is divided between the two agents $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, and, at each time step h, the game is in one state s_h and the agent that controls this state chooses an action from its action space $\mathcal{A}$. We define a Turn-based Stochastic game more formally as follows.

Definition 4.2.4 (Turn-based Stochastic games)**.** *A Turn-based Stochastic game is a tuple $\mathcal{TSG} = (N, \mathcal{S}_1, \dots, \mathcal{S}_N, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu, H)$ specified by:*

- *N agents,*
- *A state space $\mathcal{S} = \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_N$, which may be finite or infinite,*
- *An action space $\mathcal{A}$,*
- *A transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, where with $\mathcal{P}(s'|s,a)$ we indicate the probability of transitioning to state s' from state s taking action a,*
- *A set of reward functions $\mathcal{R} = (\mathcal{R}_1, \dots, \mathcal{R}_N)$ where $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for the agent i,*
- *A set of discount factors $\gamma = (\gamma_1, \dots, \gamma_N)$, where $\gamma_i \in [0,1]$ for all $i \in [N]$,*
- *An initial state distribution $\mu \in \Delta(\mathcal{S})$ which describes the probability of starting from any state.*
- *The horizon H of the problem that can be infinite or a finite integer.*

For Turn-based Stochastic games, we can define an opportune notion of action-value function derived from the one of the stochastic games. Since this is an easy modification, we do not report it here.

The reward functions of each agent characterize every game. Different reward functions generate different game dynamics. We can identify three main *classes* of games: cooperative, zero-sum, and general-sum games.

Best response evaluation Given a joint policy π_{-i} of the other agents, the problem of finding the best response is the same as solving an MDP. In fact, given the other agent policies, the Stochastic games reduce to a single-agent Markov Decision Process. In this case, we can use methods such as Value-iteration or Policy-iteration to solve this task.

4.3 Cooperative stochastic games

Cooperative games are characterized by full cooperation between the agents. All the agents have the same reward function $\mathcal{R}_1 = \mathcal{R}_2 = \cdots = \mathcal{R}_N$. This setting is referred in literature as *Multi-agent MDPs* [Boutilier, 1996, Lauer and Riedmiller, 2000] in the AI community, *team Markov games* [Yoshikawa, 1978, Wang and Sandholm, 2002] in the control/game theory community, and *exact Potential games* from a game theoretic perspectives [González-Sánchez and Hernández-Lerma, 2013, Lã et al., 2016].

In these games, if we apply the Bellman operator for single-agent RL over the joint action space, and from its unique solution $V^\star$, we can derive a Nash equilibrium policy.

Value-based methods In this setting, the action-value functions are identical between all agents, and for this reason, we could use Q-learning taking the maximum over the joint action space $\mathcal{A}$. In [Littman, 2001] converge to NE policy was established to convergence to the Nash equilibrium policy if there is a unique equilibrium solution, or the agents are coordinated in its selection. An example in which the agent are coordinated in the equilibrium selection was proposed in *optimal adaptive learning* [Wang and Sandholm, 2002], which is the first cooperative MARL algorithm with provable convergence to a Nash equilibrium.

Policy gradient methods In the cooperative framework, since each agent optimizes the same function, we can apply single-agent RL policy gradient algorithms.

4.4 Competitive stochastic games

Competitive (zero-sum) games are characterized by the property that for every state $s \in \mathcal{S}$, actions $\mathbf{a} = (a_1, \dots, a_n)$ the sum of the agents rewards is equal to 0, i.e. $\sum_{i=1}^{N} \mathcal{R}_i(s, a) = 0$. This class of games receives much attention since it models natural settings such as two agents that compete against each other [Littman, 1994], and *robust learning* since the other agent can be seen as the *uncertainty* that makes the learning process difficult [Jacobson, 1973, Başar and Bernhard, 2008]. However, there is a great difference between solving a two-agent or multi-agent stochastic game; in fact, also finding a Nash equilibrium for a three-player zero-sum stochastic game is demonstrated to be PPAD-complete [Papadimitriou, 1992, Daskalakis et al., 2009]. For this reason, most algorithms are concentrated on two-player zero-sum games. In the rest of this section, we focus on the two-player setting.

In the zero-sum games we have that the expected discounted return of the first agent is equal to the negated of the expected discounted return of the second agent, i.e., $V_1^{\pi_1,\pi_2}(s) = -V_2^{\pi_1,\pi_2}(s)$ for each $s \in \mathcal{S}$. From the Von Neumann's minimax theorem [Von Neumann and Morgenstern, 2007]

Algorithm 7 Minimax-Q

```
Input: learning rate $\alpha$, number of rolls out $N$
Initialize $s \sim \mu$, $Q_1 \in \mathbb{R}$, $V = \mathbf{0}$, $\pi_1$
for $i = 1, 2, \dots, N$ do
    $a_1 \leftarrow \pi_1(\cdot|s)$
    Take action $a_1$ observe reward $\mathcal{R}$, next state $s'$ and opponent action $a_2$
    $V_1(s) = \max_{\pi(s)\in\Delta(\mathcal{A}_1)} \min_{a'_2\in\mathcal{A}_2} \sum_{a'_1\in\mathcal{A}'_1} \pi(a_1|s)Q_1(s, a'_1, a'_2)$
    $Q_1(s, a_1, a_2) \leftarrow (1-\alpha)Q_1(s, a_1, a_2) + \alpha(\mathcal{R} + \gamma V(s))$
    $\pi_1(s) \leftarrow \arg\max_{\pi(s)\in\Delta(\mathcal{A}_1)} \min_{a'_2\in\mathcal{A}_2} \sum_{a'_1\in\mathcal{A}_1} \pi(a'_1|s)Q_1(s, a'_1, a'_2)$
    $s \leftarrow s'$
end for
```

we define the optimal value function $V^\star : \mathcal{S} \to \mathbb{R}$ as, for every $s \in \mathcal{S}$:

$$V^\star(s) = \max_{\pi_1} \min_{\pi_2} V_1^{\pi_1,\pi_2}(s) = \min_{\pi_2} \max_{\pi_1} V_1^{\pi_1,\pi_2}(s).$$

This implies that the minimax solution is also a Nash equilibrium solution. [Shapley, 1953] shows that $V^\star$ is the unique solution of a Bellman equation, and from that, we can construct a Nash equilibrium joint policy:

$$(T^\star V^\star)(s) = \max_{\pi_1(\cdot|s)} \min_{\pi_2(\cdot|s)} \sum_{a_1\in\mathcal{A}_1} \sum_{a_2\in\mathcal{A}_2} \pi_1(a_1|s)\pi_2(a_2|s)Q_1(s, a_1, a_2).$$

To solve the max-min optimization problem we can use a linear program [Vanderbei et al., 2015]. This Bellman operator is γ-contractive in the infinite norm and has $V^\star$ as unique solution. From this Bellman operator we can derive, as in MDPs, a Value-iteration algorithm [Littman, 2001], and policy-iteration algorithms [Hoffman and Karp, 1966, Van Der Wal, 1978, Rao et al., 1973].

Value-based methods In [Littman, 1994] was proposed Minimax-Q, an algorithm based on Q-learning to solve two-player zero-sum stochastic games. As we said before, the minimax equilibrium is equal to the Nash equilibrium in this class of games. For this reason, the author simply uses the minimax operator to take into account the opponent actions:

$$\pi_1(\cdot|s) = \arg\max_{\pi(\cdot|s)\in\Pi_1} \min_{a_2\in\mathcal{A}_2} \sum_{a_1\in\mathcal{A}_1} \pi(a_1|s)Q(s, a_1, a_2)$$

This algorithm is guaranteed to converge in zero-sum stochastic games to a stationary Nash equilibrium under similar assumption as Q-learning [Szepesvári and Littman, 1999]. The pseudocode of Minimax-Q is presented in 7. Minimax-Q was also extended to function-approximation setting [Lagoudakis and Parr, 2002]. Other value-based algorithms were proposed to solve the zero-sum stochastic game setting, based on different principles such as actor-critic fictitious play [Perolat et al., 2018].

Policy-gradient methods Recently there was a growing interest into *continuous* games. In this case finding a Nash Equilibrium in a continuous game reduces to a nonconvex-nonconcave problem [Chasnov et al., 2020a]. In this setting, it was proved that policy gradient algorithms, such as REINFORCE, fail to converge to a Nash equilibrium solution, due to the cyclic behavior of the game dynamics. We provide more information about these algorithms in Chapter 11.

4.5 General-sum stochastic games

The general-sum setting is notorious by the most challenging one, and the less understood. In this setting, finding a Nash Equilibrium is PPAD compete even for a two-player normal-form game [Chen et al., 2009]. Moreover, it was proved that the value-iteration method fails to find Nash equilibria and, also, policy-gradient methods avoid a subset of Nash equilibria in general-sum continuous games [Zinkevich et al., 2006, Kearns et al., 2013].

However, in finite-horizon undiscounted stochastic games we can construct a value-iteration like algorithm that converges to Nash-equilibrium policies. The algorithm, described in [Kearns et al., 2013], computes for every timestep h the Q-value functions for each state s, actions $a_1 \in \mathcal{A}_1$, $a_2 \in \mathcal{A}_2$ and agent $i \in \{1, 2\}$:

$$
\begin{aligned}
Q_{i,h}^{\pi_1,\pi_2}(s, a_1, a_2) = \mathcal{R}_i(s, a_1, a_2) + \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a_1, a_2) \\
\sum_{a_1' \in \mathcal{A}_1} \pi_1(a_1'|s') \sum_{a_2' \in \mathcal{A}_2} \pi_2(a_2'|s') V_{i,h-1}^{\pi_1,\pi_2}(s', a_1', a_2').
\end{aligned}
$$

Then it computes a Nash equilibrium for the state game $(Q_{1,h}^{\pi_1,\pi_2}(s, \cdot, \cdot)$, $Q_{1,h}^{\pi_1,\pi_2}(s, \cdot, \cdot))$. This algorithm is proved to find a Nash equilibrium solution policies [Kearns et al., 2013].

Value-based methods Under some strict assumptions some value-based algorithms were proposed for the general-sum setting. The algorithm Nash-Q [Hu and Wellman, 2003], similar to Minimax-Q, generalized the Q-learning algorithm to the multi-agent setting. The algorithm at each iteration

Algorithm 8 Nash-Q

```
**Input**: learning rate $\alpha$, number of rolls out $K$
Initialize $s \sim \mu$, $Q_1, \dots, Q_N$, $V_1, \dots, V_n$, $\pi_1, \dots \pi_N$
for $i = 1, 2, \dots, N$ do
    $a_1 \leftarrow \pi_1(\cdot|s)$
    Take action $a_i$ observe rewards $\mathcal{R}_1, \dots, \mathcal{R}_n$, next state $s'$, other agent actions $a_2, \dots, a_n$
    for every agent $j \in [N]$ do
        Update $Q_j(s, a_1, \dots, a_n) \leftarrow (1-\alpha)Q_j(s, a_1, \dots, a_n) + \alpha(\mathcal{R}_j(s, a_1, \dots, a_n) + \gamma\pi_1 \cdot \dots \cdot \pi_N V_j(s'))$
    end for
    $\pi_1(s), \dots, \pi_N(s) \leftarrow$ Nash$(Q_1(s), \dots, Q_N(s))$
    $s \leftarrow s'$
end for
```

i for every state $s \in \mathcal{S}$ evaluates the policies of all the agents computing the Nash equilibrium for the current stage game:

$$(\pi_1(\cdot|s), \dots, \pi_N(\cdot|s)) = \text{Nash}(Q_{i,1}(s, \cdot), \dots, Q_{i,N}(s, \cdot)),$$

where the function Nash computes the Nash equilibrium of the game. This algorithm is proved to converge to a Nash equilibrium solution under the assumption that there exists a unique Nash equilibrium every time it is necessary to compute the Nash function. Moreover, every agent has to see the other agents actions and rewards. The pseudocode of Nash-Q is presented in Algorithm 8.

Recently Perolat et al. have shown that the minimizer of the empirical Bellman residual is an approximate Nash equilibrium, and the authors use this finding to construct an algorithm for the general-sum setting [Pérolat et al., 2017].

Policy-gradient methods Policy-based algorithms were proposed to solve general-sum games. However the convergence is guaranteed only in specific classes of games: policy prediction in two-player two-action bimatrix games [Zhang and Lesser, 2010, Song et al., 2019]; WoLF in two-player two-action games [Bowling and Veloso, 2002]; AWESOME in repeated games [Conitzer and Sandholm, 2007]. Recently, many algorithms have been studied to apply policy gradient methods to general-sum continuous games. As for zero-sum games, we cannot apply policy-gradient algorithms from single-agent RL, due to the cyclic dynamics of these games [Chasnov et al., 2020a]. However, recently many algorithms which take inspiration from dynamical system theory were proposed [Balduzzi et al., 2018, Letcher et al., 2019, Foerster et al., 2018, Fiez et al., 2020a, Mescheder et al., 2017]. The convergence guarantees of these algorithms are not strong as for the zero-sum and cooperative games, but, in most cases, are limited to a subset of local Nash equilibrium points. We provide a broader overview of these methods in Chapter 11.

Part II

Inverse Reinforcement Learning in Multi-Agent Systems

The IRL [Ng et al., 2000, Osa et al., 2018] framework aims at recovering the reward function of an optimal agent. In the classical setting, an expert, i.e., an agent that has already learned a task, makes available a dataset of its interactions with the environment. From this, the IRL algorithm recovers the reward function that the expert is optimizing. When there are multiple agents in the environment, also the IRL framework changes its objective. For example, there could be multiple experts who show their possible different behaviors, leading to an increase in available data, but a necessity to cluster them by their intentions. Or, an agent can be interested in learning the adversary reward function, to use it to compute its strategy or to cooperate; however, it has to discover it without waiting for the other agent's convergence to an optimal policy.

In this part, we address the problems associated with IRL in MA environments. We start by presenting in Chapter 5 the new chances and issues that the multi-agent setting creates, and we revise the literature connected to these problems. Then in Chapter 6, we present the first contribution of this thesis, introducing a new algorithm for the IRL about Multiple Intentions [Babes et al., 2011] setting. The proposed algorithm [Ramponi et al., 2020b, Likmeta et al., 2021] aims at clustering a set of experts by their intentions, recovering them for each cluster. Finally, in Chapter 7 we present an algorithm to solve the IRL from a learning agent problem [Jacq et al., 2019]. The proposed algorithm [Ramponi et al., 2020a] infers the reward function while observing an agent that is actually learning a task.

CHAPTER 5

Inverse Reinforcement Learning for multi-agent systems

The IRL framework is widely studied in the standard setting, i.e., when there is an observer with access to a dataset of expert's interactions with the environment. These interactions, encoded by the expert's policy, are optimizing an *unknown* reward function. The IRL goal consists of finding a reward function that explains the expert's demonstrations, i.e., that makes the expert's policy optimal [Ng et al., 2000]. In the literature [Argall et al., 2009, Hussein et al., 2017, Osa et al., 2018] there are other methods beyond IRL to address an imitation learning problem, such as Behavioral Cloning [Bain and Sammut, 1995], which outputs an imitating policy (e.g., Behavioral Cloning [Argall et al., 2009]). IRL, however, compared to them, explicitly provides a succinct representation of the expert's *intention*. This produces two main advantages: first, the reward function offers a generalization of the expert's policy to unobserved situations, also permitting to use it in different environments; second, the recovered reward has an explainability power, since its description of the expert's behavior can also help in system design issues.

However, in real-world situations in which it could be useful to use the

IRL setting, there could be more than one agent performing a task. The imitation-learning paradigm is used, for example, in autonomous driving research, but there is more than one driver on the streets. Moreover, if we are competing with opponents on some board games, we would like to know their intentions before they become experts. These novel problems cannot be solved using the algorithms presented in Chapter 3. Exploiting the knowledge that we are in a multi-agent environment creates possibilities and new challenges which we describe below.

Inverse Reinforcement Learning about Multiple Intentions An interesting and practical opportunity that a natural multi-agent system can provide is the possibility to observe more than one expert optimizing the same reward function. For example, we can have access to more driving demonstrations from different drivers or download data from different social-network users. The usage of this larger dataset can help batch model-free IRL algorithms, which leverage only the demonstrations' information to recover the reward function. Moreover, we can use this data to cluster different users by their intentions. For example, in a social network, identifying users' intents and using them to cluster the users can provide valuable information for designing effective marketing or advertising strategies. This more general problem is called Inverse Reinforcement Learning about Multiple Intentions [Babes et al., 2011, Choi and Kim, 2012, Almingol and Montesano, 2015, MI-IRL]. As said before, in this framework, we aim to identify groups of experts who share the same goal.

Inverse Reinforcement Learning from a learning agent In a multi-agent scenario, in most cases, it is not possible to wait for the convergence of the agent's learning process . In fact, an agent has to infer the unknown reward functions that the other agents are learning before actually becoming "experts" to either cooperate or compete with them. The knowledge of the other agents' reward function is necessary for most situations to have a learning algorithm with *good* convergence properties or to minimize the regret, as we will show in Chapters 9 and 12. Moreover, in many situations, we can learn something useful by observing an agent's learning process. These observations contain essential information about the agent's intentions and can be used to infer its interests. Imagine a driver learning a new circuit. During its training, we can observe how it behaves in various situations (even dangerous ones), which is useful for understanding which states are good and which should be avoided. Instead, only a small sub-region of the state space could be explored when observing expert behavior, thus leaving the observer unaware of what to do in situations that are unlikely under the expert policy. This framework, which we called Inverse Reinforcement Learning from a learner (IRLfL), was recently proposed by Jacq et al. in [Jacq et al., 2019]. This setting involves two agents: a *learner* who is currently learning a task (who takes the place of the expert) and an *observer* who wants to infer the learner's intentions.

Multi-agent Inverse Reinforcement Learning The first challenge that could come to our mind in a multi-agent setting is the necessity to change the objective of the IRL problem. In a multi-agent environment, an agent is optimizing a multi-agent expected reward function, so it could be useful to change the IRL goal to, for example, finding a set of reward functions that explain the Nash equilibrium of the system. This problem is defined in the literature as Multi-agent IRL [Natarajan et al., 2010, Hadfield-Menell et al., 2016, Lin et al., 2019a], but it is out of the scope of this thesis.

5.1 Preliminaries

In this section we provide the preliminaries necessary for the next chapters. As commonly done in the IRL literature [Pirotta and Restelli, 2016, Ziebart et al., 2008, Abbeel and Ng, 2004], we assume that the expert's reward function can be represented by a linear combination with unknown weights $\boldsymbol{\omega}$ of q basis functions $\boldsymbol{\phi}$:

$$\mathcal{R}_{\boldsymbol{\omega}}(s,a) = \boldsymbol{\omega}^T \boldsymbol{\phi}(s,a), \quad \boldsymbol{\omega} \in \mathbb{R}^q, \tag{5.1}$$

where $\boldsymbol{\phi} : \mathcal{S} \times \mathcal{A} \to [-M_r, M_r]^q$ is a known bounded feature vector function.

We define the *feature expectations* of a policy $\pi_{\boldsymbol{\theta}}$ as:

$$\boldsymbol{\psi}(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu, \\ a_h \sim \pi_{\boldsymbol{\theta}}(\cdot|s_h), \\ s_{h+1} \sim P(\cdot|s_h, a_h)}} \left[\sum_{h=0}^{+\infty} \gamma^h \boldsymbol{\phi}(s_h, a_h) \right]. \tag{5.2}$$

The *expected discounted return*, under the linear reward model, is defined as:

$$J(\boldsymbol{\theta}, \boldsymbol{\omega}) = \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu, \\ a_h \sim \pi_{\boldsymbol{\theta}}(\cdot|s_h), \\ s_{h+1} \sim P(\cdot|s_h, a_h)}} \left[\sum_{h=0}^{+\infty} \gamma^h \mathcal{R}_{\boldsymbol{\omega}}(s_h, a_h) \right] = \boldsymbol{\omega}^T \boldsymbol{\psi}(\boldsymbol{\theta}). \tag{5.3}$$

Then the gradient of expected discounted return [Pirotta and Restelli, 2016] is equal to:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}) &= \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu, \\ a_h \sim \pi_{\boldsymbol{\theta}}(\cdot|s_h), \\ s_{h+1} \sim P(\cdot|s_h, a_h)}} \left[\sum_{h=0}^{+\infty} \gamma^h \mathcal{R}_{\boldsymbol{\omega}}(s_h, a_h) \sum_{l=0}^{h} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_l|s_l) \right] \\ &= \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega},\end{aligned}$$

where $\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) = (\nabla_{\boldsymbol{\theta}} \psi_1(\boldsymbol{\theta}) | \dots | \nabla_{\boldsymbol{\theta}} \psi_q(\boldsymbol{\theta})) \in \mathbb{R}^{d \times q}$ is the Jacobian matrix of the feature expectations $\boldsymbol{\psi}(\boldsymbol{\theta})$ w.r.t. the policy parameters $\boldsymbol{\theta}$.

Behavioral cloning step In order to use gradient-based IRL approaches is necessary the knowledge of the functional form of the expert's policy, in order to compute the score $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}^E}(a|s)$. However, in many cases the policy is unknown and we have to estimate it by the demonstrations. We assume that the expert's policy belongs to a parametric policy space Π_Θ made up of differentiable policies, so it is possible to recover an approximation of the expert's parameters $\boldsymbol{\theta}^E$ through behavioral cloning, exploiting the trajectories in D. We reduce this problem to a maximum-likelihood estimation, solving the following optimization problem:

$$\max_{\boldsymbol{\theta} \in \Theta} \frac{1}{n} \sum_{l=1}^{n} \sum_{h=0}^{H-1} \log \pi_{\boldsymbol{\theta}}(a_{l,h}|s_{l,h}). \tag{5.4}$$

From this, we obtain an estimate $\widehat{\boldsymbol{\theta}}^E$ of $\boldsymbol{\theta}^E$. It is known that the maximum likelihood estimation is consistent under mild regularity conditions on the policy space Π_Θ and assuming the identifiability property [Casella and Berger, 2002]. Some finite-sample guarantees on the concentration of the distance $\|\widehat{\boldsymbol{\theta}}^E - \boldsymbol{\theta}^E\|_p$ were also derived under stronger assumptions [Spokoiny et al., 2012].

Reward features In this thesis, we consider linear reward features that are built by practitioners. However, in recent years, there is a growing interest in learning the representation in an autonomous way. For example, [Munk et al., 2016, Ota et al., 2020, Rajeswaran et al., 2017, Zhang et al., 2018] authors learn state representation to improve Deep RL performance. In order to integrate these new techniques with the algorithms presented in the next chapters, a possible solution could be to learn the representation of the reward features that reduces the loss function of the IRL algorithm. However, this method does not guarantee that the recovered reward function is better (in performance) than others. Achieving a smarter solution to unify the two methods requires more in-depth analysis.

5.2 Related work

In this section, we present the related work for the two settings mentioned before: IRL about Multiple Intentions and IRLfL. We describe in details the algorithms most related to ours, which are used as baselines for our approaches.

5.2.1 Inverse Reinforcement Learning about Multiple Intentions

The Inverse Reinforcement Learning about Multiple Intentions problem was proposed by [Babes et al., 2011]. The authors solved it via an Expectation-Maximization (EM) approach considering a Boltzmann policy for the expert. The algorithm requires that the number of clusters is known. To relax this assumption, [Choi and Kim, 2012] employed a Bayesian non-parametric approach. These algorithms work only for finite MDPs. Instead, some recent works addressed the MI-IRLproblem in continuous state-action spaces [Almingol and Montesano, 2015, Rajasekaran et al., 2017]. In [Tateo et al., 2017] the authors presented a model-free method for solving a single IRL problem given a set of trajectories generated by different experts' policies. In this section, we explain more in detail the Maximum Likelihood IRL for Multiple Intentions algorithm [Babes et al., 2011], since it was used as a baseline for our experiments.

Maximum Likelihood IRL for Multiple Intentions

The authors assume there exist K intentions, each one represented by reward weights $\boldsymbol{\omega}_i$, with $i \in [K]$. The observer receives a dataset of $N > K$ trajectories $D = \{\tau_1, \dots, \tau_n\}$, and its goal is to recover the K intentions from D. The Expectation-Maximization Maximum Likelihood IRL (EM-MLIRL) algorithm adopts EM to compute a maximum likelihood model in the face of missing data, which are, in this case, the clusters' labels. The algorithm is model-based, i.e., it needs to have access to the transition model to compute the reward parameters. The expectation step calculates the probability that a trajectory is generated by an expert who is optimizing the i-th intention with $i \in [K]$. Then, in the maximization step, for every intention $i \in [K]$, the MLIRL algorithm (see Chapter 3) is used to recover the i-th intention, weighting each trajectory with the probabilities computed in the expectation step. The process is repeated until convergence.

5.2.2 Inverse Reinforcement Learning from a Learner

The problem of estimating the reward function of an agent who is learning is relatively new. This setting was proposed by Jacq et al. [Jacq et al., 2019] and, to the best of our knowledge, it is studied only in that work.

In another line of work, sub-optimal demonstrations are used in preference-based IRL [Christiano et al., 2017, Ibarz et al., 2018] and in ranking-based IRL [Brown et al., 2019, Castro et al., 2019]. Some of these works require the algorithm to ask a human to compare the possible agent's trajectories to learn the underlying reward function of the task. Instead, in [Balakrishna et al., 2020] an Imitation Learning setting was proposed where the observer tries to imitate the behavior of a supervisor who demonstrates a convergent sequence of policies.

Other works related to the IRLfL setting are the ones on *theory of minds* [Rabinowitz et al., 2018, Shum et al., 2019]. In these papers, the authors propose algorithms that use meta-learning to build a system that learns how to model other agents. It is not required that agents are experts, but they must be stationary. This assumption cannot be satisfied in the IRLfL setting since the observed agent changes its policy during the learning process.

Learning from a Learner

In [Jacq et al., 2019] the authors proposed a method based on entropy-regularized reinforcement learning, in which they assumed that the learner is performing soft policy improvements. The authors start by considering that every policy update generates a strict policy improvement, where the policy update π_2 from π_1 is performed by:

$$\forall s \in \mathcal{S} \quad \pi_2(\cdot|s) = \operatorname*{arg\,max}_{\pi'(\cdot|s)\in\Pi} \mathop{\mathbb{E}}_{a\sim\pi'(\cdot|s)} \left[Q^{\pi^1}(s,a) \right].$$

Since these policies can be only deterministic they proposed to solve this issue by placing the algorithm in an entropy-regularized framework. The entropy-regularized framework permits to construct the greedily-improving policies as:

$$\pi_2(a|s) \propto \exp\left(Q^{\pi_1}(s,a)\right).$$

Such greedy improvements, called soft policy improvements (SPIs) are used in well-known algorithms such as Soft-Actor-Critic [Haarnoja et al., 2018]. The objective of the algorithm is now to extract the reward function that explains all the observed SPIs. Previously to do that, the LfL algorithm recovers the policies by behavioral cloning on the dataset of observed trajectories.

Ranking-based Inverse Reinforcement Learning

The Trajectory-ranked Reward EXploration (T-REX) [Brown et al., 2019] algorithm uses ranked trajectories to learn a reward function from sub-optimal demonstrations. The algorithm, given the ranked trajectories, infers the reward using a neural network which optimizes:

$$\mathrm{P}(J(\boldsymbol{\omega})(\tau_i) < J(\boldsymbol{\omega})(\tau_j)) \sim \frac{\exp\left(\sum_{s\in\tau_j} r_{\boldsymbol{\omega}}(s)\right)}{\exp\left(\sum_{s\in\tau_i} r_{\boldsymbol{\omega}}(s)\right) + \exp\left(\sum_{s\in\tau_j} r_{\boldsymbol{\omega}}(s)\right)}$$

if $i < j$, where $J(\boldsymbol{\omega})(\tau_i)$ is the return of the trajectory τ_i with the reward $r_{\boldsymbol{\omega}}$. In order to augment the dataset size the authors train the network on partial couples of trajectories.

CHAPTER 6

Inverse Reinforcement Learning about Multiple Intentions

As we explained in the previous chapter, the problem of MI-IRL [Babes et al., 2011] involves an observer who has access to the demonstrations performed by multiple experts. The observer has to recover the reward functions and use them to cluster the observed agents. Solving this problem is helpful for an explainability reason since it could be used to understand the similarity between apparently different agents. Moreover, as an immediate benefit, grouping experts who show other behaviors but share the same intent allows enlarging the set of demonstrations available for the reward recovery process. This has a significant impact on several realistic scenarios, where the only information available is the demonstration dataset, and no further interactions with the environment are allowed, such as in batch model-free IRL (see Chapter 3). This setting is particularly challenging but quite common. Consider, for instance, the problem of inferring the intentions of a group of car drivers, given a set of demonstrations. We cannot perform forward learning (at least in a non-simulated environment) for safety reasons, and, typically, the amount of data available is not enough to perform off-line learning. However, in the literature, there are not many approaches to solve the batch model-free IRL problem.

In this chapter, we propose a novel batch model-free IRL algorithm, named Σ*-Gradient Inverse Reinforcement Learning* (Σ-GIRL), and then we extend it to the multiple-intention setting. Σ-GIRL is derived from Gradient Inverse Reinforcement Learning [Pirotta and Restelli, 2016, GIRL]. GIRL, as explained in Chapter 5, is an IRL algorithm that searches for a reward function that makes the estimated *policy gradient* [Deisenroth et al., 2013] vanish. Such reward function is a stationary point of the expected return and, under suitable conditions, it makes the policy demonstrated by the expert an optimal policy. The algorithm we present, instead, explicitly considers the uncertainty in the gradient estimation process. In particular, we cast the IRL problem as a constrained maximum likelihood problem, in which we look for the reward function that maximizes the likelihood of the estimated policy gradients, under the constraint that such reward is a stationary point of the expected return. The resulting objective function accounts for the variance of the gradient and reduces to the GIRL case for a specific choice of the covariance model. Then, we embed Σ-GIRL into the multiple-intention framework by proposing a *clustering algorithm* that, by exploiting the likelihood model of Σ-GIRL, groups the experts according to their intentions. The optimization of the multiple-intention objective is performed in an *expectation-maximization* (EM) fashion, in which the (soft) agent-cluster assignments and the reward functions are obtained through an alternating optimization process (Section 6.4). Finally, in Section 6.6 we present an experimental evaluation aimed at highlighting the performance of Σ-GIRL, compared with state-of-the-art methods on simulated domains and on a real-world experiment in which we recover and cluster the intents of a group of Twitter users (Section 6.6).

6.1 Problem statement

In Inverse Reinforcement Learning about Multiple Intentions [Babes et al., 2011] there are a set of experts $\mathbf{E} = (E_1, \dots, E_m)$ and a set of (unknown) reward functions $\boldsymbol{\mathcal{R}} = (\mathcal{R}_{\boldsymbol{\omega}_1}, \dots, \mathcal{R}_{\boldsymbol{\omega}_k})$, with $k \leq m$. Each expert $E_i \in \mathbf{E}$ demonstrates a policy $\pi_{E_i} \in \Pi_\Theta$ (i.e., there exists $\boldsymbol{\theta}^{E_i} \in \Theta$ such that $\pi_{E_i} = \pi_{\boldsymbol{\theta}^{E_i}}$). The policy π_{E_i} optimizes a reward $R_{\boldsymbol{\omega}_i}$, with $i \in \{1, \dots, k\}$.

The observer receives a set of datasets $\mathcal{D} = (D_1, \dots, D_m)$, where each $D_i = \{\tau_1, \dots, \tau_{n_i}\}$ is the set of n_i trajectories demonstrated by E_i. The observer's goal is to recover the reward functions optimized by each of the m experts. An illustration of the model is given in Figure 6.1.

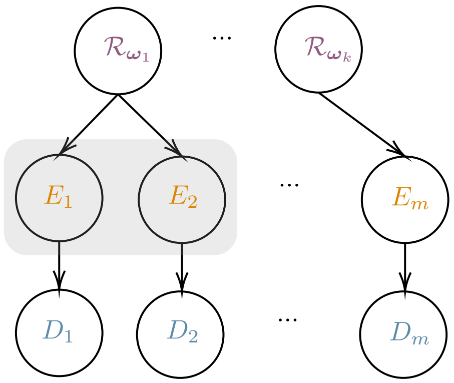


Figure 6.1: *Graphical model of the MI-IRL problem: from $\mathcal{R}_1, \dots, \mathcal{R}_k$, m different expert policies are generated, which in turn generate m demonstrations' datasets.*

A naïve approach to this problem would be to solve m independent IRL problems. However, since, in practice, we typically have $k \ll m$, this solution would be highly sub-optimal since it would produce an excessively large number of distinct reward functions. Moreover, if we have few demonstrations per-agent, then the reward recovered with this solution can be highly suboptimal. A more interesting and efficient solution will be to cluster the experts' dataset to recover k reward functions that explain the experts' behaviors.

We assume to know the identity of the expert generating each trajectory and the number of reward functions k, but not the policy parameters $\boldsymbol{\theta}^{E_i}$.

6.2 Σ-Gradient Inverse Reinforcement Learning

We start by introducing a new batch model-free IRL algorithm, named *Σ-Gradient Inverse Reinforcement Learning* (Σ-GIRL). Σ-GIRL extends GIRL [Pirotta and Restelli, 2016] (see Chapter 3) to account for the uncertainties injected by the Jacobian estimation process.

The GIRL algorithm finds the reward function that minimizes the gradient of the expected discounted return i.e., it seeks weights ω that belong to the null-space of the Jacobian $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$. However, in most cases, we do not have access to the true Jacobian, but we have to estimate it by samples. For this reason, it could be possible that the null-space of the estimated Jacobian is empty. GIRL, also in this situation, treats every estimated component in the same way without using the information about the uncertainty due to the Jacobian estimation. We take a different perspective employing directly the knowledge of the variance estimates: we allow the components of the estimated Jacobian $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ to move in order to generate a new Jacobian $\mathbf{M} \in \mathbb{R}^{d\times q}$ that has a non-empty null space. The idea of our algorithm is that the more a component $\widehat{\nabla}_{\boldsymbol{\theta}}\psi_{ij}(\boldsymbol{\theta})$ is uncertain, the more we are allowed to change it since it is more *plausible* that we have made some errors in its estimation. To formalize this intuition, we reformulate the minimization problem of GIRL into a constrained maximum-likelihood problem. Since the Jacobian is a mean of n samples, by the Central Limit Theorem, its distribution approaches a normal as n grows to infinity [Casella and Berger, 2002]. So, the idea is to model the Jacobian estimation as a (matrix) Gaussian distribution [Gupta and Nagar, 2018]: $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) \sim \mathcal{N}\left(\mathbf{M}, \frac{1}{n}\boldsymbol{\Sigma}\right)$, where $\mathbf{M}$ is the mean of the Jacobian estimation and $\boldsymbol{\Sigma} \in \mathbb{R}^{dq\times dq}$ is the covariance matrix.The corresponding likelihood function, given the trajectory dataset $D = \{\tau_1, \dots \tau_n\}$ is:

$$\mathcal{L}_{\boldsymbol{\Sigma}}(\mathbf{M}|D) = \frac{\sqrt{n}}{\sqrt{(2\pi)^{dq}|\boldsymbol{\Sigma}|}} e^{-\frac{n}{2}\left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})-\mathbf{M}\right)\right\|^2_{\boldsymbol{\Sigma}^{-1}}}, \tag{6.1}$$

where, for a matrix $\mathbf{A}$, $\mathrm{vec}(\mathbf{A})$ denotes the vectorization of $\mathbf{A}$, i.e. the vector obtained by stacking the columns of $\mathbf{A}$. We now formulate the IRL problem as the problem of finding the weight vector $\boldsymbol{\omega}$ and the new Jacobian $\mathbf{M}$ that, jointly, maximize the likelihood, while $\boldsymbol{\omega}$ belongs to the null space of $\mathbf{M}$:

$$\min_{\substack{\boldsymbol{\omega}\in\mathbb{R}^q_+ \\ \|\boldsymbol{\omega}\|_1=1}} \min_{\substack{\mathbf{M}\in\mathbb{R}^{d\times q} \\ \mathbf{M}\boldsymbol{\omega}=\mathbf{0}}} \left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M}\right)\right\|^2_{\boldsymbol{\Sigma}^{-1}}. \tag{6.2}$$

The inner minimization of this optimization problem can be solved in closed form, leveraging on a weighted low-rank approximation of the matrix $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ [Manton et al., 2003].

Theorem 6.2.1. *If* $\boldsymbol{\Sigma}$ *is positive definite, the optimization problem* (6.2) *can be restated as:*

$$\min_{\substack{\boldsymbol{\omega}\in\mathbb{R}^q_+ \\ \|\boldsymbol{\omega}\|_1=1}} \left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right\|^2_{\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\boldsymbol{\Sigma}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}}, \tag{6.3}$$

where $\otimes$ *denotes the Kronecker product and* $\mathbf{I}_d$ *is the identity matrix of order d. Furthermore, the approximating Jacobian* $\mathbf{M}(\boldsymbol{\omega})$ *is given by:*

$$\begin{aligned}\mathrm{vec}\left(\mathbf{M}(\boldsymbol{\omega})\right) = &\Big\{\mathbf{I}_{dq} - \boldsymbol{\Sigma}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\boldsymbol{\Sigma}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1} \\ &\times (\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\Big\}\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right).\end{aligned}$$

Proof. The proof is analogous to that of Theorem 1 of [Manton et al., 2003]. Let $\boldsymbol{\omega} \in \mathbb{R}^q$ be a fixed vector, we solve the following optimization problem:

$$\min_{\substack{\mathbf{M} \in \mathbb{R}^{d \times q} \\ \mathbf{M}\boldsymbol{\omega} = \mathbf{0}}} \left\| \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M} \right) \right\|_{\boldsymbol{\Sigma}^{-1}}^2 .$$

Now we employ Lagrange multipliers, leading to the Lagrangian function:

$$\begin{aligned} \mathcal{L}(\mathbf{M}, \boldsymbol{\lambda}) &= \left\| \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M} \right) \right\|_{\boldsymbol{\Sigma}^{-1}}^2 + \boldsymbol{\lambda}^T \mathbf{M} \boldsymbol{\omega} \\ &= \left\| \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) - \operatorname{vec}(\mathbf{M}) \right\|_{\boldsymbol{\Sigma}^{-1}}^2 + \boldsymbol{\lambda}^T (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}(\mathbf{M}), \end{aligned}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ is the Lagrange multiplier and we exploited the properties of the vectorization operator and the Kronecker product to derive the second equation.

Since the Lagrangian function $\mathcal{L}$ is convex with respect to $\operatorname{vec}(\mathbf{M})$, to solve the optimization problem is sufficient to make the gradient vanish:

$$\frac{\partial \mathcal{L}}{\partial \operatorname{vec}(\mathbf{M})} = 2\boldsymbol{\Sigma}^{-1} \left(\operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) - \operatorname{vec}(\mathbf{M}) \right) + (\boldsymbol{\omega} \otimes \mathbf{I}_d) \boldsymbol{\lambda} = \mathbf{0} \tag{6.2.1.1}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}(\mathbf{M}) = \mathbf{0}. \tag{6.2.1.2}$$

From Equation (6.2.1.1), we obtain an expression for $\operatorname{vec}(\mathbf{M})$ as a function of $\boldsymbol{\lambda}$:

$$\operatorname{vec}(\mathbf{M}) = \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) + \frac{1}{2} \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \boldsymbol{\lambda}.$$

By substituting into Equation (6.2.1.2), we get the value of $\boldsymbol{\lambda}$:

$$\begin{aligned} &(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) + \frac{1}{2} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \boldsymbol{\lambda} = \mathbf{0} \\ &\implies \boldsymbol{\lambda} = -2 \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right). \end{aligned}$$

Finally, we get the expression for $\operatorname{vec}(\mathbf{M})$:

$$\begin{aligned} \operatorname{vec}(\mathbf{M}) &= \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) - \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) \\ &= \left\{ \mathbf{I}_{dq} - \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\} \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right). \end{aligned}$$

We can now substitute the value of $\operatorname{vec}(\mathbf{M})$ into the loss function:

$$\begin{aligned} \left\| \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M} \right) \right\|_{\boldsymbol{\Sigma}^{-1}}^2 &= \left\| \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) \right\|_{\boldsymbol{\Sigma}^{-1}}^2 \\ &= \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right)^T (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-1} \\ &\quad \times \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) \\ &= \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right)^T (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} \\ &\quad \times (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \operatorname{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right) \\ &= \left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega} \right)^T \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} \left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega} \right) \\ &= \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega} \right\|_{\left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1}}^2, \end{aligned}$$

where we employed the properties of the vectorization operator and the Kronecker product in the last but one line and the definition of norm in the last line. □

Unfortunately, the objective function (6.2) is non-convex for a generic choice of Σ. However, for specific choices of Σ we are able to prove the convexity and recover the objective function optimized by GIRL.

Corollary 6.2.1. *Let* $\mathbf{Q} \in \mathbb{R}^{d\times d}$ *be a positive definite matrix and let* $\mathbb{1}_q$ *denote the* q*-dimensional vector of all ones. If* $\mathbf{\Sigma} = \mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}$*, then objective function* (6.3) *is convex. Furthermore, if* $\mathbf{Q} = \mathbf{I}_d$*, then the objective function* (6.3) *is equivalent to* (3.2) *with* $p = 2$*.*

Proof. When $\mathbf{\Sigma} = \mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}$, we can provide the following derivation exploiting the properties of the Kronecker product and recalling that $\boldsymbol{\omega}^T\mathbb{1}_q = 1$ because of the enforced constraints:

$$\begin{aligned}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{\Sigma}(\boldsymbol{\omega}\otimes\mathbf{I}_d) &= (\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\left(\mathbb{1}_q\mathbb{1}_q^T\otimes\mathbf{Q}\right)(\boldsymbol{\omega}\otimes\mathbf{I}_d)\\ &= (\boldsymbol{\omega}^T\mathbb{1}_q\mathbb{1}_q^T\otimes\mathbf{I}_d\mathbf{Q})(\boldsymbol{\omega}\otimes\mathbf{I}_d)\\ &= (\mathbb{1}_q^T\otimes\mathbf{Q})(\boldsymbol{\omega}\otimes\mathbf{I}_d)\\ &= \mathbb{1}_q^T\boldsymbol{\omega}\otimes\mathbf{Q}\mathbf{I}_d\\ &= \mathbf{Q}.\end{aligned}$$

Therefore, the objective function becomes $\left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right\|^2_{\mathbf{Q}^{-1}}$ which is clearly convex in $\boldsymbol{\omega}$, as $\mathbf{Q}$ is positive definite. Moreover, if we take $\mathbf{Q} = \mathbf{I}_d$, then we have $\left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right\|^2_{\mathbf{I}_d} = \left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right\|^2_2$, that is the objective function (3.2) optimized by GIRL when $p = 2$. □

Since in real situations we do not have access to the true covariance matrix $\mathbf{\Sigma}$, we can approximate it with the empirical covariance $\widehat{\mathbf{\Sigma}}$.[1]

In the following, we are going to denote with $p(D|\boldsymbol{\omega}) = \mathcal{L}_{\mathbf{\Sigma}}(\mathrm{M}(\boldsymbol{\omega})|D)$ the value of the objective as a function of the weight vector attained by the optimal mean matrix $\mathrm{M}(\boldsymbol{\omega})$. As intuition suggests, this quantity can be interpreted as the likelihood of the dataset D, given a weight vector $\boldsymbol{\omega}$. We will employ it in the clustering procedure (Section 6.4).

6.3 Theoretical analysis of Σ-GIRL

In this section, we provide two theoretical analyses of the proposed method Σ-GIRL. We start by analyzing how to approximate the covariance matrix $\mathbf{\Sigma}$ in order to make the minimization problem (6.3) convex, as we defined in Corollary 6.2.1. Then, we analyze what is the “error" that we introduced with this approximation. As the second analysis, we provide a finite-sample

[1]The empirical covariance matrix $\widehat{\mathbf{\Sigma}}$ might be singular when $dq \gg n$. In such cases, we resort to standard corrections to enforce well-conditioning [Ledoit and Wolf, 2004].

analysis on the correctness of the recovered weights, assuming that $\mathbf{\Sigma}$ is the true covariance matrix of the distribution that generated the estimated Jacobian $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$.

6.3.1 Approximation of Σ as in Corollary 6.2.1

We can approximate a generic matrix $\mathbf{\Sigma}$ as a matrix of the form $\mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}$, as in Corollary 6.2.1, getting a closed form for $\mathbf{Q}$. To do this approximation, we want to minimize the Frobenius-norm distance between $\mathbf{\Sigma}$ and $\mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}$:

$$\min_{\substack{\mathbf{Q}\in\mathbb{R}^{d\times d}\\ \mathbf{Q}\succ\mathbf{0}_d}} \frac{1}{2}\left\|\mathbf{\Sigma} - \mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}\right\|_F^2, \tag{6.4}$$

where we required that $\mathbf{Q} \succ \mathbf{0}_d$, i.e. that $\mathbf{Q}$ is positive definite whenever $\mathbf{\Sigma}$ is.

We solve the problem ignoring the constraint and then we prove that the resulting matrix is indeed positive definite whenever $\mathbf{\Sigma}$ is.

Lemma 6.3.1. *Let $\mathbf{\Sigma} = n\,\mathbb{C}\mathrm{ov}[\mathrm{vec}(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}))]$, the problem* (6.4) *admits a unique solution that is:*

$$\begin{aligned}\mathbf{Q} &= n\,\mathbb{C}\mathrm{ov}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\mathbb{1}_q\right]\\ &= \frac{1}{q^2}\sum_{i=1}^{q}\sum_{j=1}^{q}\mathbf{\Sigma}_{iq:(i+1)q,jq:(j+1)q} &(6.5)\\ &= \frac{1}{q^2}\left(\mathbb{1}_q\otimes\mathbf{I}_d\right)^T\mathbf{\Sigma}\left(\mathbb{1}_q\otimes\mathbf{I}_d\right), &(6.6)\end{aligned}$$

where we denoted with $\mathbf{\Sigma}_{i:i',j:j'}$ the submatrix obtained by taking the rows between i and i' and the columns between j and j'. Furthermore, $\mathbf{Q}$ is positive definite whenever $\mathbf{\Sigma}$ is.

Proof. Recall that the Kroneker product $\mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}$ constructs a matrix in which $\mathbf{Q}$ is repeated $q \times q$ times, arranged in a square matrix. Thus, it follows that we can rewrite the norm as:

$$\frac{1}{2}\left\|\mathbf{\Sigma} - \mathbb{1}_q\mathbb{1}_q^T \otimes \mathbf{Q}\right\|_F^2 = \frac{1}{2}\sum_{i=1}^{q}\sum_{j=1}^{q}\left\|\mathbf{Q} - \mathbf{\Sigma}_{iq:(i+1)q,jq:(j+1)q}\right\|_F^2.$$

This is a least-squares problem, that can be solved in closed form, yielding to a matrix $\mathbf{Q}$ which is the mean of the blocks $\mathbf{\Sigma}_{iq:(i+1)q,jq:(j+1)q}$.

To get the expression (6.5) we observe that each block can be rewritten as:

$$\mathbf{\Sigma}_{iq:(i+1)q,jq:(j+1)q} = \mathbb{C}\mathrm{ov}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_j(\boldsymbol{\theta})\right].$$

Given the linearity of the covariance we have:

$$\begin{aligned}\frac{1}{q^2}\sum_{i=1}^{q}\sum_{j=1}^{q}\mathbf{\Sigma}_{iq:(i+1)q,jq:(j+1)q} &= \frac{1}{q^2}\sum_{i=1}^{q}\sum_{j=1}^{q}\mathbb{C}\mathrm{ov}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i(\boldsymbol{\theta}),\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_j(\boldsymbol{\theta})\right]\\ &= \frac{1}{q^2}\,\mathbb{C}\mathrm{ov}\left[\sum_{i=1}^{q}\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i(\boldsymbol{\theta}),\sum_{j=1}^{q}\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_j(\boldsymbol{\theta})\right]\\ &= \frac{1}{q^2}\,\mathbb{C}\mathrm{ov}\left[\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i(\boldsymbol{\theta})\mathbb{1}_q,\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_j(\boldsymbol{\theta})\mathbb{1}_q\right].\end{aligned}$$

Then the equality 6.6 follows from the properties of the Kroneker product.

We need now to prove that matrix $\mathbf{Q}$ is positive definite whenever $\mathbf{\Sigma}$ is. $\mathbf{Q}$ is positive definite if and only if:

$$\inf_{\mathbf{x}\in\mathbb{R}^d:\mathbf{x}\neq\mathbf{0}}\mathbf{x}^T\mathbf{Q}\mathbf{x} > 0.$$

Let us now consider the following derivation:

$$\begin{aligned}\inf_{\mathbf{x}\in\mathbb{R}^d:\mathbf{x}\neq\mathbf{0}}\mathbf{x}^T\mathbf{Q}\mathbf{x} &= \frac{1}{q^2}\inf_{\mathbf{x}\in\mathbb{R}^d:\mathbf{x}\neq\mathbf{0}}\mathbf{x}^T\left(\mathbb{1}_q\otimes\mathbf{I}_d\right)^T\mathbf{\Sigma}\left(\mathbb{1}_q\otimes\mathbf{I}_d\right)\mathbf{x}\\ &\geq \frac{1}{q^2}\inf_{\mathbf{x}\in\mathbb{R}^{dq}:\mathbf{x}\neq\mathbf{0}}\mathbf{x}^T\mathbf{\Sigma}\mathbf{x} > 0.\end{aligned}$$

having observed that $\left(\mathbb{1}_q\otimes\mathbf{I}_d\right)\mathbf{x}$ is never null unless $\mathbf{x}$ is null. $\square$

We can notice that this approximation is equivalent to take a specific choice for the weights $\boldsymbol{\omega} = \frac{1}{q}\mathbb{1}_q$.

We now upper bound the gap between the objective function value attained by the optimum when we consider either matrix $\mathbf{\Sigma}$ or its approximation of the form $\mathbb{1}_q\mathbb{1}_q^T\otimes\mathbf{Q}$. We call this value gap.

First of all, let us denote with $l_{\mathbf{A}}(\boldsymbol{\omega})$ the objective function in problem (6.3), when using $\mathbf{A}$ as covariance model. Let $\mathbf{A}$ and $\mathbf{B}$ be two covariance matrices and let $\boldsymbol{\omega}_{\mathbf{A}}$ and $\boldsymbol{\omega}_{\mathbf{B}}$ be any of the optimal weight for the corresponding covariances. Supposing that $\mathbf{A}$ is the true covariance, we want to bound $0 \leq l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{A}}) - l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{B}})$. Using a standard argument from empirical risk minimization:

$$\begin{aligned}l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{A}}) - l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{B}}) &= l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{A}}) - l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{B}}) \pm l_{\mathbf{B}}(\boldsymbol{\omega}_{\mathbf{B}})\\ &\geq l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{A}}) - l_{\mathbf{B}}(\boldsymbol{\omega}_{\mathbf{A}}) + l_{\mathbf{B}}(\boldsymbol{\omega}_{\mathbf{B}}) - l_{\mathbf{A}}(\boldsymbol{\omega}_{\mathbf{B}})\\ &\geq -2\sup_{\boldsymbol{\omega}}\left|l_{\mathbf{A}}(\boldsymbol{\omega}) - l_{\mathbf{B}}(\boldsymbol{\omega})\right|,\end{aligned}$$

where we exploited the fact that $l_{\mathbf{B}}(\boldsymbol{\omega}_{\mathbf{B}}) \leq l_{\mathbf{B}}(\boldsymbol{\omega}_{\mathbf{A}})$. Thus, it sufficies to prove an upper bound on $\left|l_{\mathbf{A}}(\boldsymbol{\omega}) - l_{\mathbf{B}}(\boldsymbol{\omega})\right|$ that is uniform over $\boldsymbol{\omega}$.

Theorem 6.3.1. *Let $\mathbf{\Sigma}$ be the true covariance matrix and $\mathbb{1}_q\mathbb{1}_q^T\otimes\mathbf{Q}$ be its approximation. Then, it holds that:*

$$\mathrm{gap} \leq \frac{2dq}{s_{\min}(\mathbf{\Sigma})^2}\left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right\|_F^2\left\|\mathbf{\Sigma} - \mathbb{1}_q\mathbb{1}_q^T\otimes\mathbf{Q}\right\|_F, \qquad (6.7)$$

where $\sigma_{\min}$ is the minimum singular values of true covariance matrix $\mathbf{\Sigma}$.

Proof. We use Lemma B.0.2 with $\mathbf{A} = \Sigma$ and $\mathbf{B} = \mathbb{1}_q \mathbb{1}_q^T \otimes \mathbf{Q}$:

$$\begin{aligned}\left| l_\Sigma(\boldsymbol{\omega}) - l_{\mathbb{1}_q \mathbb{1}_q^T \otimes \mathbf{Q}}(\boldsymbol{\omega}) \right| \le & \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \mathbb{1}_q \mathbb{1}_q^T \otimes \mathbf{Q} \, (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & \times \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \Sigma \, (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & \times \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \right\|_F^2 \left\| \left(\mathbb{1}_q \mathbb{1}_q^T \otimes \mathbf{Q} \right) - \Sigma \right\|_F .\end{aligned}$$

Let us now consider the following identity:

$$\mathbf{I}_d = \frac{1}{q} \left(\mathbb{1}_q \otimes \mathbf{I}_d \right)^T \left(\mathbb{1}_q \otimes \mathbf{I}_d \right). \tag{6.8}$$

For the norm involving $\mathbf{B}$ we employ Lemma B.0.1 and for the other we directly derive:

$$\begin{aligned}& \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) \left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \left(\mathbb{1}_q \mathbb{1}_q^T \otimes \mathbf{Q} \right) (\boldsymbol{\omega} \otimes \mathbf{I}_d) \right]^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & = \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) \, \mathbf{Q}^{-1} (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & = \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) \, \mathbf{I}_d \mathbf{Q}^{-1} \mathbf{I}_d \, (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & = \frac{1}{q^2} \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) (\mathbb{1}_q \otimes \mathbf{I}_d)^T (\mathbb{1}_q \otimes \mathbf{I}_d) \, \mathbf{Q}^{-1} (\mathbb{1}_q \otimes \mathbf{I}_d)^T (\mathbb{1}_q \otimes \mathbf{I}_d) (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & = \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) (\mathbb{1}_q \otimes \mathbf{I}_d)^T (\mathbb{1}_q \otimes \mathbf{I}_d) \left[(\mathbb{1}_q \otimes \mathbf{I}_d)^T \mathbf{\Sigma} \, (\mathbb{1}_q \otimes \mathbf{I}_d) \right]^{-1} (\mathbb{1}_q \otimes \mathbf{I}_d)^T (\mathbb{1}_q \otimes \mathbf{I}_d) (\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \right\|_F \\ & \le \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) (\mathbb{1}_q \otimes \mathbf{I}_d)^T \right\|_2^2 \left\| (\mathbb{1}_q \otimes \mathbf{I}_d) \left[(\mathbb{1}_q \otimes \mathbf{I}_d)^T \mathbf{\Sigma} \, (\mathbb{1}_q \otimes \mathbf{I}_d) \right]^{-1} (\mathbb{1}_q \otimes \mathbf{I}_d)^T \right\|_F \\ & \le \left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) (\mathbb{1}_q \otimes \mathbf{I}_d)^T \right\|_2^2 \frac{\sqrt{d}}{\sigma_{\min}(\mathbf{\Sigma})},\end{aligned}$$

where we exploited Lemma B.0.2. To bound the remaining term we have:

$$\begin{aligned}\left\| (\boldsymbol{\omega} \otimes \mathbf{I}_d) (\mathbb{1}_q \otimes \mathbf{I}_d)^T \right\|_2 & \le \| \boldsymbol{\omega} \otimes \mathbf{I}_d \|_2 \, \| \mathbb{1}_q \otimes \mathbf{I}_d \|_2 \\ & \le \| \boldsymbol{\omega} \|_2 \, \| \mathbf{I}_d \|_2 \, \| \mathbb{1}_q \|_2 \, \| \mathbf{I}_d \|_2 \\ & \le \| \boldsymbol{\omega} \|_1 \sqrt{q} \le \sqrt{q}.\end{aligned}$$

□

6.3.2 Correctness of the recovered weights

In this section, we provide a finite-sample analysis of Σ-GIRL, under the assumption that $\mathbf{\Sigma}$ is the true covariance matrix of the distribution having generated the estimated Jacobian $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})$. We consider the case in which the weight vector $\boldsymbol{\omega}^E$ is unique under the simplex constraint (Equation (5.1)), to avoid multiple solutions. Similar to what was done in [Pirotta and Restelli, 2016], we can evaluate the norm of the difference between the expert's weights $\boldsymbol{\omega}^E$ and the recovered ones $\widehat{\boldsymbol{\omega}}$. The following result provides a finite-sample bound for this quantity.

Theorem 6.3.2. *Let $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ be an unbiased estimate of the Jacobian $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ obtained with the trajectories $D = \{\tau_1, \dots, \tau_n\}$. Let $\frac{1}{n}\boldsymbol{\Sigma} = \mathbb{C}\mathrm{ov}[\mathrm{vec}(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}))]$ be the true covariance matrix of the estimated Jacobian. Let $\widehat{\boldsymbol{\omega}}$ be the weight vector recovered by Σ-GIRL run with covariance matrix $\boldsymbol{\Sigma}$ and $\boldsymbol{\omega}^E$ be the expert's weight vector. If $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ and $\mathbf{M}(\widehat{\boldsymbol{\omega}})$ have rank $q-1$ and $s_{q-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})) = s > 0$, where $s_{q-1}(\cdot)$ denotes the $(q-1)$-th singular value, then it holds that:*

$$\mathbb{E}\left[\left\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^E\right\|_2\right] \le \sqrt{\frac{16dq\,\|\boldsymbol{\Sigma}\|_2}{s^2 n}},$$

where the expectation is taken w.r.t. the randomness of the trajectories in D used to compute $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$.

Proof. From the proof of Theorem 13.2 of [Pirotta, 2016], we know that:

$$\left\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^E\right\|_2 \le \sqrt{2(1-\cos\alpha)}, \tag{6.9}$$

where α is the angle between the two vectors $\widehat{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}^E$. We now provide a bound for $\cos\alpha$. Since $\widehat{\boldsymbol{\omega}}$ and $\boldsymbol{\omega}^E$ belong to the orthogonal complements of the column spaces generated by $\mathbf{M}(\widehat{\boldsymbol{\omega}})$ and $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ respectively. From Lemma B.0.4, we have that:

$$\cos\alpha \ge 1 - \frac{2}{s_{q-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}))^2} \min_{\boldsymbol{\Pi}\in\mathrm{Perm}_q} \|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M}(\widehat{\boldsymbol{\omega}})\boldsymbol{\Pi}\|_F^2 .$$

We now consider the following sequence of derivations:

$$\begin{aligned}
\min_{\boldsymbol{\Pi}\in\mathrm{Perm}_q} \|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \mathbf{M}(\widehat{\boldsymbol{\omega}})\boldsymbol{\Pi}\|_F^2 &\le \|\mathbf{M}(\widehat{\boldsymbol{\omega}}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\|_F^2 && (6.10)\\
&\le \|\mathrm{vec}\left(\mathbf{M}(\boldsymbol{\omega}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\|_2^2 && (6.11)\\
&\le 4\,\|\boldsymbol{\Sigma}\|_2 \left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\right\|_{\boldsymbol{\Sigma}^{-1}}^2, && (6.12)
\end{aligned}$$

where line (6.10) is obtained from selecting $\boldsymbol{\Pi} = \mathbf{I}_q$. Line (6.11) derives from observing that the Frobenius norm of a matrix equals the L^2-norm of the corresponding vectorization. Finally, line (6.12) follows from Lemma B.0.5. Putting this latter result into Equation (6.9), we have:

$$\left\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^E\right\|_2 \le \sqrt{\frac{16\,\|\boldsymbol{\Sigma}\|_2}{s_{q-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}))^2}\left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\right\|_{\boldsymbol{\Sigma}^{-1}}^2}.$$

Now we compute the expectation of the norm of the difference:

$$\begin{aligned}
\mathbb{E}\left[\left\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^E\right\|_2\right] &\le \mathbb{E}\left[\sqrt{\frac{16\,\|\boldsymbol{\Sigma}\|_2}{s_{q-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}))^2}\left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\right\|_{\boldsymbol{\Sigma}^{-1}}^2}\right]\\
&\le \sqrt{\frac{16\,\|\boldsymbol{\Sigma}\|_2}{s_{q-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}))^2}\mathbb{E}\left[\left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\right\|_{\boldsymbol{\Sigma}^{-1}}^2\right]},
\end{aligned}$$

where the last passage follows from Jensen's inequality. To conclude, we compute the expectation inside the square root by observing that it is the expectation of a zero-mean random vector under the norm induced by its true covariance matrix. Thus, by renaming $\mathbf{x} = \mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right) \in \mathbb{R}^{dq}$ and recalling that $\mathbb{E}[\mathbf{x}\mathbf{x}^T] = \mathbb{C}\mathrm{ov}[\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})] = \frac{\boldsymbol{\Sigma}}{n}$ we have:

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{x}\|^2_{\boldsymbol{\Sigma}^{-1}}\right] &= \mathbb{E}\left[\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x}\right] = \mathbb{E}\left[\mathrm{tr}(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x})\right] \\ &= \mathbb{E}\left[\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{x}\mathbf{x}^T)\right] = \mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right]\right) = \mathrm{tr}\left(\boldsymbol{\Sigma}^{-1}\frac{\boldsymbol{\Sigma}}{n}\right) = \frac{dq}{n}.\end{aligned}$$

□

Our result extends Theorem 13.2 of [Pirotta, 2016] in a few aspects. The result of [Pirotta, 2016] is clearly applicable to Σ-GIRL, since it assumes that the estimated Jacobian $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ has already rank $q-1$. In such case, GIRL and Σ-GIRL behave in the same way as $\mathbf{M}(\widehat{\boldsymbol{\omega}}) = \widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$. However, Theorem 6.3.2 is more general and applies for Σ-GIRL even for a full-rank estimated Jacobian $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$. Furthermore, although Σ-GIRL is presented considering a Gaussian likelihood model, Theorem 6.3.2 makes no assumption on the distribution of the Jacobian, but just requires that $\boldsymbol{\Sigma}$ is the true covariance matrix.

6.4 Multiple-Intention Σ-GIRL

In this section, we consider the problem of IRL about Multiple Intention (MI-IRL), using Σ-GIRL as a building block for solving the single-intention IRL problem. As said before, a naïve solution would be to solve m independent IRL problems, one for each expert agent E_i. However, since we typically have $k \ll m$ (i.e. fewer intentions than experts), this solution would be highly sub-optimal, especially in situations where few demonstrations are available per agent. In such cases, it is much wiser to cluster the trajectories during the IRL step, also from a practical point of view. To this end, we adopt an *expectation-maximization* (EM) approach [Dempster et al., 1977] to find the parameters $\boldsymbol{\omega}_j$, together with the agent-cluster assignments that maximize the overall likelihood. We introduce the hidden random variable $Y_i \in \{1, \dots, k\}$ that, for each expert E_i with $i \in \{1, \dots, m\}$, indicates to which cluster E_i is assigned. In other words, $Y_i = j$ means that agent E_i is optimizing the reward $R_{\boldsymbol{\omega}_j}$. For these random variables, we assume a prior distribution $\alpha_j = p(Y_i = j)$, independent of i, where $\alpha_j \geq 0$ and $\sum_{j=1}^k \alpha_j = 1$. We will denote with $\mathbf{Y} = (Y_1, \dots Y_m)$ the concatenation of all the Y_is. The collection of parameters we are going to optimize on is given by the concatenation of the weight vectors $\boldsymbol{\omega}_j$ and the prior probabilities α_j, i.e. $\Omega = (\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_k, \alpha_1, \dots, \alpha_k)$.

The crucial observation, for applying EM, is that we can compute the likelihood of a dataset D_i, once we know the cluster assignment of agent E_i, i.e. Y_i:

$$p(D_i|Y_i = j; \Omega) = p(D_i|\boldsymbol{\omega}_j), \tag{6.13}$$

where the latter is defined in Section 6.2 (Equation (6.1)). Exploiting the independence of the datasets $\mathbf{D} = (D_1, \dots, D_m)$ and recalling that $p(D_i, Y_i|\Omega) = p(D_i|Y_i; \Omega)p(Y_i|\Omega)$, we can define the likelihood $\mathcal{L}(\Omega|\mathbf{D}, \mathbf{Y}) = p(\mathbf{D}, \mathbf{Y}|\Omega)$ of all the data as:

$$\mathcal{L}(\Omega|\mathbf{D}, \mathbf{Y}) = \prod_{i=1}^{m} p(D_i, Y_i|\Omega) = \prod_{i=1}^{m} \alpha_{Y_i} p(D_i|\boldsymbol{\omega}_{Y_i}).$$

According to [Bilmes et al., 1998], in the expectation step (E-step), we compute the probability of the assignment Y_i, conditioned by the data D_i, in terms of the old parameters Ω^{old}, i.e. $z_{ij} = p(Y_i = j|D_i; \Omega^{\text{old}})$. We derive z_{ij} using Bayes theorem:

$$\begin{aligned} z_{ij} &= \frac{p(D_i|Y_i = j; \Omega^{\text{old}})p(Y_i = j|\Omega^{\text{old}})}{p(D_i|\Omega^{\text{old}})} \\ &= \frac{\alpha_j^{\text{old}} p(D_i|\boldsymbol{\omega}_j^{\text{old}})}{\sum_{h=1}^{k} \alpha_h^{\text{old}} p(D_i|\boldsymbol{\omega}_h^{\text{old}})}. \end{aligned}$$

In the maximization step (M-step), instead, we look for the new parameters Ω that maximize the expectation of the log-likelihood $\log \mathcal{L}(\Omega|\mathbf{D}, \mathbf{Y})$ under the previously found distribution over the assignments Y_i:

$$\begin{aligned} Q(\Omega, \Omega^{\text{old}}) &= \mathop{\mathbb{E}}_{\mathbf{Y} \sim p(\cdot|\mathbf{X}; \Omega^{\text{old}})} [\log \mathcal{L}(\Omega|\mathbf{D}, \mathbf{Y})] \\ &= \sum_{\mathbf{y}} \log(\mathcal{L}(\Omega|\mathbf{D}, \mathbf{y}))p(\mathbf{y}|\mathbf{D}, \Omega^{\text{old}}) \\ &= \sum_{\mathbf{y}} \sum_{i=1}^{m} \log(\alpha_{y_i} p_{y_i}(D_i|\boldsymbol{\omega}_{y_i})) \prod_{i'=1}^{m} p(y_{i'}|D_{i'}, \Omega^{\text{old}}) \\ &= \sum_{y_1=1}^{k} \cdots \sum_{y_m=1}^{k} \sum_{i=1}^{m} \log(\alpha_{y_i} p_{y_i}(D_i|\boldsymbol{\omega}_{y_i})) \prod_{i'=1}^{m} p(y_{i'}|D_{i'}, \Omega^{\text{old}}) = \end{aligned}$$

Algorithm 9 Multiple-Intention Σ-GIRL

```
input: datasets $\mathbf{D} = (D_1, \dots, D_m)$, number of clusters $k$, number of iterations $N_{\text{ite}}$
output: optimal parameters $\Omega = (\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_k, \alpha_1, \dots, \alpha_k)$
  Initialize $\Omega^0$ randomly
  for $t = 1, \dots, N_{\text{ite}}$ do
    E-step: Compute $z_{ij} = \frac{\alpha_j^{t-1} p(D_i|\boldsymbol{\omega}_j^{t-1})}{\sum_{h=1}^{k} \alpha_h^{t-1} p(D_i|\boldsymbol{\omega}_h^{t-1})}$
    M-step: Optimize $\Omega^t \in \arg\max_{\boldsymbol{\omega}_j, \alpha_j} Q(\Omega, \Omega^{t-1})$
                $= \sum_{j=1}^{k} \sum_{i=1}^{m} z_{ij} (\log \alpha_j + \log p(D_i|\boldsymbol{\omega}_j))$
  end for
  return $\Omega^{N_{\text{ite}}}$
```

$$
\begin{aligned}
&= \sum_{j=1}^{k} \sum_{i=1}^{m} \log(\alpha_j p(D_i|\boldsymbol{\omega}_j)) \sum_{y_1=1}^{k} \cdots \sum_{y_m=1}^{k} \mathbb{1}_{\{j=y_i\}} \prod_{i'=1}^{m} p(y_{i'}|D_{i'}, \Omega) \\
&= \sum_{j=1}^{k} \sum_{i=1}^{m} \log(\alpha_j p(D_i|\boldsymbol{\omega}_j)) z_{ij} \\
&= \sum_{j=1}^{k} \sum_{i=1}^{m} z_{ij} (\log \alpha_j + \log p(D_i|\boldsymbol{\omega}_j)) \\
&= \sum_{j=1}^{k} \sum_{i=1}^{m} z_{ij} \log \alpha_j + \sum_{j=1}^{k} \sum_{i=1}^{m} z_{ij} \log p(D_i|\boldsymbol{\omega}_j),
\end{aligned}
$$

where we obtain this expression following a derivation analogous to the one presented in [Bilmes et al., 1998] and denoting with $\mathbf{y} = (y_1, \dots, y_m)$ the realization of the random vector $\mathbf{Y}$.

Thus, the EM algorithm keeps alternating the E-step by computing the probabilities z_{ij} and the M-step by computing the new parametrization Ω and α optimizing $Q(\Omega, \Omega^{\text{old}})$. Algorithm 9 reports an overview of Multiple-Intention Σ-GIRL. It is worth noting that for the computation of the new Ω it is required the solution of k IRL problems. Indeed, for a fixed $j \in \{1, \dots, k\}$, denoting with $\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}_i(\boldsymbol{\theta})$ the estimated Jacobian of agent E_i, the objective function to be optimized is given by:

$$
\min_{\substack{\boldsymbol{\omega}_j \in \mathbb{R}^q_+ \\ \|\boldsymbol{\omega}_j\|_1 = 1}} \sum_{i=1}^{m} z_{ij} n_i \left\| \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}_i(\boldsymbol{\theta}) \boldsymbol{\omega}_j \right\|^2_{[(\boldsymbol{\omega}_j \otimes \mathbf{I}_d) \boldsymbol{\Sigma}_i (\boldsymbol{\omega}_j \otimes \mathbf{I}_d)^T]^{-1}}.
$$

6.4.1 Computational Complexity Analysis

In this section we provide the computational complexity analysis of Σ-GIRL and MI-Σ-GIRL.

The computational cost of the Jacobian estimation is linear in the number of policy parameters d, reward parameters q, samples N, and horizon H; while the cost of estimating the full covariance matrix is quadratic in d and q, and linear in N and H. Thus, for a given $\boldsymbol{\omega}$, evaluating the objective in Equation (6.3) has a cost of $\mathcal{O}\left(d^3 + d^2q^2\right)$, where the d^3 term comes from the inversion of matrix $\left[(\boldsymbol{\omega} \otimes \mathbf{I}_d)^T \boldsymbol{\Sigma} \left(\boldsymbol{\omega} \otimes \mathbf{I}_d\right)\right]$.

The cost of an EM step is $O(MkC)$, where M and k are the number of agents and clusters respectively and C is the cost of optimizing the function, which depends on the optimizer.

6.5 Discussion on the related work

As we exposed in Chapter 3, there has been a growing interest in making IRL algorithms scale over real-world continuous domains, where only a few or no environment interactions are allowed [Boularias et al., 2011, Jain et al., 2019]. However, these algorithms have some limitations: [Boularias et al., 2011] requires a dataset collected under an explorative policy; instead, [Jain et al., 2019]repeatedly needs to solve batch RL problems to compute (approximately) optimal Q-functions. On the other hand, [Klein et al., 2012], and [Klein et al., 2013] (see Chapter 6) reduced IRL to a structured classification problem which, similarly to GIRL [Pirotta and Restelli, 2016], can be solved efficiently using only the observed trajectories and without further interaction with the environment. These algorithms require a good estimate of the expert's feature expectations for each action, even those that the expert has not demonstrated. Σ-GIRL mitigates this issue by considering an explicit model of the uncertainty $\boldsymbol{\Sigma}$ of the gradient estimate while importing all the advantages of GIRL.

In its first formulation, the MI-IRL problem was solved via an EM algorithm considering a Boltzmann policy for the expert [Babes et al., 2011]. Then Bayesian non-parametric approaches were proposed in the case k is unknown. These methods, however, require access to the environment (which must have a finite a state-action space) and need a careful tuning of the temperature parameter. Our multiple-intention version of MI-Σ-GIRL, instead, can be employed in continuous environments and just requires the selection of the number of clusters k as a unique hyperparameter.

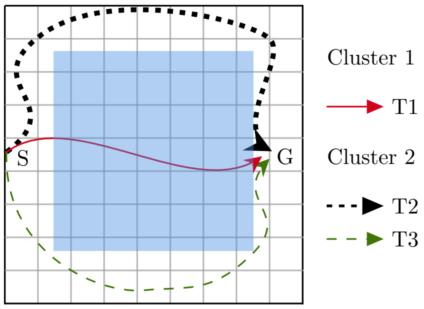


Figure 6.2: *Gridworld with puddles showing the start state (S), the goal state (G), puddles (light blue) and three sample trajectories: T1 (red) from the first cluster, T2 (black) and T3 (green) from the second cluster.*

6.6 Experiments

This section is devoted to the experimental evaluation of Σ-GIRL in both single-intention (Section 6.6.1) and multiple-intention (Section 6.6.2- 6.6.3) settings.[2] For the single intention case, Σ-GIRL is compared with some batch model-free IRL algorithms in two continuous domains: the Linear Quadratic Gaussian regulator [Dorato et al., 2000, LQG] and a Gridworld with puddles domain (Figure 6.2). For the multiple-intention case, we test the quality of the clusters identified by Multiple-Intention Σ-GIRL compared to Maximum-Likelihood IRL [Babes et al., 2011, MLIRL] in the Gridworld with puddles. Finally, we evaluate Σ-GIRL in a real-world case study in which we infer and cluster the intentions of a group of Twitter users.

Optimization of Σ-GIRL objective function The objective function optimized by Σ-GIRL is, in the general case, non-convex. In the experiments, we optimize this function using the implementation of SLSQP (Sequential Least SQuares Programming) from scipy Python package[3]. We used the default parameters and tolerance value $1e-8$. We took the best of 25 in the LQG experiment and 5 in the Gridworld experiment different random initializations.

6.6.1 Single-IRL experiments

We start evaluating the performance of Σ-GIRL compared with state-of-the-art model-free IRL algorithms in the single-intention IRL problem.

[2]The code is available at github.com/sigma-girl-MIIRL.

[3]`https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`

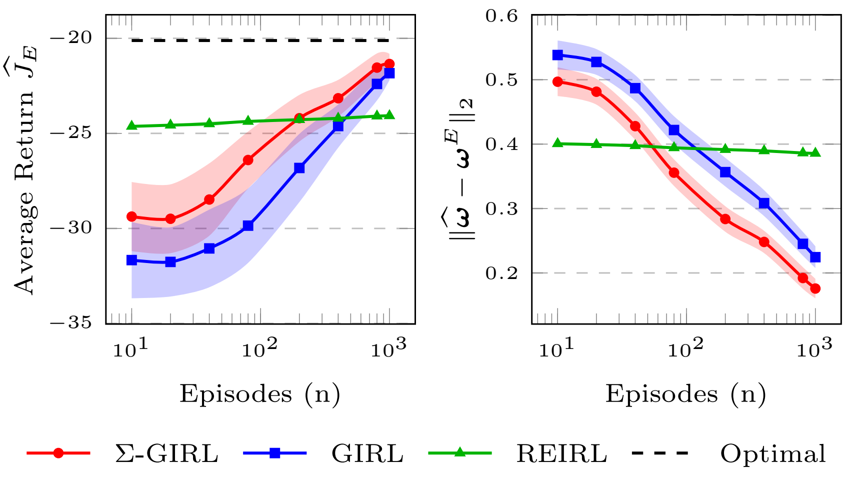


Figure 6.3: *Average return in the original environment $\widehat{J}_E$ of the optimal policy given the recovered weights $\widehat{\boldsymbol{\omega}}$ (left) and distance between the recovered and expert's weights $\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^E\|_2$ (right) in the LQG experiment. 100 runs, 95 % c.i.*

Linear Quadratic Gaussian regulator We consider the two-dimensional LQG environment in which the agent has to reach the origin, limiting the magnitude of the actions. The reward features are the state and actions squared $\boldsymbol{\phi}(\mathbf{s}, \mathbf{a}) = (-s_1^2, -s_2^2, -a_1^2, -a_2^2)^T$ and the weights are $\boldsymbol{\omega}^E = (1, 1, 1, 1)^T$. The expert plays a Gaussian linear policy, in which the control matrix $\mathbf{K}$ is computed in closed form and with fixed diagonal covariance $\mathrm{diag}(0.1, 2)$.[4] We show, for each of the algorithms considered, the performance of the optimal policies with the recovered reward function in the original environment $\widehat{J}_E$ and the distance between the weights found and the expert weights $\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^E\|_2$, as a function of the number of trajectories. In Figure 6.3, we notice that Σ-GIRL outperforms both GIRL and REIRL[5] in both indexes, achieving better performance and weights closer to the original ones. REIRL requires, besides the expert demonstrations, a dataset collected using a second uniform policy. This requirement partially violates the assumption of no interaction with the environment. The full sample covariance matrix was used in this experiment since it resulted well-conditioned.

Gridworld The second experiment aims at evaluating the performance of Σ-GIRL in a continuous Gridworld environment. The agent is initialized in

[4]This asymmetric choice induces a higher variance in the second dimension of the state and action spaces; so that we can easily see the benefits of Σ-GIRL in modeling the gradient uncertainty.

[5]REIRL is implemented following the original paper.

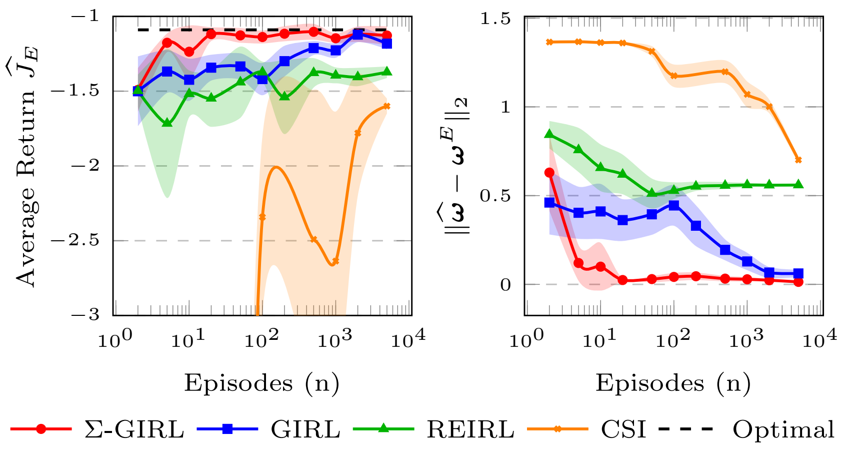


Figure 6.4: *Average return in the original environment $\widehat{J}_E$ of the optimal policy trained with G(PO)MDP with the recovered weights $\widehat{\omega}$ (left) and distance between the recovered and expert's weights $\|\widehat{\omega} - \omega^E\|_2$ (right) in the Gridworld experiment. 20 runs, 95 % c.i.*

a random position and has to reach the goal in the minimum number of steps by playing a bivariate Gaussian policy, linear in a set of 9×9 radial basis functions, that generates the x and y displacement. There is a region in the border of the environment that should be avoided. To make the environment more challenging, the agent is also penalized for performing high magnitude actions. The reward feature space is given by three features: two binary features indicating whether the agent is at the border or in the central region and $-\|\boldsymbol{a}\|_2^2$ to penalize the magnitude of actions. The expert's weights are $\boldsymbol{\omega}^E = (1, 100, 0)^T$. In Figure 6.4, we compare the performance of Σ-GIRL with GIRL, REIRL, and CSI [Klein et al., 2013, Cascade Supervised IRL]. CSI is implemented following the authors' implementation[6]. We report the results using SVM as classifier and SVR as regressor (as in the original implementation, with the same library[7]). We notice that Σ-GIRL is able to recover weights that are almost identical to the expert's ones even with a small (30) number of trajectories. This, as expected, reflects on the performance, in the original environment, of the optimal policy learned using the recovered weights. While GIRL is still able to obtain a good weighting, REIRL and CSI's performance are significantly suboptimal.

[6]The repository is in: https://github.com/edouardklein/RL-and-IRL.
[7]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

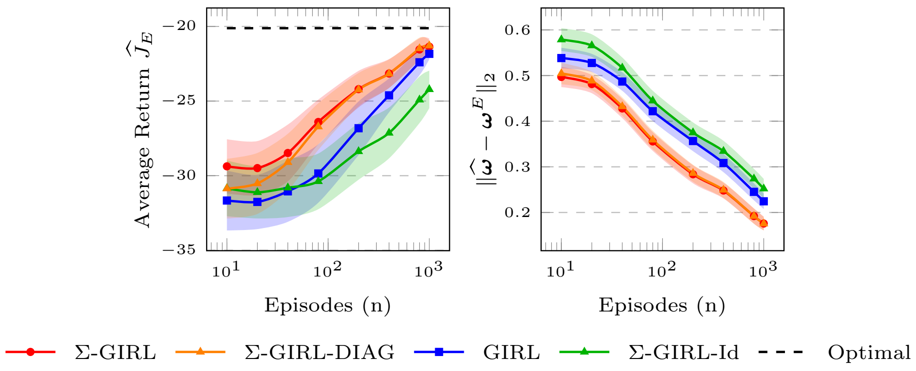


Figure 6.5: *Comparison on the LQG experiment with different choices of covariance model. 100 runs, 95% c.i.*

Σ-GIRL Comparison In this section, we compare different choices of the covariance matrix used in Σ-GIRL in the LQG environment. Apart from the full sample covariance matrix and the matrix of Corollary 6.2.1 (which reduces our algorithm to GIRL), we also consider using a diagonal sample covariance matrix and the identity matrix. The former considers only the uncertainty in each of the Jacobian matrix entries, while the latter does not consider the uncertainty.

Figure 6.5 shows the results in the LQG environment. We can see that using the uncertainty of the gradient estimation clearly achieves better performance. In the environment considered, using the full sample covariance matrix offers only a slight improvement when considering few trajectories compared to the diagonal case. In larger problems, where estimating the full covariance matrix might be prohibitive, using a diagonal covariance model offers improvements with respect to not using the uncertainty at all in the gradient estimation. The use of an identity matrix instead do not produce satisfying results.

6.6.2 Multiple-intentions experiments

In this experiment, we compare the Multiple-Intention Σ-GIRL with MLIRL [Babes et al., 2011], to test the capability of clustering agents which demonstrate multiple intentions. MLIRL is implemented following the original paper and the thesis of the main author [Vroman, 2014]. Unfortunately (as far as we know) the authors do not provide a public repository. We consider a Gridworld-puddles consisting of an initial state, a goal state, and some states are puddles, as in Figure 6.2. The world is characterized by a three-feature reward: one for the goal state, one for the puddles, and one for the other

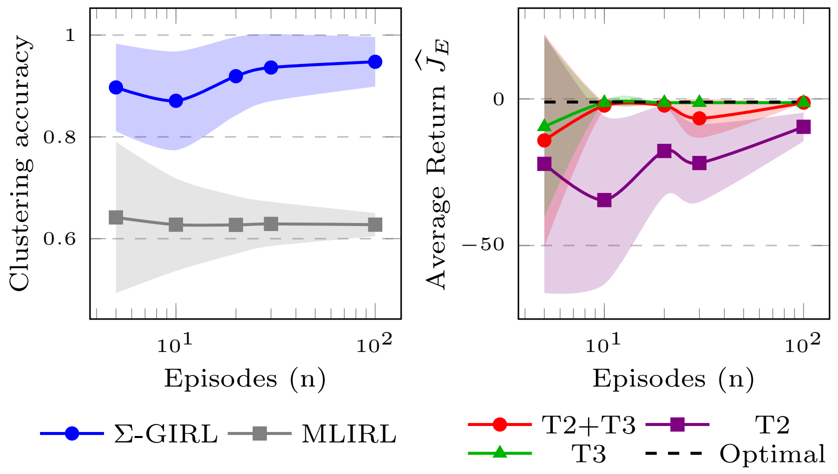


Figure 6.6: *Clustering accuracy of MI-Σ-GIRL and MLIRL (left) and average return in the original environment $\widehat{J}_E$ of the policies trained given the weights recovered separately for T2 and T3 and their cluster (right) in the Gridworld experiment. 20 runs, 98% c.i.*

	Demonstrations per agent			
Running Time	5	10	30	100
Σ-GIRL	1.25s	1.28s	1.21s	1.51s
MLIRL	60.96s	62.20s	69.22s	93.23s

Table 6.1: *Running time of MI-Σ-GIRL and MI-MLIRL with increasing number of trajectories per agent.*

states. In this setting, we consider two clusters of agents which demonstrate different behaviors: the first cluster (T1) has the goal of ignoring the puddles, the second (T2+T3) has the goal of avoiding puddles. For the second cluster, we have three agents that have three different but equivalently optimal policies (Figure 6.2). The first agent (T2) always chooses to go up as the first action, the second (T3) to go down, and the third one randomly performs *up* or *down* as the first action, and then they all follow the border. The first cluster weights are $\boldsymbol{\omega}^{\text{T1}} = (1, 1, 1)^T$ and the second cluster weights are $\boldsymbol{\omega}^{\text{T2,T3}} = (1, 10, 1)^T$. The agents are initialized in the initial state S, and they play bivariate Gaussian policies linear in the state space. We set the number of clusters to 2 and increase the number of trajectories per agent (from 5 to 100). For MLIRL, we set the same hyperparameters as in [Babes et al., 2011], and we appropriately discretize the actions.

Then, we perform an empirical analysis of the algorithm MLIRL [Babes et al., 2011]. The previous results show that the algorithm is unable to

	Reward Features			N. agents
	Popularity	N. retweets	δ_{time}	
Cluster 1	0.56	0.00	0.44	4
Cluster 2	0.16	0.19	0.65	6
Cluster 3	0.78	0.03	0.19	4

Table 6.2: *The reward weights learned by Σ-GIRL: popularity score of a retweet, number of retweets in a window T, and retweet proximity (δ_{time}).*

cluster the agents properly. We perform two experiments. In the first one, we have two agents with two different intentions. In the second one, we have two agents with the same intention but different optimal policies and one agent with another intention. As shown in Figure 6.7, in the first experiment MLIRL succeeds in the clustering task (left). When we add trajectories performed by two agents with the same intentions but different optimal policies, the algorithm decreases its performance (right). This behavior explains the results of Section 6.6 where we have a dataset with three agents sharing the same intention (but different optimal policies) and two agents with the other reward function.

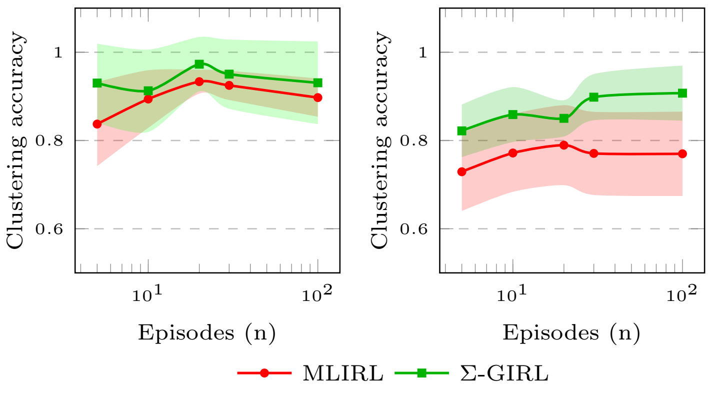


Figure 6.7: *Clustering accuracy in the case of two agents and two clusters (left) and in the case of three agents and two clusters (right). 20 runs 98 % c.i.*

6.6.3 Twitter experiment

In the last experiment, we employ the MI-Σ-GIRL algorithm to cluster and infer Twitter users' intentions.[8] In particular, we turn to the questions:

"Why does a user decide to retweet a post? What is their intention in deciding to post the tweet?"

Data The dataset consists of 14 Twitter accounts (agents) and their followings. The total number of followings is 5745. We collected their tweets from November 2018 to the end of January 2019 using a crawling process. The total number of followings' tweets is 468304. We suppose that each user can see only the tweets of whom she follows. We assume that a user sees a tweet with a probability of 0.01 to simulate the real behavior of a social network's user. After receiving a tweet, the agent must select between two actions: she can re-post it on her page or not re-post it.

Features We represent the state space with three features: the *popularity* of a tweet, the *number of retweets* out of the last $T = 10$ (retweet window) tweets seen by the agent, and the *retweet proximity*. The popularity score is the weighted sum of the number of likes collected by the tweet and the number of retweets:

$$\text{Popularity-score} = \alpha N_{\text{like}} + (1 - \alpha) N_{\text{retweet}}$$

where $\alpha = 0.5$ and then normalized by the average of the popularity score of user's tweets. The retweet proximity is computed as $\delta_{\text{time}} = 0.1(t - t_0) - 1$ where t is the earliest time at which the agent receives a tweet that she decides to retweet after having retweeted at time $t_0 < t$. When an agent performs a retweet, she goes to a next state s' which is composed of the *popularity* of the new tweet, $\delta_{\text{time}} = 0$ (since the last action was a retweet) and the number of retweet mod10. The reward features are the same as the state ones, but the Popularity-score is set to 0 when the agent does not re-tweet the tweet.

Clustering results We perform behavioral cloning on the agents' demonstrations employing a two-layer neural network (8 neurons each). Then, we divide the demonstrations into trajectories of size 10 to have one retweet window in every trajectory. We apply Multiple-Intention MI-Σ-GIRL with $k = 3$ clusters. The results are shown in Table 6.2, while Figure 6.8 reports

[8]It is worth noting that MLIRL [Babes et al., 2011] cannot be applied in this experiment as we are in a fully batch setting and we cannot interact with the environment.

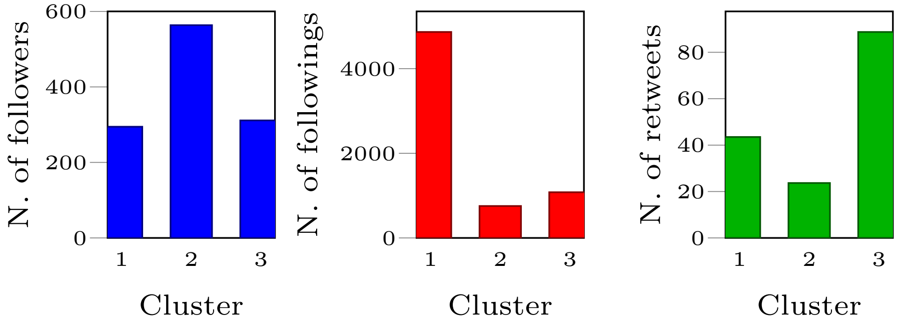


Figure 6.8: *Twitter clustering statistics. Average number of followers (left), followings (center) and retweets (right) for each cluster.*

some statistics on the three clusters found. The results underline that the first cluster is interested in retweeting posts with high popularity at a high frequency. Indeed, this cluster represents a standard Twitter user who follows many users and has fewer followers. The second cluster shows a different behavior: these agents do not want to retweet too often. They have not used the social network much, as they have few retweets and follow a small number of people. The last cluster is the most interesting one: these agents tend to retweet all popular tweets. Upon inspecting them, we discover that they are commercial accounts (a bot, a company, and two HR managers). It is not surprising that they show the intention to post popular tweets, but they are uninterested in following other accounts.

CHAPTER 7

Inverse Reinforcement Learning from a Learner

The standard IRL setting assumes that an observer receives the interactions between another agent, the expert, which already knows how to perform the task, and the environment. However, as we have introduced in Chapter 5, in some cases, the observer can observe the learning process of this other agent, and so it can try to infer the agent's reward function beforehand. This setting, called Inverse Reinforcement Learning from a Learner (IRLfL), was proposed in [Jacq et al., 2019] and posed new opportunities and new challenges. In fact, IRLfL violates the common IRL assumption that the perceived demonstrations come from an expert. In [Jacq et al., 2019] the authors assume that the learner is learning under an entropy-regularized framework, motivated by the assumption that the learner is showing a sequence of constantly improving policies. However, many Reinforcement Learning (RL) algorithms [Deisenroth et al., 2013] do not satisfy this assumption, and also human learning is characterized by mistakes that may lead to a non-monotonic learning process. In this chapter, we propose an algorithm for this relatively new setting, IRLfL, called Learning Observing a Gradient not-Expert Learner (LOGEL), which is not affected by the violation of the constantly improving assumption. Given that many successful RL algorithms are gradient-based [Deisenroth et al., 2013] and there is some evidence that the human learning process is similar to a gradient-based method [Shteingart and Loewenstein, 2014], we assume that the learner is following the gradient direction of her expected discounted return. The algorithm learns the reward function that minimizes the distance between the actual policy parameters of the learner and the policy parameters that should be obtained if she were following the policy gradient using that reward function.

After a formal introduction of the IRLfL setting in Section 7.1, we provide in Section 7.2 a first solution to the IRLfL problem when the observer has full access to the learner's policy parameters and learning rates. Then, in Section 7.3 we extend the algorithm to the more realistic case in which the observer can identify the optimized reward function only by analyzing the learner's trajectories. For each problem setting, we provide a finite-sample analysis to give to the reader an intuition on the correctness of the recovered weights. Finally, we consider discrete and continuous simulated domains to empirically compare the proposed algorithm with state-of-the-art baselines in this setting [Jacq et al., 2019, Brown et al., 2019]. Moreover, we report preliminary results on a simulated autonomous driving task (see Section 7.5.3).

7.1 Problem statement

The Inverse Reinforcement Learning from a Learner setting (IRLfL), proposed in [Jacq et al., 2019], involves two agents (as shown in Figure 7.1):

- a *learner* which is learning a task defined by the reward function $R_{\boldsymbol{\omega}^L}$,
- and an *observer* which wants to infer the learner's reward function.

More formally, the learner is an RL agent which is learning a policy $\pi_{\boldsymbol{\theta}} \in \Pi_\Theta$ in order to maximize its *discounted expected return* $J(\boldsymbol{\theta}, \boldsymbol{\omega}^L)$. The learner is improving its own policy by an update function $f(\boldsymbol{\theta}, \boldsymbol{\omega}^L) : \mathbb{R}^d \times \mathbb{R}^q \to \mathbb{R}^d$, i.e., at time t, $\boldsymbol{\theta}_{t+1} = f(\boldsymbol{\theta}_t, \boldsymbol{\omega}^L)$. The observer, instead, perceives a sequence of learner's policy parameters $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{m+1}\}$ and/or a dataset of trajectories for each policy $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_{m+1}\}$, where $\mathcal{D}_i = \{\tau_1^i, \cdots, \tau_n^i\}$. Her goal is to recover the reward function $R_{\boldsymbol{\omega}^L}$ that explains $\pi_{\boldsymbol{\theta}_i} \to \pi_{\boldsymbol{\theta}_{i+1}}$ for all $1 \leq i \leq m$, i.e., the updates of the learner's policy. We denote by $\boldsymbol{\omega}^L$ ($\boldsymbol{\theta}^L$) the reward (policy) parameters of the learner, and by $\widehat{\boldsymbol{\omega}}$ ($\widehat{\boldsymbol{\theta}}$) the reward (policy) parameters recovered by the observer.

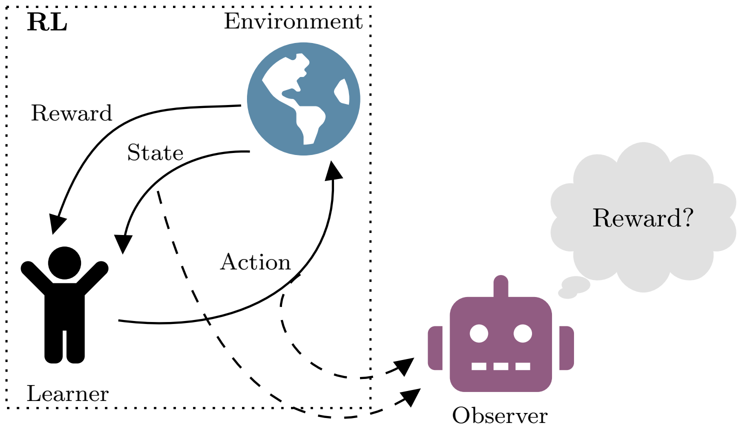


Figure 7.1: *The learner-environment-observer interaction in the Inverse Reinforcement Learning from a Learner framework.*

IRL algorithms for IRLfL problem It is easy to notice that this problem has the same intention as Inverse Reinforcement Learning since the demonstrating agent is motivated by some reward function. On the other hand, in classical IRL, the learner agent is an expert and not a non-stationary agent. For this reason, we cannot simply apply standard IRL algorithms to this problem or use Behavioral Cloning [Pomerleau, 1989, Argall et al., 2009, Osa et al., 2018] algorithms, which mimic a suboptimal behavior.

7.2 Learning from a learner following the gradient

Many algorithms that are the state of the art of reinforcement learning are policy-gradient methods [Deisenroth et al., 2013, Peters and Schaal, 2008b, Sutton et al., 2000], i.e., approaches which optimize the expected discounted return with gradient updates of the policy parameters. Moreover, recently it has been proved that even standard RL algorithms such as Value Iteration or Q-learning have strict connections with policy gradient methods [Goyal and Grand-Clement, 2019, Schulman et al., 2017a], and some neuroscience works established that the human learning process follows the gradient direction [Shteingart and Loewenstein, 2014]. For the above reasons, in our work, we assume that the learner is optimizing the expected discounted return using gradient descent.

For the sake of presentation, we start by considering the simplified case in which we assume that the observer can perceive the sequence of the learner's policy parameters $(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{m+1})$, the associated gradients of the feature expectations $(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}_1), \ldots, \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}_m))$, and the learning rates $(\alpha_1, \cdots, \alpha_m)$. Then, we will replace the exact knowledge of the gradients with estimates built on a set of demonstrations $\mathcal{D}_i$ for each learner's policy $\pi_{\boldsymbol{\theta}_i}$ (Section 7.2.2). Finally, we introduce our algorithm LOGEL, which, using behavioral cloning and an alternate block-coordinate optimization [Tseng, 2001], is able to estimate the reward's parameters without requiring as input the policy parameters and the learning rates (Section 7.3).

7.2.1 Exact gradient

We express the gradient of the expected return as [Sutton et al., 2000, Peters and Schaal, 2008b]:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}, \boldsymbol{\omega}) &= \mathop{\mathbb{E}}_{\substack{s_0 \sim \mu,\\ a_h \sim \pi_{\boldsymbol{\theta}}(\cdot|s_h),\\ s_{h+1} \sim P(\cdot|s_h, a_h)}} \left[\sum_{h=0}^{+\infty} \gamma^t R_{\boldsymbol{\omega}}(s_h, a_h) \sum_{l=0}^{t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_l|s_l) \right] \\ &= \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega},\end{aligned}$$

where $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) = (\nabla_{\boldsymbol{\theta}}\psi_1(\boldsymbol{\theta})| \ldots |\nabla_{\boldsymbol{\theta}}\psi_q(\boldsymbol{\theta})) \in \mathbb{R}^{d \times q}$ is the Jacobian matrix of the feature expectations $\boldsymbol{\psi}(\boldsymbol{\theta})$ w.r.t. the policy parameters $\boldsymbol{\theta}$. In the rest of the chapter, with some abuse of notation, we will indicate $\boldsymbol{\psi}(\boldsymbol{\theta}_t)$ with $\boldsymbol{\psi}_t$.

We define the gradient-based learner updating rule at time t as:

$$\boldsymbol{\theta}^L_{t+1} = \boldsymbol{\theta}^L_t + \alpha_t \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^L_t, \boldsymbol{\omega}^L) = \boldsymbol{\theta}^L_t + \alpha_t \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^L_t \boldsymbol{\omega}^L, \qquad (7.1)$$

where α_t is the learning rate. Given a sequence of consecutive policy parameters $(\boldsymbol{\theta}^L_1, \cdots, \boldsymbol{\theta}^L_{m+1})$, and of learning rates $(\alpha_1, \cdots, \alpha_m)$ the observer has to find the reward function $R_{\boldsymbol{\omega}}$ such that the improvements are explainable by the update rule in Eq. (7.1). This implies that the observer has to solve the following minimization problem:

$$\min_{\boldsymbol{\omega} \in \mathbb{R}^q} \sum_{i=1}^{m} \left\| \Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i \boldsymbol{\omega} \right\|_2^2, \qquad (7.2)$$

where $\Delta_i = \boldsymbol{\theta}_{i+1} - \boldsymbol{\theta}_i$. This optimization problem can be easily solved in closed-form under the assumption that $\left(\sum_{i=1}^m \alpha_i^2 \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i\right)^{-1}$ is invertible.

Lemma 7.2.1. *If the matrix* $\left(\sum_{i=1}^m \alpha_i^2 \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i\right)^{-1}$ *is full-rank then the optimization problem* (7.2) *is solved in closed form by*

$$\widehat{\boldsymbol{\omega}} = \left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i \right)^{-1} \left(\sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i^T \Delta_i \right). \qquad (7.3)$$

Proof. Taking the derivative of (7.2) with respect to ω:

$$\begin{aligned}
\nabla_{\boldsymbol{\omega}} \sum_{i=1}^{m} \|\Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}\|_2^2 &= \sum_{i=1}^{m} \nabla_{\boldsymbol{\omega}} (\Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega})^T (\Delta_i - \alpha \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}) \\
&= \sum_{i=1}^{m} \nabla_{\boldsymbol{\omega}} (\Delta_i^T \Delta_i + (\alpha \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega})^T (\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}) - (2\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega})^T \Delta_i) \\
&= 2 \left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \right) \boldsymbol{\omega} - 2 \sum_{i=1}^{m} \left(\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i \right).
\end{aligned}$$

Taking it equal to zero:

$$\begin{aligned}
\left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \right) \boldsymbol{\omega} - \sum_{i=1}^{m} \left(\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i \right) &= 0 \\
\boldsymbol{\omega} &= \left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \right)^{-1} \left(\sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i \right)
\end{aligned}$$

□

When problem (7.2) has no unique solution or when the matrix to be inverted is nearly singular, in order to avoid numerical issues, we can resort to a regularized version of the optimization problem. In the case we add an L2-norm penalty term over weights $\boldsymbol{\omega}$ we can still compute a closed-form solution.

Lemma 7.2.2. *The regularized version of* (7.2) *is equal to:*

$$\min_{\boldsymbol{\omega}} \sum_{i=1}^{m} \|\Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}\|_2^2 + \lambda \|\boldsymbol{\omega}\|_2^2,$$

where $\lambda > 0$. *We can solve the regularized problem in closed form:*

$$\boldsymbol{\omega} = \left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i + \lambda \mathbf{I}_q \right)^{-1} \left(\sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i \right).$$

Proof. Taking the derivative respect to ω:

$$\begin{aligned}
&\nabla_{\omega} \sum_{i=1}^{m} \|\Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}\|_2^2 + \lambda \|\boldsymbol{\omega}\|_2^2 \\
&\quad = \sum_{i=1}^{m} \nabla_{\boldsymbol{\omega}} (\Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega})^T (\Delta_i - \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}) + \nabla_{\boldsymbol{\omega}} \lambda \boldsymbol{\omega}^T \boldsymbol{\omega} \\
&\quad = \sum_{i=1}^{m} \nabla_{\boldsymbol{\omega}} (\Delta_i^T \Delta_i + (\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega})^T (\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega}) - 2\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \boldsymbol{\omega})^T \Delta_i) + 2\lambda \boldsymbol{\omega} \\
&\quad = 2 \left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i \right) \boldsymbol{\omega} - 2 \sum_{i=1}^{m} \left(\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i \right) + 2\lambda \boldsymbol{\omega}.
\end{aligned}$$

Taking it equal to zero:

$$\left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i + \lambda \mathbf{I}_q\right) \boldsymbol{\omega} - \sum_{i=1}^{m} \left(\alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i\right) = 0$$
$$\boldsymbol{\omega} = \left(\sum_{i=1}^{m} \alpha_i^2 \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i + \lambda \mathbf{I}_q\right)^{-1} \left(\sum_{i=1}^{m} \alpha_i \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i^T \Delta_i\right).$$

□

7.2.2 Approximate gradient

In practice, we do not have access to the Jacobian matrix $\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}$, but the observer has to estimate it using the learner's dataset of learning demonstrations D and some unbiased policy gradient estimator, such as REINFORCE [Williams, 1992] or G(PO)MDP [Baxter and Bartlett, 2001] (see Chapter 2). The estimation of the Jacobian will introduce errors on the optimization problem (7.2). Obviously the estimation of the reward weights $\boldsymbol{\omega}$ becomes more accurate when more data are available [Pirotta et al., 2013]. On the other hand, during the learning process, the learner will produce more than one policy improvement, and the observer can use these improvements to get better estimates of the reward weights.

In order to have an insight on the relationship between the amount of data needed to estimate the gradient and the number of learning steps, we provide a finite-sample analysis on the norm of the difference between the learner's weights $\boldsymbol{\omega}^L$ and the recovered weights $\widehat{\boldsymbol{\omega}}$. The analysis takes into account the number of learning steps and the number of demonstrations for each learning step, without having any assumption on the policy of the learner. We denote with $\boldsymbol{\Psi} = [\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_1, \cdots, \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_m]^T$ the concatenation of the Jacobians and $\widehat{\boldsymbol{\Psi}} = \left[\widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}}_1, \cdots, \widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}}_m\right]^T$ the concatenation of the estimated Jacobians.

Theorem 7.2.1. *Let $\boldsymbol{\Psi}$ be the true Jacobians and $\widehat{\boldsymbol{\Psi}}$ the estimated Jacocobian from n trajectories $\{\tau_1, \cdots, \tau_n\}$. Assume that $\boldsymbol{\Psi}$ is bounded by a constant M and $\lambda_{\min}(\widehat{\boldsymbol{\Psi}}^T \widehat{\boldsymbol{\Psi}}) \geq \lambda > 0$. Then, with probability at least $1 - \delta$:*

$$\left\|\boldsymbol{\omega}^L - \widehat{\boldsymbol{\omega}}\right\|_2 \leq O\left(\frac{1}{\lambda} M \sqrt{\frac{dq}{2n}} \left(\sqrt{\frac{\log dq}{m}} + \sqrt{dq}\right)\right).$$

Proof. We decompose the estimated Jacobian $\widehat{\boldsymbol{\Psi}} = \boldsymbol{\Psi} + E$, where E is the random variable component caused by the estimation of the $\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}$. Since we estimate the jacobians with an unbiased estimator the mean of E is 0. We reshape $\boldsymbol{\Psi}$ and E as $\boldsymbol{\Psi} \in \mathbb{R}^{m \times dq}$ and $E \in \mathbb{R}^{m \times dq}$. Now E, since its mean is 0 and all lines are independent of each other, is a sub-Gaussian matrix with parameters $(\frac{1}{m}\sigma_E, \frac{1}{m}\Sigma_E)^1$.

The proof follows similar argument of the proof of Theorem 1 in [McWilliams et al., 2014].

$$\begin{aligned}
&\left\|(\mathbf{\Psi}+E)^T(\mathbf{\Psi}\boldsymbol{\omega}^L) - (\mathbf{\Psi}+E)^T(\mathbf{\Psi}+E)\boldsymbol{\omega}^L\right\|_2 \\
&= \left\|\mathbf{\Psi}^T\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T\boldsymbol{\omega}^L + E^T\mathbf{\Psi}^T\boldsymbol{\omega}^L - \mathbf{\Psi}^T\mathbf{\Psi}\boldsymbol{\omega}^L - \mathbf{\Psi}^TE\boldsymbol{\omega}^* - E\mathbf{\Psi}^T\boldsymbol{\omega}^L - E^TE\boldsymbol{\omega}^L\right\|_2 \\
&= \left\|-\mathbf{\Psi}^TE\boldsymbol{\omega}^L - E^TE\boldsymbol{\omega}^L\right\|_2.
\end{aligned}$$

Now we bound separately these two terms, using Lemma C.1.3:

$$\begin{aligned}
&\left\|\mathbf{\Psi}^TE\boldsymbol{\omega}^L\right\|_2 \leq \|\mathbf{\Psi}\|_2\,\sigma_E\left\|\boldsymbol{\omega}^L\right\|_2\sqrt{\frac{\log dq}{m}} \\
&\left\|E^TE\boldsymbol{\omega}^L\right\|_2 = \left\|(E^TE+\sigma_E^2\mathbf{I}_{qd}-\sigma_E^2\mathbf{I}_{qd})\boldsymbol{\omega}^L\right\|_2 \leq \sigma_E^2\left(C\sqrt{\frac{\log dq}{m}}+\sqrt{dq}\right)\left\|\boldsymbol{\omega}^L\right\|_2
\end{aligned}$$

with probability $1-c_1\exp(-c_2\log dq)$ where c_1, c_2 are positive constants that do not depend on σ_E, n, q. Now applying Lemma C.1.2:

$$\left\|\boldsymbol{\omega}^L-\widehat{\boldsymbol{\omega}}\right\|_2 \leq \frac{1}{\lambda}\left(\|\mathbf{\Psi}\|_2\,\sigma_E\left\|\boldsymbol{\omega}^L\right\|_2\sqrt{\frac{\log dq}{m}} + \sigma_E^2\left(C\sqrt{\frac{\log dq}{m}}+\sqrt{dq}\right)\left\|\boldsymbol{\omega}^L\right\|_2\right)$$

We need, now, to bound the random variable σ_E. Remember that $E_i = \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i - \widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}_i$. Since $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ are assumed to be bounded by M, by applying Hoeffding's inequality, with probability $1-\delta_1$,

$$\|E_i\|_2 = \left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}_i - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}_i\right\|_2 \leq M\sqrt{\frac{dq\log(\frac{2}{\delta_1})}{2n}}.$$

So E is a subgaussian random variable where each component is bounded by $M\sqrt{\frac{\bullet dq\log(\frac{2}{\delta_1})}{2n}}$.

Then,

$$\begin{aligned}
&\mathrm{P}\left[\left\|\boldsymbol{\omega}^L-\widehat{\boldsymbol{\omega}}\right\|_2 \geq \frac{1}{\lambda}M\sqrt{\frac{dq\log(\frac{2}{\delta_1})}{2n}}\left\|\boldsymbol{\omega}^L\right\|_2\|\mathbf{\Psi}\|_2\sqrt{\frac{\log dq}{m}} + M\frac{dq\log(\frac{2}{\delta_1})}{2n}C\sqrt{\frac{\log dq}{m}}+\sqrt{dq}\right] \\
&\leq \mathrm{P}\left[\left\|\boldsymbol{\omega}^L-\widehat{\boldsymbol{\omega}}\right\|_2 \geq \frac{1}{\lambda}M\sqrt{\frac{dq\log(\frac{2}{\delta_1})}{2n}}\left\|\boldsymbol{\omega}^L\right\|_2\left(\|\mathbf{\Psi}\|_2\sqrt{\frac{\log dq}{m}}+C\sqrt{\frac{\log dq}{m}}+\sqrt{dq}\right)\right] \\
&\leq \delta_1 + c1\exp(-c2\log dq)
\end{aligned}$$

So the result follows after renaming $\delta = 1-(\delta_1+c1\exp(-c2\log dq))$ as in [McWilliams et al., 2014]. □

The theorem shows how we can reduce the error on the recovered weights increasing the dataset size n for each learning step and/or increasing the

[1]A zero-mean matrix E is called sub-Gaussian with parameter $(\frac{1}{m}\sigma_E, \frac{1}{m}\Sigma_E)$, if each row e^T is sampled independently and has $\mathbb{E}[e_ie_i^T] = \frac{1}{m}\Sigma_E$ and for any unit vector $v \in \mathbb{R}^p$, v^Te_i is a sub-Gaussian random variable with parameter at most $\frac{1}{\sqrt{p}}\sigma_E$ [McWilliams et al., 2014].

number of learning steps m. This fact is quite important for this problem. In fact, the number of policy improvement steps of the learner is finite as the learner will eventually achieve an optimal policy. Knowing the finite number of learning improvements m, we can estimate how much data n we need for each policy to get an estimate with a certain accuracy.

Intrinsic bias Another important aspect to take into account is the intrinsic bias [McWilliams et al., 2014] due to the gradient estimation error that cannot be solved by increasing the number of learning steps, but only with a more accurate estimation of the gradient. However, we show in Section 9.8 that, experimentally, this intrinsic bias, i.e. the component of the bound that does not depend on the number of learning steps, does not influence the capability of recovered correct weights.

7.3 Learning from improvement trajectories

In a more realistic scenario, the observer has access only to a dataset $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_{m+1})$ of trajectories generated by each policy, such that $\mathcal{D}_i = \{\tau_1, \cdots, \tau_n\} \sim \pi_{\boldsymbol{\theta}_i}$. Furthermore, the learning rates are unknown and possibly the learner applies an update rule other than (7.1). The observer has to infer the policy parameters $\Theta = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{m+1})$, the learning rates $A = (\alpha_1, \dots, \alpha_m)$, and the reward weights $\boldsymbol{\omega}$. If we suppose that the learner is updating its policy parameters with gradient ascent on the discounted expected return, the natural way to see this problem is to maximize the log-likelihood of $p(\boldsymbol{\theta}_1, \boldsymbol{\omega}, A | \mathcal{D})$:

$$\max_{\boldsymbol{\theta}_1, \boldsymbol{\omega}, A} \sum_{(s,a)\in\mathcal{D}_1} \log \pi_{\boldsymbol{\theta}_1}(a|s) + \sum_{i=2}^{m+1} \sum_{(s,a)\in\mathcal{D}_i} \log \pi_{\boldsymbol{\theta}_i}(a|s),$$

where $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \alpha_{i-1} \nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_{i-1}$. Unfortunately, solving this problem directly is not practical as it involves evaluating gradients of the discounted expected return up to the m-th order. To deal with this, we break down the inference problem into two steps: the first one consists in recovering the policy parameters Θ of the learner and the second in estimating the learning rates A and the reward weights $\boldsymbol{\omega}$ (see Algorithm 10). To some extent, this process resembles the expectation-maximization approach of the previous Chapter 6.

To recover the policy parameter we apply the same approach as exposed in Section 5.1. So, we apply a behavioral cloning procedure exploiting the trajectories in $\mathcal{D} = \{\mathcal{D}_1, \cdots, \mathcal{D}_{m+1}\}$ and casting the problem of finding the parameter $\boldsymbol{\theta}_i$ to a maximum-likelihood estimation.

Algorithm 10 LOGEL

```
Require: Dataset $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_{m+1}\}$ with $\mathcal{D}_j = \{(\tau_1, \dots, \tau_{n_j}) \mid \tau_i \sim \pi_{\boldsymbol{\theta}_j}\}$
Ensure: Reward weights $\boldsymbol{\omega} \in \mathbb{R}^q$
1: Estimate policy parameters $(\hat{\boldsymbol{\theta}}_1, \dots, \hat{\boldsymbol{\theta}}_{m+1})$ with Equation 5.4
2: Initialize $A$ and $\boldsymbol{\omega}$
3: Compute learning rates $A$ and reward weights $\boldsymbol{\omega}$ by alternating Equation (7.6) and Equation (7.3) up to convergence
```

7.3.1 Recovering learning rates and reward weights

Given the parameters $(\widehat{\boldsymbol{\theta}}_1, \dots, \widehat{\boldsymbol{\theta}}_{m+1})$ learned by behavioral cloning, if the learner is updating its policy with a constant (even unknown) learning rate we can simply apply Eq. (7.2). On the other hand, with an unknown learner, we cannot make this assumption and it is necessary to estimate also the learning rates $A = (\alpha_1, \dots, \alpha_m)$. The optimization problem in Eq. (7.2) becomes:

$$\min_{\boldsymbol{\omega} \in \mathbb{R}^q, A \in \mathbb{R}^m} \sum_{t=1}^{m} \left\| \widehat{\Delta}_t - \alpha_t \widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_t} \boldsymbol{\omega} \right\|_2^2 \tag{7.4}$$

$$\text{s.t.} \quad \alpha_t \geq \epsilon \quad 1 \leq t \leq m. \tag{7.5}$$

where $\widehat{\Delta}_t = \widehat{\boldsymbol{\theta}}_{t+1} - \widehat{\boldsymbol{\theta}}_t$ and ϵ is a small constant. To optimize this function we use alternate block-coordinate descent [Tseng, 2001]. We alternate the optimization of parameters A and the optimization of parameters $\boldsymbol{\omega}$. Furthermore, we notice that these two steps can be solved in closed form. When we optimize on $\boldsymbol{\omega}$, the optimization can be done using Lemma 7.2.1. Instead, when we search for the learning rates A we can solve for each parameter $\alpha_i \in A$, with $1 \leq i \leq m$, in closed form.

Lemma 7.3.1. *The minimizer of* (7.4) *with respect to* $\alpha_i \in A$ *is equal to:*

$$\hat{\alpha}_i = \max\left(\epsilon, \left((\widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i} \boldsymbol{\omega})^T (\widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i} \boldsymbol{\omega})\right)^{-1} (\widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}_i} \boldsymbol{\omega})^T \hat{\Delta}_i\right). \tag{7.6}$$

The inner matrix cannot be inverted only if the vector $\widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi} \boldsymbol{\omega}}$ is equal to $\mathbf{0}$. This would happen only if the expert is at a stationary point, so $\widehat{\nabla_{\boldsymbol{\theta}} \boldsymbol{\psi}}$ is $\mathbf{0}$. The alternated block-coordinate optimization converges under the assumption that there exists a unique minimum for each variable A and $\boldsymbol{\omega}$ [Tseng, 2001].

7.3.2 Theoretical result

In this section, we provide a finite-sample analysis of LOGEL when only one learning step is observed. In this setting, the observer has access to two datasets $\mathcal{D}_\infty, \mathcal{D}_\in$, where D_1 is generated by an unknown policy $\pi_{\boldsymbol{\theta}_1}$ and D_2 by the policy $\pi_{\boldsymbol{\theta}_2}$, where $\boldsymbol{\theta}_2 = \boldsymbol{\theta}_1 + \alpha\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_1, \boldsymbol{\omega}^L)$. We assume that the Jacobian matrix $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ is bounded and the learner's policy is a Gaussian policy $\pi \sim \mathcal{N}(\boldsymbol{\theta}^T\boldsymbol{\varphi}(s), \sigma^2)$. The analysis evaluates the norm of the difference between the learner's weights $\boldsymbol{\omega}^L$ and the recovered weights $\hat{\boldsymbol{\omega}}$. Without loss of generality, we consider the case where the learning rate $\alpha = 1$. The analysis takes into account the bias introduced by the Behavioral Cloning and the gradient estimation.

Theorem 7.3.1. *Let $\pi_{\boldsymbol{\theta}_1}, \pi_{\boldsymbol{\theta}_2}$ be two Gaussian policies $\pi_{\boldsymbol{\theta}_i}(\cdot|s) \sim \mathcal{N}(\boldsymbol{\theta}_i^T\boldsymbol{\varphi}(s), \sigma^2)$ with $i \in \{1, 2\}$, such that $\pi_{\boldsymbol{\theta}_2}$ is the improvement of $\pi_{\boldsymbol{\theta}_1}$. Let $i \in [1, 2]$. Given datasets $D_i = \{\tau_1^i, \dots, \tau_n^i\}$ of trajectories generated by π_i, such that $S_i \in \mathbb{R}^{nH\times d}$ is the matrix of corresponding states features, let the minimum singular value of $\sigma_{\min}(S_i^TS_i) \geq \eta > 0$, $\widehat{\nabla_{\boldsymbol{\theta}_1}\boldsymbol{\psi}}$ uniformly bounded by M, the state features bounded by M_S, and the reward features bounded by M_R. Then with probability $1 - \delta$:*

$$\left\|\boldsymbol{\omega}^L - \widehat{\boldsymbol{\omega}}\right\|_2 \leq O\left(\frac{(M + M_S^2 M_R)}{\sigma_{\min}(\nabla_{\boldsymbol{\theta}_1}\boldsymbol{\psi})}\sqrt{\frac{\log(\frac{2}{\delta})}{n\eta}}\right)$$

where ω^L are the true reward parameters and $\widehat{\omega}$ are the parameters recovered using Lemma 7.2.1.

The theorem, that relies on perturbation analysis [Wedin, 1973] and least squares with fixed design [Rigollet, 2015], underlines how LOGEL, with a sufficient number of samples to estimate the policy parameters and the gradients, succeeds in recovering the correct reward parameters.

Proof of Theorem 7.3.1

In this section we prove the theoretical result on the recovered weights from our algorithm LOGEL.

We start by bounding the bias introduced by the behavioral cloning procedure.

Lemma 7.3.2. *Given a dataset $\mathcal{D} = \{(s_1, a_1), \cdots, (s_{nH}, a_{nH})\}$ of state-action couples sampled from a Gaussian linear policy $\pi_{\boldsymbol{\theta}}(\cdot|s) \sim \mathcal{N}(\boldsymbol{\theta}^T\boldsymbol{\varphi}(s), \sigma^2)$ such that $S \in \mathbb{R}^{nH\times p}$ is the matrix of states features and let the minimum* *singular value of $(S^T S)$ be $\sigma_{\min} \geq \eta > 0$, then the error between the maximum likelihood estimator $\boldsymbol{\theta}^{MLE}$ and the mean $\boldsymbol{\theta}$ is, with probability $1 - \delta$:*

$$\left\|\boldsymbol{\theta}^{MLE} - \boldsymbol{\theta}\right\|_2 \leq \sigma\sqrt{\frac{r + \log(\frac{1}{\delta})}{nT\eta}},$$

where r is rank($S^T S$).

Proof. We start by stating that the maximum likelihood for linear Gaussian policies can be recast as an ordinary least-squares problem. We write the log-likelihood $\log L(\boldsymbol{\theta})$.

$$\begin{aligned}\log L(\boldsymbol{\theta}) = \log\left(\prod_{i=1}^{nH} \pi(a_i|s_i)\right) &= \sum_{i=1}^{nH} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(a_i - \boldsymbol{\theta}^T \boldsymbol{\varphi}(s_i))^2}{2\sigma^2}\right)\right) \\ &= nH \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{nH} \frac{(a_i - \boldsymbol{\theta}^T \boldsymbol{\varphi}(s_i))^2}{2\sigma^2}\end{aligned}$$

The resulting maximum likelihood problem is given by:

$$\arg\max_{\boldsymbol{\theta}} \log L(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{nH} (a_i - \boldsymbol{\theta}^T \boldsymbol{\varphi}(s_i))^2$$

Then having the following linear least-squares problem:

$$\min_{\theta} \|S\boldsymbol{\theta} - A + \epsilon\|_2,$$

where ϵ is an error with mean 0 and variance σ^2, $S \in \mathbb{R}^{nH \times p}$ is the matrix of states features and $A \in \mathbb{R}^{nH}$ is the vector of actions. Using Theorem C.1.1, we can say that with probability $1 - \delta$:

$$\left\|\boldsymbol{\theta}^{\text{MLE}} - \boldsymbol{\theta}\right\|_2 \leq \sigma\sqrt{\frac{r + \log(\frac{1}{\delta})}{nH\eta}},$$

where r is rank($S^T S$). □

We need now to add an auxiliary lemma to bound the 2-norm differences between the gradient of the logarithm of two Gaussian policies in the terms of the 2-norm difference between the means of the Gaussians.

Lemma 7.3.3. *Given two Gaussian policies $\pi_{\boldsymbol{\theta}_1}(\cdot|s) \sim \mathcal{N}(\boldsymbol{\theta}_1^T \boldsymbol{\varphi}(s), \sigma^2)$ and $\pi_{\boldsymbol{\theta}_2}(\cdot|s) \sim \mathcal{N}(\boldsymbol{\theta}_2^T \boldsymbol{\varphi}(s), \sigma^2)$ with same variance and and with state features bounded by M_S:*

$$\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_1}(a|s) - \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_2}(a|s)\|_2 \leq \frac{M_S^2}{\sigma^2} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2.$$

Proof. The gradient of the log policy of a general policy $\pi_{\boldsymbol{\theta}}(a|s)$ is:

$$\nabla_{\boldsymbol{\theta}} \log \pi(a|s) = \frac{\boldsymbol{\varphi}(s)^T(a - \boldsymbol{\theta}^T\boldsymbol{\varphi}(s))}{\sigma^2}.$$

Now we apply this result to the difference in norm between two Gaussian log policies:

$$\begin{aligned}
\left\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_1}(a|s) - \nabla_{\boldsymbol{\theta}} \log \pi_(a|s)\right\|_2 &= \left\| \frac{\boldsymbol{\varphi}(s)^T(a - \boldsymbol{\theta}_1^T\boldsymbol{\varphi}(s))}{\sigma^2} - \frac{\boldsymbol{\varphi}(s)^T(a - \boldsymbol{\theta}_2^T\boldsymbol{\varphi}(s))}{\sigma^2} \right\|_2 \\
&= \left\| \frac{\boldsymbol{\varphi}(s)}{\sigma^2}(\boldsymbol{\theta}_1^T\boldsymbol{\varphi}(s) - \boldsymbol{\theta}_2^T\boldsymbol{\varphi}(s)) \right\|_2 \\
&\leq \left\| \frac{\boldsymbol{\varphi}(s)}{\sigma^2} \right\|_2 \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2 \|\boldsymbol{\varphi}(s)\|_2 &(7.7)\\
&\leq \frac{M_S^2}{\sigma^2} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2. &(7.8)
\end{aligned}$$

In line 7.7 we use the Cauchy-Schwartz inequality, and in line 7.8 the assumption that the state features are bounded by M_S. □

Then we bound the 2-norm differences between the estimated Jacobian of a Gaussian policy parametrized by the maximum likelihood value, and the real Jacobian.

Lemma 7.3.4. *Given a dataset $\mathcal{D} = \{\tau_1, \cdots, \tau_n\}$ of trajectories such that every trajectory $\tau_i = \{(s_1, a_1), \cdots, (s_H, a_H)\}$ is sampled from a Gaussian linear policy $\pi_{\boldsymbol{\theta}}(\cdot|s) \sim \mathcal{N}(\boldsymbol{\theta}^T\boldsymbol{\varphi}(s), \sigma^2)$, the maximum likelihood estimator $\boldsymbol{\theta}^{MLE}$ computed on $\mathcal{D}$, the condition of Lemma 7.3.2 holds, the $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ uniformly bounded by M, the state features bounded by M_S, the reward features bounded by M_R. Then with probability $1-\delta$:*

$$\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\boldsymbol{\theta}^{MLE}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right\|_2 \leq M\sqrt{qd}\sqrt{\frac{\log(\frac{2}{\delta})}{2n}} + \frac{HM_S^2M_R}{(1-\gamma)\sigma}\sqrt{\frac{r + \log(\frac{1}{\delta})}{n\eta}},$$

where γ is the discount factor and r is the rank of S^TS.

Proof. We start by decomposing the norm of the difference in two components, using the triangular inequality:

$$\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\widehat{\boldsymbol{\theta}}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right\|_2 \leq \left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right\|_2 + \left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\boldsymbol{\theta}) - \widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\widehat{\boldsymbol{\theta}})\right\|_2.$$

The first component is bounded by Lemma C.2.1. We will bound now the second component, using the Reinforce estimator for the gradient:

$$\begin{aligned}
&\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\boldsymbol{\theta}) - \widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}(\widehat{\boldsymbol{\theta}})\right\|_2 = \\
&= \left\| \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t}) R_{i,t}\gamma^t - \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\widehat{\boldsymbol{\theta}}}(a_{i,t}|s_{i,t}) R_{i,t}\gamma^t \right\|_2 \\
&= \frac{1}{n}\left\| \sum_{i=1}^{n}\sum_{t=1}^{H} (\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t}) - \nabla_{\boldsymbol{\theta}} \log \pi_{\widehat{\boldsymbol{\theta}}}(a_{i,t}|s_{i,t})) R_{i,t}\gamma^t \right\|_2 \\
&\leq \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{H} \left\|(\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_{i,t}|s_{i,t}) - \nabla_{\boldsymbol{\theta}} \log \pi_{\widehat{\boldsymbol{\theta}}}(a_{i,t}|s_{i,t}))\right\|_2 \left\|R_t\gamma^t\right\|_2 && (7.9) \\
&\leq \frac{1}{n}\frac{M_R}{(1-\gamma)}\sum_{i=1}^{n}\sum_{t=1}^{H}\frac{M_S^2}{\sigma^2}\left\|\boldsymbol{\theta}-\widehat{\boldsymbol{\theta}}\right\|_2 && (7.10) \\
&\leq \frac{HM_S^2M_R}{\sigma^2(1-\gamma)}\sigma\sqrt{\frac{r+\log(\frac{1}{\delta})}{n\eta}}. && (7.11)
\end{aligned}$$

In line 7.9 we apply the Cauchy-Schwartz inequality. In line 7.10 we apply lemma 7.3.2 and in line 7.11 we apply lemma 7.3.2. Merging the two results the proof follows. □

Finally we derive the bound on the error on the recovered weights.

Theorem 7.3.1. *Let $\pi_{\boldsymbol{\theta}_1}, \pi_{\boldsymbol{\theta}_2}$ be two Gaussian policies $\pi_{\boldsymbol{\theta}_i}(\cdot|s) \sim \mathcal{N}(\boldsymbol{\theta}_i^T\boldsymbol{\varphi}(s), \sigma^2)$ with $i \in \{1,2\}$, such that $\pi_{\boldsymbol{\theta}_2}$ is the improvement of $\pi_{\boldsymbol{\theta}_1}$. Let $i \in [1,2]$. Given datasets $D_i = \{\tau_1^i, \ldots, \tau_n^i\}$ of trajectories generated by π_i, such that $S_i \in \mathbb{R}^{nH\times d}$ is the matrix of corresponding states features, let the minimum singular value of $\sigma_{\min}(S_i^TS_i) \geq \eta > 0$, $\widehat{\nabla_{\boldsymbol{\theta}_1}\boldsymbol{\psi}}$ uniformly bounded by M, the state features bounded by M_S, and the reward features bounded by M_R. Then with probability $1-\delta$:*

$$\left\|\boldsymbol{\omega}^L - \widehat{\boldsymbol{\omega}}\right\|_2 \leq O\left(\frac{(M+M_S^2M_R)}{\sigma_{\min}(\nabla_{\boldsymbol{\theta}_1}\boldsymbol{\psi})}\sqrt{\frac{\log(\frac{2}{\delta})}{n\eta}}\right)$$

where ω^L are the true reward parameters and $\widehat{\omega}$ are the parameters recovered using Lemma 7.2.1.

Proof. First we have to bound the error on Δ created by the behavioral cloning. Given $\Delta = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1$ and $\widehat{\Delta} = \widehat{\boldsymbol{\theta}}_2 - \hat{\boldsymbol{\theta}}_1$:

$$\left\|\Delta - \widehat{\Delta}\right\| = \left\|\boldsymbol{\theta}_2 - \boldsymbol{\theta}_1 - \widehat{\boldsymbol{\theta}}_2 + \hat{\boldsymbol{\theta}}_1\right\| \leq \left\|\boldsymbol{\theta}_1 - \hat{\boldsymbol{\theta}}_1\right\| + \left\|\boldsymbol{\theta}_2 - \widehat{\boldsymbol{\theta}}_2\right\| \leq 2\sigma\sqrt{\frac{r+\log(\frac{1}{\delta})}{n\eta}}.$$

So we can bound the difference in norm between the real weights $\boldsymbol{\omega}^L$ and the estimated weights $\widehat{\boldsymbol{\omega}}$. We indicate with κ the condition number of $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}$, with $\chi = \frac{\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}$ and $y = \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^H\boldsymbol{\omega}$. We apply the pertubation Lemma C.1.1.

$$
\begin{aligned}
\left\|\boldsymbol{\omega}^L-\widehat{\boldsymbol{\omega}}\right\|_2 &\leq \frac{\kappa}{(1-\kappa\chi)\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}(\chi\left\|\boldsymbol{\omega}^L\right\|_2\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2+\left\|\Delta-\hat{\Delta}\right\|_2)+\chi\left\|y\right\|_2\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2 \qquad (7.12)\\
&\leq \frac{\kappa\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}-\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}{c\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}\left\|\boldsymbol{\omega}^L\right\|_2+\frac{\kappa\left\|\Delta-\widehat{\Delta}\right\|_2}{c\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}+\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2\left\|y\right\|_2 \qquad (7.13)\\
&=\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}-\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}\right\|_2\left(\frac{\kappa\left\|\boldsymbol{\omega}^L\right\|_2}{c\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}+\left\|y\right\|_2\right)+\left\|\Delta-\widehat{\Delta}\right\|_2\frac{\kappa}{c\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}\\
&=\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}-\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}\right\|_2\left(\frac{\left\|\boldsymbol{\omega}^L\right\|_2}{c\sigma_{\min}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi})}+\left\|y\right\|_2\right)+\left\|\Delta-\widehat{\Delta}\right\|_2\frac{1}{c\sigma_{\min}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi})}\\
&\quad+2\sigma\sqrt{\frac{r+\log(\frac{1}{\delta})}{n\eta}}\frac{1}{c\sigma_{\min}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi})}\\
&\leq O\left(\frac{(M+M_S^2M_R)}{\sigma_{\min}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi})}\sqrt{\frac{\log(\frac{2}{\delta})}{n\eta}}\right),
\end{aligned}
$$

where in line 7.12 we apply Lemma C.1.1 and in line 7.13 we apply Equation 7.3.2 and Lemma 7.3.4.

□

7.4 Discussion on the related works

This setting was proposed in [Jacq et al., 2019]. The authors proposed a method based on entropy-regularized reinforcement learning, in which they assumed that the learner is performing soft policy improvements. In order to derive their algorithm, the authors also assume that the learner satisfies the policy improvement condition. We remove this assumption, taking only that the learner is changing its policy parameters along the gradient direction (which can result in a performance loss). Other works such as T-REX [Brown et al., 2019, Castro et al., 2019] also recover the reward function from suboptimal demonstrations, although they require to have supplementary information about the observed trajectories such as a performance ranking or human labeling.

7.5 Experiments

This section is devoted to the experimental evaluation of LOGEL. The algorithm LOGEL is compared to the state-of-the-art baseline Learner From a Learner (LfL) [Jacq et al., 2019] and T-REX [Brown et al., 2019] in a gridworld navigation task and in two MuJoCo environments. In these experiments the assumption that the learner is gradient-based is violated

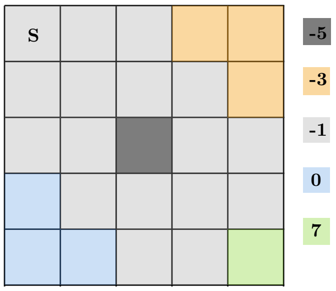


Figure 7.2: *Gridworld environment: every area has a different reward weight. In the green area the agent is reset to the starting state.*

and in the MuJoCo task the reward features are constructed by states and actions features. Therefore we can argue that in this experiment the reward linearity assumption is violated, since we use a different reward space for the recovered reward function.

7.5.1 Gridworld

The first set of experiments aims at evaluating the performance of LOGEL in a discrete Gridworld environment. The Gridworld, represented in Figure 7.2, is composed of five regions with a different reward for each area. The agent starts from the cell top left and when it reaches the green state, then it returns to the starting state. The reward feature space is composed of the one-hot encoding of five areas: the orange, the light grey, the dark grey, the blue, and the green. The learner weights for the areas are $(-3, -1, -5, 7, 0)$ respectively.

As a first experiment, we want to verify in practice the theoretical finding exposed in Section 7.2. In this experiment, the learner uses a Boltzmann policy and she is learning with the G(PO)MDP policy gradient algorithm. The observer has access to the true policy parameters of the learner. Figure 7.3 shows the performance of LOGEL in two settings: a single learning step and increasing batch size (5, 10, 20, 30, 40, 50); a fixed batch size (batch size 5 and trajectory length 20) and an increasing number of learning steps (2, 4, 6, 8, 10). The figure shows the expected discounted return (evaluated in closed form) and the difference in norm between the learner's weights and the recovered weights [2]. We note that, as explained in Theorem 7.2.1, with a more accurate gradient estimate, the observer succeeds in recovering the reward weights by observing even just one learning step. On the other hand, as we can deduce from Theorem 7.2.1, if we have a noisy estimation of the

[2]To perform this comparison, we normalize the recovered weights and the learner's weights

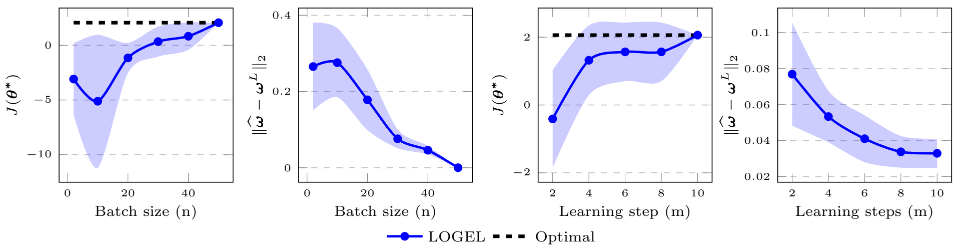


Figure 7.3: *Gridworld experiment with known policy parameters. The learner is using G(PO)MDP. From left the expected discounted return and the norm difference between the true weights and the recovered ones with one learning step; the same measures with fixed batch size (*5 *trajectories with length* 20*). The performance of the observers are evaluated on the learner's reward weights. Results are averaged over 20 runs.* 98% *c.i as shaded area.*

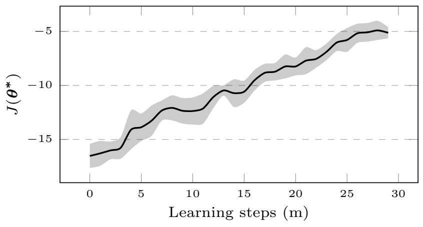


Figure 7.4: *Learning performance of G(PO)MDP. 20 runs, 98%c.i.*

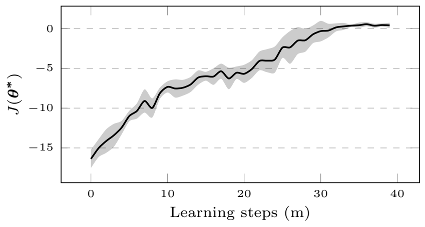


Figure 7.5: *Learning performance of Q-Learning. 20 runs, 98%c.i.*

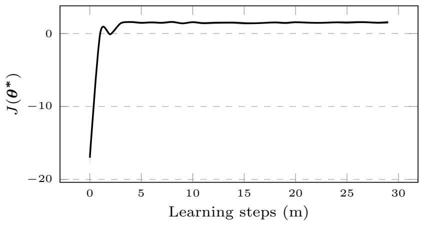


Figure 7.6: *Learning performance of SPI. 20 runs, 98%c.i.*

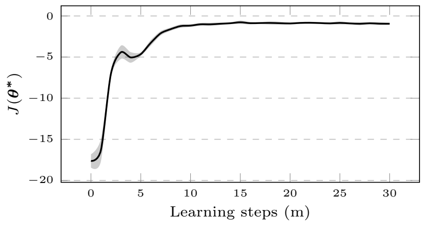


Figure 7.7: *Learning performance of SVI. 20 runs, 98%c.i.*

gradient, with multiple learning steps, the observer succeeds in recovering the learner's weights. It is interesting to notice that, from this experiment, it seems that the bias component, which does not vanish as the learning steps increase (see Theorem 7.2.1), does not affect the correctness of the recovered weights.

In the second experiment we consider four different learners using: Q-learning [Sutton et al., 1998], G(PO)MDP [Deisenroth et al., 2013], Soft policy improvement (SPI) [Jacq et al., 2019] and Soft Value Iteration

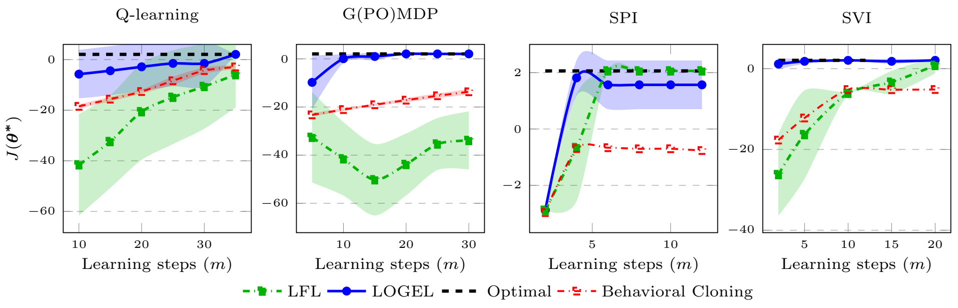


Figure 7.8: *Gridworld experiment with estimated policy parameters and four learners: from left Q-learning, G(PO)MDP, SPI, SVI. The green line is the LfL observer, the blue one is the LOGEL observer and the red one Behavioral Cloning. The performance of the observers are evaluated on the learner's reward weights. Results are everaged over 20 runs.* 98% *c.i. as shaded area.*

(SVI) [Haarnoja et al., 2018]. First of all we analyze the learning process of each learning agent, because the number of learning steps that we employ in the experiment depends on the number of policy updates that the learner takes to become an expert. In the following plots, we report the expected discounted return for each learner: Q-Learning (Figure 7.5), G(PO)MDP (Figure 7.4), SPI (Figure 7.6), SVI (Figure 7.7). In these plots, the expected discounted return is estimated using a batch of 50 trajectories for each learner. The discount factor used in all experiments is 0.96. After this step we perform the experiment by setting the number of learning steps looking at the learning process (as in Figures 7.5,7.4,7.6,7.7): Q-learning and G(PO)MDP requires more steps to learn the task, instead SPI and SVI become optimal in less than 10 steps. For this experiment, we compare the performance of LOGEL , LfL [Jacq et al., 2019] and Behavioral Cloning. In Figure 7.8 we can notice that LOGEL succeeds in recovering the learner's reward weights even with learner algorithms other than gradient-based ones. In fact, SPI, SVI, and Q-learning are not gradient-based learners (although it was proved recently that Q-learning and SVI follow the gradient direction [Goyal and Grand-Clement, 2019]). On the other hand, LfL does not recover the reward weights of the G(PO)MDP learner, which does not use Bellman updates but gradient ones, and needs more learning steps than LOGEL to learn the reward weights when Q-learning learner and SVI are observed. However, it achieves comparable performance to LOGEL when the learner is SPI. Behavioral Cloning only mimics the last seen policy, which can be suboptimal.

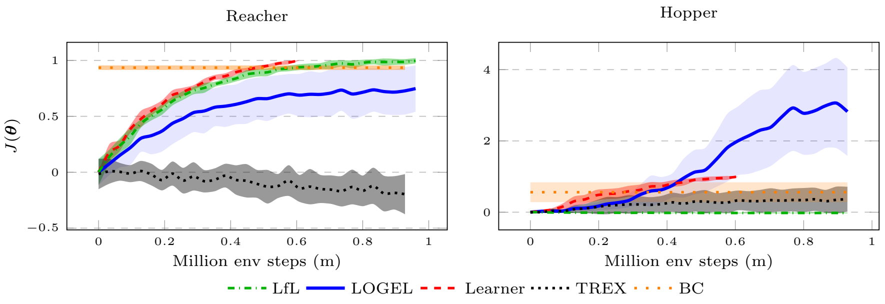


Figure 7.9: *From the left, the Reacher and the Hopper MuJoCo environments. The red line is the performance of the learner during* 20 *learning steps. The observers, LfL, LOGEL, T-REX and Behavioral Cloning (BC) observe the trajectories of the learning steps from* 10 *to* 20*. The performance of the observers are evaluated on the learner's reward weights. Scores are normalized setting to* 0 *the first return of the learner and to* 1 *the last one. The results are averaged over* 10 *runs.* 98% *c.i. are shown as shaded areas.*

7.5.2 MuJoCo environments

In the second set of experiments, we show the ability of LOGEL to infer the reward weights in more complex and continuous environments. We use two environments from the MuJoCo control suite [Brockman et al., 2016]: Hopper and Reacher. As in [Jacq et al., 2019], the learner is trained using Policy Proximal Optimization (PPO) [Schulman et al., 2017b][3], with 16 parallel agents for each learning step. For each step, the length of the trajectories is 2000. Then we use LOGEL , LfL or T-REX [Brown et al., 2019] to recover the reward parameters. In the end, the observer is trained with the recovered weights using PPO and the performances are evaluated on the learner's weights, starting from the same initial policy of the learner for a fair comparison. The scores are normalized by setting to 1 the score of the last observed learner policy and to 0 the score of the initial one (as done in [Jacq et al., 2019]). In both environments, the observer learns using the learning steps from 10 to 20 as the first learning steps are too noisy. The reward function of LfL is the same as the one used in the original paper, where the reward function is a neural network equal to the one used for the learner's policy. Instead, for LOGEL we used linear reward functions derived only from state and action features. The reward for the Reacher environment is a 26-grid radial basis function that describes the distance between the agent and the goal, plus the 2-norm squared of the action. In the Hopper environment, instead, the reward features are the distance

[3]It is important to notice that PPO violates the gradient learning assumption of LOGEL.

between the previous and the current position and the 2-norm squared of the action. The T-REX algorithm aims to recover a reward function from ranked trajectories, where the rank is given by an oracle and is based on the expected discounted return. We use the algorithm in the LfL setting, where we approximate the ranking with the temporal updates of the policies, as was done in an example in the original paper. We implement the reward function as in the original paper with a three layer neural network with 256 as hidden size.

The results are shown in Figure 7.9, where we reported results averaged over 10 runs. For Behavioral Cloning we report the performance of the policy learnt at the 20th step; in fact Behavioral Cloning cannot improve its performance with learning steps. We can notice that LOGEL succeeds in identifying a good reward function in both environments, although in the Reacher environment the recovered reward function causes slower learning. Instead, LfL fails to recover an effective reward function for the Hopper environment [Jacq et al., 2019]. The T-REX algorithm, as in the original paper, succeeds in recovering a good approximation of the reward weights in the Hopper domain; instead, it does not succeed into recovering the reward function of the Reacher environment.

Implementation details For the MuJoCo experiments, we use the same hyperparameters as in [Jacq et al., 2019], apart from that we use 16 parallel agents for PPO, due to resource constraints. The number of forward steps are settled to 2000. As in [Jacq et al., 2019], we select a subset of the learner's trajectories and we do not use the first 10 trajectories because the first phase of learning is too noisy. We evaluate the algorithms on the first 1 million environment steps.

7.5.3 Autonomous driving scenario

In this section, we report a preliminary experiment that we perform on a driving simulator scenario. We employ the SUMO simulator, an open-source, continuous road traffic simulation package designed to handle large road networks. SUMO focuses on the high-level control of the car, integrating an internal system that controls the vehicle dynamics. During the simulation, SUMO provides information on the other vehicles around the ego vehicle.

We consider a crossroad scenario which consists of an intersection with an arbitrary number of roads. The vehicle coming from the source road has to reach a target road that has a higher priority. The goal of the agent is to drive the ego car and enter the target road, avoiding dangerous manoeuvres.

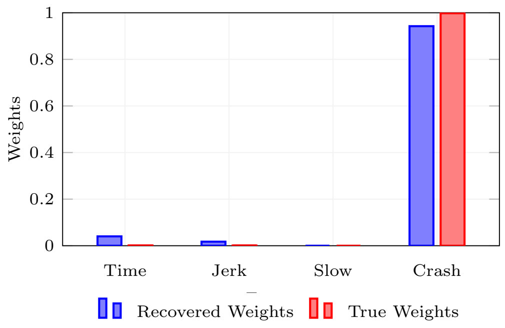


Figure 7.10: *Reward weights for the autonomous simulate driving scenario.*

	Recovered Weights	Real Weights
Time	0.0401	0.0017
Jerk	0.0174	0.0003
Slow	0.0001	0.0000
Crash	0.9424	0.9980

Table 7.1: *Reward weights for the autonomous simulate driving scenario.*

The reward features consists of four components:

- *Time*, a constant feature at each decision step;
- *Jerk*, the absolute value of the instantaneous jerk, i.e., the finite- difference derivative of the acceleration;
- *Harsh Slow Down*, a binary feature, which activates whenever the velocity is lower than a threshold;
- *Crash*, a binary feature which activates when the vehicle violates the safety constraints or performs a crash.

The agent's policy is a rule-based policy, i.e., a set of parametrized rules, which is learned using Policy Gradients with Parameter-based Exploration (PGPE) [Sehnke et al., 2008]. It is important to notice that the agent's policy is not differentiable. We perform 10 PGPE updates of the agents and then we use the learning trajectories with LOGEL. In the behavioural cloning step, we use a linear layer to approximate the policy of the learner. Table 7.1 shows the normalized weights recovered by LOGEL and the normalized real weights. As shown in Table 7.1 and in Figure 7.10, the reward weights recovered are quite similar to the real ones.

Part III

Online Learning in Multi-Agent Reinforcement Learning

Approaching the RL problem from an online learning perspective, we are not only interested in finding the optimal policies but also in measuring the performance of our algorithm during the learning process. The performance is measured using the *regret*, i.e., comparing the performance of the agent's actual policy with the optimal one; or evaluating the sample complexity, i.e., studying how many interactions with the environment are necessary to converge to an optimal solution. In the last two years, there is an increasing interest in studying the MARL problem from an online learning perspective, approaching the problem from two different perspectives: when we control all the involved agents and, instead, when we can control only one of them.

In Chapter 8 we present the problem formally and revise the related literature. In Chapter 9 we propose a new algorithm [Ramponi et al., 2021] to solve the problem of online learning in Configurable MDP. We start by introducing the concept of Non-cooperative MDP, i.e., a Configurable MDP where the agent and the configurator can have different reward functions. Then we propose two algorithms based on the feedback that the configurator perceives from the interaction between agent-environment. Finally, in Chapter 10, we propose a new lower bound on the expected regret for the General-sum Stochastic Game setting. Then we propose a new algorithm that nearly matches the proposed lower bound.

CHAPTER *8*

Online Learning in Stochastic Games

The RL goal that we have considered so far consists of finding a policy that maximizes the cumulative expected sum of rewards. In the online learning setting, instead, we are not only interested in finding the optimal policy but also in measuring the performance of our algorithm during the learning process. To evaluate the algorithm behavior, the reward that it collects is compared with the ones of an optimal policy. This performance measure, called regret, well characterizes the exploration-exploitation dilemma that we have introduced in Chapter 2, i.e., the problem of finding the best trade-off between following a decision rule that seems optimal and exploring different solutions. The exploration, clearly, can lead to instantaneous lower performances but can also provide information to learn the optimal policy [1].

In RL, the online learning setting was extensively studied in the Bandit literature [Auer et al., 2002, Lattimore and Szepesvári, 2020]. We have not introduced the Bandit framework in the preliminaries since they are out of the scope of this thesis. However, a Bandit is informally an MDP without dynamics, where the agent playing an action receives a stochastic realization of its reward. The objective is to learn the best action (or action in the

[1]We refer the reader to the Prediction, Learning & Games [Cesa-Bianchi and Lugosi, 2006] and Bandit Algorithms [Lattimore and Szepesvári, 2020] books for a thorough introduction to the online learning problem.

contextual setting) to maximize the expected return. Besides its apparent simplicity, this problem captures the difficulties of learning when we are in the presence of uncertainty and models many real-world problems.

There are also many works that treat the online learning problem in Markov Decision Processes [Jaksch et al., 2010, Bartlett and Tewari, 2009a, Fruit et al., 2018, Osband et al., 2013, Osband and Van Roy, 2017, Agrawal and Jia, 2017]. In the MDP setting, new challenges have to be solved. In fact, the exploration-exploitation dilemma in MDPs is clearly more complicated since we also have to account for the transition probability model and for its estimation. Moreover, when we move to the multi-agent setting, we add other difficulties since we also need to control more than one agent or/and to account for the non-controllable agents' decisions.

In this chapter, we introduce the online learning problem in Stochastic games, analyzing the two settings that were proposed in the literature: the *Online* and the *Offline* setting. Then, we review the works that addressed this problem.

In the rest of this part, we expose two algorithms to solve the online setting in two different contexts. First, we analyze the online learning in the Configurable Markov Decision Process (in Chapter 9), where we control only the configurator entity, and our algorithm does not handle the agent. Then, we account for the Turn-based Stochastic Games setting (in Chapter 10) where we have to face an unknown opponent. With these two contributions, we make a step forward to the theoretical understanding of the general-sum MARL problem.

8.1 Problem setting

In the Stochastic Games [Shapley, 1953] literature, the theoretical problem of learning are studied under two different setting, that we call as in [Wei et al., 2017, Xie et al., 2020a, Tian et al., 2020], *offline* and *online*.

In the offline setting [Szepesvári and Littman, 1996, Lagoudakis and Parr, 2002, Perolat et al., 2015], all the agents are controlled by a central learner, which decides what actions they have to take. It can also be thought of as learning in a self-play, i.e., where we are against ourselves. In this framework, the goal, in general, is finding an ϵ-Nash Equilibrium. The algorithm is usually measured by its sample-complexity, i.e., how many interactions with the environment are necessary to learn an ϵ-Nash Equilibrium. Obviously, if we are in a zero-sum game, it is equivalent to estimate how many samples are required to learn a maximin (or minimax) policy.

In the *online* setting [Littman, 1994, Bowling and Veloso, 2002, Brafman and Tennenholtz, 2002, Conitzer and Sandholm, 2007], instead, we can control only one agent, which has to maximize its own expected cumulative sum of rewards in a multi-agent environment. This leads to new challenges with respect to the single-agent setting since the environment is no longer stationary and, so our exploration process is highly dependent on the other agents' policies. In the online setting, we are interested in studying the algorithm's regret, i.e., the difference between the cumulative sum of rewards obtained by a benchmark policy and the one obtained by our algorithm during the learning process.

Since we propose two algorithms to deal with the online setting, we continue this section introducing the preliminaries for this one.

In this thesis, we consider the two-player finite-horizon (episodic) Stochastic Game setting, $SG = (N, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mu, H)$, described in Chapter 4, i.e., an SG such that $N = 2$, $H < \infty$ and $\gamma = 1$. As we have introduced in Chapter 4, for this setting, we can construct a value-iteration-like algorithm that converges to an Equilibrium policy, under the assumption that the admissible policies are only the Markovian ones. The interaction between the two agents proceeds in episodes, where at the beginning of each one, the agents decide what policy to play. We indicate with K the number of episodes played by the two agents.

We define as $\pi_1 = (\pi_{1,1}, \dots, \pi_{1,K})$ and $\pi_2 = (\pi_{2,1}, \dots, \pi_{2,K})$ the two sequences of policies that, respectively, are played by our agent (agent 1) and the other agent (agent 2). Moreover, we denote with $V_{1,h}^{\pi_1,\pi_2}(s)$ and $V_{2,h}^{\pi_1,\pi_2}(s)$ the value functions of agent 1 and agent 2 in the state $s \in \mathcal{S}$ at time step h and with $V_1^{\pi_1,\pi_2} = \mathbb{E}_{s\sim\mu}[V_{1,1}^{\pi_1,\pi_2}(s)]$ and $V_2^{\pi_1,\pi_2} = \mathbb{E}_{s\sim\mu}[V_{2,1}^{\pi_1,\pi_2}(s)]$ the expected returns for the two agents.

We do not know the agent 2 policies π_2 a priori. and our goal is to learn a sequence of policies π_1 which minimizes the total (expected) *regret*, which is defined by:

$$\mathbb{E}[\text{Regret}(K)] = \sum_{k=1}^{K} V_1^\star - V_1^{\pi_{1,k},\pi_{2,k}}, \tag{8.1}$$

where $V_1^\star$ corresponds to the benchmarks couple of policies $\pi_1^\star, \pi_2^\star$ used to compare our algorithm. In literature there are common ways of defining $V_1^\star$ [Xie et al., 2020a]. The most used is the minmax policy defines as:

$$V^{\text{minmax}} = \min_{\pi_2\in\Pi_2} \max_{\pi_1\in\Pi_1} V_1^{\pi_1,\pi_2}, \tag{8.2}$$

which corresponds to play in a zero-sum games.

The second is the Stackelberg Equilibrium of the game:

$$V^{\mathrm{SE}} = \max_{\pi_1 \in \Pi_1} V_1^{\pi_1, \mathrm{br}(\pi_1)}, \tag{8.3}$$

where $\mathrm{br} : \Pi_1 \to \Pi_2$ corresponds to a function that select a best response policy for the agent 2 to each policy of the agent 1.

While the first way of defining the benchmark policy is suitable to account for adversarial settings, where the other agent can change adversarially its policy to maximize the regret, or we are in a zero-sum game, it cannot be used in a general-sum setting where the agent 2 only wants to maximize its own reward function. In fact, in the latter setting, agent 1 could hope to achieve better performance than when facing an adversarial opponent. However, it was shown that in some cases, like the one where we cannot observe anything of the other agent interactions, it is necessary to adopt the minimax benchmark since, otherwise, the regret minimization problem would be too difficult [Tian et al., 2020].

On the other hand, in many cases, we can observe the other agent interactions with the environment, and in other situations, it is possible that we retrieve its rewards (for example, using IRL approaches). In these problems, it is more reasonable to use the more challenging Stackelberg Equilibrium benchmark.

8.2 Related works

After the introduction of the concept of Stochastic Games (aka Markov Games) [Shapley, 1953], many RL algorithms were proposed to learn in this setting. Some of them were discussed in the preliminary Chapter 4.

However, the theoretical study of this setting is quite poor, compared to the empirical one (see the survey [Zhang et al., 2019a, Da Silva and Costa, 2019, Hernandez-Leal et al., 2019, Papoudakis et al., 2019]). Only in the last years there has been a growing interest in providing algorithms with strong sample-complexity and regret guarantees for the two theoretical MARL settings that we introduced in Section 8.1: the *offline* and *online* ones. In this section, we review the algorithms that were proposed to address these two settings.

Offline setting The majority of the works in theoretical MARL provide results in the zero-sum offline setting. In the offline zero-sum setting both model-free [Bai et al., 2020, Zhang et al., 2020c] and model-based [Bai and Jin, 2020, Sidford et al., 2020, Li et al., 2020, Liu et al., 2020, Zhang et al., 2020b] algorithms were proposed with near-optimal sample complexity and regret guarantees. For the model-based setting, the prevalent approach is to assume to have access to a generative model [Sidford et al., 2020, Zhang et al., 2020b] where the authors provide non-asymptotic results on the number of query to the generator. However, in [Liu et al., 2020] the authors proposed a model-based algorithm for the zero-sum setting without access to a generative model and that matches the information-theoretic lower bound. Moreover, in this recent work, the authors also proposed the first line of provably sample-efficient algorithms for multi-player general-sum games. In [Li et al., 2020] the authors introduced, instead, an algorithm to learn a Nash Equilibrium in the multi-player general-sum setting. Very recently, the first algorithm to deal with sample complexity in the general-sum games that achieves an ϵ-Stackleberg Equilibrium was introduced [Bai et al., 2021]. In this paper the authors consider the *bandit feedback* setting i.e., they can see only the random samples of the rewards received by the two players. In the paper it is identified a fundamental gap between the exact value of the Stackelberg equilibrium and its estimated version using finite samples. This result gives insights into the hardness of learning in General-sum games also when the setting is state-less and the algorithm has the control of the leader and the follower.

Online setting The online setting is only studied, as far as we know, in the zero-sum setting. The first work that analyzes the problem of online learning in Stochastic Games is [Brafman and Tennenholtz, 2002]. In this paper the authors propose the famous R-MAX algorithm that deals with the zero-sum average-reward setting and provides the first regret bound for the setting. In [Wei et al., 2017] the authors provide an algorithm for zero-sum Stochastic games that extends UCRL2, but that works under strong reachability assumptions. This algorithm significantly improve the R-MAX regret bound. [Xie et al., 2020a] propose an algorithm with a "weak" regret notion (the minimax defined before) which is compatible with a zero-sum game, using function approximation. This work is analyze under the finite-horizon setting and it achieves near-optimal regret bounds. Instead, [Tian et al., 2020] introduce the online setting with bandit-feedback. In this setting the agent cannot observe any interaction between the adversarial agent and the environment. The authors extend the method of [Bai et al., 2020] to deal with this setting. In [Xie et al., 2020a] the authors leave as open question how to construct an algorithm to achieve optimal regret exploiting a "weak opponent", i.e., an opponent that is not totally adversarial (as in zero-sum games). We address this open question in Chapter 10.

Adversarial MDPs The adversarial MDP problem is strictly related to the Stochastic Game setting. Most of the works in this setting consider adversarial rewards [Even-Dar et al., 2009, Gergely Neu et al., 2010, Zimin and Neu, 2013, Dick et al., 2014, Rosenberg and Mansour, 2019, Jin et al., 2020a]. i.e. the presence of an adversary that can change the received rewards. Obviously this setting is quite different from the Stochastic Games, since the adversary can affect only the rewards and not the transitions model. There are also some works that consider adversarial transitions [Yu and Mannor, 2009, Neu et al., 2012, Lykouris et al., 2019]. We can apply these approaches in the bandit setting where we cannot see any feedback from the other agent, with the scope of constructing an algorithm robust to these perturbations.

CHAPTER 9

Non-Cooperative Configurable Markov Decision Processess

In this chapter, we solve an online learning problem in the Configurable Markov Decision process framework, which involves two entities, the configurator and the agent.

As we have explained in the previous chapters, the standard RL framework concerns an agent whose objective is to maximize the reward collected during its interaction with the environment. However, there are real-world scenarios in which the agent itself or an external supervisor (configurator) can *partially* modify the environment. For example, in an autonomous driving scenario, it is possible to alter the car setup to suit its needs better. Recently, the Configurable Markov Decision Processes [Metelli et al., 2018, Conf-MDPs] were introduced to model these scenarios and analyze the interactions between the agent and the configurator.

Originally, solving a Conf-MDP consists of simultaneously optimizing a set of environmental parameters and the agent's policy to reach the maximum expected return. In many scenarios, however, the configurator does not know the agent's reward and its intention differs from that of the agent, leading to new appealing schemes. For instance, imagine we are the owner of a supermarket, and we have to decide how to arrange the products on the shelves. We intend to increase the company's final profit; instead, a customer aims to spend the shortest time possible inside the supermarket and only buy the indispensable products. Since we do not know the customer's reward function, the only possibility is to try different dispositions and see their reactions. Nevertheless, what if we knew what buyers are most interested in? In this case, we can *strategically* decide how to position other products close to the popular ones to induce the customer in a more profitable behavior.

In this chapter, we introduce the Non-Cooperative Configurable Markov Decision Processes (NConf-MDP), a new framework that handles the possibility of having different reward functions for the agent and the configurator. While a Conf-MDP assumes that the configurator acts to help the agent to optimize its expected reward, an NConf-MDP, instead, allows modeling a more extensive set of scenarios, including all the cases in which agent and configurator display a non-cooperative behavior, modeled by the individual reward functions. Obviously, this setting cannot be solved with straightforward application of the algorithms designed for Conf-MDPs that focus on the case in which both entities share the same interests. In fact, if the configurator and the agent optimize separately different objectives, they might not converge to an equilibrium strategy [Mertikopoulos et al., 2018b, Papadimitriou and Piliouras, 2016]. Moreover, accounting for the agent's interests would be, as in Stochastic games, advantageous for the configurator [Hu and Wellman, 2003].

In this novel setting, we consider an online learning problem, where the configurator has to select a configuration, within a finite set of possible configurations, in order to maximize its own return. This framework can be seen as a *leader-follower* game, in which the *follower* (the agent) is selfish and optimizes its own reward function, and the *leader* (the configurator) has to decide the best configuration w.r.t. the best response of the agent. Clearly, to adapt its decisions, the configurator has to receive some form of feedback related to the agent's behavior. In this chapter, we analyze two settings based on whether the configurator observes just the agent's actions or also a noisy version of the agent's reward function. For the two settings, we propose algorithms based on the Optimism in the Face of Uncertainty [Auer et al., 2002, OFU] principle.

- We extend the Configurable Markov Decision Process setting to deal with situations where the configurator and the agent have different reward functions. We call this new framework the Non-Cooperative Markov Decision Process (Section 9.2).

- We provide an algorithm, Action-feedback Optimistic Configuration Learning (AfOCL), which maximizes the configurator's performance under the assumption that it observes the agent's actions only (Section 9.4). We show AfOCL achieves finite expected regret, scaling linearly with the number of admissible configurations. Moreover, as far as we know, we provide the first problem-dependent regret analysis in Multi-Agent RL.
- Then, we introduce an algorithm Reward-feedback Optimistic Configuration Learning (RfOCL) that assumes to observe a noisy version of the agent's reward in addition to the agent's actions (Section 9.5). We prove that, under suitable conditions, it is possible to further exploit the *structure* underlying the decision process, removing the dependence on the number of configurations. This is the first proof that takes into account the suboptimality gap of the controllable agent (the configurator) as well as the suboptimality gaps of the uncontrollable agent (the RL agent).
- Finally, we provide an experimental evaluation on benchmark domains, inspired by the motivational scenarios of NConf-MDPs (Section 9.8).

9.1 Configurable MDPs

We start by introducing the Configurable Markov Decision Process (Conf-MDPs).

Definition 9.1.1 (Configurable MDP [Metelli et al., 2018]). *A Configurable Markov Decision Process (Conf-MDP) is defined as $\mathcal{CM} = (\mathcal{S}, \mathcal{A}, \mathfrak{P}, \gamma, \mu, \mathcal{R}, H)$[1], where $(\mathcal{S}, \mathcal{A}, \gamma, \mu, \mathcal{R}, H)$ is an MDP without transition function, and $\mathfrak{P}$ is a configuration space.*

A Conf-MDP extends the MDP concept by considering a configuration space $\mathfrak{P}$ (i.e., a set of transition models) instead of a single transition model $\mathcal{P}$. This allows us to model situations in which we can modify some environmental parameters to optimize the expected discounted return.

The Q-value of a policy $\pi \in \Pi$ and configuration $\mathcal{P} \in \mathfrak{P}$ is the expected sum of the rewards starting from $(s, a) \in \mathcal{S} \times \mathcal{A}$ at step $h \in [H]$:

$$Q_h^{\pi,\mathcal{P}}(s,a) = \mathcal{R}(s) + \mathop{\mathbb{E}}_{s_{h'} \sim \mathcal{P},\pi} \left[\sum_{h'=h+1}^{H} \mathcal{R}(s_{h'}) | s_h = s, a_h = a \right],$$

[1]In this chapter we consider rewards which are function of the state only.

having denoted with $\mathbb{E}_{s_{h'} \sim \mathcal{P}, \pi}$ the expectation w.r.t. the distribution $\mathcal{P}(\cdot|s_{h'-1}, \pi_{h'-1}(s_{h'-1}))$. The value function is given by $V_h^{\pi,\mathcal{P}}(s) = Q_h^{\pi,\mathcal{P}}(s, \pi_h(s))$ and the expected return is defined as $V^{\pi,\mathcal{P}} = \mathbb{E}_{s\sim\mu}[V_1^{\pi,\mathcal{P}}(s)]$. In a Conf-MDP the goal consists in finding a policy π^* together with an environment configuration $\mathcal{P}^*$ so as to maximize the expected return, i.e.

$$(\pi^*, \mathcal{P}^*) \in \operatorname*{arg\,max}_{\pi\in\Pi, \mathcal{P}\in\mathfrak{P}} V^{\pi,\mathcal{P}}.$$

9.2 Non-Cooperative Configurable MDPs

The definition of Conf-MDP (see section 9.1) allows modeling scenarios in which agent and configurator share the same objective, encoded in a single reward function $\mathcal{R}$. In this section, we introduce an extension of this framework to account for the presence of a configurator having interests that might differ from those of the agent.

Definition 9.2.1 (Non-Cooperative Configurable MDP)**.** *A Non-Cooperative Configurable Markov Decision Process (NConf-MDP) is defined by a tuple* $\mathcal{NCM} = (\mathcal{S}, \mathcal{A}, \mathfrak{P}, \mu, \mathcal{R}_c, \mathcal{R}_o, \gamma_c, \gamma_o, H)$, *where* $(\mathcal{S}, \mathcal{A}, \mathfrak{P}, \mu, H)$ *is a Conf-MDP without reward and discount factors,* $\mathcal{R}_c, \mathcal{R}_o : \mathcal{S} \to [0, 1]$ *are the configurator and agent (opponent) reward functions, and* $\gamma_c, \gamma_o \in [0, 1]$ *are the configurator and agent discount factors.*

In this chapter we assume that the discount factors are equal to 1, since we consider the finite-horizon (episodic) setting. Moreover, we consider finite state and action spaces. Given a policy $\pi \in \Pi$ and a configuration $\mathcal{P} \in \mathfrak{P}$, for every $(s, a) \in \mathcal{S} \times \mathcal{A}$ and $h \in [H]$ we define the configurator and agent Q-values as:

$$Q_{c,h}^{\pi,\mathcal{P}}(s, a) = \mathcal{R}_c(s) + \mathop{\mathbb{E}}_{s_{h'}\sim\mathcal{P}, a_{h'}\sim\pi} \left[\sum_{h'=h+1}^{H} \mathcal{R}_c(s_{h'}) | s_h = s, a_h = a \right],$$

$$Q_{o,h}^{\pi,\mathcal{P}}(s, a) = \mathcal{R}_o(s) + \mathop{\mathbb{E}}_{s_{h'}\sim\mathcal{P}, a_{h'}\sim\pi} \left[\sum_{h'=h+1}^{H} \mathcal{R}_o(s_{h'}) | s_h = s, a_h = a \right].$$

We denote with

$$V_{c,h}^{\pi,\mathcal{P}}(s) = Q_{c,h}^{\pi,\mathcal{P}}(s, \pi_h(s)), \quad V_{o,h}^{\pi,\mathcal{P}} = Q_{o,h}^{\pi,\mathcal{P}}(s, \pi_h(s)),$$

the value functions and with

$$V_c^{\pi,\mathcal{P}} = \mathop{\mathbb{E}}_{s\sim\mu}[V_{c,1}^{\pi,\mathcal{P}}(s)], \quad V_o^{\pi,\mathcal{P}} = \mathop{\mathbb{E}}_{s\sim\mu}[V_{o,1}^{\pi,\mathcal{P}}(s)],$$

the expected returns for the configurator and the agent respectively.

Connections with Robus RL There are some natural connections between the Robust RL [Morimoto and Doya, 2005] problem and Non-cooperative Conf-MDPs. The intent of Robust RL is to learn policies that are *robust* to transition model shifts. The literature regarding Robust control problems is wide [Nilim and El Ghaoui, 2005, Mannor et al., 2012, Lim et al., 2013, Mannor et al., 2016, Tessler et al., 2019]. Many works tract the problem considering an adversary that can change the environment transitions during the learning process, leading to a min-max optimization problem. However, we have to notice that this setting is quite different from Non-cooperative Conf-MDPs, where the reward functions are not the necessary opposite. However, the main difference between the NConf-MDP and Robust MDP problem is in the control that we have on the entities: in NConf-MDPs, we can control the configurator and, in some cases also the agent.

9.3 Problem Formulation

While for classical Conf-MDPs [Metelli et al., 2018, Metelli et al., 2019a] a notion of optimality is straightforward as agent and configurator share the same objective, in an NConf-MDP, they can display possibly conflicting interests. We assume a *sequential* interaction between the configurator and the agent that resembles the leader-follower protocol [Breton et al., 1988, Balcan et al., 2015, Peng et al., 2019]. First, the configurator (leader) selects an environment configuration $\mathcal{P} \in \mathfrak{P}$ and then the agent (follower) plays a best response policy $\pi^*_{\mathcal{P}} \in \Pi$, i.e., an optimal policy for the MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu, \mathcal{R}_o, H)$:

$$\pi^*_{\mathcal{P}} \in \operatorname*{arg\,max}_{\pi \in \Pi^H_D} V_o^{\pi, \mathcal{P}}.$$

Before proceeding we make the following assumption.

Assumption 9.3.1. *For every environment configuration $\mathcal{P} \in \mathfrak{P}$, the agent will always play the same best response policy $\pi^*_{\mathcal{P}}$. Furthermore, $\pi^*_{\mathcal{P}}$ is deterministic.*

We assume that the agent will react with the same optimal policy $\pi^*_{\mathcal{P}}$ whenever facing configuration $\mathcal{P}$. This is a common assumption in standard Stackelberg Games [Balcan et al., 2015, Peng et al., 2019, Sessa et al., 2020]. Moreover, we require that the best response is a deterministic policy. This assumption is justified since, for every MDP, there exists at least one deterministic optimal policy [Sutton et al., 1998, Puterman, 2014].

Under Assumption 9.3.1, the goal of the configurator is well-defined and consists in finding a configuration $\mathcal{P}^* \in \mathfrak{P}$ that is optimal under the agent's best response policy:

$$\mathcal{P}^* \in \arg\max_{\mathcal{P}\in\mathfrak{P}} V_c^{\pi^*_{\mathcal{P}},\mathcal{P}}.$$

From a game theoretic perspective, the pair $(\mathcal{P}^*, \pi^*_{\mathcal{P}^*})$ can be regarded as a Stackelberg equilibrium of the corresponding game [Breton et al., 1988].

The configurator knows everything about the NConf-MDP, except for the agent reward function $\mathcal{R}_o$. At each episode $k \in [K]$, the configurator selects a configuration $\mathcal{P}_k \in \mathfrak{P}$ and observes a trajectory of H steps generated by the agent's best response policy $\pi^*_{\mathcal{P}_k}$. In this chapter we study two types of feedback from the agent:

- *Action-feedback* (Af). The configurator observes the states and the actions played by the agent $(s_1, a_1, \dots, s_{H-1}, a_{H-1}, s_H)$, where $a_h = \pi^*_{\mathcal{P}_k,h}(s_h)$.
- *Reward-feedback* (Rf). The configurator observes the states, the actions played by the agent, and a noisy feedback of the agent reward function $(s_1, \widetilde{r}_1, a_1, \dots, s_{H-1}, \widetilde{r}_{H-1}, a_{H-1}, s_H, \widetilde{r}_H)$, where $a_h \sim \pi^*_{p_k,h}(s_h)$ and $\widetilde{r}_h$ is sampled from a distribution with mean $\mathcal{R}_o(s_h)$ and support $[0,1]$.[2]

The Rf tries to model situations in which the agent's reward is either known under uncertainty or it is obtained in an approximate way through IRL [Osa et al., 2018].

From an online learning perspective, the goal of the configurator is to minimize the expected regret:

$$\mathbb{E}[\text{Regret}(K)] = \mathbb{E}\left[\sum_{k=1}^{K} \max_{\mathcal{P}\in\mathfrak{P}} V_c^{\pi^*_{\mathcal{P}},\mathcal{P}} - V_c^{\pi^*_{\mathcal{P}_k},\mathcal{P}_k}\right]. \tag{9.1}$$

We assume that the configuration space $\mathfrak{P}$ is a finite set made of M stochastic transition models $\mathfrak{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$. To lighten the notation, in the following, we will denote with π_i the agent's best response policy to the configuration $\mathcal{P}_i$, i.e., $\pi^*_{\mathcal{P}_i}$ and with V^i the configurator expected return attained with configuration $\mathcal{P}_i$ and $\pi^*_{\mathcal{P}_i}$, i.e., $V_c^{\pi^*_{\mathcal{P}_i},\mathcal{P}_i}$. Finally, we denote with $V^* = \max_{i\in[M]} V^i$. This setting is quite similar to the one presenting in the previous chapter where, instead of a configurator and an agent, we have two agents. As in the setting defined in Chapter 8 we can control only one of the entities of the game (in this case the configurator).

[2]Clearly, the results we present can be directly extended to subgaussian distributions on the reward.

On the optimality of the agent's policy In our setting, we assume that the agent's policy at every episode is an optimal policy. It might be argued that the agent, whenever experiencing a modification of the environment configuration, needs some time to adjust its policy before reaching optimality. However, in real-world situations, environment configuration and agent learning typically happen on different time scales. Indeed, the configuration changes slowly, giving the agent the time to converge to an optimal policy. For instance, in the supermarket example, the time interval between two changes of product disposition might be more than one month; instead, a buyer takes less time (few visits) to learn the disposition and their best policy.

In the next section, we present two algorithms for the online learning problem introduced above. The first algorithm uses only the collected agent decisions to optimistically learn the best configuration (Section 9.4). In the second algorithm, we also use the noisy reward feedback to construct an algorithm that leverages the structure that links together all the transition probability models: the agent's reward function $\mathcal{R}_o$. We show that, under suitable assumptions, the regret of the second algorithm removes the dependencies on the number of configurations (Section 9.5).

9.4 Action-feedback Optimistic Configuration Learning

We start with the action-feedback (Af) setting in which the configurator observes the agent's actions only. The idea at the basis of the algorithm we propose, *Action-feedback Optimistic Configuration Learning* (AfOCL), is to maintain, for each configuration, a set of *plausible* policies that contains the agent's best response policy. The configurator plays the transition model that maximizes an optimistic approximation of its value function. Specifically, for every $i \in [M]$, $k \in [K]$, and $h \in [H]$ we denote with $\mathcal{A}^i_{k,h}(s) \subseteq \mathcal{A}$ the set of plausible actions in state s at step h for configuration $\mathcal{P}_i$ at the beginning of episode k. Since the agent's best response policy π_i is deterministic, if state s is visited at step h before episode k, we know the agent's action in the current model $\mathcal{P}_i$ and therefore we set $\mathcal{A}^i_{k,h}(s) = \{\pi_{i,h}(s)\}$, otherwise we have no knowledge and we set $\mathcal{A}^i_{k,h}(s) = \mathcal{A}$.

Based on this, we can compute an optimistic approximation $\widetilde{V}^i_{k,h}$ of the configurator value function V^i_h:

$$\widetilde{V}^i_{k,h}(s) = \mathcal{R}_c(s) + \max_{a \in \mathcal{A}^i_{k,h}(s)} \sum_{s' \in \mathcal{S}} p_i(s'|s,a)\widetilde{V}^i_{k,h+1}(s'), \tag{9.2}$$

Algorithm 11 Action-feedback Optimistic Configuration Learning (AfOCL).

```
1: Input: $\mathcal{S}$, $\mathcal{A}$, $H$, $\mathfrak{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$
2: Initialize $\mathcal{A}^i_{1,h}(s) = \mathcal{A}$ for all $s \in \mathcal{S}$, $h \in [H]$, and $i \in [M]$
3: for episodes $1, 2, \dots, K$ do
4:     Compute $\widetilde{V}^i_k$ for all $i \in [M]$
5:     Play $\mathcal{P}_{I_k}$ with $I_k \in \arg\max_{i \in [M]} \widetilde{V}^i_k$
6:     Observe $(s_{k,1}, a_{k,1}, \dots, s_{k,H-1}, a_{k,H-1}, s_{k,H})$
7:     Compute the plausible actions for all $s \in \mathcal{S}$ and $h \in [H]$:
           $\mathcal{A}^i_{k+1,h}(s) = \begin{cases} \{a_{k,h}\} & \text{if } i = I_k \text{ and } s = s_{k,h} \\ \mathcal{A}^i_{k,h}(s) & \text{otherwise} \end{cases}$
8: end for
```

and $\widetilde{V}^i_{k,H}(s) = \mathcal{R}_c(s)$. For visited pairs (s, h) the maximization over the actions reduces to the evaluation of the transition model in the agent's action $\pi_{i,h}(s)$. Clearly, we have that $\widetilde{V}^i_{k,h}(s) \geq V^i_h(s)$ for all $s \in \mathcal{S}$, $h \in [H]$, and $i \in [M]$. Thus, at each episode $k \in [K]$ the configurator plays the transition model $\mathcal{P}_{I_k}$ maximizing the optimistic approximation $\widetilde{V}^i_k$:

$$I_k \in \arg\max_{i \in [M]} \widetilde{V}^i_k.$$

The pseudocode of AfOCL is reported in Algorithm 11. The computation of the optimistic approximation $\widetilde{V}^i_{k,h}$ can be simply performed applying a value-iteration-like algorithm [Puterman, 2014] that employs the iterate as in Equation (9.2). Notice that the computational complexity decreases as the number of visited states increases and, in any case, is bounded by that of value iteration $\mathcal{O}(HS^2A)$. Therefore, the time complexity of AfOCL is $\mathcal{O}(KMHS^2A)$.

9.4.1 Regret Guarantees

In this section, we provide an expected regret bound for the AfOCL algorithm. Since the policy is deterministic, in every episode $k \in [K]$, we acquire the information about which action the agent plays, in the chosen model $\mathcal{P}_{I_k}$, for every visited state. So the main effort is to estimate the agent's policies for every model. In fact, after that, the algorithm will be able to compute the correct expected return for each transition model. However, due to the models' stochasticity $\mathcal{P}_i$ for $i \in [M]$, some states might be visited with low frequency. The following result exploits the determinism of the agent's best response policy to prove that the regret AfOCL suffers is constant, independent on the number of episodes K.

Theorem 9.4.1 (Regret of AfOCL). *Let $\mathcal{NCM} = (\mathcal{S}, \mathcal{A}, \mathfrak{P}, \mu, \mathcal{R}_c, \mathcal{R}_o, H)$ with $\mathfrak{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$ be the M finite-horizon MDPs of the problem. The expected regret of AfOCL at every episode $K > 0$ is bounded by:*

$$\mathbb{E}[\text{Regret}(K)] \leq 3MH^3S^2. \tag{9.3}$$

Discussion on the AfOCL regret bound The result might be surprising as the regret is constant and independent of the suboptimality gaps between the configurations, i.e., $\Delta_i = V^* - V^i$ for every $i \in [M]$. As supported by intuition, we need to spend more time to discard MDPs that are more similar in performance to the optimal one. Formally, the maximum number of times a suboptimal configuration $\mathcal{P}_i$ is played is proportional to $\frac{1}{\Delta_i}$ (and not proportional to $\frac{1}{\Delta_i^2}$ as in standard bandits). This is because the policies are deterministic and, to learn them, we just need *one* visit to the state. Then, since playing the sub-optimal i-th configuration costs exactly Δ_i immediate expected regret, we have that the toal regret does not scale with Δ_i.

Proof To prove the result of Theorem 9.4.1 we start by bounding the difference between the optimistic expected discounted return $\widetilde{V}_k^i$ and the true expected return V^i for every configuration $\mathcal{P}_i \in \mathfrak{P}$. We define with $d_h^i(s)$ the visitation probability of state s at step h under the transition model $\mathcal{P}_i$.

Lemma 9.4.1. *For every episode $k \in [K]$ and configuration $\mathcal{P}_i \in \mathfrak{P}$, the difference between the optimistic expected return $\widetilde{V}_k^i$ and the true expected return V^i is bounded by:*

$$\widetilde{V}_k^i - V^i \leq 2H \sum_{s \in \mathcal{S}} \sum_{h=1}^{H-1} d_h^i(s) \mathbb{1}\left\{N_{k,h}^i(s) = 0\right\}. \tag{9.4}$$

where $N_{k,h}^i(s)$ is the number of times the state $s \in \mathcal{S}$ is visited at step $h \in [H]$ with the configuration $\mathcal{P}_i \in \mathfrak{P}$ up to episode $k - 1$.

Proof. First of all, we denote with $\widetilde{d}_h^i(s)$ the visitation probability of visiting state s at step h under transition model $\mathcal{P}_i$ and playing the estimated agent's best response policy $\widetilde{\pi}_{i,k}$ (we will omit the subscript k in the following). Moreover, the visitation probabilities satisfy the following equalities for all $h \geq 2$:

$$\begin{aligned} d_h^i(s) &= \sum_{s' \in \mathcal{S}} \mathcal{P}_i(s|s', \pi_{i,h}(s')) d_{h-1}^i(s') \\ \widetilde{d}_h^i(s) &= \sum_{s' \in \mathcal{S}} \mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s')) \widetilde{d}_{h-1}^i(s'), \end{aligned} \tag{9.5}$$

with $d^i_h(s) = \widetilde{d}^i_h(s) = \mu(s)$. Thus, we have:

$$\begin{aligned}
\widetilde{V}^i_k - V^i &= \sum_{s\in\mathcal{S}} \left[\mu(s)\mathcal{R}(s) - \mu(s)\mathcal{R}(s) + \sum_{h=2}^{H} (\widetilde{d}^i_h(s) - d^i_h(s))\mathcal{R}(s) \right] && (9.6)\\
&\le \sum_{s\in\mathcal{S}} \sum_{h=2}^{H} \left| \widetilde{d}^i_h(s) - d^i_h(s) \right| && (9.7)\\
&= \sum_{s\in\mathcal{S}} \sum_{h=1}^{H-1} \left| \sum_{s'\in\mathcal{S}} \widetilde{d}^i_h(s')\mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s')) - d^i_h(s')\mathcal{P}_i(s|s', \pi_{i,h}(s')) \right| && (9.8)\\
&= \sum_{s\in\mathcal{S}} \sum_{h=1}^{H-1} \sum_{s'\in\mathcal{S}} \left| \widetilde{d}^i_h(s') - d^i_h(s') \right| \mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s'))\\
&\quad + d^i_h(s') \left| \mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s')) - \mathcal{P}_i(s|s', \pi_{i,h}(s')) \right|\\
&= \sum_{s'\in\mathcal{S}} \sum_{h=2}^{H-1} \left| \widetilde{d}^i_h(s') - d^i_h(s') \right|\\
&\quad + \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{h=1}^{H-1} d^i_h(s') \left| \mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s')) - \mathcal{P}_i(s|s', \pi_{i,h}(s')) \right| && (9.9)\\
&= \sum_{H'=2}^{H} \sum_{s\in\mathcal{S}} \sum_{s'\in\mathcal{S}} \sum_{h=1}^{H'-1} d^i_h(s') \left| \mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s')) - \mathcal{P}_i(s|s', \pi_{i,h}(s')) \right| && (9.10)\\
&\le H \sum_{s'\in\mathcal{S}} \sum_{h=1}^{H-1} d^i_h(s') \sum_{s\in\mathcal{S}} \left| \mathcal{P}_i(s|s', \widetilde{\pi}_{i,h}(s')) - \mathcal{P}_i(s|s', \pi_{i,h}(s')) \right| && (9.11)\\
&\le 2H \sum_{s'\in\mathcal{S}} \sum_{h=1}^{H-1} \mathbb{1}\left\{ N^i_{k,h}(s) = 0 \right\} d^i_h(s'), && (9.12)
\end{aligned}$$

where in line (9.6) we use the definition of expected return. In line (9.7) we bound the value of every reward with its maximum value 1. In line (9.8) we expanded the probability distribution of visiting states using Equations (9.5). In line (9.9) we observe that $\widetilde{d}^i_1(s') - d^i_1(s') = \mu(s) - \mu(s) = 0$ to make the first summation starting from $h = 2$. In line (9.10), we apply the recursion with line (9.7). In line (9.11), we bound $H' \le H$ and observe that the outer summation has less than H terms. Finally, in line (9.12) we upper bound the differences between the two probabilities with 2, and we use the fact that when we have seen a state s at step h with a configuration $\mathcal{P}_i$ we have learned its policy (that is deterministic). □

Given the previous lemma we can now assess when the configurator stops playing a suboptimal configuration $\mathcal{P}_i \neq \mathcal{P}^*$. Then we will use this stopping criteria to derive our regret bound.

Lemma 9.4.2. *A configuration $\mathcal{P}_i \in \mathfrak{P}$ is no longer played after episode $k \in [K]$ if for every state $s \in \mathcal{S}$ and $h \in [H]$, with $d^i_h(s) \ge \frac{\Delta_i - c}{2H^2S}$, we have $N^i_{k,h}(s) > 0$, where $c > 0$ is arbitrary and $\Delta_i = V^* - V^i$.*

Proof. It suffices to prove that the optimistic expected return satisfies $\widetilde{V}_k^i < V^*$, that, in turn, will satisfy $V^* \leq \widetilde{V}_k^{i^*}$ where $i^* \in \arg\max_{i \in [M]} V^i$ (this way configuration i will no longer be played):

$$\begin{aligned} \widetilde{V}_k^i &= V^i + \widetilde{V}_k^i - V^i \\ &\leq V^i + 2H \sum_{s \in \mathcal{S}} \sum_{h=1}^{H-2} d_h^i(s) \mathbb{1}\left\{N_{h,k}^i(s) = 0\right\} && (9.13) \\ &\leq V^i + 2H^2 S \frac{\Delta_i - c}{2H^2 S} && (9.14) \\ &= V^i + \Delta_i - c < V^*, && (9.15) \end{aligned}$$

where in line (9.13) we apply Lemma 9.4.1. In line (9.14) we bound the state visitation probabilities of the (s, h) pairs with $N_{h,k}^i(s) > 0$ with their maximum value, as in the statement hypothesis. In line (9.15) we use the fact that $\Delta_i = V^* - V_i$. □

Having defined in the above lemma 9.4.2 how many times the configurator has to play suboptimal configurations, we can now derive the regret of the proposed algorithm.

Theorem 9.4.1 (Regret of AfOCL). *Let $\mathcal{NCM} = (\mathcal{S}, \mathcal{A}, \mathfrak{P}, \mu, \mathcal{R}_c, \mathcal{R}_o, H)$ with $\mathfrak{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$ be the M finite-horizon MDPs of the problem. The expected regret of AfOCL at every episode $K > 0$ is bounded by:*

$$\mathbb{E}[\text{Regret}(K)] \leq 3MH^3S^2. \tag{9.3}$$

Proof. We rephrase the regret as:

$$\mathbb{E}[\text{Regret}(K)] = \sum_{i \in [M]: \Delta_i > 0} \Delta_i \, \mathbb{E}[N_i],$$

where N_i is the number of times that the algorithm plays model $\mathcal{P}_i$ which is not the optimal configuration $\mathcal{P}_{i^*}$. We start bounding for every configuration $\mathcal{P}_i$ s.t. $\Delta_i > 0$ the expected value of N_i. We denote with k_l^i the round at which model i is selected for the l-th time:

$$\begin{aligned} \mathbb{E}[N_i] &\leq \sum_{l=0}^{K} \Pr(N_i \geq l) \\ &\leq \sum_{l=0}^{\infty} \Pr(N_i \geq l) && (9.16) \\ &\leq \sum_{l=0}^{\infty} \Pr\left(\widetilde{V}_{k_l^i}^i - V^* \geq 0\right), && (9.17) \\ & && (9.18) \end{aligned}$$

where in line (9.16) we extend the sum to ∞. In line (9.17) we exploit the fact that if configuration i is selected then it must be $\widetilde{V}_{k_l^i}^i \geq \widetilde{V}_{k_l^i}^{i^*}$ and, because of optimism $\widetilde{V}_{k_l^i}^{i^*} \geq V^*$. Then, we observe that for Lemma 9.4.2, if configuration i is played at time k_l^i, then there must exists $s \in \mathcal{S}$ and $h \in [H]$

with $d_h^i(s) \geq \frac{\Delta_i - c}{2H^2S}$ that is not played yet. Formally:

$$\begin{aligned}
\mathbb{E}[N_i] &\leq \sum_{l=0}^{\infty} \Pr\left(\widetilde{V}^i_{k_l^i} - V^* \geq 0\right) \\
&\leq 1 + \sum_{l=1}^{\infty} \Pr\left(\exists s \in \mathcal{S},\, \exists h \in [H] \text{ s.t. } d_h^i(s) \geq \frac{\Delta_i - c}{2H^2S} \,:\, N^i_{k_l^i,h}(s) = 0\right) \qquad (9.19)\\
&\leq 1 + \sum_{l=1}^{\infty} \sum_{s\in\mathcal{S},h\in[H]:d_h^i(s)\geq\frac{\Delta_i-c}{2H^2S}} \Pr\left(N^i_{k_l^i,h}(s) = 0\right) \qquad (9.20)\\
&\leq 1 + SH\sum_{l=1}^{\infty}\left(1 - \frac{\Delta_i - c}{2H^2S}\right)^{l-1} \qquad (9.21)\\
&= 1 + SH\frac{1}{\frac{\Delta_i-c}{2H^2S}} \leq 1 + \frac{2H^3S^2}{\Delta_i - c}, \qquad (9.22)
\end{aligned}$$

where, in line (9.19) we use Lemma 9.4.2. In line (9.20) we use the union bound over the set employed for existential quantification. In line (9.21) we bound the probability as $\Pr\left(N^i_{k_l^i,h}(s) = 0\right) = (1 - d_h^i(s))^{l-1}$, thanks to the independence of the rounds. In line (9.22) we use the geometric series properties.

So the expected regret is bounded by:

$$\mathbb{E}[\text{Regret}(K)] = \sum_{i\in[M]:\Delta_i>0} \Delta_i\, \mathbb{E}[N_i] \leq \sum_{i\in[M]:\Delta_i>0} \Delta_i \left(\frac{2H^3S^2}{\Delta_i - c} + 1\right) \leq 3MH^3S^2,$$

having taken the infimum over $c > 0$. □

9.5 Reward-feedback Optimistic Configuration Learning

The main drawback of AfOCL is that every transition model is treated separately, preventing from employing the underlying *structure* of the environment. Such a structure is represented by the agent reward function $\mathcal{R}_o$, which is completely ignored in AfOCL. Indeed, if the configurator knew $\mathcal{R}_o$, it could find the optimal configuration with no need for interaction by simply computing the agent's best response policies.

The algorithm we propose in this section, *Reward-feedback Optimistic Configuration Learning* (RfOCL), employs the reward feedback (Rf), i.e., at every interaction, the configurator can see also a noisy version of the agent's reward function. The crucial point is that $\mathcal{R}_o$ is the same regardless of the chosen configuration, and, for this reason, it provides a link between them.

Specifically, for every $k \in [K]$ and $s \in \mathcal{S}$, RfOCL maintains a confidence interval for the agent reward function $\mathcal{R}_k(s) = [\underline{\mathcal{R}}_{o,k}(s), \overline{\mathcal{R}}_{o,k}(s)]$ obtained using the samples collected up to episode $k-1$ *regardless* of the played configuration. We apply Höeffding's concentration inequality to build the confidence intervals obtaining:

$$\widehat{\mathcal{R}}_{o,k}(s) \pm \sqrt{\frac{\log(2SHk^2)}{\max\{N_k(s),1\}}}, \tag{9.23}$$

where $N_k(s)$ is the number of visits of state s in the first $k-1$ episodes, and $\widehat{\mathcal{R}}_{o,k}(s)$ is the sample mean of the observed rewards for state s up to episode k.

Given the estimated reward, for every configuration $i \in [M]$, we can compute a confidence interval for the corresponding agent's Q-values $\mathcal{Q}_{k,h}(s,a) = [\underline{Q}^i_{o,k,h}(s,a), \overline{Q}^i_{o,k,h}(s,a)]$, by simply applying the Bellman equation:

$$\underline{Q}^i_{o,k,h}(s,a) = \underline{\mathcal{R}}_{o,k}(s) + \sum_{s' \in \mathcal{S}} \mathcal{P}_i(s'|s,a) \max_{a' \in \mathcal{A}} \underline{Q}^i_{o,k,h+1}(s',a'),$$
$$\overline{Q}^i_{o,k,h}(s,a) = \overline{\mathcal{R}}_{o,k}(s) + \sum_{s' \in \mathcal{S}} \mathcal{P}_i(s'|s,a) \max_{a' \in \mathcal{A}} \overline{Q}^i_{o,k,h+1}(s',a'),$$

and $\underline{Q}^i_{o,k,H}(s,a) = \underline{\mathcal{R}}_{o,k}(s)$ and $\overline{Q}^i_{o,k,H}(s,a) = \overline{\mathcal{R}}_{o,k}(s)$. If the true reward function belongs to the confidence interval, i.e., $\mathcal{R}_o \in \mathcal{R}_k$, then the true Q-value belongs to the corresponding confidence interval, i.e., $Q^i_h \in \mathcal{Q}_{k,h}$. Consequently, we can use $\mathcal{Q}_{k,h}$ to restrict the set of plausible actions in a state *without* actually observing the agent playing the action in that state. Indeed, the plausible actions are those that have a Q-value upper bound larger than the maximum Q-value lower bound:

$$\widetilde{\mathcal{A}}^i_{k,h}(s) = \left\{ a \in \mathcal{A} : \overline{Q}^i_{o,k,h}(s,a) \geq \max_{a' \in \mathcal{A}} \underline{Q}^i_{o,k,h}(s,a') \right\}. \tag{9.24}$$

In other words, if the upper Q-value of an action is smaller than the largest lower Q-value, it cannot be the greedy action, and it is discarded. Clearly, whenever we observe the agent playing an action in (s,h) we can reduce the plausible actions to the singleton $\{\pi_{i,h}(s)\}$, as in the action-feedback setting (Section 9.4). Based on this refined definition of plausible actions, we can compute the optimistic estimate $\widetilde{V}^i_{k,h}$ of the configurator value function V^i_h as in Equation (9.2) and proceed playing the optimistic configuration.

The pseudocode of RfOCL is reported in Algorithm 12. It is worth noting that we need to keep track of the states that have been already visited because for those, we know the agent's action, and there is no need to apply Equation (9.24). This is why we introduce the counts $N_{k,h}(s)$. The computational complexity of an individual iteration of RfOCL is dominated by the value iteration (steps 5 and 9) leading, as for AfOCL, to $\mathcal{O}(KMHS^2A)$.

Algorithm 12 Reward-feedback Optimistic Configuration Learning (RfOCL)

```
1: **Input:** $\mathcal{S}, \mathcal{A}, H, \mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$
2: Initialize $\mathcal{A}^i_{1,h}(s) = \mathcal{A}$ for all $s \in \mathcal{S}$, $h \in [H]$, and $i \in [M]$
3: Initialize $\overline{\mathcal{R}}_{o,1}(s) = 1$, $\underline{\mathcal{R}}_{o,1}(s) = 0$, and $N_{1,h}(s) = 0$ for all $s \in \mathcal{S}$ and $h \in [H]$
4: for episodes $1, 2, \dots, K$ do
5:     Compute $\widetilde{V}^i_k$ for all $i \in [M]$
6:     Play $\mathcal{P}_{I_k}$ with $I_k \in \arg\max_{i \in [M]} \widetilde{V}^i_k$
7:     Observe
       $(s_{k,1}, \widetilde{r}_{k,1}, a_{k,1}, \dots, s_{k,H-1}, \widetilde{r}_{k,H-1}, a_{k,H-1}, s_{k,H}, \widetilde{r}_{k,H})$
8:     Compute $\overline{\mathcal{R}}_{o,k+1}(s)$, $\underline{\mathcal{R}}_{o,k+1}(s)$, and $N_{k+1,h}(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$ using
       $\widetilde{r}_{k,1} \cdots \widetilde{r}_{k,H}$ as in Equation (9.23)
9:     Compute $\underline{Q}^i_{o,k+1,h}(s,a)$ and $\overline{Q}^i_{o,k+1,h}(s,a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$, $h \in [H]$, and
       $i \in [M]$
10:    Compute the plausible actions for all $s \in \mathcal{S}$ and $h \in [H]$:
       $$\mathcal{A}^i_{k+1,h}(s) = \begin{cases} \{a_{k,h}\} & \text{if } i = I_k \text{ and } s = s_{k,h} \\ \mathcal{A}^i_{k,h}(s) & \text{if } N_{k,h}(s) > 0 \\ \widetilde{\mathcal{A}}^i_{k+1,h}(s) & \text{otherwise} \end{cases}$$
       with $\widetilde{\mathcal{A}}^i_{k+1,h}(s)$ as in Equation (9.24).
11: end for
```

9.5.1 Regret Guarantees

In this section, we give a regret bound for the RfOCL algorithm. Obviously, the same arguments for AfOCL can also be applied for this extended version, and then the regret bound of Theorem 9.4.1 is valid for RfOCL. Moreover, for this algorithm, we prove that the regret, under certain conditions, does not depend on the number of configurations. In order to prove the result, we have to make the following assumption on the NConf-MDP.

Assumption 9.5.1. *There exists $\epsilon > 0$ such that:*

$$\min_{i \in [M]} \min_{s \in \mathcal{S}} \max_{h \in [H]} d^i_h(s) \geq \epsilon,$$

where $d^i_h(s)$ is the probability of visiting the state $s \in \mathcal{S}$ at time $h \in [H]$ in configuration $\mathcal{P}_i$ under the agent's best response policy π_i.

This assumption involves that in every model $\mathcal{P}_i \in \mathfrak{P}$ the agent has non-zero probability, in some step h, to visit every state s. This allows shrinking the confidence intervals for the reward of every state in order to estimate correctly the agent's policy, regardless the played configuration.

Notice that this assumption is less strict than requiring the ergodicity of the Markov process induced by *any* policy.

In the following theorem we will show how the algorithm exploits this information under Assumption 9.5.1.

Theorem 9.5.1 (Regret of RfOCL). *Let $\mathcal{NCM} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu, \mathcal{R}_c, \mathcal{R}_o, H)$ with $\mathfrak{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_M\}$ be the M finite-horizon MDPs of the problem. Under Assumption 9.5.1, the expected regret of RfOCL at every episode $K > 0$ is bounded by:*

$$\mathbb{E}[\textit{Regret}(\mathrm{K})] \leq \min\left\{3MH^3S^2, \overline{K}\Delta + \frac{\pi^2}{3}\right\},$$

where $\overline{K}$ is the smallest integer solution of the inequality

$$\overline{K} \geq 1 + \Big(\frac{2H^2S^2\log(2SH\overline{K}^2)}{2\Delta_Q^2} + \sqrt{\frac{\overline{K}-1}{2}\log(SH\overline{K}^2)}\Big)\frac{1}{\epsilon},$$

$\Delta = \max_{i\in[M]} \Delta_i$, i.e. the maximum suboptimality gap, and Δ_Q is the minimum positive gap of the agent's Q-values.

Discussion on the regret of RfOCL The regret bound removes the dependence on the number of models M, as $\overline{K}$ is clearly independent of M, but it introduces, as expected, a dependence on the minimum visitation probability ϵ. Since RfOCL exploits additional information compared to AfOCL and the set of plausible actions $\mathcal{A}^i_{k,h}$ of RfOCL are subsets of those of AfOCL, the regret bound AfOCL (Theorem 9.4.1) also holds for RfOCL. Thus, we can take as regret bound for RfOCL the minimum between $\overline{K}\Delta + \frac{\pi^2}{3}$ and $3MH^3S^2$.

Proof To prove the result of Theorem 9.5.1 we start defining the good events G_k for $k \in [K]$:

$$G_k = \left\{\forall s \in \mathcal{S}, \left|\widehat{\mathcal{R}}_{o,k}(s) - r_o(s)\right| \leq \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}}\right\}$$

This event means that, at episode $k \in [K]$, the estimated rewards of each state $s \in \mathcal{S}$ are inside the confidence intervals.

In the following two lemmas we bound the difference between the optimistic (pessimistic) action-value function and the true optimal state-action value function.

Lemma 9.5.1. *For every configuration $\mathcal{P}_i \in \mathfrak{P}$ and state action pair $(s,a) \in \mathcal{S} \times \mathcal{A}$, the difference between the optimistic state-action value function $\overline{Q}^i_{o,k,1}(s,a)$ and the true optimal state-action value function $Q^i_{o,1}(s,a)$ is bounded by:*

$$\overline{Q}^i_{o,k,1}(s,a) - Q^i_{o,1}(s,a) \leq \overline{\mathcal{R}}_{o,k}(s) - \mathcal{R}_o(s) + \sum_{s' \in \mathcal{S}} \sum_{h=2}^{H} \overline{d}^i_{k,h}(s') \left(\overline{\mathcal{R}}_{o,k}(s') - \mathcal{R}_o(s') \right),$$

where $\overline{d}^i_{k,h}$ the visitation distribution induced by a greedy policy $\overline{\pi}_{i,k}$ w.r.t. $\overline{Q}^i_{o,k}$. Similarly, the difference between the true optimal state-action value function $Q^i_{o,1}(s,a)$ and the pessimistic state-action value function $\underline{Q}^i_{o,k,1}(s,a)$ is bounded by:

$$Q^i_{o,1}(s,a) - \underline{Q}^i_{o,k,1}(s,a) \leq \mathcal{R}_o(s) - \underline{\mathcal{R}}_{o,k}(s) + \sum_{s' \in \mathcal{S}} \sum_{h=2}^{H} d^i_{k,h}(s') \left(\mathcal{R}_o(s') - \underline{\mathcal{R}}_{o,k}(s') \right).$$

Proof. The proof is basically taken from [Azar et al., 2013, Zanette et al., 2019, Tirinzoni et al., 2020]:

$$\overline{Q}^i_{o,k,1}(s,a) - Q^i_{o,1}(s,a) \leq \overline{Q}^i_{o,k,1}(s,a) - Q^{\overline{\pi}_{i,k}}_{o,1}(s,a) \tag{9.25}$$

$$= \overline{\mathcal{R}}_{o,k}(s) - \mathcal{R}_o(s) + \sum_{s' \in \mathcal{S}} \sum_{h=2}^{H} \overline{d}^i_{k,h}(s') \left(\overline{\mathcal{R}}_{o,k}(s') - \mathcal{R}_o(s') \right). \tag{9.26}$$

where line (9.25) is due to $Q^i_{o,1}(s,a) \geq Q^{\overline{\pi}_{i,k}}_{o,1}(s,a)$, recalling that $Q^i_{o,1}$ is the optimal Q-value for the agent, under configuration $\mathcal{P}_i$ and the optimal agent's policy. Line (9.25) derives form the application of the simulation lemma since $\overline{Q}^i_{o,k,1}(s,a)$ and $Q^{\overline{\pi}_{i,k}}_{o,1}(s,a)$ are under the same policy $\overline{\pi}_{i,k}$. For the second statement, we proceed analogously by simply observing that $\underline{Q}^i_{o,k,1}(s,a) \geq \underline{Q}^{\pi_i}_{o,1}(s,a)$ where π_i is a greedy policy w.r.t. $Q^i_{o,k}(s,a)$. ☐

Lemma 9.5.2. *If for all $k \in [K]$, the good events G_k hold, for all state-action pairs $(s,a) \in \mathcal{S} \times \mathcal{A}$, $h \in [H]$, and configuration $\mathcal{P}_i \in \mathfrak{P}$ it holds that:*

$$\overline{Q}^i_{o,k,1}(s,a) - Q^i_{o,1}(s,a) \leq SH \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}},$$

$$Q^i_{o,1}(s,a) - \underline{Q}^i_{o,k,1}(s,a) \leq SH \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}}.$$

Proof. We apply Lemma 9.5.1, recall that $\overline{\mathcal{R}}_{o,k}(s) = \widehat{\mathcal{R}}_{o,k}(s) + \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}}$ and $\underline{\mathcal{R}}_{o,k}(s) = \widehat{\mathcal{R}}_{o,k}(s) - \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}}$, and make use of the definition of the events G_k. Then, we bound the visitation distribution with 1. □

Then we bound the number of state visitation given its visitation probability. This bound is necessary to ensure that we have enough visitation of every state to have a good estimation of the agent's reward function.

Lemma 9.5.3. *Let $s \in \mathcal{S}$ be a state with minimum visitation probability $d(s) := \min_{i\in[M]} \max_{h\in[H]} d_h^i(s) > 0$. Then, at episode $k \in [K]$, for every $\delta_k \in (0,1)$, with probability at least $1-\delta_k$ it holds that:*

$$N_k(s) \geq (k-1)d(s) - \sqrt{\frac{k-1}{2}\log\left(\frac{1}{\delta_k}\right)}.$$

Proof. First of all, we define the random variable $N_k^u(s)$ as the count of the visits to state s, where multiple visits in the same episode are considered just once:

$$N_k^u(s) = \sum_{i=1}^{k-1} \mathbb{1}\left\{\exists h \in [H] \,:\, s_{k,h} = s\right\}.$$

Clearly, $N_k^u(s) \leq N_k(s)$ and, consequently, $\mathbb{E}[N_k^u(s)] \leq \mathbb{E}[N_k(s)]$. The expectation of $\mathbb{E}[N_k^u(s)]$ can be bounded as:

$$\begin{aligned}
\mathbb{E}[N_k^u(s)] &= \mathbb{E}\left[\sum_{i=1}^{k-1} \mathbb{1}\left\{\exists h \in [H] \,:\, s_{k,h} = s\right\}\right] \\
&= \sum_{i=1}^{k-1} \Pr\left(\exists h \in [H] \,:\, s_{k,h} = s | \mathcal{P}_{I_k}, \pi_{I_k}\right) && (9.27) \\
&= \sum_{i=1}^{k-1} \Pr\left(\bigcup_{h\in[H]} \{s_{k,h} = s\} | \mathcal{P}_{I_k}, \pi_{I_k}\right) && (9.28) \\
&\geq \sum_{i=1}^{k-1} \max_{h\in[H]} \Pr\left(s_{k,h} = s | \mathcal{P}_{I_k}, \pi_{I_k}\right) && (9.29) \\
&= \sum_{i=1}^{k-1} \max_{h\in[H]} d_h^{I_k}(s) && (9.30) \\
&\geq (k-1) \min_{i\in[M]} \max_{h\in[H]} d_h^i(s) = (k-1)d(s), && (9.31)
\end{aligned}$$

where line (9.27) and line (9.28) we simply rewrite the expectation as probability. In line (9.29) we bound the probability of the union with just one term. In line (9.30) we employ the definition of $d_h^{I_k}(s)$. Finally, in line (9.31), we take the minimum over I_k. Since $0 \leq N_k^u(s) \leq k-1$, by using Höeffding's inequality, we have that with probability at least $1-\delta_k$ it holds that:

$$N_k^u(s) \geq \mathbb{E}[N_k^u(s)] - \sqrt{\frac{k-1}{2}\log\frac{1}{\delta_k}} \geq (k-1)d(s) - \sqrt{\frac{k-1}{2}\log\frac{1}{\delta_k}},$$

having used the lower bound on $\mathbb{E}[N_k^u(s)]$. The result follows from recalling that$\mathbb{E}[N_k^u(s)] \leq \mathbb{E}[N_k(s)]$. □

Previously to give the regret bound we have to bound the number of time a state is visited to have correctly recover the agent's response policy.

Lemma 9.5.4. *If for all $k \in [K]$, the good events G_k hold, and for all $s \in \mathcal{S}$ it holds that $\sqrt{\frac{\log(2SHk^2)}{2N_k(s)}} \leq \frac{\Delta_Q - c}{2SH}$, with arbitrary $c > 0$, then for every configuration $\mathcal{P}_i \in \mathfrak{P}$ we have that $\widetilde{\pi}_{i,k} = \pi_i$.*

Proof. Let Δ_Q be the minimum gap between the Q-function in the optimal action and a different action in all transition probabilities $\mathcal{P}_i \in \mathfrak{P}$:

$$\Delta_Q = \min_{i\in[M]} \min_{s\in\mathcal{S}} \min_{h\in[H]} \left\{ \max_{a\in\mathcal{A}} Q^i_{o,h}(s,a) - \max_{a'\in\mathcal{A}\setminus\arg\max_{a\in\mathcal{A}} Q^i_{o,h}(s,a)} Q^i_{o,h}(s,a') \right\}.$$

For all $s \in \mathcal{S}$ and $h \in [H]$, we denote with $a^* = \arg\max_{a\in\mathcal{A}} Q^i_{o,h}(s,a)$ and we have for all $a \in \mathcal{A} \setminus \{a^*\}$:

$$\begin{aligned}
\overline{Q}^i_{o,k,h}(s,a) - \underline{Q}^i_{o,k,h}(s,a^*) &= \overline{Q}^i_{o,k,h}(s,a) - \underline{Q}^i_{o,k,h}(s,a^*) \pm Q^i_{o,h}(s,a) \pm Q^i_{o,h}(s,a^*) \\
&= \underbrace{\overline{Q}^i_{o,k,h}(s,a) - Q^i_{o,h}(s,a)}_{(A)} + \underbrace{Q^i_{o,h}(s,a^*) - \underline{Q}^i_{o,k,h}(s,a^*)}_{(B)} \\
&\quad + \underbrace{Q^i_{o,h}(s,a) - Q^i_{o,h}(s,a^*)}_{(C)} \\
&\leq 2SH\sqrt{\frac{\log(2SHk^2)}{2N_k(s)}} - \Delta_Q \\
&\leq 2SH\frac{\Delta_Q - c}{2SH} - \Delta_Q \leq -c,
\end{aligned}$$

where for (A) and (B) we applied Lemma 9.5.2 and for (C) we used the definition of Δ_Q. We have proved that the lower bound on the Q-value of the optimal action $\underline{Q}^i_{o,k,h}(s,a^*)$ falls above the upper bound on the Q-value of all other actions $\overline{Q}^i_{o,k,h}(s,a)$. Consequently, the greedy action will be properly identified and $\widetilde{\pi}_{i,k} = \pi_i$. □

Finally we derive the regret of the proposed algorithm RfOCL (see Theorem 9.5.1).

Proof. We rewrite the expected regret as follows:

$$\begin{aligned}
\mathbb{E}[\text{Regret}(K)] &= \sum_{k=1}^{K} \left(\mathbb{E}[\Delta_{I_k} \mathbb{1}\{G_k\}] + \mathbb{E}[\Delta_{I_k} \mathbb{1}\{\neg G_k\}]\right) \\
&\leq \underbrace{\sum_{k=1}^{K} \mathbb{E}[\Delta_{I_k} | G_k]}_{(A)} + \underbrace{H\sum_{k=1}^{K} \Pr(\neg G_k)}_{(B)},
\end{aligned}$$

where we bounded $\Pr(G_k) \leq 1$ in term (A) and Δ_{I_k} with its maximum value H in term (B). We start bounding the (B) term:

$$H \sum_{k=1}^{K} \Pr(\neg G_k) = H \sum_{k=1}^{K} \Pr\left(\exists s \in \mathcal{S} \text{ s.t. } |\widehat{\mathcal{R}}_{o,k}(s) - \mathcal{R}(s)| > \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}} \right) \tag{9.32}$$

$$\leq H \sum_{k=1}^{K} \sum_{s \in \mathcal{S}} \Pr\left(|\widehat{\mathcal{R}}_{o,k}(s) - \mathcal{R}(s)| > \sqrt{\frac{\log(2SHk^2)}{2N_k(s)}} \right) \tag{9.33}$$

$$\leq H \sum_{k=1}^{K} \sum_{s \in \mathcal{S}} \frac{1}{SHk^2} \leq \frac{\pi^2}{6}, \tag{9.34}$$

where line (9.32) follows from the definition of the good event G_k. Line (9.33) is a union bound on the states. Line (9.34) comes from Höeffding's inequality.

For the first term (A) we define the event E_k for all $k \in [K]$:

$$E_k = \left\{ \forall s \in \mathcal{S} \, : \, N_k(s) \geq (k-1)d(s) - \sqrt{\frac{k-1}{2} \log\left(SHk^2\right)} \right\}.$$

If this event holds then every state $s \in \mathcal{S}$ is visited at least $(k-1)d(s) - \sqrt{\frac{k}{2} \log\left(SHk^2\right)}$ times, where $d(s)$ is defined as in Lemma 9.5.3.

Considering the term (A), we have:

$$\sum_{k=1}^{K} \mathbb{E}[\Delta_{I_k} | G_k] \leq \underbrace{\sum_{k=1}^{K} \mathbb{E}[\Delta_{I_k} | G_k, E_k]}_{\text{(C)}} + \underbrace{H \sum_{k=1}^{K} \Pr(\neg E_k)}_{\text{(D)}},$$

where we bound the in the second term $\Delta_{I_k} \leq H$.

We start bounding the second term (D). We apply Lemma 9.5.3 after a union bound over the states:

$$H \sum_{k=1}^{K} \Pr(\neg E_k) = H \sum_{k=1}^{K} \Pr\left(\exists s \in \mathcal{S} : N_k(s) < (k-1)d(s) - \sqrt{\frac{k-1}{2} \log\left(SHk^2\right)} \right)$$

$$\leq H \sum_{s \in \mathcal{S}} \sum_{k=1}^{K} \Pr\left(N_k(s) < (k-1)d(s) - \sqrt{\frac{k-1}{2} \log\left(SHk^2\right)} \right)$$

$$\leq H \sum_{s \in \mathcal{S}} \sum_{k=1}^{K} \frac{1}{SHk^2} \leq \frac{\pi^2}{6}.$$

Now it remains to bound the term (C) that, using Lemma 9.5.4, is zero whenever $\sqrt{\frac{\log(2SHk^2)}{2N_k(s)}} \leq \frac{\Delta_Q - c}{2SH}$. Thus, under the events E_k and recalling that under Assumption 9.5.1 we have $d(s) \geq \epsilon$, we obtain:

$$\sqrt{\frac{\log(2SHk^2)}{2N_k(s)}} \leq \sqrt{\frac{\log(2SHk^2)}{2(k-1)\epsilon - \sqrt{2(k-1)\log\left(SHk^2\right)}}}.$$

From which, we derive the condition:

$$\overline{K} \geq 1 + \left(\frac{2H^2S^2 \log(2SH\overline{K}^2)}{2(\Delta_Q - c)^2} + \sqrt{\frac{\overline{K}-1}{2} \log(SH\overline{K}^2)} \right) \frac{1}{\epsilon}.$$

Then, we take the infimum over c. Thus, for the term (C), we consider the decomposition:

$$\sum_{k=1}^{K} \mathbb{E}[\Delta_{I_k} | G_k, E_k] \leq \sum_{k=1}^{\overline{K}} \mathbb{E}[\Delta_{I_k} | G_k, E_k] + \sum_{k=\overline{K}+1}^{\infty} \mathbb{E}[\Delta_{I_k} | G_k, E_k] = \overline{K}\Delta + 0,$$

where we bounded $\Delta_{I_k} \leq \Delta$ with $\Delta = \max_{i \in [M]} \Delta_i$. Then the total regret is given by:

$$\mathbb{E}[\text{Regret}(K)] \leq \overline{K}\Delta + \frac{\pi^2}{3}.$$

□

9.6 Comparison between the two algorithms

The two proposed algorithms use different types of feedback acquired by the configurator when interacting with the agent. The second algorithm allows eliminating the dependence on the number of configurations, assuming that the MDP, for each configuration, is ergodic under the agent's optimal policy. On the other hand, RfOCL is heavier than AfOCL (although the asymptotic complexity is the same) as it requires to compute, for each episode, the optimistic values of the agent Q functions for each model.

It is important to notice that the two algorithms suffer constant regret; this is due to the assumption that the agent's optimal policy is deterministic. In fact, if we remove this assumption and allow the agent's policy to be stochastic,[3] it is reasonable to believe that the regret AfOCL, suitably modified to maintain confidence intervals for the policy, would scale logarithmically with K, as in unstructured bandits. We cannot conclude the same for the corresponding adaptation of RfOCL. We conjecture that, under Assumption 9.5.1, RfOCL continues to suffer constant regret because it exploits the underlying structure given by the agent's reward function that allows linking together the different transition models. Thus, when playing any configuration, we acquire a finite piece of information that can be shared among all configurations. We leave the investigation of this case as future work.

The online problem that we are facing can be seen as a stochastic multi-armed bandit [Lattimore and Szepesvári, 2020], in which the arms are configurations, and the configurator receives a random realization of its expected return at every episode. Thus, in principle, it can be solved by standard algorithms for bandit problems, such as UCB1 [Auer et al., 2002]. These algorithms are computationally less demanding than ours but suffer regret that grows logarithmically, i.e., indefinitely, with the number of

[3]For example, the agent might optimize an entropy-regularized objective [Haarnoja et al., 2018].

episodes. Indeed, they do not exploit either the fact that the agent's policy is deterministic or the structure induced by the agent's reward function.

9.7 Discussion on the related works

More recently, it has been observed that environment configuration can be actuated even by an external entity, opening new opportunities for the application of environment configurability, including settings in which the configurator's interest conflict with those of the agent. For instance, in [Metelli et al., 2019b] the configurator acts on the environment to induce the agent revealing its capabilities in terms of perception and actuation. Instead, in [Gallego et al., 2019] a threatener entity can change the transition probabilities either in a stochastic or adversarial manner. More generally, environment configuration carried out by an external entity has been studied in the field of planning as a form of *environment design* [Zhang et al., 2009]. Thus, our NConf-MDP unifies these settings, allowing for arbitrary agent's and configurator's reward functions. An interesting connection is established with the *robust control* literature [Nilim and Ghaoui, 2003, Iyengar, 2005]. Whenever the two reward functions are opposite, i.e., the interaction between the agent and the configuration is fully *competitive*, the resulting equilibrium corresponds to a robust policy. Indeed, while the agent tries to maximize its expected return, the configurator places the agent in the worst possible environment.

The design of our approaches is inspired by classic algorithms based on the OFU principle for stochastic multi-armed bandits (e.g. [Lai and Robbins, 1985, Auer et al., 2002, Garivier and Cappé, 2011, Lattimore and Szepesvári, 2020]) and MDPs (e.g. [Auer et al., 2009, Bartlett and Tewari, 2009b]). Moreover, our learning setting with reward feedback is related to structured bandits or bandits with correlated arms.[4] Interestingly, for certain structures, it is known that bounded regret is achievable [Bubeck et al., 2013, Lattimore and Munos, 2014], a property that is enjoyed by both our algorithms. Our setting is also close to the Stochastic Games model, in which two or more agents act in an MDP to maximize their own reward functions. Recently, the stochastic game's framework gains growing interest [Bai et al., 2020, Bai and Jin, 2020, Zhang et al., 2020b], especially in the offline setting i.e., we can control all the agents. For this reason, these approaches do not apply to our setting, where we have the control of the configurator only. Although some works tract the online setting [Wei et al., 2017, Xie et al., 2020a, Tian

[4]In our case, playing a single configuration provides information about the opponent's reward, which, in turns, provides information about the value of all configurations.

et al., 2020], where we can control only one agent, all of these algorithms works in the zero-sum setting only.

9.8 Experiments

In this section, we provide the experimental evaluation of our algorithms on three different domains: Configurable Gridworld (Section 9.8.1), Student-Teacher (Section 9.8.3), and Configurable Market (Section 9.8.2). We compare the algorithms with the standard implementation of UCB1 [Auer et al., 2002].

9.8.1 Configurable Gridworld

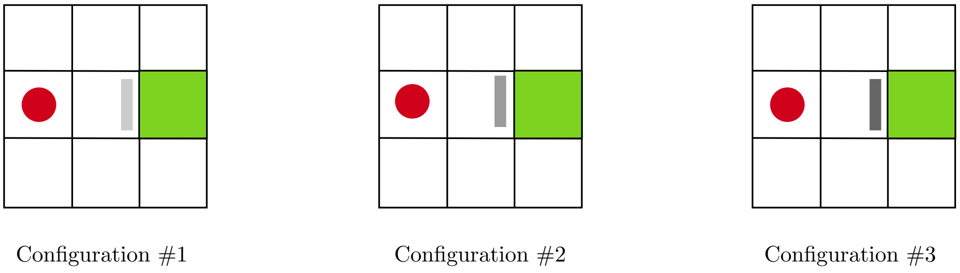


Figure 9.1: *Configurable Gridworld: from left to right the* 3 *configurations represent increasing "power" of the obstacle.*

The Configurable Gridworld is a configurable version of a classic 3×3 Gridworld (a graphical representation is shown in Figure 9.1). The agent's starting state is in the cell $(0, 1)$, and its goal is to minimize the number of steps required to reach the exit located in the cell $(2, 1)$. Instead, the configurator takes reward 1 when the agent occupies the central cell $(1, 1)$ and 0 otherwise. In a classic Gridworld, the optimal policy would be trivial, as the agent would proceed straight to the exit. In this Configurable Gridworld, instead, the configurator can set the "power" p of a stochastic obstacle located in the cell $(1, 1)$. In particular, when the agent is in that cell and performs action "go right" to reach the exit, it will hit the obstacle, and it will remain in the same position with probability p. The configurator's goal is to tune this probability to keep the agent in the central cell for the maximum number of steps. In practice, this means raising the probability p as much as possible. However, it is easy to prove that if p is too large, the agent will learn to avoid the obstacle by passing close to the boundaries, leading to unsatisfactory performance for the configurator. The M configurations

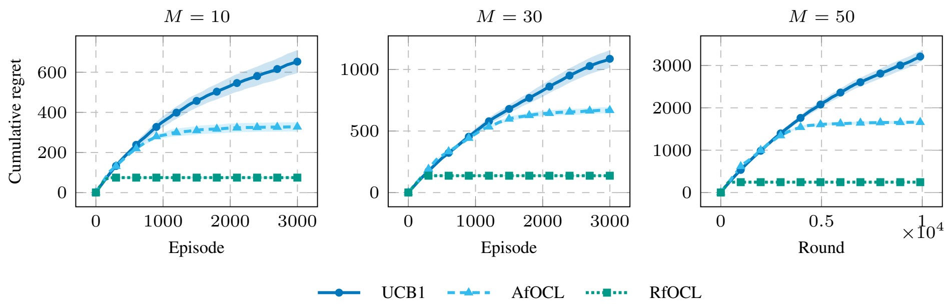


Figure 9.2: *Cumulative regret as a function of the episodes for the Gridworld experiment. 50 runs, 98% c.i.*

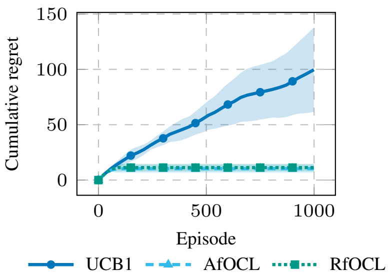


Figure 9.3: *Cumulative regret as a function of the episodes for the Gridworld experiment in the extreme setting. 50 runs, 98% c.i.*

differ in the probability p and are obtained by a regular discretization of $[0, 1]$.

The results of the experiments are shown in Figures 9.2 and 9.3. In the first experiment (Figure 9.2), we considered 10 and 30 configurations with a number of episodes $K = 3000$ and horizon $H = 10$. We can see that the two algorithms, AfOCL and RfOCL, suffer constant regret, whereas UCB1 displays a logarithmic regret, as expected. Specifically, RfOCL outperforms AfOCL and stops playing suboptimal configuration in less than 500 episodes in both cases. This can be explained because, being Assumption 9.5.1 fulfilled (in fact, the agent has the probability 0.1 of failing its action), RfOCL is able to exploit the underlying structure of the problem more effectively. Figure 9.3 presents a more extreme case in which we have only three configurations, designed so that the optimal agent's policy generates a non-ergodic Markov chain. In such a case, we violate Assumption 9.5.1 and consequently, we observe that AfOCL and RfOCL display very similar behavior, but still significantly better than UCB1.

9.8.2 Configurable Market

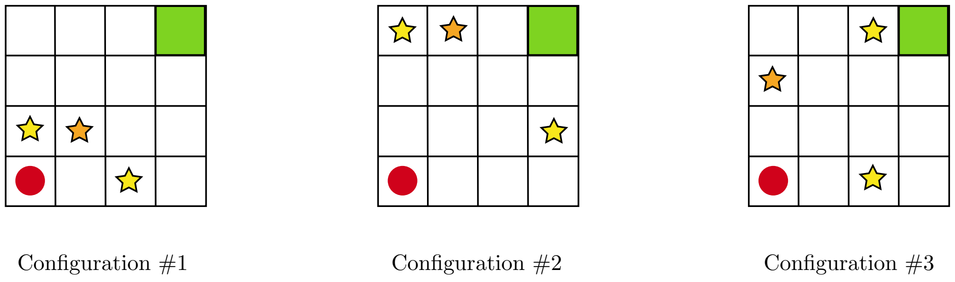


Figure 9.4: *Market: the figure shows a 5×5 market. The red state is the starting state, instead the green state is the "end" state. The stars are the product and the orange star is the only product the agent is interested in.*

A Configurable Market is a simplified model for a marketplace. The agent, namely the customer, wants to buy a given set of products Q_A in the minimum number of steps. Instead, the configurator has the role in placing all the products $Q \supset Q_A$ in the marketplace to maximize the market's revenue inducing the agent to buy other products in addition to those it would buy. The configurator's reward is 1 any time the agent passes over a state where a product is placed and 0 in all the other states. Whereas the agent's reward is -1 everywhere and gains a bonus of 0.9 when it passes over a state with a product in Q_A. In other words, the products remain fixed in the market, and the configurator can change the transition model within a set of random transition models. In Figure 9.4 the market domain with 3 different configurations is shown. The market domains consists in $K \times K$ states, where every product is assigned to a specific state. The configurator can change the transition matrix for all the states except for the starting state and the "exit" state. Each different configuration can be thought of as a

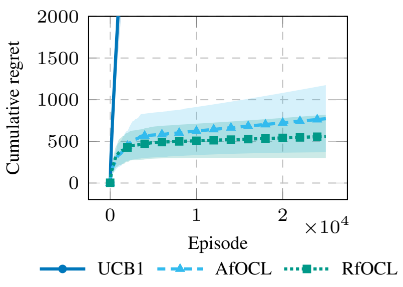


Figure 9.5: *Cumulative regret as a function of the episodes for the Configurable Market experiment. 50 runs, 98% c.i.*

different positioning of the products. So, from an abstract point of view, changing configuration is equivalent to moving products into the Gridworld.

In Figure 9.5, AfOCL and RfOCL are compared against UCB1. The number of configurations is 10, the horizon 15, and the Gridworld size is 4×4. In every run, we construct 10 different transition models, which specify the 10 configurations. Also, in this experiment, the trend is confirmed since AfOCL and RfOCL outperform UCB1. We observe that the two algorithms, in this environment, behave similarly, and this is due to the small number of configurations. However, we can notice RfOCL at the end of the considered episodes approaches the constant regret.

9.8.3 Student-Teacher

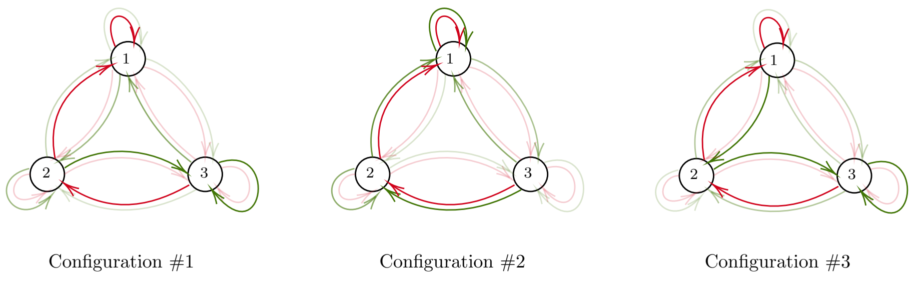


Figure 9.6: *Teacher Student environment.*

The Student-Teacher environment models a simple interaction between a student and a teacher. The teacher has several exercises available with different difficulty levels and wants to find the optimal sequence of exercises in order to make the student acquire as much knowledge as possible. On the other hand, the student perceives the exercises' level of difficulties in a different way. The student's goal is to maximize the number of exercises that he/she knows how to solve, and we model this information with an integer between $[0, S]$. The student decides whether to answer or not the exercise. In the case, he/she answers, he/she receives a reward equal to the level of "correctness" of the exercise, the teacher receives a reward corresponding to the level of exercise's "hardness", and they end up to the next exercise. If the student does not answer, the student and the teacher will receive -1, and with a probability of 0.7, the next exercise will be easier to solve. In Figure 9.6 we reported an illustrative example of the Teacher-Student domain: right arrows correspond to answer "No", and green arrows to answer "Yes". The transparency is due to the transition probability distribution. The configurator can change the transition probability for the

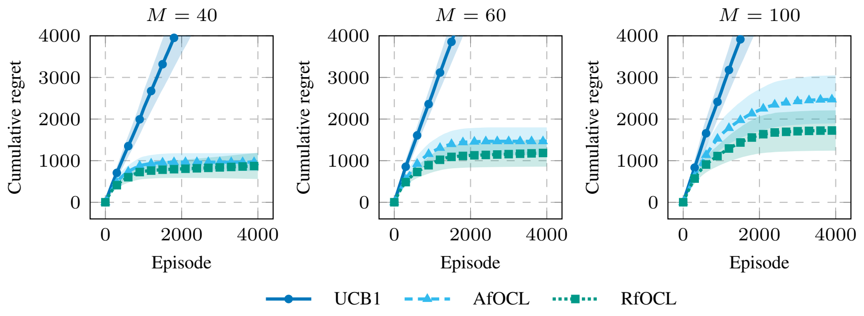


Figure 9.7: *Cumulative regret as a function of the episodes for the Student-Teacher experiment. 50 runs, 98% c.i.*

action “Yes”.

In Figure 9.7, we compare our algorithms with UCB1 for different number of configurations $M \in \{40, 60, 100\}$ and horizon $H = 10$. In every run, we construct M random different configurations that represent the distribution over the next exercise, given the current exercise and a positive answer. Moreover, in every run, we change the *mildness* of an exercise from the agent’s point of view. We observe that both AfOCL and RfOCL suffer significantly less regret compared to UCB1 and tend to converge to a constant, especially with a small number of configurations. It is interesting to observe that, in line with our analysis, the gap between AfOCL and RfOCL appears more evident as the number of configurations grows.

CHAPTER 10

Online Learning in General-sum Turn-based Stochastic Games

In this chapter, we analyze the online setting introduced in Section 8.1 for General-sum Stochastic games. We would like to mention that these are preliminary results original for this thesis. Unlike what we have done in the previous chapter, we consider the problem of learning in Stochastic games, where there is one agent that we can control and that acts in an environment observing the actions taken by the other agent. We have to underline that this chapter addresses an open-question left in [Xie et al., 2020a]:

"How to achieve optimal regret guarantees exploiting a weak opponent?"

In the paper with the notion of *weak opponent*, the authors mean that we are facing an opponent that is not totally adversarial (as in zero-sum games). As the authors suggested, in this case, the guarantee involves a stronger notion of the regret with respect to the minimax ones (see Section 8.1); in Section 8.1 we have formalized this stronger regret as the regret achieved by the Stackelberg Equilibrium policy, under the assumption that we are restricted to the setting of Markov policies.

In this chapter, we start with a formal introduction to the problem. In section 10.2, we derive a lower bound on the expected regret of any "good" learning strategy that captures the exploration challenges in this context. In particular, the lower bound clearly shows that regret minimization in Stochastic Games is significantly more complex than in standard MDPs. Then, we propose an algorithm that nearly-matches the proposed lower-bound.

10.1 Problem statement

In this section, we introduce the online learning problem in Turn-based General-sum Stochastic Games. We have already formally introduced the Turn-based Stochastic Games in Chapter 4. In these games, at each step h only one player takes an action. In fact, the state space $\mathcal{S}$ is partitioned in $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, where $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. For each state $s \in \mathcal{S}$, we denote by $I(s) \in \{1, 2\}$ the function that indicates if a state belongs to $\mathcal{S}_1$ or $\mathcal{S}_2$, i.e., which player has to play in the current state s. At each step h, the agent $I(s_h)$ has to decide the action a_h to be taken and the two players receive respectively rewards $\mathcal{R}_1(s_h, a_h)$ and $\mathcal{R}_2(s_h, a_h)$; then, the system transitions to the next state $s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)$.

In the online setting, we can control only the agent 1. In this chapter, we analyze the special case described in [Xie et al., 2020a], where the second agent is omniscient and always plays the best response. So, the agent 2, given the policy $\pi_1^i \in \Pi_1$, follows the policy $\pi_2^{*,i}$ such that:

$$\pi_2^{*,i} \in \arg\max_{\pi_2 \in \Pi_2} V_2^{\pi_1^i, \pi_2},$$

i.e. it plays its *best response*.

As in the previous chapter this creates an inherently *asymmetric* interaction: the first agent can be seen as a *leader*, which decides the policy to be played in an episode and the second agent can be seen as a *follower*, which can see the leader's policy and adapts its response to it. As is commonly done in the game-theory literature [Balcan et al., 2015, Peng et al., 2019, Sessa et al., 2020] we make the following assumption:

Assumption 10.1.1. *For every policy $\pi_1 \in \Pi_1$ the second uncontrollable agent will always play the same best response policy $br(\pi_1)$, where $br : \Pi_1 \to \Pi_2$. Furthermore, $br(\pi_1)$ is deterministic.*

Under this assumption the goal of our agent is well-defined and consists in finding the policy $\pi_1 \in \Pi_1$ that is optimal under the second agent's best response policy:

$$\pi_1^\star \in \arg\max_{\pi_1 \in \Pi_1} V_1^{\pi_1, br(\pi_1)},$$

where it corresponds to finding the Stackelberg Equilibrium of the game.

In our setting we do not know the policies that the second agent will play, i.e., the *br* function is unknown. From an online learning perspective, we are interested in minimizing the expected regret:

$$\mathbb{E}[\text{Regret}(K)] = K \max_{\pi_1 \in \Pi_1} V_1^{\pi_1, br(\pi_1)} - \sum_{k=1}^{K} V_1^{\pi_{1,k}, br(\pi_{1,k})}. \tag{10.1}$$

Obviously, this problem can be seen as solving a stochastic multiarmed bandit problem [Lattimore and Szepesvári, 2020]. In this case, the arms are the policies, and the agent at each episode receives a random realization of its expected return. So, this problem can be solved with standard bandit algorithms such as UCB1 [Auer et al., 2002]. However, as we will explain in the next section, this is not the best that we can do. In fact, the regret would not scale sublinearly with the number of possible policies, as it happens with standard bandit algorithms (where the regret is $\mathcal{O}(\sqrt{|\Pi_1|K})$. On the other hand, for our setting, we will derive lower and upper bounds on the expected regret with only a *constant* dependence on the number of policies (i.e., not multiplicative of K).

Comparison between Nash Equilibrium and Stackelberg Equilibrium In this chapter, we use the notion of Stackelberg Equilibrium (SE) rather than the Nash Equilibrium (NE) to formalize the regret. There are two reasons to adopt the SE concept, one more philosophical and the other more practical. The SE, differently from the NE, models the asymmetric interaction between the two agents and, moreover, models many real-world problems (e.g., in the security domain and network routing) that cannot be modeled with the NE. Moreover, from a practical viewpoint, at every stage game, the computation of the NE is also PPAD complete for two agents with $|\mathcal{A}| > 2$ [Papadimitriou, 1992]; on the other hand, computing the SE requires polynomial complexity [Coniglio et al., 2020].

10.2 Lower bound on the regret

We start by proposing a lower bound on the expected regret for the online Turn-based Stochastic Game problem that we have defined above. In the following result, we consider the case in which our player can see the actions taken by the second agent as well as its rewards. We start by considering the TSG shown in Figure 10.1. This TSG is composed of $N + 3$ states and A

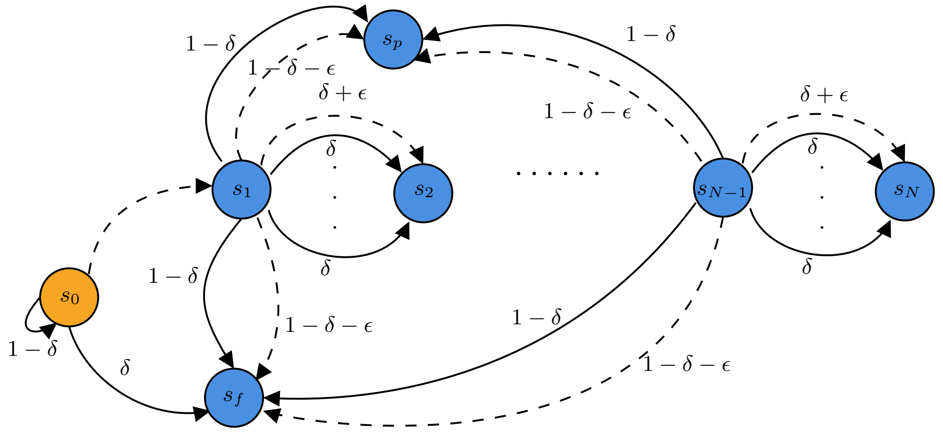


Figure 10.1: *The composite Turn-based Stochastic Game for the lower bound. The states belonging to $\mathcal{S}_2$ are in orange, the ones belonging to $\mathcal{S}_1$ in blue. The dashed lines corresponds to the transition probabilities taking action $a^\star$, the others taking any other action $a \in \mathcal{A}$ with $a \neq a^\star$. The dots indicate the chain composed by N states.*

actions where N and A are two positive integers. The agent 2 controls the starting state s_0, $\mathcal{S}_2 = \{s_0\}$, identified in the figure with the orange color. The state-space of agent 1, instead, is equal to $\mathcal{S}_1 = \{s_f, s_p, s_1, \dots, s_N\}$, i.e. it controls the blue states in Figure 10.1. The reward functions of the two agents are:

$$\mathcal{R}_1(s,a) = \begin{cases} 1 & \text{if } s = s_N \\ 0 & \text{otherwise} \end{cases},$$

$$\mathcal{R}_2(s,a) = \begin{cases} R & \text{if } s = s_N \\ R_f & \text{if } s = s_f \\ 0 & \text{otherwise} \end{cases}.$$

The transition model of the TSG is defined as follows. In state s_0 the agent can choose only two actions $a^\star$ and a_f, so $\mathcal{P}(s_1|s_0,a^\star) = 1$ and $\mathcal{P}(s_f|s_0,a_f) = \delta$, $\mathcal{P}(s_0|s_0,a_f) = 1-\delta$. From the state s_f with any action we continue to stay in state s_f. From all the other states s_i with $i \in [N]$: $\mathcal{P}(s_{i+1}|s_i,a^\star) = \delta+\epsilon$ and $\mathcal{P}(s_f|s_i,a^\star) = 1-\delta-\epsilon$; instead, for any other action $a \in \mathcal{A}$, $\mathcal{P}(s_{i+1}|s_i,a) = \delta$ and $\mathcal{P}(s_f|s_i,a) = 1-\delta$. Moreover we assume that $H = N+1$.

The second agent has only two response functions: in s_0 it can choose action $a^\star$ and continue to state s_1 or it can choose a_f. Obviously, it depends on the policy that we decide to take at the beginning of the episode.

Proposition 10.2.1. *If* $R_f = \frac{(\delta+\epsilon)^{H-2}(\delta+c)R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}}$*, there exists a constant* $0 < c < \epsilon$ *such that the only policy that induces the second agent to play* $a^\star$ *is* $\pi_1(a^\star|\cdot) = 1$ *for all* $s \in \mathcal{S}_1$.

Proof. We start by calling $\pi_2^{a^\star}$ and $\pi_2^{a_f}$ the policies of the second agent that choose respectively $a^\star$ and a_f in s_0. To be $\pi_1^\star$ the only policy that induces the second agent to play $a^\star$ two things have to happen:

1. $V_2^{\pi_1^\star,\pi_2^{a^\star}} > V_2^{\pi_1^\star,\pi_2^{a_f}}$,
2. $V_2^{\pi_1,\pi_2^{a^\star}} < V_2^{\pi_1^\star,\pi_2^{a_f}}$,

where $\pi_1 \in \Pi_1$ is every policy in Π_1 such that $\pi_1 \neq \pi_1^\star$. We now show that using the proposed two proposed rewards R and R_f fulfills these two conditions. We start by evaluating the two value functions. We remind that $H = N + 1$.

$$\begin{aligned} V_2^{\pi_1^\star,\pi_2^{a^\star}} &= (\delta+\epsilon)^{H-1}R, \\ V_2^{\pi_1,\pi_2^{a^\star}} &\leq (\delta+\epsilon)^{H-2}\delta R, \\ V_2^{\pi_1^\star,\pi_2^{a_f}} &= V_2^{\pi_1,\pi_2^{a_f}} = \sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}R_f, \end{aligned}$$

where the second equation is due the fact that the policy that achieves the greatest expected return is the one that chooses the action $a^\star$ in all the states except to state s_1.

We now show that the first condition holds.

$$\begin{aligned} &(\delta+\epsilon)^{H-1}R > \sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}R_f \\ &\frac{(\delta+\epsilon)^{H-1}R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}} > R_f \\ &\frac{(\delta+\epsilon)^{H-1}R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}} > \frac{(\delta+\epsilon)^{H-2}(\delta+c)R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}}. \end{aligned}$$

And it is always true if $\epsilon > c$.

It is easy to see that the second condition also holds since:

$$\begin{aligned} &(\delta+\epsilon)^{H-2}\delta R < \sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}R_f \\ &\frac{(\delta+\epsilon)^{H-2}\delta R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}} < R_f \\ &\frac{(\delta+\epsilon)^{H-2}\delta R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}} < \frac{(\delta+\epsilon)^{H-2}(\delta+c)R}{\sum_{h=1}^{H-1}(H-h)\delta(1-\delta)^{h-1}}. \end{aligned}$$

□

Then, for an appropriate value R_f the second agent will choose the action $a^\star$ only if we choose the optimal policy that always takes the correct action $a^\star$ at each step.

Intuition on lower bound In all the cases where the agent 1 plays a policy different from the optimal one, we cannot acquire any information about the transition model since we will visit only state s_f until the end of the episode. Intuitively, we can notice that, in the worst case, we have to play all the policies in Π_1 before finding the best policy that takes us to acquire information about all the states in the chain. In fact, only with the optimal policy the agent 2 permits us to visit states different from s_f. In the following theorem, we will formally prove this intuition.

Theorem 10.2.1 (Lower bound for online Turn-based Stochastic Game)**.** *Let $\mathfrak{A}$ be a"good" learning algorithm, where with "good" we indicate an algorithm such that its expected regret is upper bounded by $\mathcal{O}\left(CK^{\alpha}\right)$ with $\alpha < 1$ in all Turn-based Stochastic Games* [1]*. Then we can create a Turn-based Stochastic Game such that the expected regret is lower bounded by:*

$$\mathbb{E}[\textit{Regret}^{\mathfrak{A}}(K)] \geq \Omega\left(H\sqrt{SAK}\right). \tag{10.2}$$

Moreover, we can create a Turn-based Stochastic Game with S states, A actions and horizon $H = S - 2$, and a specific initial distribution μ such that the expected regret of $\mathfrak{A}$ after K steps:

$$\mathbb{E}[\textit{Regret}^{\mathfrak{A}}(K)] \geq \Omega\left(A^S\right). \tag{10.3}$$

Proof. In this part we will prove the lower bound for the online Turn-based Stochastic Game problem. We start by stating that a Markov Decision Process is a special case of a Turn-based Stochastic Game, in which the state space of the second agent $\mathcal{S}_2 = \emptyset$. Then from this consideration we can state that the worst-case lower bound for MDPs can be also applied for TSGs [Jaksch et al., 2010, Domingues et al., 2020]:

$$\mathbb{E}[\text{Regret}^{\mathfrak{A}}(K)] \geq \Omega\left(H\sqrt{SAK}\right).$$

So we have now to prove the lower bound in equation 10.3. In order to achieve that we rely on standard information-theoretic methods to prove lower bounds in episodic MDP and bandit problems. We start by stating a lemma taken from [Simchowitz and Jamieson, 2019]. Since the proof is analogous we omitted it here.

Lemma 10.2.1. *Let $\mathcal{TSG} = (\mathcal{S}, \mathcal{A}, H, \mathcal{R}, \mu, \mathcal{P})$ and $\mathcal{TSG}' = (\mathcal{S}, \mathcal{A}, H, \mathcal{R}, \mu, \mathcal{P}')$ be two TSGs with the same state space $\mathcal{S}$, action space $\mathcal{A}$, initial state distribution μ and horizon H. Fix a number of episodes $K \geq 1$ and let $\mathcal{F}_K$ be the filtration generated by all rollouts up to episode K. Then for any $\mathcal{F}_K$-measurable random variable $Z \in [0, 1]$,*

$$\sum_{s,a} \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}\left[N_K(s, a)\right] KL(\mathcal{P}(\cdot|s, a), \mathcal{P}'(\cdot|s, a)) \geq \textit{kl}(\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z], \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[Z]) \tag{10.4}$$

where $\textit{kl}(x, y) = x\log\left(\frac{x}{y}\right) + (1 - x)\log\left(\frac{1-x}{1-y}\right)$ is the binary KL-divergence and $KL(\cdot, \cdot)$ denotes the KL-divergence between two probability laws.

[1]We note that algorithms that satisfy this assumption exist. For instance, applying UCB over the set of policies Π_1 yields regret $\mathcal{O}\left(\sqrt{|\Pi_1|K}\right)$

We apply this lemma as follows. For the fixed state-action pair (s_0, a_f), we define an alternative $\mathcal{TSG}'$ to be the TSG that coincides with $\mathcal{TSG}$ except that:

$$\mathcal{P}(s_f|s_0, a_f) = \delta + \epsilon \quad \mathcal{P}(s_0|s_0, a_f) = 1 - \delta - \epsilon.$$

For this game there are no policies that induces the second agent to play $a^\star$ since it always gains more by playing a_f and taking R_f.

By construction the two games $\mathcal{TSG}$ and $\mathcal{TSG}'$ differ only at s_0, a. Thus:

$$\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}\left[N_K(s_0, a)\right] KL(\mathcal{P}(\cdot|s_0, a), \mathcal{P}'(\cdot|s_0, a)) \geq \text{kl}(\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z], \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[Z]).$$

We define $N_K(a) = \sum_{k=1}^{K} \mathbb{1}\left\{a_{k,1} = a\right\}$. We define the following two events:

$$\mathcal{E}^a_K = \{N_K(a_f) \geq \overline{K}\}, \qquad \mathcal{E}^{s_f}_K = \{\sum_{k=1}^{K} \mathbb{1}\left\{s_2 = s_f\right\} \geq \overline{K}\}$$

i.e. at episode K the number of times the second agent has played action a_f at time step 1 and s_f is visited at time step 2 is greater than $\overline{K}$, where $\bar{K}$ is a constant to be chosen later. We define $\mathbb{1}\left\{\mathcal{E}^a_K, \mathcal{E}^{s_f}_K\right\}$ as the indicator random variable that is 1 if event $\mathcal{E}^a_K$ and $\mathcal{E}^{s_f}_K$ happens and 0 otherwise. We are going to evaluate now the expectation $\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[\mathbb{1}\left\{\mathcal{E}^a_K, \mathcal{E}^{s_f}_K\right\}]$. We start by stating that:

$$\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[\mathbb{1}\left\{\mathcal{E}^a_K, \mathcal{E}^{s_f}_K\right\}] \leq \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[\mathbb{1}\left\{\mathcal{E}^a_K\right\}].$$

Then we note that we have assumed that the algorithm $\mathfrak{A}$ is “good” in the sense that its regret is bounded by $\mathcal{O}\left(CK^\alpha\right)$. From this assumption and considering that we do not pay regret only if the second agent plays $a^\star$ in state s_0, we can say that:

$$\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[\mathbb{1}\left\{\mathcal{E}^a_K, \mathcal{E}^{s_f}_K\right\}] \leq \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[\mathbb{1}\left\{\mathcal{E}^a_K\right\}] \leq \frac{CK^\alpha}{\overline{K}},$$

applying the Markov inequality.

Then we evaluate $\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\mathcal{E}^a_K, \mathcal{E}^{s_f}_K\right\}]$. We start considering the fact that in the modified $\mathcal{TSG}'$ the second agent always plays the action a, so:

$$\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\mathcal{E}^a_K, \mathcal{E}^{s_f}_K\right\}] = \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\mathcal{E}^{s_f}_K\right\}],$$

or equivalently to lower bound the value $\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\mathcal{E}^{s_f}_K\right\}]$

$$\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\mathcal{E}^{s_f}_K\right\}] = 1 - \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\neg\mathcal{E}^{s_f}_K\right\}].$$

Where $\neg\mathcal{E}^{s_f}_K = \{\sum_{k=1}^{K} \mathbb{1}\left\{s_2 = s_f\right\} < \overline{K}\}$. Noting that the random variable $\sum_{k=1}^{K} \mathbb{1}\left\{s_2 = s_f\right\}$ has a binomial distribution with probability of success δ, we upper bound $\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\neg\mathcal{E}^{s_f}_K\right\}]$ with:

$$\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[\mathbb{1}\left\{\neg\mathcal{E}^{s_f}_K\right\}] \leq \frac{(K - \overline{K})(\delta + \epsilon)}{(K(\delta + \epsilon) - \overline{K})^2},$$

applying the upper bound from [Feller, 1957] for the pdf of the binomial distribution.

It is easy to see that for small ϵ we have:

$$KL(\mathcal{P}(\cdot|s_0, a), \mathcal{P}'(\cdot|s_0, a)) \simeq \epsilon$$

Then applying lemma 10.2.1, where we call $Z = \mathbb{1}\left\{\mathcal{E}_K^a, \mathcal{E}_K^{s_f}\right\}$:

$$\begin{aligned}
\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}\left[N_K(s_0,a)\right]\epsilon &\geq \mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z]\log\left(\frac{\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z]}{\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[Z]}\right) + (1-\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z])\log\left(\frac{1-\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z]}{1-\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[Z]}\right)\\
&\geq (1-\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}}[Z])\log\left(\frac{1}{1-\mathbb{E}^{\mathfrak{A}}_{\mathcal{TSG}'}[Z]}\right) - \log(2)\\
&\geq \left(1-\frac{CK^\alpha}{\overline{K}}\right)\log\left(\frac{1}{1-\left(1-\frac{(K-\overline{K})(\delta+\epsilon)}{(K(\delta+\epsilon)-\overline{K})^2}\right)}\right) - \log(2)\\
&= \left(1-\frac{CK^\alpha}{\overline{K}}\right)\log\left(\frac{(K(\delta+\epsilon)-\overline{K})^2}{(K-\overline{K})(\delta+\epsilon)}\right) - \log(2)\\
&\geq \left(1-\frac{CK^\alpha}{\overline{K}}\right)\log\left(\frac{K^2(\delta+\epsilon)-\overline{K}}{K-\overline{K}}\right) - \log(2)
\end{aligned}$$

Setting $\overline{K} = C2K^\alpha$:

$$\mathop{\mathbb{E}}_{\mathcal{TSG}}\left[N_K(s_0,a)\right]\epsilon \geq \frac{1}{2}\log\left(\frac{K^2(\delta+\epsilon)-C2K^\alpha}{K-C2K^\alpha}\right) - \log(2)$$

Then setting $\epsilon = \frac{1}{2A^S}$.

$$\mathop{\mathbb{E}}_{\mathcal{TSG}}\left[N_K(s_0,a)\right] \geq A^S\log\left(\frac{K^2(\delta+\epsilon)-C2K^\alpha}{K-C2K^\alpha}\right) - \log(2),$$

and if $\delta > \frac{2K-C2K^\alpha}{K^2}$ then the logarithm is always greater than $\log(2)$ then:

$$\mathop{\mathbb{E}}_{\mathcal{TSG}}\left[N_K(s_0,a_f)\right] \geq A^S\log(2) - \log(2).$$

Then since every time the action a_f is taken in state s_0, we cannot reach state s_N in H steps and we pay a regret equal to 1, the expected regret is bounded by:

$$\mathbb{E}[\text{Regret}(K)] \geq \mathop{\mathbb{E}}_{\mathcal{TSG}}\left[N_K(s_0,a_f)\right] \geq \Omega\left(A^S\right).$$

Then the result follows.

□

10.2.1 Discussion on the lower bound

The lower bound implicitly says that we can create very small suboptimality gaps for the second agent, and that the regret must scale with the inverse of them regardless of the suboptimility gaps of the first agent. This can happen since our agent does not pay for the small suboptimality gap of the uncontrallable agent but for our gap that can be potentially very large. This lower bound is the first one that states the difficulties in learning in general-sum Stochastic Games with the possibility to see the other agent's reward function and actions. Other lower bounds were derived for the general-sum setting. In [Bai et al., 2020] the authors proposed a lower bound to underline the difficulties to learn against an adversarial opponent. In [Tian et al., 2020], instead, the authors show the statistical hardness of learning with only bandit feedback. However, these two settings are harder than the one proposed in this section, and, for this reason, cannot be applied.

10.3 TSG Optimistic Policies Value Iteration

In this section, we propose an algorithm, called TSG Optimistic Policies Value Iteration (TSG-OPVI), that nearly-matches the lower bound proposed in the previous section. The algorithm, given the set of policies Π_1 for the first agent, stores a table recording the policy that is played by the second player. We assume that Π_1 is any set of policies (similarly to [Abbasi-Yadkori et al., 2013]), not necessarily corresponding to the full set of all deterministic policies More formally, let M be the cardinality of the policy set Π_1. Similarly to what we have done before in Chapter 9, for every $i \in [M]$, $k \in [K]$, $h \in [H]$ we denote with $\mathcal{A}^i_{k,h}(s) \subseteq \mathcal{A}$ the set of plausible actions in state s at step h for policy $\pi_i \in \Pi_1$ at the beginning of episode k. Since, given a agent 1 policy, the response policy of the other agent is deterministic and unique, when we play the policy π_i and we observe in state $s \in \mathcal{S}_2$, at time step h, the policy $\pi_{2,h}(s)$, we can set $\mathcal{A}^i_{k,h}(s) = \{\pi_{2,h}(s)\}$.

As common in optimisitic value iteration algorithms [Azar et al., 2013], we shall build upper confidence bounds to the value function of each policy by adding bonus terms based on confidence intervals on the rewards and transition probabilities. Formally, for every $k \in [K]$, state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ we derive the bonus term, based on Hoeffding's concentration inequality, for the reward function and the expected value function:

$$b^r_k(s,a) = \sqrt{\frac{2\log\left(\frac{4SAHk}{\delta}\right)}{N_k(s,a)}} \qquad b^{\mathcal{P}}_k(s,a) = H\sqrt{\frac{2S\log\left(\frac{4SAHk}{\delta}\right)}{N_k(s,a)}}.$$

Moreover we indicate with $\widehat{\mathcal{R}}_{1,k}(s,a)$ and $\widehat{\mathcal{P}}_{1,k}(s'|s,a)$ the sample means of respectively the observed rewards and transitions up to (and not including) episode k.

Based on this, at the beginning of each episode $k \in [K]$ we can compute for each policy π^i_1 with $i \in [M]$ an optimistic approximation $\widetilde{V}^i_{1,k,h}$ of the value function $V^i_{1,h}$:

$$\widetilde{V}^i_{1,k,h}(s) = \begin{cases} \widehat{\mathcal{R}}(s,\pi^i_{1,h}(s)) + \sum_{s' \in \mathcal{S}} \widehat{\mathcal{P}}(s'|s,\pi^i_{1,h}(s)) \widetilde{V}^i_{1,k,h+1}(s') + b_k(s,\pi^i_{1,h}(s)) & \text{if } I(s) = 1 \\ \max_{a \in \mathcal{A}^i_{k,h}(s)} \widehat{\mathcal{R}}(s,a) + \sum_{s' \in \mathcal{S}} \widehat{\mathcal{P}}(s'|s,a) \widetilde{V}^i_{1,k,h+1}(s') + b_k(s,a) & \text{if } I(s) = 2\text{'} \end{cases} \tag{10.5}$$

Algorithm 13 TSG-OPVI

```
1: Input: $\mathcal{S}$, $\mathcal{A}$, $H$, $\Pi_1 = \{\pi_1^1, \dots, \pi_1^M\}$
2: Initialize $\mathcal{A}_{1,h}^i(s) = \mathcal{A}$ for all $s \in \mathcal{S}$, $h \in [H]$, and $i \in [M]$
3: for episodes $1, 2, \dots, K$ do
4:     Compute $\widetilde{V}_{1,k}^i$ (Equation 10.5) for all $i \in [M]$
5:     Play $\pi_1^{I_k}$ with $I_k \in \arg\max_{i \in [M]} \widetilde{V}_{1,k}^i$
6:     Observe $(s_{k,1}, a_{k,1}, \dots, s_{k,H-1}, a_{k,H-1}, s_{k,H})$
7:     Compute the plausible actions for all $s \in \mathcal{S}$, $h \in [H]$ and $i \in [M]$:
           $$\mathcal{A}_{k+1,h}^i(s) = \begin{cases} \{a_{k,h}\} & \text{if } i = I_k \text{ and } s = s_{k,h} \\ \mathcal{A}_{k,h}^i(s) & \text{otherwise} \end{cases}$$
8: end for
```

where $b_k(s, \pi_{1,h}^i(s)) = b_k^r(s, \pi_{1,h}^i(s)) + b_k^{\mathcal{P}}(s, \pi_{1,h}^i(s))$. Note that we use two levels of optimism: one for the unknown transition probabilities and rewards, and one for the unknown actions of the second agent. More precisely, if we have already seen the action that the second agent will play in a state s with a policy π_1^i we use this information to estimate the value function, otherwise we act optimistically by taking the maximum over all plausible actions. The pseudocode of TSG-OPVI is reported in Algorithm 13.

10.3.1 Regret Guarantees

In this section, we give a regret bound for the proposed algorithm. The result exploits the determinism of the other agent's policies in order to match the lower bound derived in the previous section. The main idea behind the proof is that after having played a finite number of time every policy, we know in every state that is reachable what action the agent 2 will play. At this point we have reduced our problem to an MDP. In fact when we know the best response function of agent 2, for every policy we can create a policy that is the union of the policy of agent 1 and agent 2. At this point the uncertainty comes only from the transition model and the reward function.

Theorem 10.3.1. *Let $TSG = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mu, \mathcal{R}_1, \mathcal{R}_2, H)$ with $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ be the finite-horizon TSG of our problem. Then the expected regret of TSG-OPVI at every episode $K > 0$ is bounded by:*

$$\mathbb{E}[\textit{Regret}(K)] \leq \mathcal{O}\left(MSH\overline{K} + SH\sqrt{AHK\log\left(SAK^2H\right)}\right),$$

where where $\bar{K}$ is the first integer such that $\overline{K} > \frac{\log\left(MS\overline{K}^2\right)}{-\log(1-d)}$.

Discussion on Theorem 10.3.1 The following regret nearly-matches the proposed lower bound. In fact, if we instantiate the set of policies of agent 1 equal to all the possible deterministic policies, then $M = A^S$ where A is the cardinality of the action space and S the cardinality of the state space. Instead, the second term of the regret is near to the worst-case lower bound for MDPs.

Proof of Theorem 10.3.1

Proof. We start by defining for every state $s \in \mathcal{S}$, policy π_1^i with $i \in [M]$, opponent's policy $\text{br}(\pi_1^i)$, and time step $h \in H$, $d_h^i(s)$ as the probability of visiting s under these policies. Then we define:

$$S_{2,h}^{+,i} = \{s \in \mathcal{S}_2 \text{ such that } d_h^i(s) > 0\}.$$

We define as $d = \min_{i\in[M]} \min_{h\in[H]} \min s \in S^{+,i} d_h^i(s)$, i.e., the minimum probability of visiting a "reachable" state.

Moreover, we define $N_{k,h}^i(s)$ as the number of time a state $s \in \mathcal{S}$ is visited playing policy π_1^i up to iteration $k - 1 \leq K$ and time step h.

Then we define for each policy π_1^i with $i \in [M]$ the following event:

$$\mathcal{E}_i = \{\forall h \in [H]\, \forall s \in S_{2,h}^{+,i} \text{ such that } N_{k,h}^i(s) > 0\},$$

i.e. under the event $\mathcal{E}_i$ every opponent's state is visited at least one time. Then we introduce the indicator random variable $\mathbb{1}\{\mathcal{E}_i\}$ which is equal to one if the event $\mathcal{E}_i$ is verified and 0 otherwise.

We can then decompose the regret as:

$$\begin{aligned}\text{Regret} &= \sum_{k=1}^K V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \\ &= \underbrace{\sum_{k=1}^K V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\{\neg\mathcal{E}_{\mathrm{I}_k}\}}_{A} + \underbrace{\sum_{k=1}^K V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\{\mathcal{E}_{\mathrm{I}_k}\}}_{B}\end{aligned}$$

We start by bounding the A part. We rewrite the regret making explicit its dependence on the policy i.

$$\begin{aligned}&\sum_{k=1}^K V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\{\neg\mathcal{E}_{\pi_{1,k}}\} \\ &= \sum_{k=1}^K \sum_{i\in[M]} V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\{\neg\mathcal{E}_{\pi_{1,k}}\} \mathbb{1}\{\pi_{1,k} = \pi_i\}. \qquad (10.6)\end{aligned}$$

Fixing a policy π_i we can say that:

$$\begin{aligned}
&\sum_{k=1}^{K} V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\left\{\neg\mathcal{E}_{\pi_{1,k}}\right\} \mathbb{1}\left\{\pi_{1,k} = \pi_i\right\} \\
&\leq \sum_{k=1}^{\overline{K}} V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\left\{\neg\mathcal{E}_{\pi_{1,k}}, \pi_{1,k} = \pi_i, N_k(\pi_i) \leq \overline{K}\right\} \\
&+ \sum_{k=\overline{K}}^{\infty} V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\left\{\neg\mathcal{E}_{\pi_{1,k}}, \pi_{1,k} = \pi_i, N_k(\pi_i) > \overline{K}\right\} \leq \overline{K}
\end{aligned}$$

where we use lemma 10.3.1 and $\overline{K}$ is the first $\overline{K}$ such that it fulfills the inequality $\overline{K} > \frac{\log\left(MSH^2\overline{K}\right)}{\log\left(\frac{1}{1-d}\right)}$.

Then we can bound equation 10.6.

$$\sum_{i\in[M]} \sum_{k=1}^{K} V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \mathbb{1}\left\{\neg\mathcal{E}_{\pi_{1,k}}\right\} \mathbb{1}\left\{\pi_{1,k} = \pi_i\right\} \leq MH\overline{K}.$$

Then we bound the term B. When the event $\mathcal{E}_i$ is verified we know what is the policy $br(\pi_1^i)$ that the opponent will play in every state $s \in \mathcal{S}_2^{+,i}$. Then, in practice, we are facing a single-agent problem where the single-agent policy is derived by the union of the two policies of the agents, i.e., the policy that we use is equal to:

$$\pi^i(s) = \begin{cases} \pi_1^i(s) & \text{if} \quad I(s) = 1 \\ \mathcal{A}_{k,h}^i(s) & \text{if} \quad I(s) = 2 \end{cases}, \tag{10.7}$$

where in this case $\mathcal{A}_{k,h}^i(s)$ is a singleton for every state $s \in \mathcal{S}_2$ and policy π_1^i with $i \in [M]$.

$$\sum_{k=1}^{K} V_1^{\pi_1^\star, br(\pi_1^\star)} - V_1^{\pi_{1,k}, br(\pi_{1,k})} \leq \sum_{k=1}^{K} \widetilde{V}_1^{\pi_{1,k}, br(\pi_{1,k})} - V_1^{\pi_{1,k}, br(\pi_{1,k})}$$

with probability $1-\delta$, since we used the optimism to bound the regret and the confidence intervals must be verified. In lemma 10.3.2 we proved that the confidence intervals are verified with probability $1-\delta$. Then, for a specific episode $k \leq K$ and time step $h \leq H$:

$$\begin{aligned}
&\widetilde{V}_{1,k,h}^{\pi_k}(s_{k,h}) - V_{1,k,h}^{\pi_k}(s_{k,h}) \\
&= \widehat{\mathcal{R}}_1(s_{k,h}, a_{k,h}) + b_{k,h}^r - \mathcal{R}_1(s_{k,h}, a_{k,h}) + \widehat{\mathcal{P}}(\cdot|s_{k,h}, a_{k,h})\widetilde{V}_{1,k,h}^{\pi_k} + b_{k,h}^{\mathcal{P}} - \mathcal{P}(\cdot|s_{k,h}, a_{k,h})V_{1,k,h}^{\pi_k} \\
&= \underbrace{b_{k,h}^r + \widehat{\mathcal{R}}_1(s_{k,h}, a_{k,h}) - \mathcal{R}_1(s_{k,h}, a_{k,h})}_{\Delta_{k,h}^R} + \underbrace{b_{k,h}^{\mathcal{P}} + (\widehat{\mathcal{P}}(\cdot|s_{k,h}, a_{k,h}) - \mathcal{P}(\cdot|s_{k,h}, a_{k,h}))\widetilde{V}_{1,k,h}^{\pi_k}}_{\Delta_{v,k}^P} \\
&\quad + \underbrace{\mathcal{P}(\cdot|s_{k,h}, a_{k,h})(\widetilde{V}_{1,k,h}^{\pi_k} - V_{1,k,h}^{\pi_k}) - (\widetilde{V}_{1,k,h+1}^{\pi_k}(s_{k,h+1}) - V_{1,k,h+1}^{\pi_k}(s_{k,h+1}))}_{\Delta_{k,h}^V} \\
&\quad + \widetilde{V}_{1,k,h+1}^{\pi_k}(s_{k,h+1}) - V_{1,k,h+1}^{\pi_k}(s_{k,h+1})
\end{aligned}$$

We are going to bound the different Δs terms. We call $\Delta_{k,h+1}^S(s) = \widetilde{V}_{1,k,h+1}^{\pi_k}(s) - V_{1,k,h+1}^{\pi_k}(s)$, and we can say that with probability $1-\delta$:

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\Delta^{V}_{k,h} = \sum_{k=1}^{K}\sum_{h=1}^{H}\mathop{\mathbb{E}}_{s\sim\mathcal{P}(\cdot|s_{k,h},a_{k,h})}[\Delta^{S}_{k,h+1}(s)] - \Delta^{S}_{k,h+1}(s_{k,h+1})$$
$$\leq \sqrt{2KH\log\left(\frac{1}{\delta}\right)}$$

where we apply Azuma-Hoeffding inequality since it is a martingale difference sequence.

For the second term $\Delta^{R}_{k,h}$ we apply the confidence intervals on the reward function:

$$\begin{aligned}\sum_{k=1}^{K}\sum_{h=1}^{H}\Delta^{R}_{k,h} &\leq \sum_{k=1}^{K}\sum_{h=1}^{H}2b^{R}_{k,h} = \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{s,a\in\mathcal{S}\times\mathcal{A}}2b^{R}_{k,h}\mathbb{1}\{s_{k,h}=s\}\mathbb{1}\{a_{k,h}=a\}\\ &= \sum_{k=1}^{K}\sum_{h=1}^{H}\sum_{s,a\in\mathcal{S}\times\mathcal{A}}2\sqrt{\frac{2\log(\frac{4SAHk}{\delta})}{N_k(s,a)}}\mathbb{1}\{s_{k,h}=s\}\mathbb{1}\{a_{k,h}=a\}\\ &\leq 2\sqrt{2\log(\frac{4SAHK}{\delta})}\sum_{s,a\in\mathcal{S}\times\mathcal{A}}\sum_{k=1}^{K}\sum_{h=1}^{H}\sqrt{\frac{1}{N_k(s,a)}}\mathbb{1}\{s_{k,h}=s\}\mathbb{1}\{a_{k,h}=a\}\\ &\leq 2\sqrt{2\log\left(\frac{4SAHK}{\delta}\right)}\left(\sum_{s,a\in\mathcal{S}\times\mathcal{A}}\sum_{i=1}^{N_K(s,a)}\sqrt{\frac{1}{i}}\right)\\ &= 4\sqrt{2SAKH\log\left(\frac{4SAHK}{\delta}\right)},\end{aligned}$$

with probability $1-\delta$.

The term Δ^{P}s can be bounded:

$$\sum_{k=1}^{K}\sum_{h=1}^{H}\Delta^{P}_{k,h} \leq \sum_{k=1}^{K}\sum_{h=1}^{H}2b^{P}_{k,h} = 2H\sum_{k=1}^{K}\sum_{h=1}^{H}\sqrt{\frac{2S\log\left(\frac{4SAHk}{\delta}\right)}{N_k(s,a)}} \leq 4HS\sqrt{2AKH\log\left(\frac{4SAHK}{\delta}\right)},$$

with probability $1-\delta$.

Putting everything together, and including the regret suffered on the events where the confidence intervals do not hold (which occur with probability at most δ):

$$\mathbb{E}[\text{Regret}(K)] \leq MH\overline{K} + (1-\delta)8SH\sqrt{2AHK\log\left(\frac{4SAHK}{\delta}\right)}$$
$$+ 2\delta KH + (1-\delta)\sqrt{2KH\log\left(\frac{1}{\delta}\right)} + \delta KH$$

Setting $\delta = \frac{1}{3KH}$ the result follows.

$\square$

Auxiliar lemmas In the following lemma we prove that if for every policy π_i with $i \in [M]$, every state $s \in \mathcal{S}_2^{+,i}$, at each time step $h \in [H]$ has the probability to be visited equal at least to d then after "enough" times the policy is played then every state reachable is visited at least one time.

Lemma 10.3.1. *For each policy π_i with $i \in [M]$ if $N_k(\pi_i) \geq \bar{K}$, where $\bar{K} \geq \frac{\log\left(\frac{MSH}{\delta}\right)}{\log\left(\frac{1}{1-d}\right)}$, then every state $s \in \mathcal{S}_2^{+,i}$ is visited at least one time with probability $1-\delta$.*

Proof. We start by bounding the probability that there is at least one state reachable that is not already visited at least one policy:

$$\mathrm{P}\{\exists s \in \mathcal{S}_2^{+,i}, \quad \exists \pi_1^i \text{ with } i \in [M] \text{ such that } N_{k,h}(s) = 0 \text{ and } N_k(\pi_1^i) \geq \overline{K}\} \leq \\ \sum_{i \in [M]} \sum_{s \in \mathcal{S}_2^{+,i}} \sum_{h \in [H]} \mathrm{P}\{N_{k,h}(s) = 0 \text{ and } N_k(\pi_1^i) \geq \overline{K}\} \leq MSH(1-d)^{\overline{K}}.$$

Since we want to say that this probability is less than δ:

$$\begin{aligned} &MSH(1-d)^{\overline{K}} \leq \delta \\ &\overline{K} \log\left(\frac{1}{1-d}\right) \geq \log\left(\frac{MSH}{\delta}\right) \\ &\overline{K} \geq \frac{\log\left(\frac{MSH}{\delta}\right)}{\log\left(\frac{1}{1-d}\right)}. \end{aligned}$$

Then the result follows. □

Then we provide the lemma for the confidence intervals:

Lemma 10.3.2. *The confidence intervals derived by the bonus b^r and $b^{\mathcal{P}}$ are verified with probability $1-\delta$.*

Proof. We recall that the bonus terms used are respectively:

$$b_k^r(s,a) = \sqrt{\frac{2\log\left(\frac{4SAHk}{\delta}\right)}{N_k(s,a)}} \qquad b_{k,h}^{\mathcal{P}}(s,a) = h\sqrt{\frac{2S\log\left(\frac{4SAHk}{\delta}\right)}{N_k(s,a)}}.$$

The bonus term are directly derived by Hoeffding concentration inequality and union bound.

□

10.4 Discussion

In this Chapter, we propose the first insights to the online learning problem in general-sum Stochastic Games. Although there are some recent results in solving the problem in the zero-sum (aka competitive) setting, there are no works that take into account that we could face a *weaker* opponent. We have shown that the problem is much more complicated then in a zero-sum game and an MDP. The main problems arise from the limited control on the environment's exploration. We underline this difficulties in our lower bound (Section 10.2) and we show how to build a provably efficient algorithm in Section 10.3.

Part IV

Policy Optimization in Multi-Agent Reinforcement Learning

When approaching the MARL problem from an optimization point of view, the objective is to provide efficient algorithms that, based on the system's dynamics, can converge to interesting solutions concepts (e.g., Nash Equilibrium points). For the single-agent setting, many policy-gradient algorithms [Deisenroth et al., 2013] were proposed to solve the RL problem. On the other hand, it was shown that the usage of these algorithms in the multi-agent setting is unsuccessful (e.g., they may not converge) [Mescheder et al., 2017, Mertikopoulos et al., 2018b, Adolphs et al., 2019, Mazumdar et al., 2019].

In Chapter 11 we introduce the setting, formalizing the solution concepts and extensively reviewing the state of the art. We start by introducing the Continuous Games and then we establish the connection with MARL and the relative policy (first order and second order) gradient estimators. Then, in Chapter 12 we propose a new algorithm [Ramponi and Restelli, 2021], called Newton Optimization on Helmholtz Decomposition, to solve the MARL problem. The algorithm is based on the classical Helmholtz decomposition. Initially, we propose two algorithms based on Newton optimization respectively for (exact) Potential Games and Hamiltonian Games. Then we propose an algorithm for General Games that *approximates* the game, at each step of the learning process, to its Potential or Hamiltonian component. For this algorithm, we provide an extensive experimental evaluation.

CHAPTER 11

Continuous games and gradient-based approaches

Thanks to their ability to learn in the stochastic policy space and their effectiveness in solving high-dimensional, continuous state and action problems, policy-gradient algorithms [Deisenroth et al., 2013] are natural candidates for adoption in MARL problems. Nonetheless, the interaction between multiple policy-gradient agents has proven unsuccessful in learning a set of policies that converges to a (local) Nash Equilibrium [Mertikopoulos et al., 2018b, Papadimitriou and Piliouras, 2016]. This problem is more general than MARL and includes learning in *continuous games*. Learning in continuous games is interesting not only from a MARL point of view but also for optimization problems with multiple loss functions. Examples are in Generative Adversarial Networks (GANs) [Goodfellow et al., 2014], which achieve successful results in Computer Vision [Isola et al., 2017, Ledig et al., 2017] and Natural Language Generation [Nie et al., 2018, Yu et al., 2017]; robust supervised learning [Madry et al., 2018, Tramèr et al., 2018] to contrast adversarial attacks to neural networks; and hyperparameter optimization [Maclaurin et al., 2015, Ravi and Larochelle, 2016] to tuning the hyperparameters of machine learning algorithms.

In this chapter, we start by formally introducing Continuous Games and the Helmholtz Decomposition for games [Candogan et al., 2011, Balduzzi et al., 2018], that we shall use in the next chapter to construct a learning algorithm for multi-agent problems. Then, we revise the related work on continuous games, and, finally, we expose how to cast MARL problems in the continuous game's setting.

11.1 Continuous Games

In this section, after introducing Continuous Games [Ratliff et al., 2013, Ratliff et al., 2016], we describe the desired convergence solutions. Then we recall the Hamiltonian game decomposition. The definitions are intended to be close to literature and remain close to the notation used in this thesis.

Definition 11.1.1 (Continuous Games). *A n-agents continuous game is defined by $(\Theta, C_1, \dots, C_n)$ where $\Theta = (\Theta_1, \dots, \Theta_n)$ are the parameters' space of each agent and $C_i : \Theta \rightarrow \mathbb{R}$, with $C_i \in C^q$ $q \geq 2$, is the cost function of the agent i.*

Moreover is usually assumed that C_i is globally Lipschitz continuous and all the derivatives in all its arguments are globally Lipschitz continuous.

We define the *simultaneous gradient* as the concatenation of the gradient of each cost function respect to the parameters of each agent:

$$\xi(\boldsymbol{\theta}) = (\nabla_{\boldsymbol{\theta}_1} C_1^T(\boldsymbol{\theta}), \dots, \nabla_{\boldsymbol{\theta}_n} C_n(\boldsymbol{\theta})^T)^T, \tag{11.1}$$

where with $\boldsymbol{\theta}$ we intend the concatenation of every agent's parameter, i.e., $\boldsymbol{\theta} \in \Theta$. Performing gradient descent on a stochastic game implies following the simultaneous gradient, as every player i updates its parameters with the component $\nabla_{\boldsymbol{\theta}_i} C_i(\boldsymbol{\theta})$.

The game *Jacobian* [Ratliff et al., 2013] $\mathcal{J}$ [1] is an $nd \times nd$ matrix, where n is the number of agents and d the number of policy parameters for each agent. The game Jacobian describes the dynamics of the game. $\mathcal{J}$ is composed by the matrix of the gradients of the simultaneous gradient, i.e., for each player i the i-th row of its hessian:

$$\mathcal{J} = \nabla_{\boldsymbol{\theta}} \xi(\boldsymbol{\theta}) = \begin{pmatrix} \nabla^2_{\boldsymbol{\theta}_1} C_1(\boldsymbol{\theta}) & \nabla_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2} C_1(\boldsymbol{\theta}) & \cdots & \nabla_{\boldsymbol{\theta}_1, \boldsymbol{\theta}_n} C_1(\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}_2, \boldsymbol{\theta}_1} C_2(\boldsymbol{\theta}) & \nabla^2_{\boldsymbol{\theta}_2} C_2(\boldsymbol{\theta}) & \cdots & \nabla_{\boldsymbol{\theta}_2, \boldsymbol{\theta}_n} C_2(\boldsymbol{\theta}) \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_{\boldsymbol{\theta}_n, \boldsymbol{\theta}_1} C_n(\boldsymbol{\theta}) & \nabla_{\boldsymbol{\theta}_n, \boldsymbol{\theta}_2} C_n(\boldsymbol{\theta}) & \cdots & \nabla^2_{\boldsymbol{\theta}_n} C_n(\boldsymbol{\theta}) \end{pmatrix}.$$

[1] We indicate the game Jacobian with $\mathcal{J}$ instead of J to be not confused with the expected discounted return.

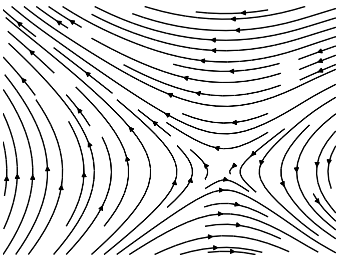


Figure 11.1: *Dynamics of gradient descent on a Potential game.*

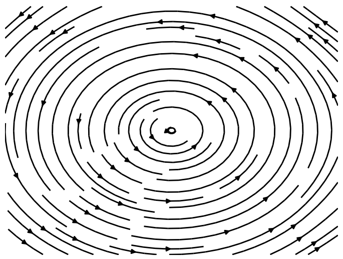


Figure 11.2: *Dynamics of gradient descent on an Hamiltonian game.*

11.1.1 Helmhotz game decomposition

The game Jacobian $\mathcal{J}(\boldsymbol{\theta})$ is a square matrix, not necessarily symmetric. The antisymmetric part of $\mathcal{J}(\boldsymbol{\theta})$ is caused by each agent's different cost functions and can cause cyclical behavior in the game (even in simple cases as bimatrix zeros-sum games, see Figure 11.2). On the other hand, the symmetric part represents the "cooperative" part of the game. The Jacobian $\mathcal{J}(\boldsymbol{\theta})$ can be decomposed into its symmetric and antisymmetric component using the Generalized Helmholtz decomposition [Wills, 1958] [2].

Proposition 11.1.1. *The Jacobian of a game decomposes uniquely into two components $\mathcal{J}(\boldsymbol{\theta}) = S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})$, where $S(\boldsymbol{\theta}) = \frac{1}{2}(\mathcal{J}(\boldsymbol{\theta}) + \mathcal{J}(\boldsymbol{\theta})^T)$ and $A(\boldsymbol{\theta}) = \frac{1}{2}(\mathcal{J}(\boldsymbol{\theta}) - \mathcal{J}(\boldsymbol{\theta})^T)$.*

Components $S(\boldsymbol{\theta})$ and $A(\boldsymbol{\theta})$ represent the irrotational (Potential), $S(\boldsymbol{\theta})$, and the solenoidal (Hamiltonian), $A(\boldsymbol{\theta})$, part of the game, respectively. The irrotational component is its curl-free component, and the solenoidal one is the divergence-free one. This decomposition naturally generates two classes of games: the one with the component $A(\boldsymbol{\theta}) = \mathbf{0}$ and the one with $S(\boldsymbol{\theta}) = \mathbf{0}$. We explain more in details these two classes below.

Potential games Potential games are a class of games introduced by [Monderer and Shapley, 1996]. A game is a potential game if there exists a potential function $\phi : \mathbb{R}^{n\times d} \to \mathbb{R}$, and a scalar $\alpha > 0$ such that: $\phi(\boldsymbol{\theta}_i^{'}, \boldsymbol{\theta}_{-i}) - \phi(\boldsymbol{\theta}_i^{''}, \boldsymbol{\theta}_{-i}) = \alpha(C(\boldsymbol{\theta}_i^{'}, \boldsymbol{\theta}_{-i}) - C(\boldsymbol{\theta}_i^{''}, \boldsymbol{\theta}_{-i}))$. A potential game is an *exact* potential game if $\alpha = 1$; exact potential games have $A = \mathbf{0}$. In these games, $\mathcal{J}$ is symmetrical and it coincides with the hessian of the potential function.

[2]The Helmholtz decomposition applies to any vector field [Wills, 1958].

This class of games is widely studied because in these games gradient descent converges to a Nash Equilibrium [Rosenthal, 1973, Lee et al., 2016]. In the rest of the document when we refer to potential games we refer to exact potential games.

Hamiltonian games Hamiltonian games are games with $S = \mathbf{0}$. A Hamiltonian game is described by a Hamiltonian function, which specifies the conserved quantity of the game. Formally, a Hamiltonian system is fully described by a scalar function, $\mathcal{H}(\mathbf{p}, \mathbf{q}) : \mathbb{R}^{n \times d} \to \mathbb{R}$. The state of a Hamiltonian system is represented by the generalized coordinates, i.e, momentum $\mathbf{q}$ and position $\mathbf{p}$, which are vectors of the same size. The evolution of the system is given by Hamilton's equations: $\frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}, \frac{d\mathbf{q}}{dt} = +\frac{\partial \mathcal{H}}{\partial \mathbf{p}}$. The gradient of $\mathcal{H}$ corresponds to $(S + A^T)\xi$ [Balduzzi et al., 2018]. In bimatrix games, Hamiltonian games coincide with zero-sum games, but this is not true in general games.

Discussion on previous works on Game dynamics decomposition As far as we know, the first work that proposed a decomposition for games is the work by [Candogan et al., 2011]. In this paper, the authors introduce how to apply the Hodge decomposition to finite games[3]. The paper shows that the game's flow can be decomposed into three components: the potential, the harmonic, and the non-strategic ones. Similar to what we have explained in the previous sections, the potential component represents the common interests between the players. The harmonic part, on the other hand (similarly to the hamiltonian), represents the conflicts of the players. Interestingly the authors obtain explicit expressions for the projection of games into their potential and hamiltonian components. Then in [Balduzzi et al., 2018] the authors propose to decompose a continuous game into its potential and hamiltonian components and exploit this decomposition to propose a novel algorithm that adjusts the gradient with the hamiltonian component of the game. In [Chasnov et al., 2020b] the authors propose a method to learn in two-players continuous games. In this paper, the authors decompose the game Jacobian to reflect the dynamic interaction between the two players. The idea is to decompose the matrix into two components, such that they perform a change of coordinates. The alternative coordinates more directly assess the conditions for stability of a Nash Equilibrium.

[3]The Hodge decomposition is related to the Helmholtz decomposition since it *generalized* the Helmholtz decomposition to differential forms on Riemannian manifold. However, it is not a real generalization since the Hodge decomposition requires that the manifold is compact.

11.1.2 Desired convergence points

In classic game theory, the standard solution concept is the Nash Equilibrium [Nash et al., 1950][4]. Since we focus on gradient-based methods and make no assumptions about the convexity of the cost functions, we consider the concept of local Nash Equilibrium [Ratliff et al., 2013].

Definition 11.1.2 (Local Nash Equilibrium)**.** *A point $\boldsymbol{\theta}^*$ is a local Nash Equilibrium if, for each agent $1 \leq i \leq n$, there is a neighborhood B_i of $\boldsymbol{\theta}_i^*$ such that $C_i(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_{-i}^*) \leq C_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*)$ for any $\boldsymbol{\theta}_i \in B_i$. If the above inequalities are strict, then we say $\boldsymbol{\theta}^*$ is a strict local Nash equilibrium.*

Gradient-based methods can reliably find local (not global) optima even in single-agent non-convex problems [Lee et al., 2016, Lee et al., 2017], but they may fail to find local Nash equilibria in non-convex games.

Another important solution concept is the concept of stability of the learning process. In this manuscript we adapt the concept of stability in dynamical system theory to stability of the learning process. The (local) stability is defined as follows [La Salle, 1976, Hespanha, 2018, Mazumdar et al., 2020b].

Definition 11.1.3 (Stable points)**.** *A fixed point $\boldsymbol{\theta}$ is locally (asymptotically) stable if and only if all the eigenvalues of $\mathcal{J}(\boldsymbol{\theta})$ have strictly positive real parts, i.e., all the eigenvalues live in the open right-half complex plane.*

We introduce now another desirable solution concept, used previously in the Machine Learning community [Balduzzi et al., 2018, Letcher et al., 2018, Letcher et al., 2019], that in this manuscript we call *symmetric stable fixed points*.

Definition 11.1.4 (Symmetric stable, symmetric unstable and saddle point)**.** *A fixed point $\boldsymbol{\theta}^*$ with $\xi(\boldsymbol{\theta}^*) = 0$ is symmetric stable if $S(\boldsymbol{\theta}^*) \succeq 0$, and $S(\boldsymbol{\theta}^*)$ is invertible, symmetric unstable if $S(\boldsymbol{\theta}^*) \prec 0$ and a saddle if $S(\boldsymbol{\theta}^*)$ has an eigenvalue with negative real part, but not all.*

It is important to notice that this notion of (local) *symmetric stability* is only a restrictive sufficient condition to the notion of (local) stability in dynamical system theory defined before. When considering only the symmetric stable points we are possibly removing many stable points. However, since this is an open active research area, considering the convergence also to these points can be of interest.

[4]We review basic solution concepts in appendix A

Lemma 11.1.1. *If $\boldsymbol{\theta}^\star$ is a symmetric stable point then $\boldsymbol{\theta}^\star$ is a local strict Nash Equilibrium.*

It is important, to notice that the symmetric stable points are a (possible very small) subset of all the local Nash Equilibria of a game.

11.2 Related work

In this section, we revise the literature on gradient-based learning algorithms for continuous games. We would like to mention that this is a research area that started to be of interest to the Machine Learning community in the last years, but the study of the dynamic property of continuous games is not new. In fact, the majority of the results in this area were achieved by the optimization, control and game theory community.

Convex and convex-concave continuous games The study of convergence in classic convex multi-player games has been extensively studied and analyzed [Rosen, 1965, Facchinei and Kanzow, 2007, Facchinei and Pang, 2007]. These algorithms are based on extragradient methods [Facchinei and Pang, 2007]. More recently, in [Bravo et al., 2018] the authors studied the learning behavior under the bandit feedback. The paper shows that no-regret learning based on mirror descent with bandit feedback converges to Nash equilibrium if the game is concave. In [Zhou et al., 2018] the authors study a setting where the feedback can be lost during the learning process and propose an algorithm based on online gradient descent that converges to the set of Nash Equilibrium points. Other recent works consider the problem of learning in convex or convex-concave games with different assumptions on the game [Héliou et al., 2020, Wan et al., 2021, Hsieh et al.,]. A very recent work [Hsieh et al., 2021] propose a new no-regret algorithm based on optimistic mirror descent, that in variationally stable games (so also in convex-concave zero-sum games) converges to Nash equilibrium with $O(1)$ individual regret. Unfortunately, the same algorithms cannot be used with neural networks due to the non-convexity of the objective functions.

Linear quadratic games There are many works that study linear-quadratic (LQ) continuous games. In LQ games, the particularity is that the cost function is quadratically dependent on the states and joint control actions. This games are particularly interesting since there are applications of them in robust control synthesis [Başar and Bernhard, 2008, Zhang et al., 2020a, Zhang et al.,], and risk-sensitive control [Jacobson, 1973, Whittle, 1981]. It was shown that even asymptotically, basic gradient-based approaches may not converge to (local) Nash equilibria or stationary points [Mazumdar et al., 2019, Mazumdar et al., 2020a]. In [Zhang et al., 2019b] the authors develop a projected nested-gradient-based algorithm that converges to the Nash Equilibrium of the game. In [Gravell et al., 2020] the authors analyze policy iteration algorithms to compute equilibrium strategies and value functions. In [Bu et al., 2019] Bu et al. propose a projection-free sequential algorithm to solve LQ games. The idea is to use the Stackelberg leadership model where the leader follows the natural gradient.

Zero-sum games Due to the great success of Generative Adversarial Networks [Goodfellow et al., 2014] many works on continuous games optimization are concentrated to two-player zero-sum continuous games setting. In this case the problem becomes a nonconvex-nonconcave saddle-point problem in general [Gemp and Mahadevan, 2018, Heusel et al., 2017, Adolphs et al., 2019, Mazumdar et al., 2019, Schäfer and Anandkumar, 2019, Bu et al., 2019], and various techniques to solve this problem were proposed [Adolphs et al., 2019, Mertikopoulos et al., 2018a, Mescheder et al., 2017, Yang et al., 2020, Diakonikolas et al., 2021]. In [Schäfer and Anandkumar, 2019] the authors provide an algorithm that, using second-order gradients, converges exponentially for convex-concave zero-sum games. Consensus Optimization [Mescheder et al., 2017] and Optimistic Mirror Descent [Mertikopoulos et al., 2018a] study methods to adjust the oscillatory dynamic. [Adolphs et al., 2019] and [Mazumdar et al., 2019] exploit the curvature of the functions to converge to local Nash Equilibria. In [Fiez and Ratliff, 2020] it was shown that gradient descent-ascent with finite timescale separation converges to local Minmax Equilibria. Moreover, algorithms for convergence in GANs were studied: [Heusel et al., 2017] use a two-timescale procedure to converge to local NE; [Gemp and Mahadevan, 2018] rely on variational inequalities to guarantee global convergence properties; [Nagarajan and Kolter, 2017] study the convergence properties of gradient descent, showing that it could find stable points. In [Lin et al., 2020] the authors study the dynamics of two-time-scale GDA for solving nonconvex-concave minimax games. It is an interesting result for training GANs. Instead, in [Jin et al., 2020b] the authors propose interesting insights into what is locally optimal for nonconvex-nonconcave minimax games (e.g, for GANs). In the paper, it is proposed the concept of *local minimax*, a proper mathematical definition of local optimality. Other recent works propose methods to learn in particular classes of zero-sum games: smooth-markets [Balduzzi et al., 2020], sequential imperfect information games [Perolat et al., 2020], zero-sum linear-quadratic games [Zhang et al., 2019b].

General sum games Recently there was a great interest in study gradient-optimization algorithms for non-convex games. This setting is also very challenging in the single-agent optimization case. However, the complexity increases in the multi-agent setting. In fact, it was proved that successful single-optimization techniques such as gradient-descent cannot always converge to a Local Nash Equilibrium due to the cyclic behavior of the dynamics around the NE points [Mescheder et al., 2017, Daskalakis et al., 2009, Mertikopoulos et al., 2018b, Adolphs et al., 2019, Mazumdar et al., 2020a, Letcher, 2020]. In [Chasnov et al., 2020a] using dynamic system theory tools, the authors study the converge properties to local Nash Equilibria of multi-agent gradient-based algorithms, also with non-uniform learning rates. Some recent works have developed learning algorithms also for general sum continuous games. The first example in literature is the Iterated Gradient Ascent Policy Prediction (IGA-PP) algorithm, renamed LookAhead [Zhang and Lesser, 2010]. In [Letcher et al., 2018] it is proved that IGA-PP converges to local Nash Equilibria not only in two-player two-action bimatrix games but also in general games. Learning with opponent learning awareness (LOLA) [Foerster et al., 2018] is an attempt to use the other agents' costs to account for the impact of one agent's policy on the anticipated parameter update of the other agents. The empirical results show the effectiveness of LOLA, but no convergence guarantees are provided. Indeed, [Letcher et al., 2018] has shown that LOLA may converge to non-fixed points and proposed Stable Opponent Shaping [Letcher et al., 2018], an algorithm that maintains the theoretical convergence guarantees of IGA-PP, also exploiting the opponent dynamics like LOLA. In [Balduzzi et al., 2018, Letcher et al., 2019] the authors studied game dynamics by decomposing a game into its Potential and Hamiltonian components using the generalized Helmholtz decomposition. The authors propose Symplectic Gradient Adjustment (SGA), an algorithm for general games, which converges locally to symmetric stable fixed points, using the Hamiltonian part to adjust the gradient update. [Fiez et al., 2020b] considered a two-timescale algorithm that converges to local Stackelberg Equilibrium points in general games, providing finite-time convergence analysis and stochastic convergence analysis. In [Mazumdar et al., 2020b] the authors propose a general framework for competitive gradient-based learning, such that it includes many multi-agent learning algorithms. In [Chasnov et al., 2019] the same authors report in an useful table many multi-agent learning updates rules.

11.3 MARL and policy-gradient algorithms

The described algorithms can be used as policy-gradient methods for MARL problems, casting the problems as a Continuous Stochastic Game.

Definition 11.3.1 (Continuous Stochastic Game). *A n-Continuous Stochastic Game is a tuple $\mathcal{CSG} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma_1, \dots, \gamma_n)$ where:*

- *n is the number of agents,*
- *$\mathcal{A}_i$, $1 \leq i \leq n$, is the set of actions of agent i and $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the joint action set;*
- *$\mathcal{S}$ is the set of states;*
- *$\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{A})$ is the state transition probability function,*
- *$\mathcal{R} = (\mathcal{R}_1, \dots, \mathcal{R}_n)$ is the set of cost (reward) functions for each agent, where $\mathcal{R}_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function of agent i [5],*
- *$\gamma = (\gamma_1, \dots, \gamma_n)$, where $\gamma_i \in [0, 1)$ is the discount factor for the agent i.*

The agent's behavior is described by means of a parametric twice differentiable policy $\pi_{\boldsymbol{\theta}_i} : \mathcal{S} \to \Delta(\mathcal{A}_i)$, where $\boldsymbol{\theta}_i \in \Theta_i \subseteq \mathbb{R}^d$ and $\pi_{\boldsymbol{\theta}_i}(\cdot|s)$ specifies for each state s a distribution over the action space $\mathcal{A}_i$.[6] We denote by $\boldsymbol{\theta}$ the vector of length nd obtained by stacking together the parameters of all the agents: $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^T, \dots, \boldsymbol{\theta}_n^T)^T$.

In continuous stochastic games, all agents try to minimize their expected discounted cost separately. It is defined for the i-th agent as:

$$C_i(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h \mathcal{R}(s_h, \mathbf{a}_h)\right] = \mathbb{E}\left[\sum_{h=0}^{H-1} \gamma^h \mathcal{R}(s_h, (a_{1,h}, \dots, a_{n,h}))\right]$$

where the expectation is taken with respect to $s_0 \sim \mu$, $s_{h+1} \sim \mathcal{P}(\cdot|s_h, a_h)$, $a_{i,h} \sim \pi_i(\cdot|s_h)$ for each $i \in [N]$. The continuous stochastic cost is the expected discounted return defined before in Chapter 2. However in this part we define it as *expected discounted cost* because the objective of the agent is to minimize it, instead of maximize it. We do not assume the convexity of the functions $C_i(\boldsymbol{\theta})$.

Usually, agents do not have access to the full gradient or the Jacobian, and we need to estimate them as for single-agent RL (see Chapter 2). Following

[5]We call the cost function of each agent $\mathcal{R}_i$ since then we define the expected discounted cost with C_i.

[6]To ease the notation, we will drop $\boldsymbol{\theta}$ (e.g., π_i instead of $\pi_{\boldsymbol{\theta}_i}$) when not necessary.

the derivation of G(PO)MDP [Peters and Schaal, 2008b], we estimate the gradient of the expected cost function $\nabla_{\theta_i} C_i(\boldsymbol{\theta})$:[7]

Proposition 11.3.1. *Given a differentiable parametric policy π_{θ_i} the policy gradient of the expected discounted cost of agent i respect to its own parameters θ_i can be estimated by:*

$$\widehat{\nabla}^M_{\boldsymbol{\theta}_i} C_i(\boldsymbol{\theta}) = \frac{1}{M}\sum_{m=1}^{M}\sum_{h=0}^{H-1}\left(\sum_{h'=0}^{h}\nabla_{\boldsymbol{\theta}_i}\log\pi_{\boldsymbol{\theta}_i}(a^m_{i,h'}|s^m_{h'})\gamma^h\mathcal{R}_i(s^m_h,\boldsymbol{a}^m_h)\right). \tag{11.2}$$

Proof. We derive the second order gradient when there are only two agents to ease readability. The extension to n players is trivial. We define with $\pi_i(\tau) = \prod_{h=0}^{H-1}\pi_i(a_h|s_h)$ and $\mathcal{P}(\tau) = \mu(s_0)\times\prod_{h=1}^{H-1}\mathcal{P}(s_h,a_h)$.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_1} C_1(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}_1}\mathop{\mathbb{E}}_{\tau}[\mathcal{R}_1(\tau)] \\
&= \nabla_{\boldsymbol{\theta}_1}\int_\tau \pi_1(\tau)\pi_2(\tau)\mathcal{P}(\tau)\mathcal{R}_1(\tau)\mathrm{d}\tau \\
&= \int_\tau \nabla_{\boldsymbol{\theta}_1}\pi_1(\tau)\pi_2(\tau)\mathcal{P}(\tau)\mathcal{R}_1(\tau)\mathrm{d}\tau \\
&= \int_\tau \frac{\pi_1(\tau)}{\pi_1(\tau)}\nabla_{\boldsymbol{\theta}_1}\pi_1(\tau)\pi_2(\tau)\mathcal{P}(\tau)\mathcal{R}_1(\tau)\mathrm{d}\tau \qquad (11.3)\\
&= \int_\tau \pi_1(\tau)\pi_2(\tau)\mathcal{P}(\tau)\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(\tau))\mathcal{R}_1(\tau)\mathrm{d}\tau \\
&= \mathop{\mathbb{E}}_{\tau}[\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(\tau))\mathcal{R}_1(\tau)] \\
&= \int_{s_0}\mu(s_0)\cdots\int_{s_{H-1}}\mathcal{P}(s_{H-1}|s_{H-2},a_{1,H-2},a_{2,H-2})\int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1}) \\
&\times\int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1})\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(a_{1,h}|s_h))\right) \\
&\times\left(\sum_{t=0}^{H-1}\gamma^t\mathcal{R}_1(s_t,a_{1,t},a_{2,t})\right)\mathrm{d}s_0\times\ldots\mathrm{d}a_{2,H-1} \\
&= \sum_{t=0}^{H-1}\Big(\int_{s_0}\mu(s_0)\cdots\int_{s_{H-1}}\mathcal{P}(s_{H-1}|s_{H-2},a_{1,H-2},a_{2,H-2})\int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1}) \\
&\times\int_{a_{2,H-1}}\pi(a_{2,H-1}|s_{H-1})\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(a_{1,h}|s_h))\right) \\
&\times\gamma^t\mathcal{R}_1(s_t,a_{1,t},a_{2,t})\mathrm{d}s_0\times\ldots\mathrm{d}a_{2,H-1}\Big), \qquad (11.4)
\end{aligned}$$

where in line 11.3 we use the log trick, in line 11.4 we write explicitly the expectation. Then, picking

[7]With $\widehat{\nabla}_{M,\boldsymbol{\theta}_i}$ we intend the estimator of $\nabla_{\boldsymbol{\theta}_i}$ over M samples.

the inner equation and selecting a specific $0 \leq t \leq H-1$:

$$
\begin{aligned}
&\int_{s_0} \mu(s_0) \cdots \int_{s_{H-1}} \mathcal{P}(s_{H-1}|s_{H-2}, a_{1,H-2}, a_{2,H-2}) \int_{a_{1,H-1}} \pi(a_{1,H-1}|s_{H-1}) \\
&\times \int_{a_{2,H-1}} \pi(a_{2,H-1}|s_{H-1}) \left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_1} \log(\pi_1(a_{1,h}|s_h)) \right) \mathcal{R}_1(s_t, a_{1,t}, a_{2,t}) \mathrm{d}s_0 \times \ldots \mathrm{d}a_{2,H-1} \\
&= \int_{s_0} \mu(s_0) \cdots \int_{s_t} \mathcal{P}(s_t|s_{t-1}, a_{1,t-1}, a_{2,t-1}) \int_{a_{1,t}} \pi(a_{1,t}|s_t) \\
&\times \int_{a_{1,t}} \pi(a_{1,t}|s_t) \gamma^t \mathcal{R}_1(s_t, a_{1,t}, a_{2,t}) \cdots \int_{a_{2,H-1}} \pi(a_{2,H-1}|s_{H-1}) \\
&\times \left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_1} \log(\pi_1(a_{1,h}|s_h)) \right) \mathrm{d}s_0 \times \ldots \mathrm{d}a_{2,H-1}
\end{aligned}
$$

Now the reader can verify that the integrals from $t+1$ to $H-1$ (multiplied by the log policy gradients in the same range) evaluate to 1. So the result follows.

□

Then, to estimate the Jacobian $\mathcal{J}$, we have to compute the second-order gradient $\nabla_{\boldsymbol{\theta}_i} \nabla_{\boldsymbol{\theta}_j} C_i$, with $1 \leq i \leq n$, $1 \leq j \leq n$ and $i \neq j$. We derive, as in [Foerster et al., 2018], the second-order gradient, exploiting the independence of agents' policies.

Proposition 11.3.2. *Given two diffentiable parametric policies $\pi_{\boldsymbol{\theta}_i}, \pi_{\boldsymbol{\theta}_j}$ the second-order policy gradient of the expected discounted cost of agent i respect to $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ can be estimated by:*

$$
\begin{aligned}
\widehat{\nabla}^M_{\boldsymbol{\theta}_i,\boldsymbol{\theta}_j} \sum_{h'=0}^{h} C_i = &\frac{1}{M} \sum_{m=1}^{M} \sum_{h=0}^{H-1} \left(\widehat{\nabla}^M_{\boldsymbol{\theta}_i} \log(\pi_{\boldsymbol{\theta}_i}(a^m_{i,h'}|s^m_{h'})) \right) \left(\widehat{\nabla}^M_{\boldsymbol{\theta}_j} \log(\pi_{\boldsymbol{\theta}_j}(a^m_{j,h'}|s^m_{h'})) \right)^T \\
&\times \gamma^h \left(\mathcal{R}_i(s^m_h, \boldsymbol{a}^m_h) \right).
\end{aligned}
$$

Proof. We derive the second order gradient when there are only two agents to ease readability. The extension to n players is trivial. We define with $\pi_i(\tau) = \prod_{h=0}^{H-1} \pi_i(a_h|s_h)$ and $\mathcal{P}(\tau) = \mu(s_0) \times \prod_{h=1}^{H-1} \mathcal{P}(s_h, a_h)$.

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} C_1(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} \mathop{\mathbb{E}}_{\tau} [\mathcal{R}_1(\tau)] \\
&= \nabla_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} \int_{\tau} \pi_1(\tau)\pi_2(\tau)\mathcal{P}(\tau)\mathcal{R}_1(\tau)\mathrm{d}\tau \\
&= \int_{\tau} \nabla_{\boldsymbol{\theta}_1}\pi_1(\tau)\nabla_{\boldsymbol{\theta}_1}\pi_2(\tau)\mathcal{P}(\tau)\mathcal{R}_1(\tau)\mathrm{d}\tau \\
&= \int_{\tau} \frac{\pi_1}{\pi_1}\nabla_{\boldsymbol{\theta}_1}\pi_1(\tau)\frac{\pi_1}{\pi_1}\nabla_{\boldsymbol{\theta}_2}\pi_2(\tau)\mathcal{P}(\tau)\mathcal{R}_1(\tau)\mathrm{d}\tau
\end{aligned}
$$

$$
\begin{aligned}
&= \int_{\tau} \pi_1(\tau)\pi_2(\tau)\mathcal{P}(\tau)\left(\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(\tau))\right)^T \nabla_{\boldsymbol{\theta}_2}\log(\pi_2(\tau))\mathcal{R}_1(\tau)\mathrm{d}\tau \\
&= \mathop{\mathbb{E}}_{\tau}\left[\left(\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(\tau))\right)^T \nabla_{\boldsymbol{\theta}_2}\log(\pi_2(\tau))\mathcal{R}_1(\tau)\right] \\
&= \int_{s_0}\mu(s_0)\cdots\int_{s_{H-1}}\mathcal{P}(s_{H-1}|s_{H-2},a_{1,H-2},a_{2,H-2})\int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1}) \\
&\times \int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1})\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(a_{1,h}|s_h))\right)^T\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_2}\log(\pi_2(a_{2,h}|s_h))\right) \\
&\times\left(\sum_{t=0}^{H-1}\gamma^t\mathcal{R}_1(s_t,a_{1,t},a_{2,t})\mathrm{d}s_0\times\ldots\mathrm{d}a_{2,H-1}\right) \\
&= \sum_{t=0}^{H-1}\Big(\int_{s_0}\mu(s_0)\cdots\int_{s_{H-1}}\mathcal{P}(s_{H-1}|s_{H-2},a_{1,H-2},a_{2,H-2})\int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1}) \\
&\times \int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1})\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(a_{1,h}|s_h))\right)^T\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_2}\log(\pi_2(a_{2,h}|s_h))\right) \\
&\times\gamma^t\mathcal{R}_1(s_t,a_{1,t},a_{2,t})\mathrm{d}s_0\times\ldots\mathrm{d}a_{2,H-1}\Big)
\end{aligned}
$$

picking the inner equation and selecting a specific $0 \leq t \leq H-1$:

$$
\begin{aligned}
&\int_{s_0}\mu(s_0)\cdots\int_{s_{H-1}}\mathcal{P}(s_{H-1}|s_{H-2},a_{1,H-2},a_{2,H-2})\int_{a_{1,H-1}}\pi(a_{1,H-1}|s_{H-1}) \\
&\times \int_{a_{2,H-1}}\pi(a_{2,H-1}|s_{H-1})\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(a_{1,h}|s_h))\right)^T\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_2}\log(\pi_2(a_{2,h}|s_h))\right) \\
&\times\gamma^t\mathcal{R}_1(s_t,a_{1,t},a_{2,t})\mathrm{d}s_0\times\ldots\mathrm{d}a_{2,H-1} \\
&= \int_{s_0}\mu(s_0)\cdots\int_{s_{H-1}}\mathcal{P}(s_t|s_{t-1},a_{1,t-1},a_{2,t-1})\int_{a_{1,t}}\pi(a_{1,t}|s_t) \\
&\times\int_{a_{1,t}}\pi(a_{1,t}|s_t)\gamma^t\mathcal{R}_1(s_t,a_{1,t},a_{2,t})\cdots\int_{a_{2,H-1}}\pi(a_{2,H-1}|s_{H-1})\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_1}\log(\pi_1(a_{1,h}|s_h))\right)^T \\
&\times\left(\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}_2}\log(\pi_2(a_{2,h}|s_h))\right)\mathrm{d}s_0\times\ldots\mathrm{d}a_{2,H-1}
\end{aligned}
$$

Now the reader can verify that the integrals from $t+1$ to $H-1$ (multiplied by the log policy gradients in the same range) evaluate to 1. So the result follows. □

When $i = j$ we are evaluating the second-order gradient of $\pi_{\boldsymbol{\theta}_i}$. To evaluate this part we derive the second-order gradient as done for the single-agent case [Shen et al., 2019].

Proposition 11.3.3. *Given a twice differentiable parametric policy π_{θ_1} the second-order policy gradient of the expected discounted cost function of agent i respect to its own parameters $\boldsymbol{\theta}_i$ can be estimated by:*

$$\widehat{\nabla}^{2,M}_{\boldsymbol{\theta}_i} C_i = \frac{1}{M}\sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}_i} g_i(\boldsymbol{\theta}_i, \tau_m)\left(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_i} \log \pi_{\boldsymbol{\theta}_i}(a_h^m | s_h^m)\right)^T + \nabla^2_{\boldsymbol{\theta}_i} g_i(\boldsymbol{\theta}_i, \tau_m)$$

where $g_i(\boldsymbol{\theta}_i, \tau) = \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}_i} \log(\pi_{\boldsymbol{\theta}_i}(s_h, a_{i,h})) \sum_{h'=h}^{H-1} \mathcal{R}(s_{h'}, a_{i,h'})$.

Proof. We derive the second order gradient when there are only two agents to ease readability. The extension to n players is trivial. The derivation follow similar steps of [Shen et al., 2019].

As we have shown previously in Proposition 11.3.1:

$$\nabla_{\boldsymbol{\theta}_1} C_1(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau}\left[\left(\sum_{h=0}^{H-1} \log(\pi_1(a_h|s_h))\right)\left(\sum_{h=0}^{H-1} \gamma^h \mathcal{R}_1(s_h, a_h)\right)\right]$$

We start calling:

$$g_1(\boldsymbol{\theta}_1, \tau) = \sum_{h=0}^{H-1} \log(\pi_1(a_{1,h}|s_h) \sum_{h'=h}^{H-1} \gamma^h \gamma^h \mathcal{R}_1(s_{h'}, a_{1,h'})$$

Using this notation we can rewrite the gradient of the expected discounted cost as:

$$\nabla_{\boldsymbol{\theta}_1} C_1(\boldsymbol{\theta}) = \mathop{\mathbb{E}}_{\tau}\left[\nabla_{\boldsymbol{\theta}_1} g_1(\boldsymbol{\theta}_1, \tau)\right] = \int_\tau \mathcal{P}(\tau)\pi_1(\tau)\pi_2(\tau)\nabla_{\boldsymbol{\theta}_1} g_1(\boldsymbol{\theta}_1, \tau)\mathrm{d}\tau$$

So we can compute the second order gradient as:

$$\begin{aligned}\nabla^2_{\boldsymbol{\theta}_1} C_1(\boldsymbol{\theta}) &= \int_\tau \mathcal{P}(\tau)\pi_2(\tau)\nabla_{\boldsymbol{\theta}_1} g_1(\boldsymbol{\theta}_1, \tau)(\nabla_{\boldsymbol{\theta}_1}\pi_1(\tau))^T + \mathcal{P}(\tau)\pi_1(\tau)\pi_2(\tau)\nabla^2_{\boldsymbol{\theta}_1} g_1(\boldsymbol{\theta}_1, \tau)\mathrm{d}\tau \\ &= \mathop{\mathbb{E}}_{\tau}\left[g_1(\boldsymbol{\theta}_1, \tau)(\nabla_{\boldsymbol{\theta}_1}\pi_1(\tau))^T + \nabla^2_{\boldsymbol{\theta}_1} g_1(\boldsymbol{\theta}_1, \tau)\right]\end{aligned}$$

□

Convergence results for MARL The convergence result of the algorithms described in Section 11.2 are in general developed for general continuous games, where the functions can have various forms, as long as they are differentiable and in most of the cases Lipschitz continuous with respect to agent's policy parameters. In a MARL context this means that the expected discounted cost has to have these differentiability/continuity properties. These properties are very restrictive in general, e.g. even the smoothness assumption fails to hold for LQ games [Zhang et al., 2019b, Mazumdar et al., 2020a, Bu et al., 2019]. For LQ games in some works [Zhang et al., 2019b, Bu et al., 2019] are convergence results are proved working on the particular structure of the game. Moreover, the most of the results are proved for the exact setting and do not study the stochastic one. However [Fiez et al., 2020b] provide convergence results also in the stochastic framework leveraging on stochastic approximation and dynamic system theory [Borkar, 2009].

CHAPTER 12

Newton Optimization On Helmhotz Decomposition

In this chapter, we study how to build a Newton-based algorithm for learning policies in multi-agent problems. First of all, we start by analyzing two specific game classes: Potential Games and Hamiltonian Games. Then we propose a Newton-based update for these two classes of games, for which linear-rate algorithms that guarantee convergence are known, proving quadratic convergence rates. Then, we extend the algorithm to the general case, neither Hamiltonian nor Potential. We show that the proposed algorithm, called Newton Optimization on Helmhotz Decomposition (NOHD), fulfills some desiderata, similar to those proposed in [Balduzzi et al., 2018]: the algorithm has to guarantee convergence to (local) Nash Equilibria in (D1) Potential and (D2) Hamiltonian games; (D3) the algorithm has to be attracted by symmetric stable fixed points and (D4) repelled by symmetric unstable fixed points. Finally, in Section 12.3, we analyze the empirical performance of NOHD when agents optimize either a linear policy or a Boltzmann policy, in three bimatrix games: Prisoner's Dilemma, Matching Pennies, and Rock-Paper-Scissors. Then we study the learning performance of NOHD in two continuous gridworld environments. Finally, we show

how the algorithm performs in a simple GAN problem. In all experiments, NOHD achieves great results confirming the quadratic nature of the update.

12.1 Newton method for non-convex functions

Newton's method [Nocedal and Wright, 2006] guarantees, under mild assumptions, a quadratic convergence rate to the root of a function. In optimization this technique is used to obtain a quadratic convergence rate to a fixed point of a function. Obviously, if the function is convex, it corresponds to converge to the global minimum of the function. The Newton method is based on a second-order approximation of the twice differentiable function $g(\boldsymbol{\theta})$ that we are optimizing. Starting from an initial guess $\boldsymbol{\theta}_0$, Newton's method updates the parameters $\boldsymbol{\theta}$ by setting the derivative of the second-order Taylor approximation of $g(\boldsymbol{\theta})$ to 0:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \nabla^2 g(\boldsymbol{\theta}_t)^{-1} \nabla g(\boldsymbol{\theta}_t). \tag{12.1}$$

For non-convex functions, the hessian $\nabla^2 g(\boldsymbol{\theta})$ is not necessarily positive semidefinite and all fixed points are possible solutions for Newton's method. So Newton's update in non-convex functions may converge to a local minimum, a saddle, or a local maximum. A possible way to avoid this shortcoming is to use a modified version of the inverse of the hessian, called Positive Truncated inverse (PT-inverse) [Nocedal and Wright, 2006, Paternain et al., 2019]:

Definition 12.1.1 (PT-inverse)**.** *Let $H \in \mathbb{R}^{n\times n}$ be a symmetric matrix, $Q \in \mathbb{R}^{n\times n}$ a basis of orthogonal eigenvectors of H, and $\Lambda \in \mathbb{R}^{n\times n}$ a diagonal matrix of corresponding eigenvalues. The $|\Lambda|_m \in \mathbb{R}^{n\times n}$ is the positive definite truncated eigenvalue matrix of Λ with parameter m:*

$$(|\Lambda|_m)_{ii} = \begin{cases} |\Lambda_{ii}| & if |\Lambda_{ii}| \geq m, \\ m & \textit{otherwise.} \end{cases} \tag{12.2}$$

The PT-inverse of H with parameter m is the matrix $|H|_m^{-1} = Q|\Lambda|_m^{-1}Q^T$.

The PT-inverse flips the sign of negative eigenvalues and truncates small eigenvalues by replacing them with m. Then the usage of the PT-inverse, instead of the real one, guarantees convergence to a local minimum even in non-convex functions [Paternain et al., 2019]. These properties are necessary to obtain a convergent Newton-like method for non-convex functions.

12.2 Newton for Games

In this section, we describe how to apply Newton-based methods to Continuous Games. We start by showing how Newton's method can be applied to two-game classes, which we extensively described in section 11.1: Potential games and Hamiltonian Games. Then we describe an algorithm to extend Newton's method in general games.

12.2.1 Newton's method for Potential games

In this section, with the word Potential games, we are indicating Exact Potential games. Potential games are a class of games characterized by the existence of a potential function $\phi : \mathbb{R}^{n\times d} \to \mathbb{R}$ which describes the dynamics of the system. In these games, gradient descent on the simultaneous gradient or the potential function converges to a (local) Nash Equilibrium as shown in Figure 11.1. The Jacobian of a Potential game is equal to its symmetric component, i.e, $\mathcal{J}(\boldsymbol{\theta}) = S(\boldsymbol{\theta})$, since the anti-symmetric component $A(\boldsymbol{\theta}) = \mathbf{0}$. Considering the existence of the potential function ϕ (see Chapter 11), it is sufficient to apply the Newton PT-inverse method (see Definition 12.1.1) on it to guarantee a quadratic convergence rate to a local Nash Equilibrium (see Figure 11.1). Therefore, the Newton's update for Potential games is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha S_m(\boldsymbol{\theta})^{-1}\xi(\boldsymbol{\theta}_t), \tag{12.3}$$

where α is a learning rate and m is the parameter of the PT-inverse (see Section 12.1). So, in Potential games, we have the same convergence properties as in single-function optimization. The convergence into the local minima follows Newton's convergence proofs for non-convex functions [Paternain et al., 2019].

12.2.2 Newton's method for Hamiltonian games

Hamiltonian games are characterized by an Hamiltonian function $\mathcal{H}(\mathbf{p}, \mathbf{q}) : \mathbb{R}^{n\times d} \to \mathbb{R}$. In these games, the gradient descent does not converge to a symmetric stable fixed point but causes cyclical behavior. Instead, the gradient descent on the Hamiltonian function converges to a Nash equilibrium. Figure 11.2 shows the dynamics of gradient descent w.r.t. ξ and $\nabla\mathcal{H}$ on a Hamiltonian game: the figure points out that a gradient descent on ξ cycles.

Example 12.2.1. *Take a two-player bilinear game with agents with parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ minimizing respectively $f(\boldsymbol{\theta}) : \mathbb{R}^{n\times d} \to \mathbb{R}$ and $g(\boldsymbol{\theta})$:*

$\mathbb{R}^{n\times d} \to \mathbb{R}$ respectively. A point in this class of games is a Nash Equilibrium if $\xi(\boldsymbol{\theta}) = 0$, i.e., $\nabla_{\boldsymbol{\theta}_1} f = 0$ and $\nabla_{\boldsymbol{\theta}_2} g = 0$, because $\nabla^2_{\boldsymbol{\theta}_1} f = 0$ and $\nabla^2_{\boldsymbol{\theta}_2} g = 0$. Considering this, the Nash Equilibrium can be calculated in closed form (if the inverse exists[1]) by setting the gradient equal to zero:

$$\begin{bmatrix}\nabla_{\boldsymbol{\theta}_1} f\\ \nabla_{\boldsymbol{\theta}_2} g\end{bmatrix} + \begin{bmatrix}0 & \nabla_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} f\\ \nabla_{\boldsymbol{\theta}_2,\boldsymbol{\theta}_1} g & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_1\\ \boldsymbol{\theta}_2\end{bmatrix} = 0$$

$$\begin{bmatrix}\boldsymbol{\theta}_1\\ \boldsymbol{\theta}_2\end{bmatrix} = -\begin{bmatrix}0 & \nabla_{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2} f\\ \nabla_{\boldsymbol{\theta}_2,\boldsymbol{\theta}_1} g & 0\end{bmatrix}^{-1}\begin{bmatrix}\nabla_{\boldsymbol{\theta}_1} f\\ \nabla_{\boldsymbol{\theta}_2} g\end{bmatrix}.$$

The example above provides the intuition that the solution to quadratic Hamiltonian games is achieved by the following update rule (in Hamiltonian games $S = 0$):

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha A(\boldsymbol{\theta}_k)^{-1}\xi(\boldsymbol{\theta}_k). \tag{12.4}$$

In the following theorem, we state that even in this class of games the local convergence to a local Nash Equilibrium (using the above update) is quadratic (see Figure 11.2).

Theorem 12.2.1. *Suppose that ξ is twice continuosly differentiable and that A is invertible in the (local) Nash Equilibrium $\boldsymbol{\theta}^*$. Then, there exists $\epsilon > 0$ such that iterations starting from any point in the ball $\boldsymbol{\theta}_0 \in B(\boldsymbol{\theta}^*, \epsilon)$ with center $\boldsymbol{\theta}^*$ and ray ϵ converge to $\boldsymbol{\theta}^*$. Furthermore, the convergence rate is quadratic.*

Proof. The proof is the standard proof of convergence for Newton's methods. We report here the steps for completeness.

In the following derivations with $||M||$ we indicate the operator norm of a matrix M defined as:

$$||M|| := \max_x\{||Mx|| \text{ such that } ||x|| = 1\}$$

Since ξ is twice differentiable, its Taylor series expansion in $\boldsymbol{\theta}_0$ is:

$$\xi(\boldsymbol{\theta}) = \xi(\boldsymbol{\theta}_0) + A(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \mathcal{O}(||\boldsymbol{\theta} - \boldsymbol{\theta}_0||^2) \tag{12.5}$$

Because A is continuously differentiable then $\exists\epsilon_1, M > 0$ s.t. if we take $\boldsymbol{\theta}_0, \boldsymbol{\theta} \in B(\boldsymbol{\theta}^*, \epsilon_1)$ then we have that:

$$||\xi(\boldsymbol{\theta}) - \xi(\boldsymbol{\theta}_0) - A(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)|| \leq M\,||\boldsymbol{\theta} - \boldsymbol{\theta}_0||^2. \tag{12.6}$$

Since A is twice continuously differentiable and by assumption it is invertible in $\boldsymbol{\theta}^*$ then there exists ϵ_2, N s.t. $\forall\boldsymbol{\theta} \in B(\boldsymbol{\theta}^*, \epsilon_2)$ A^{-1} exists and $\left\|A^{-1}(\boldsymbol{\theta})\right\| \leq N$ (see lemma 5.3 in [Chong and Zak, 2004]).

Let $\epsilon = \min(\epsilon_1, \epsilon_2)$. Now, we substitute $\boldsymbol{\theta}$ with $\boldsymbol{\theta}^*$ in 12.6 and we use the assumption that $\xi(\boldsymbol{\theta}^*) = 0$:

$$||\mathrm{A}(\boldsymbol{\theta}_0)(\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*) - \xi(\boldsymbol{\theta}_0)|| \leq M\,||\boldsymbol{\theta} - \boldsymbol{\theta}_0||^2. \tag{12.7}$$

[1]If the inverse does not exist and a Nash Equilibrium exists, the system is indeterminate. Using the Moore-Penrose inverse we find an approximate Nash Equilibrium (the one with the smallest Euclidean norm).

Algorithm 14 NOHD

```
input: discounted costs $C = \{C\}_{i=1}^{n}$, PT inverse parameter $m$
output: update rule
  Compute $\xi, \mathcal{J}, S, A, S_m^{-1}$
  if $\cos \nu_S \geq 0$ then
    if $\cos \nu_S \geq \cos \nu_A$ then (12.3) else (12.4)
  else
    if $\cos \nu_S \leq \cos \nu_A$ then (12.3) else (12.4)
  end if
```

If we use the update rule and we take $\boldsymbol{\theta}_1 \in B(\boldsymbol{\theta}^*, \epsilon)$ we have that:

$$\begin{aligned} \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}^*\| &= \left\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^* - \mathrm{A}^{-1}(\boldsymbol{\theta}_0)\xi(\boldsymbol{\theta}_0)\right\| \\ &= \left\|\mathrm{A}(\boldsymbol{\theta}_0)^{-1}(\mathrm{A}(\boldsymbol{\theta}_0)(\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*) - \xi(\boldsymbol{\theta}_0))\right\| \\ &\leq \left\|\mathrm{A}(\boldsymbol{\theta}_0)^{-1}\right\| \|(\mathrm{A}(\boldsymbol{\theta}_0)(\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*) - \xi(\boldsymbol{\theta}_0))\|, \end{aligned} \tag{12.8}$$

where in the last step we used the triangular inequality. If we used the inequalities 12.6 we have that:

$$\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}^*\| \leq MN \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2 . \tag{12.9}$$

If we suppose that $\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\| \leq \frac{\alpha}{MN}$ with $\alpha \in (0, 1)$, then:

$$\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}^*\| \leq \alpha \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|^2 . \tag{12.10}$$

Then by induction we obtain that:

$$\|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}^*\| \leq \alpha \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\|^2 . \tag{12.11}$$

Hence $\lim_{k\to\infty} \|\boldsymbol{\theta}_k - \boldsymbol{\theta}^*\| = 0$ and therefore the sequence $\boldsymbol{\theta}_k$ converges to $\boldsymbol{\theta}^*$ if we take $\epsilon \leq \frac{\alpha}{MN}$, and the order of convergence is at least 2. □

12.2.3 Newton's method for General games

In general games, it is not yet known whether and how the system's dynamics can be reduced to a single function as for Potential and Hamiltonian games. Thus, finding a Newton-based update is more challenging: if we apply Newton's Method with the Jacobian PT-transformation, we can alter the Hamiltonian dynamics of the game. Instead, applying the inverse of the Jacobian as in the Hamiltonian games can lead to local maxima. In this section, we show how to build a Newton-based learning rule that guarantees desiderata similar to those considered in [Balduzzi et al., 2018]:

(D1): the update rule has to be compatible with Potential dynamics if the game is a Potential game;

(D2): the update rule has to be compatible with Hamiltonian dynamics if the game is a Hamiltonian game;

(D3): the update rule has to be attracted by symmetric stable fixed points;

(D4): the update rule has to be repelled by symmetric unstable ones.

With compatible we mean that given two vectors u, v then $u^T v > 0$.

The algorithm that we propose (see Algorithm 14) chooses the update to perform between the two updates in (12.3) and (12.4). The choice is based on the angles between the gradient of the Hamiltonian function $\mathcal{H}$ and the two candidate updates' directions. In particular, we compute

$$\cos \nu_S = \frac{(S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T \nabla \mathcal{H}(\boldsymbol{\theta})}{\|S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta})\|_2 \, \|\nabla \mathcal{H}(\boldsymbol{\theta})\|_2}, \ \cos \nu_A = \frac{(A^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T \nabla \mathcal{H}(\boldsymbol{\theta})}{\|A^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta})\|_2 \, \|\nabla \mathcal{H}(\boldsymbol{\theta})\|_2}.$$

When the cosine is positive, the update rule follows a direction that reduces the value of the Hamiltonian function (i.e., reduces gradient norm), otherwise the update rule points in an increasing direction of the Hamiltonian function. This is an important fact since there is a connection between the positive/negative definiteness of $S(\boldsymbol{\theta})$ and the sign of $\cos \nu_S$.

Lemma 12.2.1. *Given the Jacobian $\mathcal{J}(\boldsymbol{\theta}) = S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})$ and the simultaneous gradient ξ, if $S(\boldsymbol{\theta}) \succeq 0$ then $\cos \nu_S \geq 0$; instead if $S(\boldsymbol{\theta}) \prec 0$ then $\cos \nu_S < 0$.*

Proof. We know that $\cos \nu_S = \frac{(S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T \nabla \mathcal{J}}{\|S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta})\|_2 \|\nabla \mathcal{J}(\boldsymbol{\theta})\|_2}$; then the sign of $\cos \nu_S$ depends on $(S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T \nabla \mathcal{J}(\boldsymbol{\theta})$. Suppose that $S(\boldsymbol{\theta}) \succeq 0$. We show that if $S(\boldsymbol{\theta}) \succeq 0$ then $S_m^{-1}(\boldsymbol{\theta})(S^T(\boldsymbol{\theta}) + A^T(\boldsymbol{\theta})) \succeq 0$:

$$\begin{aligned} S_m^{-1}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T) &= S_m^{-\frac{1}{2}}(\boldsymbol{\theta}) S_m^{-\frac{1}{2}}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T) \\ &= S_m^{-\frac{1}{2}}(\boldsymbol{\theta})(S_m^{-\frac{1}{2}}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T) S_m^{-\frac{1}{2}})(\boldsymbol{\theta}) S_m^{\frac{1}{2}}(\boldsymbol{\theta}). \end{aligned}$$

We use the fact that $S_m^{-1}(\boldsymbol{\theta})$ is positive definite by construction. So there exists a unique square root matrix $S_m^{-1/2}(\boldsymbol{\theta})$ that is symmetric. Then the matrix $S_m^{-1}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T)$ is similar to $S_m^{-1/2}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T) S_m^{-1/2}(\boldsymbol{\theta})$. For every vector $u \in \mathbb{R}^{n \times d}$:

$$u^T S_m^{-1/2}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T) S_m^{-1/2}(\boldsymbol{\theta}) u = z(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T) z \geq 0,$$

where $z = u^T S_m^{-1/2}(\boldsymbol{\theta}) = S_m^{-1/2}(\boldsymbol{\theta}) u$ because $S_m^{-1/2}(\boldsymbol{\theta})$ is symmetric. Using the same reasoning it is shown that if $S(\boldsymbol{\theta}) \prec 0$ then $S_m^{-1}(\boldsymbol{\theta})(S^T(\boldsymbol{\theta}) + A^T(\boldsymbol{\theta})) \prec 0$. □

The idea of NOHD is to use the sign of $\cos \nu_S$ to decide whether to move in a direction that reduces the Hamiltonian function (aiming at converging to a symmetric stable fixed point) or not (aiming at getting away from unstable points), since the angles indicate also if $S(\boldsymbol{\theta})$ is positive semidefinite or not. In case $\cos \nu_S$ is positive, the algorithm chooses the update rule with the largest cosine value (i.e., which minimizes the angle with $\nabla \mathcal{H}$), otherwise, NOHD tries to point in the opposite direction by taking the update rule that minimizes the cosine. In the following theorem, we show that the update performed by NOHD satisfies the desiderata described above.

Theorem 12.2.2. *The NOHD update rule satisfies requirements (D1), (D2), (D3), and (D4).*

Proof. **D1** NOHD has to be compatible with Potential game dynamics if the game is a Potential game: $(S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T\nabla\phi > 0$ and $(A^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T\nabla\phi > 0$, ϕ is the potential function. We notice that $\nabla\phi = \xi(\boldsymbol{\theta})$ and that $A(\boldsymbol{\theta}) = 0$ because the game is a Potential game. $\xi(\boldsymbol{\theta})^T S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}) > 0$ for every $\xi(\boldsymbol{\theta}) \neq \mathbf{0}$ since $S_m^{-1}(\boldsymbol{\theta})$ is positive definite by construction.

D2 NOHD has to be compatible with Hamiltonian game dynamics if the game is a Hamiltonian game: $(S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}))^T\nabla\mathcal{H}(\boldsymbol{\theta}) > 0$ and $(A^{-1}\xi)^T(\boldsymbol{\theta})\nabla\mathcal{H}(\boldsymbol{\theta}) > 0$. We know that $\nabla\mathcal{H}(\boldsymbol{\theta}) = (S(\boldsymbol{\theta})^T + A(\boldsymbol{\theta})^T)\xi(\boldsymbol{\theta})$ and that $S(\boldsymbol{\theta}) = 0$ because the game is a Hamiltonian game. Then $\xi(\boldsymbol{\theta})^T(A^{-1}(\boldsymbol{\theta}))^T(A(\boldsymbol{\theta})^T)\xi(\boldsymbol{\theta}) = \|\xi(\boldsymbol{\theta})\|^2$.

D3 NOHD has to be attracted to symmetric stable fixed points. It means that if $S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})$ is positive definite, and so, $\xi(\boldsymbol{\theta})^T(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta}))\xi(\boldsymbol{\theta}) > 0$. Then $S(\boldsymbol{\theta}) \succ 0$. From Lemma 12.2.1 we know that also $\xi(\boldsymbol{\theta})^T S(\boldsymbol{\theta})_m^{-1}(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T)\xi > 0$ so $\cos_{\nu_S} > 0$. The update rule takes the $\max(\cos_{\nu_S}, \cos_{\nu_A})$ that from the previous consideration is always positive.

D4 NOHD has to be repelled by symmetric unstable fixed points. It means that if $S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})$ is negative definite, and so, $\xi(\boldsymbol{\theta})^T(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta}))\xi(\boldsymbol{\theta}) \leq 0$. Then $S(\boldsymbol{\theta}) \prec 0$. From Lemma 12.2.1 we know that also $\xi(\boldsymbol{\theta})^T S_m^{-1}(\boldsymbol{\theta})(S(\boldsymbol{\theta}) + A(\boldsymbol{\theta})^T)\xi \leq 0$ so $\cos_{\nu_S} < 0$. The update rule take the $\min(\cos_{\nu_S}, \cos_{\nu_A})$ that from the previous consideration is always strictly negative. □

Given the results from Theorem 12.2.2 and Lemma 12.2.1, we can argue that if $\xi(\boldsymbol{\theta})$ points at a symmetric stable fixed point then NOHD points also to the symmetric stable fixed point otherwise if $\xi(\boldsymbol{\theta})$ points away from the fixed point also NOHD points away from it.
Then we prove that NOHD converges only to fixed points and, under mild conditions, it converges locally to symmetric stable fixed points.

Lemma 12.2.2. *If NOHD converges to a $\boldsymbol{\theta}^*$ then $\xi(\boldsymbol{\theta}^*) = \mathbf{0}$.*

Proof. Suppose that $\boldsymbol{\theta}^*$ is not a fixed point, so $\xi(\boldsymbol{\theta}^*) \neq 0$. The process is stopped in $\boldsymbol{\theta}^*$ if and only if $S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta}) = \mathbf{0}$ and $A(\boldsymbol{\theta})^{-1}\xi(\boldsymbol{\theta}) = \mathbf{0}$, because if $\xi(\boldsymbol{\theta})^T A^{-1}(\boldsymbol{\theta}) = \mathbf{0}$ and $\xi(\boldsymbol{\theta})^T S_m^{-1}(\boldsymbol{\theta}) \neq \mathbf{0}$ we always take $-S_m^{-1}(\boldsymbol{\theta})\xi(\boldsymbol{\theta})$ as update since $\|\cos_{\nu_S}\| \geq \|\cos_A\|$. $\xi(\boldsymbol{\theta})^T S_m^{-1}(\boldsymbol{\theta}) = \mathbf{0}$ only if $\xi(\boldsymbol{\theta}) = \mathbf{0}$ because $S_m^{-1}(\boldsymbol{\theta})$ is positive definite by construction, then we contradict the hypothesis.

We have to mention that with this lemma we prove only that the convergence points of the game are fixed points, i.e. $\xi(\boldsymbol{\theta}) = \mathbf{0}$. □

Theorem 12.2.3. *Suppose $\boldsymbol{\theta}^*$ is a symmetric stable fixed point, and suppose $A, S, \mathcal{J}$ are bounded and Lipschistz continuous with modulus respectively $M_A, M_S, M_{\mathcal{J}}$ in the region of attraction of the symmetric stable fixed point $\xi(\boldsymbol{\theta}^*)$. Furthermore assume that $A(\boldsymbol{\theta}^*)$ is invertible. Then there exists $\epsilon > 0$ such that the iterations starting from any point $\boldsymbol{\theta}_0 \in B(\boldsymbol{\theta}^*, \epsilon)$ converge to $\boldsymbol{\theta}^*$.*

Proof. Since ξ is differentiable by assumption we can write:

$$\xi(\boldsymbol{\theta}) - \xi(\boldsymbol{\theta}^*) = \int_0^1 [\mathcal{J}(\boldsymbol{\theta}_n + t(\boldsymbol{\theta}^* - \boldsymbol{\theta}_n))](\boldsymbol{\theta}^* - \boldsymbol{\theta}_n)dt.$$

Since for construction, $S(\boldsymbol{\theta}_0)_m^{-1}$ always exists, we can say that there exists a constant N_S such that $S(\boldsymbol{\theta})_m^{-1} \leq N_S$. Moreover, since A is twice continuously differentiable and by assumption it is invertible in $\boldsymbol{\theta}^*$ then there exists ϵ, N_A s.t. $\forall \boldsymbol{\theta} \in B(\boldsymbol{\theta}^*, \epsilon)$ $A(\boldsymbol{\theta})^{-1}$ exists and $\left\| A^{-1}(\boldsymbol{\theta}) \right\| \leq N_A$ (see lemma 5.3 in [Chong and Zak, 2004]). So we have:

$$\begin{aligned}
\boldsymbol{\theta}_1 - \boldsymbol{\theta}^* &= \boldsymbol{\theta}_0 - S(\boldsymbol{\theta}_0)_m^{-1}\xi(\boldsymbol{\theta}_0) - \boldsymbol{\theta}^* \\
&= \boldsymbol{\theta}_0 + S(\boldsymbol{\theta}_0)_m^{-1}(\xi(\boldsymbol{\theta}_0) - \xi(\boldsymbol{\theta}^*)) - \boldsymbol{\theta}^* \\
&= \boldsymbol{\theta}_0 - \boldsymbol{\theta}^* S(\boldsymbol{\theta}_0)_m^{-1} \int_0^1 \left(\mathcal{J}(\boldsymbol{\theta}_0 + t(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0))\right)(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0)dt \\
&= S(\boldsymbol{\theta}_0)_m^{-1} \int_0^1 \left(\mathcal{J}(\boldsymbol{\theta}_0 + t(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0)) - S(\boldsymbol{\theta}_0)\right)(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0)dt.
\end{aligned}$$

Taking the norm and supposing that the current update is with $S(\boldsymbol{\theta})_m^{-1}$

$$\begin{aligned}
\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}^*\| &\leq \left\|S(\boldsymbol{\theta}_0)_m^{-1}\right\| \int_0^1 \|\mathcal{J}(\boldsymbol{\theta}_0 + t(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0)) - S(\boldsymbol{\theta}_0)\| \, \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_n\| \, dt \\
&= \left\|S(\boldsymbol{\theta}_0)_m^{-1}\right\| \int_0^1 \|\mathcal{J}(\boldsymbol{\theta}_0 + t(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0)) - \mathcal{J}(\boldsymbol{\theta}_0) + A(\boldsymbol{\theta}_0)\| \, \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \, dt \\
&\leq \left\|S(\boldsymbol{\theta}_0)_m^{-1}\right\| \left(\int_0^1 \|\mathcal{J}(\boldsymbol{\theta}_0 + t(\boldsymbol{\theta}^* - \boldsymbol{\theta}_0)) - \mathcal{J}(\boldsymbol{\theta}_0)\| \, \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \, dt + \right. \\
&\qquad \left. + \int_0^1 \|[A(\boldsymbol{\theta}_0)]\| \, \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \, dt \right) \\
&\leq \left\|S(\boldsymbol{\theta})_m^{-1}\right\| \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \left(\int_0^1 Lt \, \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \, dt + \int_0^1 M_A dt \right) \\
&\leq N_S L \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\|^2 + N_A M_A \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \leq (N_S L + N_A M_A) \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| .
\end{aligned}$$

If we suppose that $\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| \leq \frac{\alpha}{(N_S L + N_A M_A)}$ with $\alpha \in (0, 1)$, then:

$$\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_1\| \leq \alpha \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\| .$$

Then by induction:

$$\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_n\| \leq \alpha \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_{n-1}\| .$$

Hence, $\lim_{n\to\infty} \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_n\| = 0$, so if we take $\epsilon \leq \frac{\alpha}{\max(N_S L + N_A M_A, N_A L + N_S M_S)}$ the sequence converges. □

12.2.4 Discussion on desired convergence points

In this paper, we focus our attention on convergence towards symmetric stable fixed points. As we have discussed in Chapter 11, these points are only a subset of the stable fixed points of the learning dynamics. On the other hand, the field of policy optimization methods for MARL is an active research area that is far from being completely understood. We think that analyzing the dynamics of the system and providing an algorithm with interesting (although limited) convergence properties and solid empirical results is a step forward. Moreover, symmetric stable fixed points are an interesting solution concept since in two-player zero-sum games (e.g., GANs [Goodfellow et al., 2014]), all local Nash Equilibria are also symmetric stable fixed points [Balduzzi et al., 2018].

In the following experiments, we show that NOHD achieves a fast convergence to equilibrium also in the General games setting. Motivated by the empirical successes, we think that the analysis can be improved to show better convergence rates.

Estimation of jacobians The proposed algorithm uses estimations of second-order statistics, which have, as it is well known, the problem of high variance. However, in the experimental evaluation, we do not see significant damage due to the variance (also in the continuous gridworld environment, which is stochastic), maybe because every algorithm for policy optimization in games uses second-order quantities. However, reducing the variance of gradient and jacobians estimation acquires great importance in the MARL context.

12.3 Experiments

This section is devoted to the experimental evaluation of NOHD. The proposed algorithm is compared with Consensus Optimization (CO) [Mescheder et al., 2017], Stable Opponent Shaping (SOS) [Letcher et al., 2018], Learning with Opponent-Learning Awareness (LOLA) [Foerster et al., 2018], Competitive Gradient Descent (CGD) [Schäfer and Anandkumar, 2019], Iterated Gradient Ascent Policy Prediction (IGA-PP) [Zhang and Lesser, 2010] and Symplectic Gradient Adjustment (SGA) [Balduzzi et al., 2018]. We start comparing the algorithm in simple matrix games. Then we show the behavior of all the algorithms in two continuous Gridworlds. After having empirically compared the approaches also from a computational point of view, we show an experiment using Generative Adversarial Networks.

12.3.1 Matrix games

We consider three matrix games: two-agent two-action Matching Pennies (MP), and Dilemma and two-agent three-action Rock Paper Scissors (RPS) (games' rewards are reported in tables 12.1,12.2,12.3). Considering a linear parameterization of the agents' policies, MP and RPS are Hamiltonian games, while Dilemma is a Potential game. The Nash equilibria of MP,

	Head	Tail
Head	1 -1	-1 1
Tail	-1 1	1 -1

Table 12.1: *Matching pennies rewards for the two agents.*

	Head	Tail
Head	-1 -1	-3 0
Tail	0 -3	-2 -2

Table 12.2: *Dilemma rewards for the two agents.*

	Rock	Paper	Scissors
Rock	0 0	-1 -1	1 -1
Paper	1 -1	0 0	-1 1
Scissors	-1 1	1 -1	0 0

Table 12.3: *RPS rewards for the two agents.*

Dilemma, and RPS are respectively 0.5, 0, 0.333 regarding the probability of taking action 1.

Matching pennies linear parametrization We start by reporting in Figures 12.1 the behavior of NOHD and the other benchmarks in a linear parametrization of Matching Pennies game. As you can see NOHD, as CGD, converges to the Nash Equilibrium in only one step. We show the best results searching between learning rates $1.0, 0.5, 0.1, 0.05$.

Boltzmann parametrization exact gradients Then we used a Boltzmann parametrization for the agents' policies and exact computation of gradients and Jacobian. In this setting, games lose their Hamiltonian or Potential property, making the experiment more interesting and the behavior of NOHD not trivial. The results are shown in Figure 12.9. For each game, we perform experiments with learning rates $0.1, 0.5, 1.0$. In the plots are reported only the best performance for each algorithm. In Matching Pennies and Dilemma we initialize probabilities to $[0.86, 0.14]$ for the first agent and to $[0.14, 0.86]$ for the second agent; instead in Rock Paper Scissors to $[0.66, 0.24, 0.1]$. The figure shows that each algorithm is able to converge to the Nash equilibrium. In MP, CO converges with a learning rate of 0.1 and takes more than 1000 iterations. In Dilemma, all algorithms except NOHD converge in a similar number of steps, so the lines overlap. NOHD converges in less than 50 iterations across all games, outperforming other algorithms. In Table 12.4 we reported the ratio between the number of steps each algorithm takes to converge and the maximum number of steps in which the slowest algorithm converges. For this simulation, we sampled 50 random initializations of the parameters from a normal distribution with zero mean and standard

	NOHD	CGD	LOLA	IGA-PP	CO	SOS	SGA
MP	**0.49**	0.84	1.00	0.99	0.99	0.99	0.99
RPS	**0.38**	0.97	0.88	1.00	0.81	0.80	0.96

Table 12.4: *Ratio between the mean convergence steps to Nash Equilibrium and the maximum mean convergence steps.* 50 *runs, sampling from a normal distribution* $\mathcal{N}(0, 0.5^2)$.

$\lvert\boldsymbol{\theta}\rvert$	NOHD	IGAPP	LOLA	SOS	SGA	CGD	CO
4	0.7205	0.6979	0.6983	0.7186	0.7302	0.7265	0.7006
16	0.7898	0.7787	0.7735	0.7906	0.8358	0.8051	0.7758
36	1.1416	1.0625	1.0874	1.0705	1.1992	1.1444	1.1066
64	1.9486	1.6342	1.6162	1.6735	1.9555	1.8955	1.8850
100	3.4070	2.7734	2.6905	2.8169	3.4799	3.3126	3.2977
144	5.9191	4.6260	4.4351	4.8438	6.5164	5.8440	5.7876

Table 12.5: *Computation time of one learning update of each algorithm with increasing parameter space size. 20 runs.*

deviation 0.5. These results show that NOHD significantly outperforms other algorithms even when starting from random initial probabilities. Then we perform another experiment on Matching pennies 12.4 and Rock Paper Scissors with 20 different starting probabilities. It demonstrates how all the algorithms converge to the Nash Equilibrium but NOHD takes less than 100 iterations.

Boltzmann parametrization estimated gradients In the second experiment, the gradients and the Jacobian are estimated from samples. The starting probabilities are the same as in the previous experiment. We performed 20 runs for each setting. In each iteration, we sampled 300 trajectories of length 1. Figures 12.9 shows that NOHD also in this experiment converges to the equilibrium in less than 100 iterations. Instead, the other algorithms exhibit oscillatory behaviors. Then we perform another experiment on Matching pennies 12.6 and Rock Paper Scissors 12.7 with 20 different starting probabilities, sampled from a Normal distribution with mean 0 and standard deviation 1. In this case we estimate the gradient and the Jacobian using 300 sampled trajectories. The results show how all the algorithms succeed in converging to the Nash Equilibrium, but NOHD converges in less than 100. It demonstrates how all the algorithms converge to the Nash Equilibrium but NOHD takes less than 100 iterations.

12.3.2 Continuous gridworlds

This experiment aims at evaluating the performance of NOHD in two continuous Gridworld environments. The first Gridworld is the continuous version of the Gridworld 2 proposed in [Hu and Wellman, 2003]: the two agents are initialized in the two opposite lower corners and have to reach the same goal; when one of the two agents reaches the goal, the game ends, and this agent gets a positive reward. Each agent has to keep a distance of no less than 0.5 with the other agent and, if they decide to move to the same region, they cannot perform the action. In the second Gridworld the agents are initialized in the two lower corners and have to reach the same goal, but they have to reach the goal with a ball. An agent can take the opponent's ball if their distance is less than 0.5; the ball is randomly given to one of the two agents at the beginning of each episode. The agents' policies are Gaussian policies, linear in a set of respectively 72 and 68 radial basis functions, which generate the ν angle for the step's direction. For each experiment, we perform 10 runs with random initialization. The learning rate is settled at 0.01, the batch size is 100, the horizon is 30 and the discount factor is 0.96. In Figure 12.10 we compare the performance of NOHD with CO, IGA-PP, LOLA, SOS, and CGD in the two gridworlds. The figure shows the mean average win of the two players at each learning iteration of the algorithms. In figure 12.10, at the top, we can see how NOHD outperforms the other algorithms in the first gridworld, converging in less than 30 steps to the equilibrium. At the bottom, results for the second gridworld are shown: NOHD converges quickly than the other algorithms, but Consensus has comparable performance. Then we report the second experiment where the second player is always NOHD. Figure 12.11 shows that all algorithms converge to a stable policy against NOHD or NOHD learn a winner policy against the algorithm.

12.3.3 Generative Adversarial Network

Finally, compare NOHD with SGA in an experiment adapted from the one in [Balduzzi et al., 2018]. In this simple experiment data are sampled from a mixture of 4 bi-variate Gaussians with means: $(1.5, -1, 5)$, $(1.5, 1, 5)$, $(-1.5, 1.5)$, $(-1.5, -1.5)$. The generator and discriminator networks are both with two ReLu layers with 10 neurons per layer. The output of the discriminator has size 1 and the output of the generator has size 2. The learning rate is 0.01. We report that NOHD finds all the modes in 400 steps and we compare these results with SGA. The results shown below are for random seed 25.

12.3.4 Computational time

In Table 12.5 we report the computation time of an update of each algorithm with increasing policy parameter sizes $|\boldsymbol{\theta}|$, from 4 to 144. As we can see, the computation time of NOHD is comparable to that of the other algorithms. Obviously, the time to compute the inverse of the Jacobian is highly demanding. However we can approximate this computation using iterative methods as [Schäfer and Anandkumar, 2019, Nocedal and Wright, 2006] or conjugate gradient techniques [Hestenes et al., 1952]. We leave the exploration of these methods as future work.

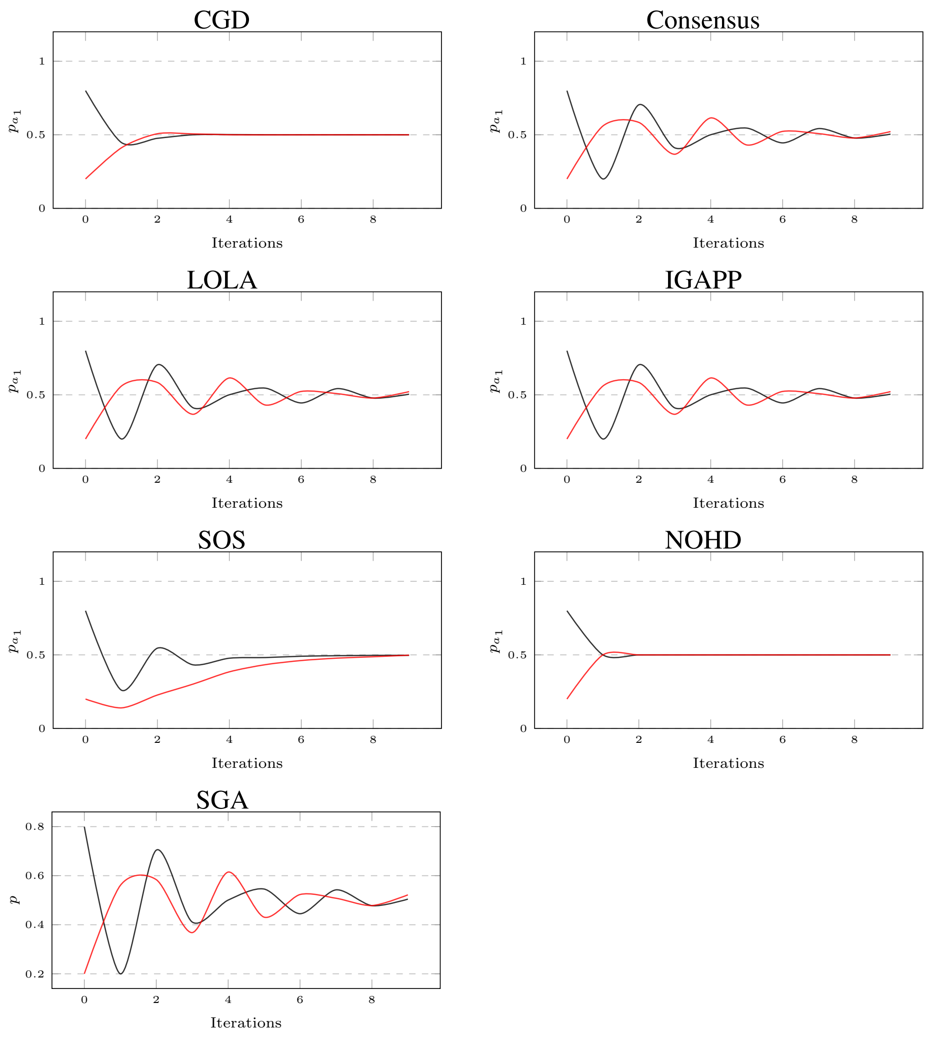


Figure 12.1: *Agents' probabily to perform action* 1 *in Matching Pennies. The initial probabilities are settled to* 0.8 *and* 0.2.

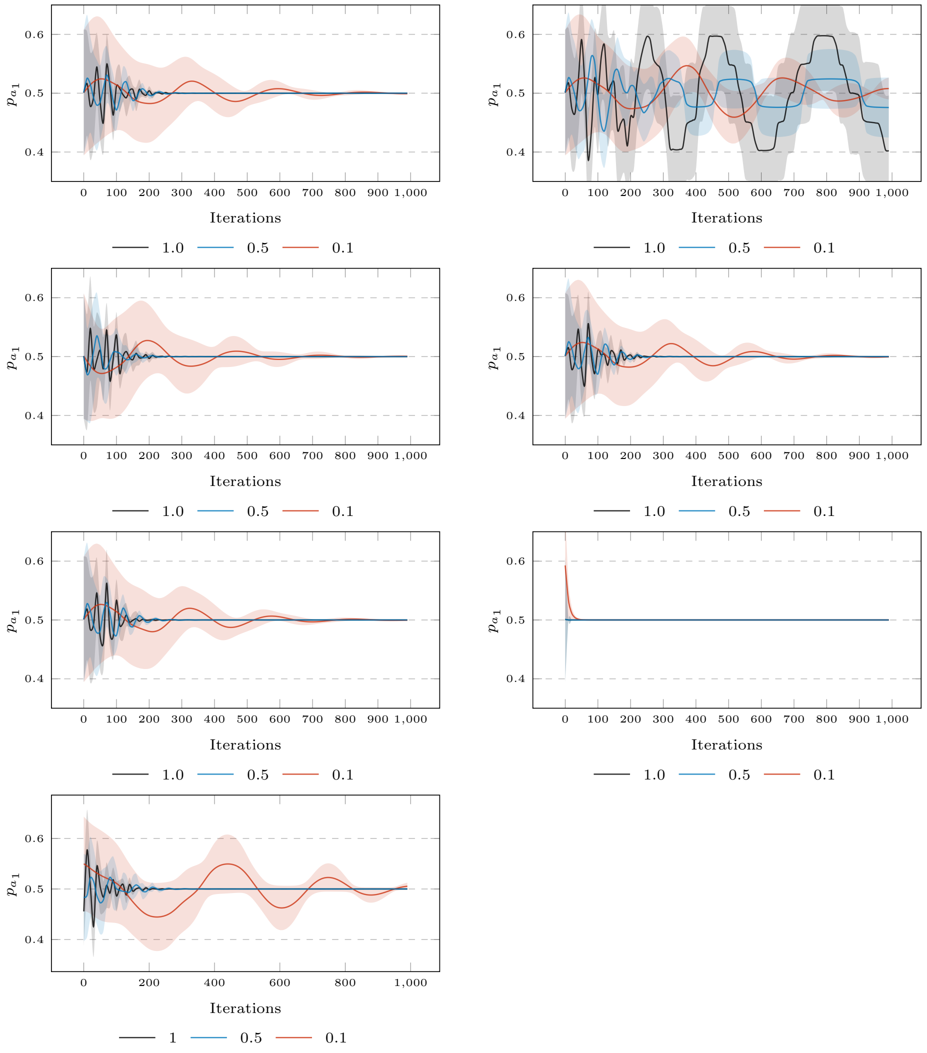


Figure 12.2: *Agent 1's probabily to perform action* 1 *in Matching Pennies. From top right to bottom left: CGD, consensus, LOLA, IGAPP, SOS, NOHD, and SGA. Exact gradients.* 20 *random sampled initial probabilities. Learning rate* 0.1, 0.5, 1.0*. 95% c.i.*

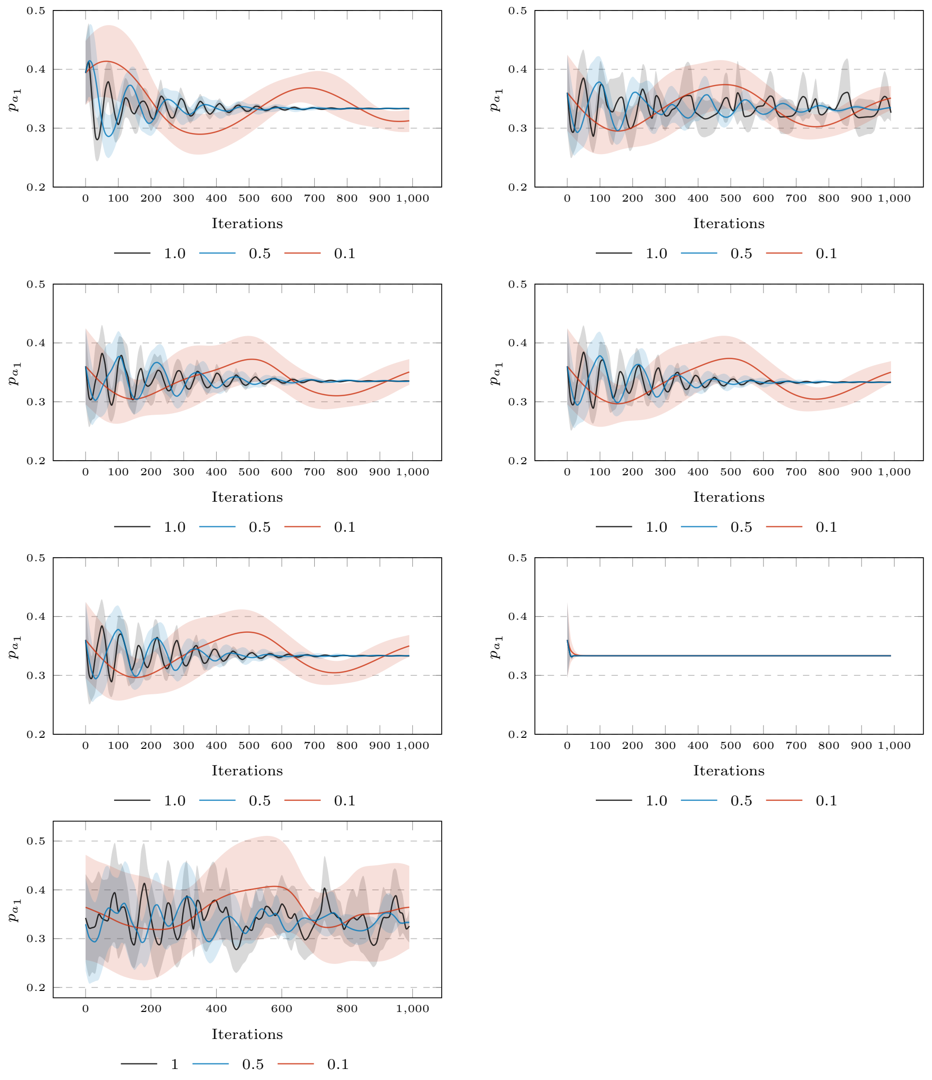


Figure 12.3: *Agent 1's probabily to perform action* 1 *in Rock Paper Scissors. From top right to bottom left: CGD, consensus, LOLA, IGAPP, SOS, NOHD, and SGA. Exact gradients batch size* 300*.* 20 *random sampled initial probabilities. Learning rate* $0.1, 0.5, 1.0$*. 95% c.i.*

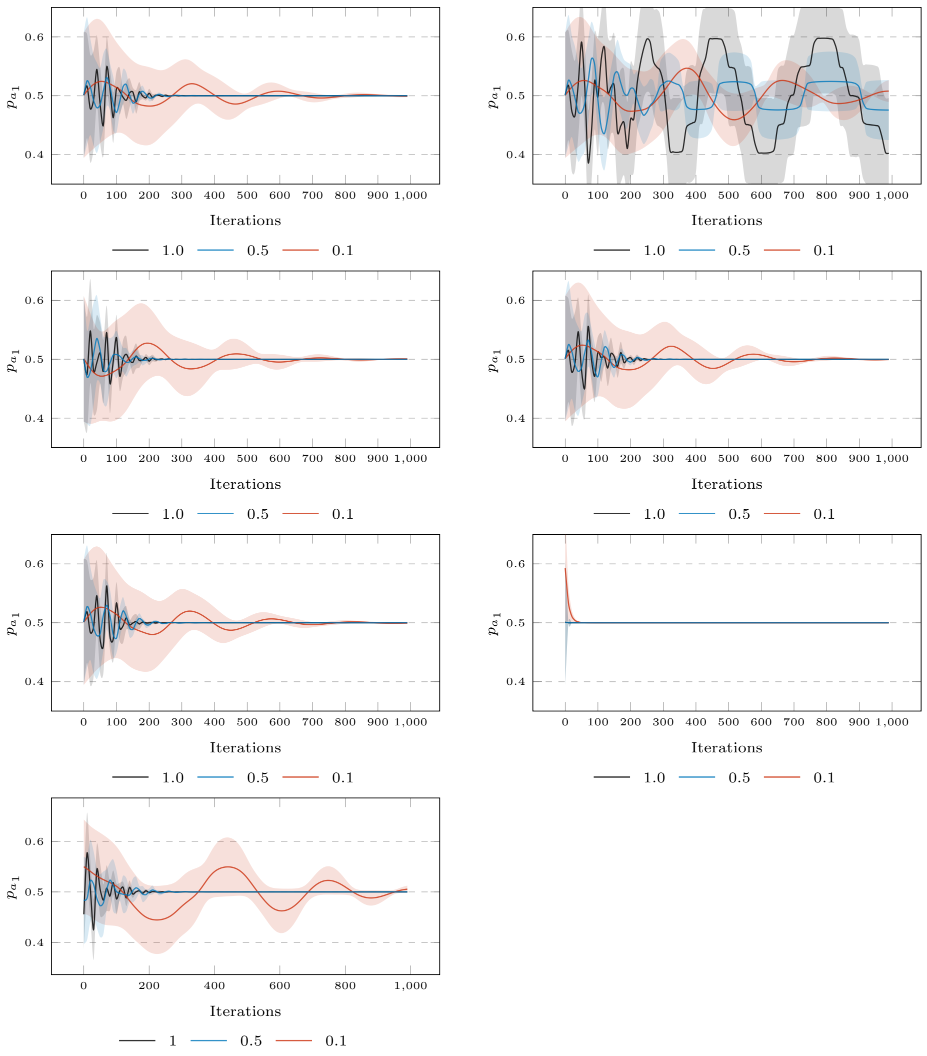


Figure 12.4: *Agent 1's probabily to perform action* 1 *in Matching Pennies. From top right to bottom left: CGD, consensus, LOLA, IGAPP, SOS, NOHD, and SGA. Exact gradients.* 20 *random sampled initial probabilities. Learning rate* 0.1, 0.5, 1.0. *95% c.i.*

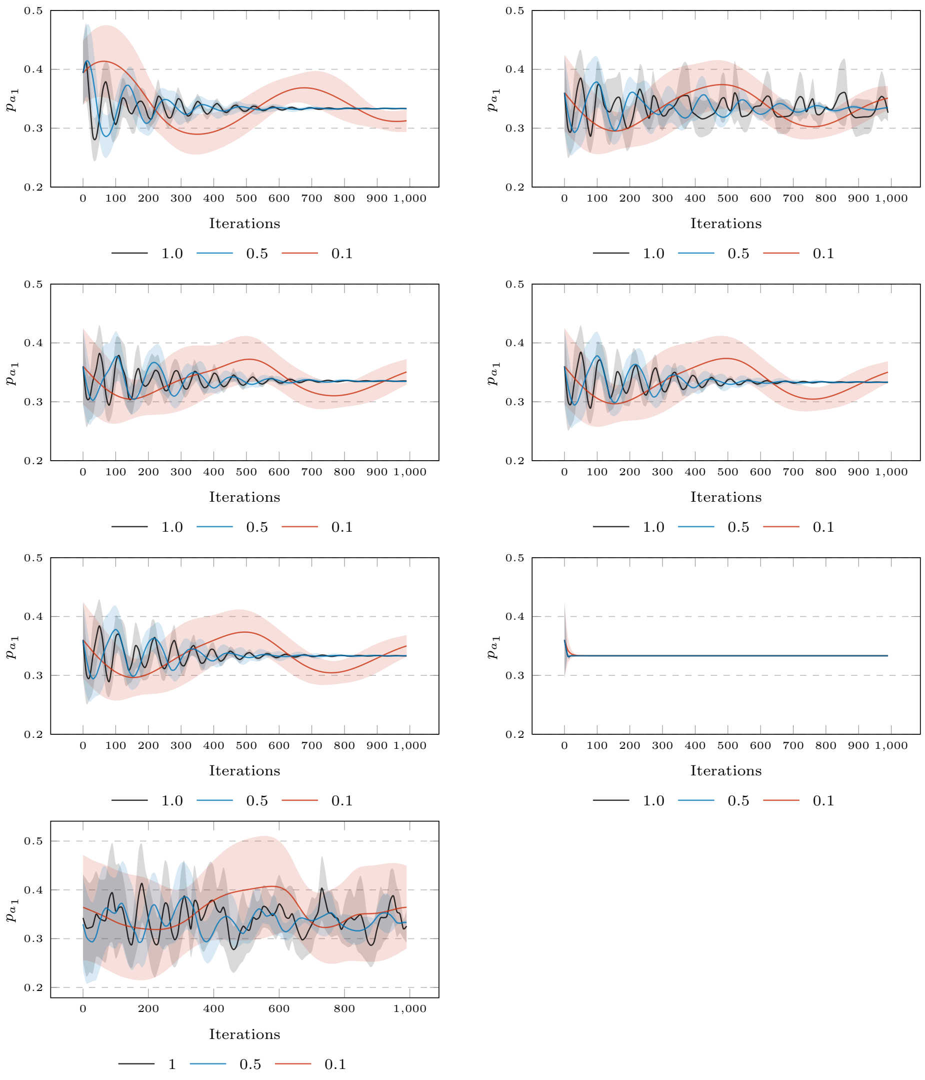


Figure 12.5: *Agent 1's probabily to perform action* 1 *in Rock Paper Scissors. From top right to bottom left: CGD, consensus, LOLA, IGAPP, SOS, NOHD, and SGA. Exact gradients batch size* 300*.* 20 *random sampled initial probabilities. Learning rate* $0.1, 0.5, 1.0$*. 95% c.i.*

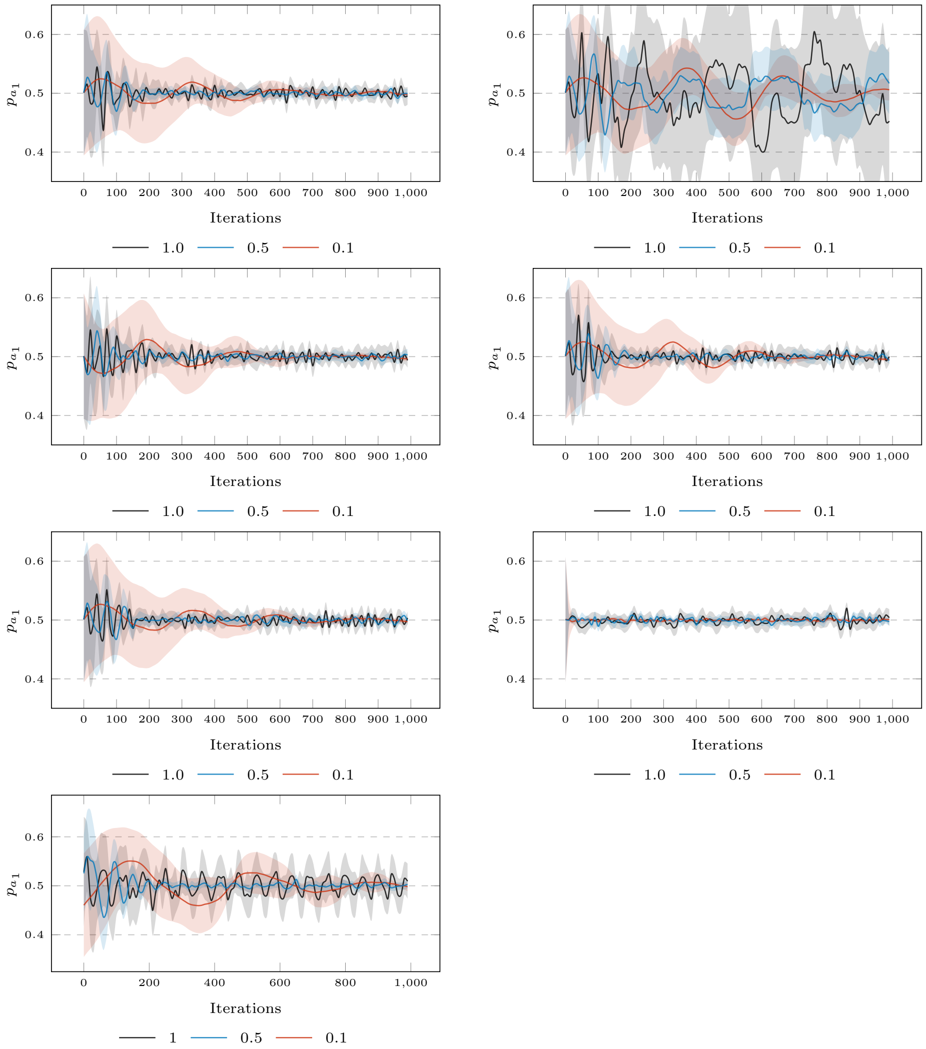


Figure 12.6: *Agent 1's probabily to perform action* 1 *in Matching Pennies. From top right to bottom left: CGD, consensus, LOLA, IGAPP, SOS, NOHD, and SGA. Estimated gradients batch size* 300. 20 *random sampled initial probabilities. Learning rate* 0.1, 0.5, 1.0. *95% c.i.*

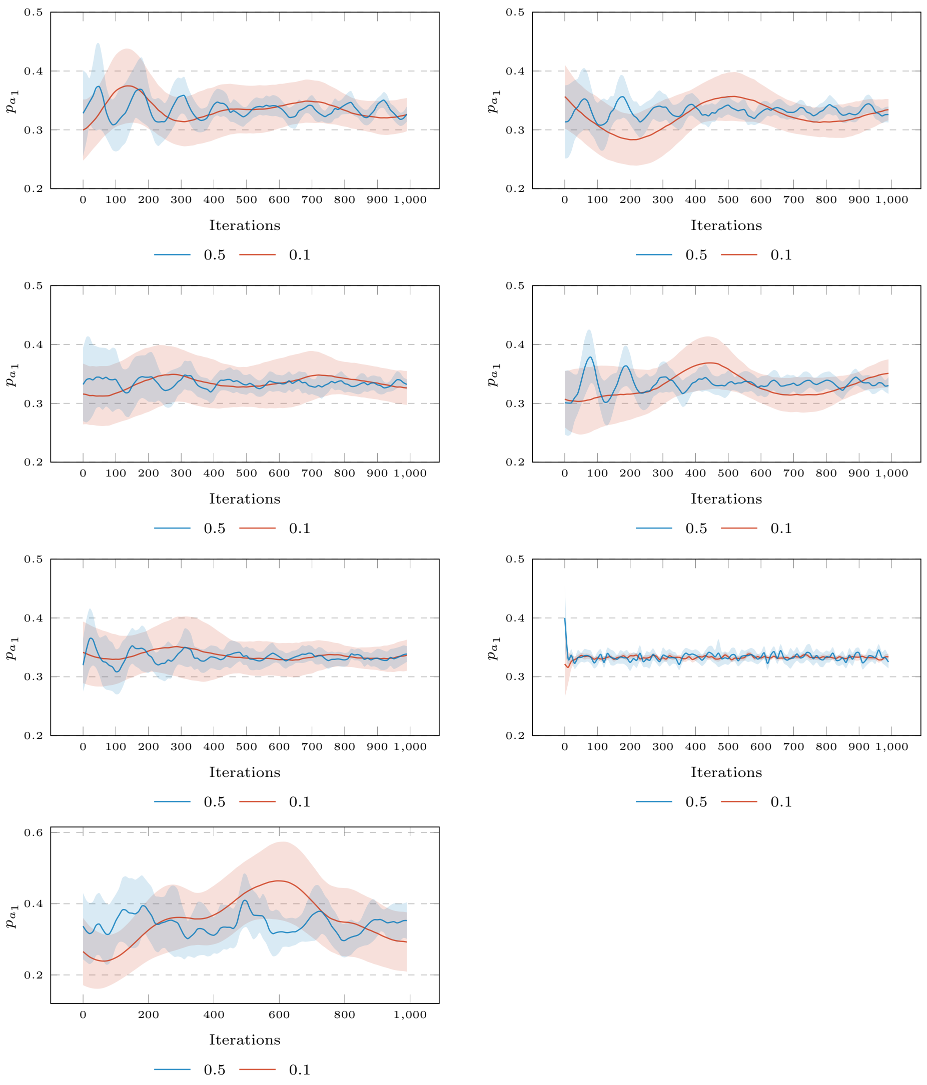


Figure 12.7: *Agent 1's probably to perform action* 1 *in Rock Paper Scissors. From top right to bottom left: CGD, consensus, LOLA, IGAPP, SOS, NOHD, and SGA. Estimated gradients.* 20 *random sampled initial probabilities. Learning rate* 0.1, 0.5. *95% c.i.*

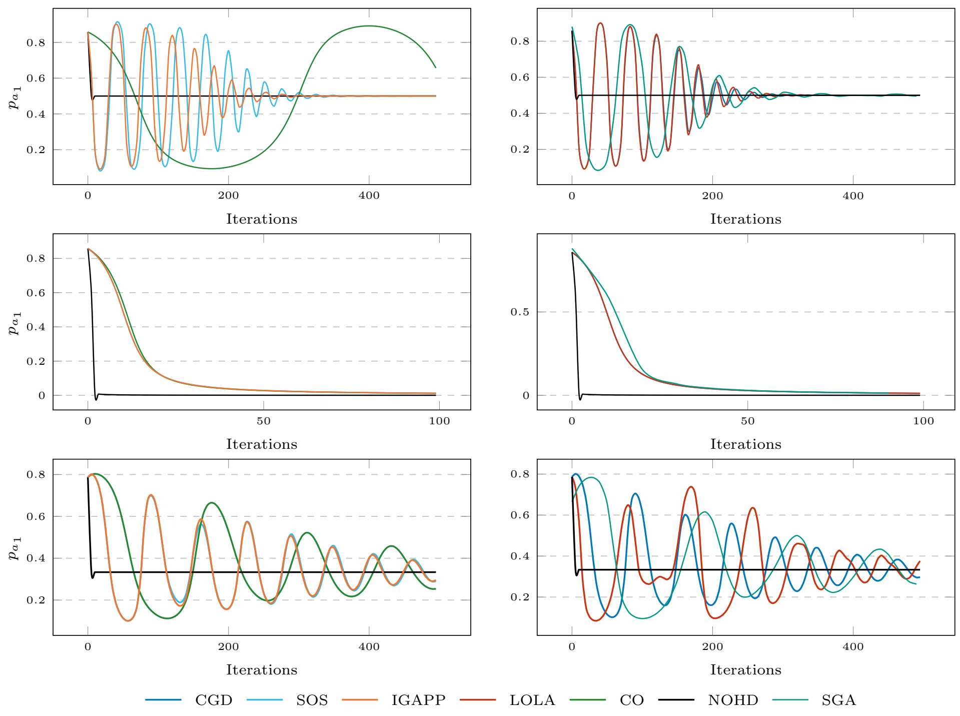


Figure 12.8: *Matching Pennies, Dilemma and Rock Paper Scissors (on the rows): probabilities of agent* 1 *to perform first action with a Boltzmann parametrization of the policies with exact gradients. Every algorithm with its best learning rate between* 0.1, 0.5, 1.0*.*

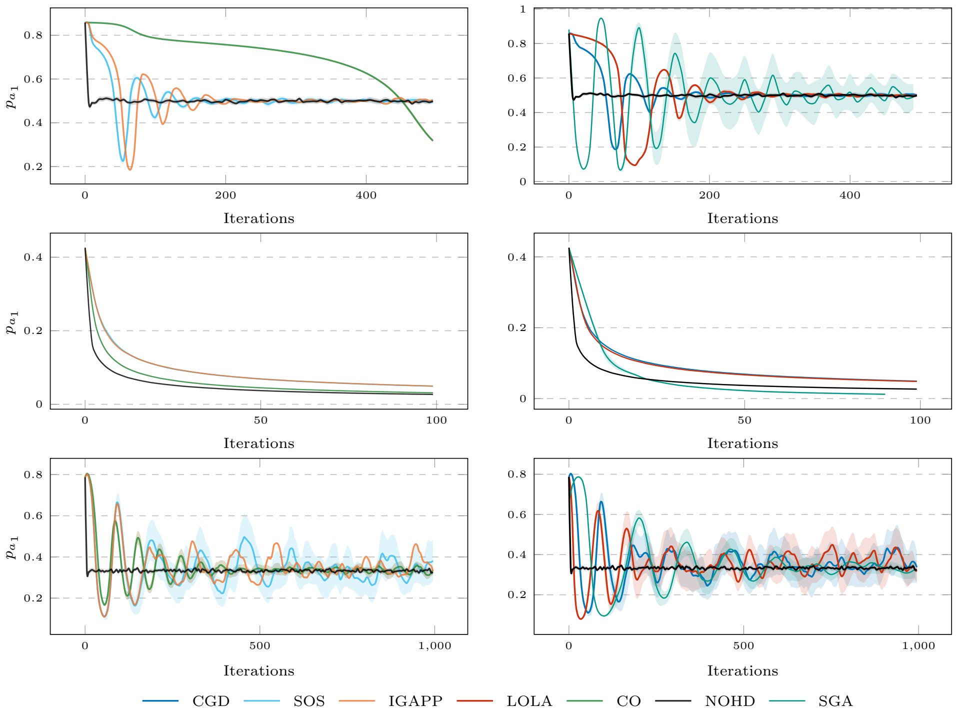
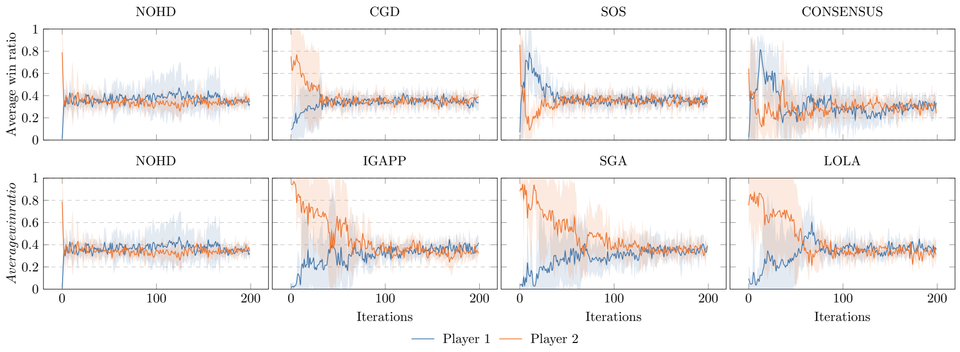


Figure 12.9: *Matching Pennies, Dilemma and Rock Paper Scissors (on the rows): probabilities of agent* 1 *to perform first action with a Boltzmann parametrization of the policies with approximated gradients. Every algorithm with its best learning rate between* $0.1, 0.5, 1.0$*. c.i. 95. %*



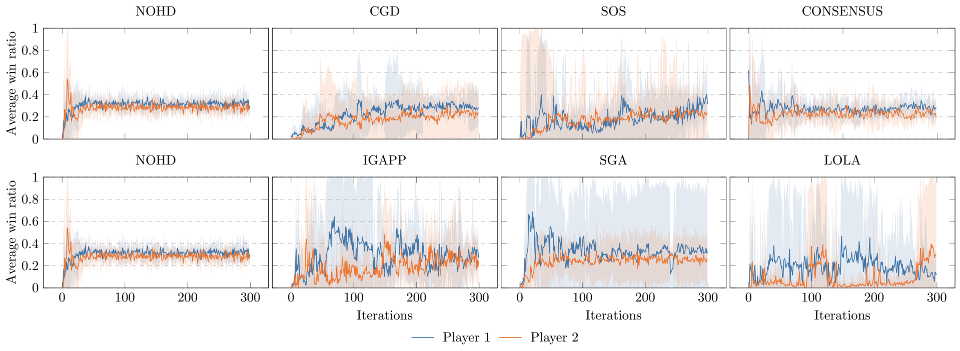


Figure 12.10: *Average wins for player* 1 *and player* 2 *in two gridworld environments.* 10 *runs, c.i. 98%*

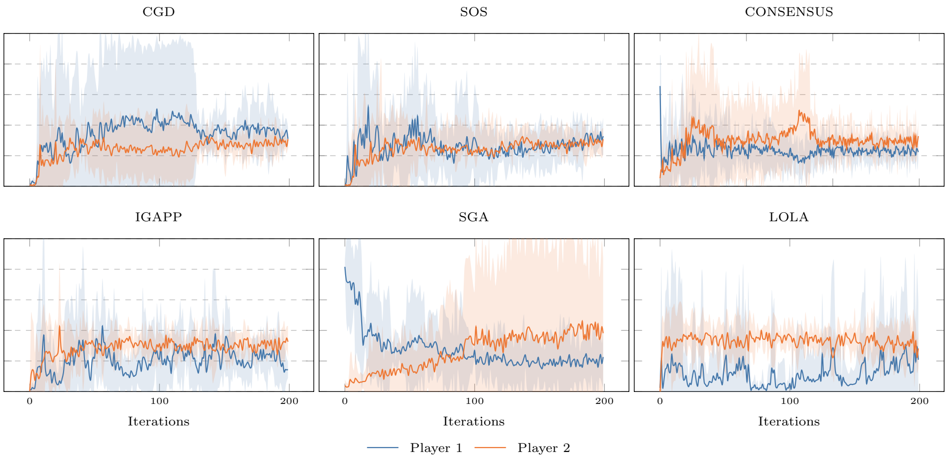


Figure 12.11: *Average wins for player* 1 *and player* 2 *in the second gridworld environments, where the second player is always NOHD.* 10 *runs, c.i. 98%*

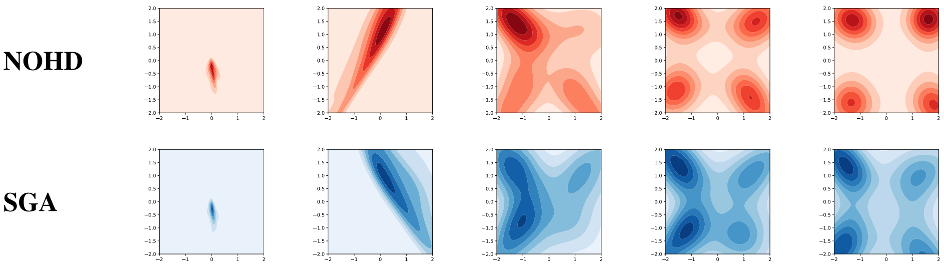


Figure 12.12: *Generator's learnt distribution for iterations* 0, 100, 200, 300, 400.

CHAPTER 13

Conclusions and Future works

This thesis addressed different problems in MARL from an algorithmic and theoretical point of view. The presence of multiple agents creates new challenges and opportunities in the RL field. The main opportunity that comes out is the possibility to model more complex environments than in RL; moreover, the multi-agent setting provides new possibilities, as we have shown in the IRL about Multiple Intentions setting (Chapter 6). On the other hand, the challenges with learning in multi-agent are various: the RL problem becomes non-stationary, the solution concept changes, and standard algorithms from single-agent RL cannot be applied in the MARL setting.

This thesis studied the MARL framework into three sub-settings: IRL in Multi-Agent Systems, Online Learning in MARL, and Optimization in MARL. These three fields are basically taken from the single-agent literature and analyzed in this thesis from the multi-agent perspective. We introduce the main concerns for all of them and propose new algorithms with (strong) theoretical guarantees. However, many problems remain unsolved, and also from our work, new open questions, practical and theoretical, come out. This (final) chapter describes the main contributions and discusses the interesting and promising future directions.

13.1 Inverse Reinforcement Learning in Multi-Agent Systems

Part II of this thesis is focused on *Inverse Reinforcement Learning in Multi-Agent Systems*. This field addressed the new opportunities and challenges that arise from switching from single agent to multi-agent problems. We identify three main new sub-problems: Inverse Reinforcement Learning about Multiple Intentions, i.e., the access to a dataset of demonstrations from a group of experts; Inverse Reinforcement Learning from a learning agent, i.e., recovering a reward function observing the agent's learning process; Multi-agent Inverse Reinforcement Learning, i.e., learning the reward functions that explain a certain equilibrium of a system. We then propose algorithms to solve the first two sub-problems. Regarding Inverse Reinforcement Learning about Multiple Intentions, we start by presenting a novel fully batch model-free IRL algorithm called Σ-GIRL. The algorithm accounts for the uncertainty in the gradient estimation and provides the reward function that maximizes the likelihood of the estimated policy gradient. Then, we extended our algorithm to the MI-IRL setting to simultaneously recover the agent-cluster assignment and the reward weights, using an Expectation-Maximization alternating process. In Chapter 7 we propose a novel algorithm for IRL from a learning agent setting. The proposed method, called LOGEL, relies on the assumption that the learner updates her policy along the direction of the gradient of the expected discounted return. For both the algorithms (Σ-GIRL and LOGEL), we provide finite-sample bounds on the algorithm performance in recovering the reward weights, and we tested the proposed approach both in simulated and real domains.

13.1.1 Future directions

The future directions in this field are several. From the MI-IRL setting, our approach presents some limitations. First of all, we have to specify the number of existing clusters, so a first (easy) extension could be to extend the approach to a non-parametric setting. Obviously, this leads to an increase in computation overhead. In the IRL from a Learner setting, the main challenge arises from the assumption that we know when the agent changes its policy. An extension could be the automatic detection of the policy change. From the learning from a learner perspective, obviously, the task becomes harder when we are in a non-stationary environment. A simple method that an agent can use if the other agent is unaware of being in a multi-agent environment is to wait without changing its policy parameter to make the environment stationary. However, if the interest is to play simultaneously and/or the other agents are not rational, it is necessary to build different algorithms to recover the reward function. In fact, another interesting future direction is certainly how to construct an IRL algorithm that learns from an agent that is optimizing a multi-agent objective. As we have shown in Part IV, the algorithms that are constructed for the MARL setting can, sometimes, be not rationally and so they could go in a direction different from the gradient one. An exciting and not straightforward problem is recovering the learning direction as well as the reward function of the observed agent. Obviously, this double-objective makes the problem "more" ill-posed, but we think it can be solved under suitable assumptions (for example, a finite-set of learning algorithms). Finally, an interesting goal to pursue is constructing IRL algorithms to learn from a dataset of multiple experts, which are, for example, in a Nash Equilibria. There are some algorithms that go in this direction [Lin et al., 2019a,Wang and Klabjan, 2018,Yu et al., 2019,Hadfield-Menell et al., 2016,Gruver et al., 2020]; however, there are no algorithms for batch-setting as far as we know.

13.2 Online Learning in Stochastic Games

In Part III we analyze the problem of online learning in Stochastic Games. In Chapter 8 we provide an introduction to the problem setting, and we underline the novel difficulties that come out from the presence of more than one agent. Then in the next two chapters (Chapter 9 and Chapter 10), we analyze the online learning problem in two different contexts: the Configurable Markov Decision Process and Turn-based Stochastic Games. For both frameworks, we consider the setting in which we can control only one entity. In Chapter 9, we generalize the Configurable MDP framework to account for possible non-cooperative interaction between the agent and the configurator. We consider an online learning problem, where we can control only the configurator. We propose two regret minimization algorithms for identifying the best environment configuration within a finite set based on the principle of optimism in the face of uncertainty. We proved that when the agent's policy is deterministic (but the configuration may not) and the configurator observes the agent's actions, it is possible to achieve finite regret that depends linearly on the admissible number configurations. Furthermore, we illustrated that it is possible to remove this dependence if the configurator observes a possibly noisy version of the agent's reward and under sufficient regularity conditions on the environment. In Chapter 10, we provide, as far as we know, the first theoretical insights on the online learning problem in the general-sum Stochastic Games. We consider the online setting in two-player Turn-based Stochastic Games, i.e., where we can control only one agent of the game. We propose a new lower bound for this setting that shows that the regret has to scale constantly with the policies' space size considered. Then we provide a regret minimization algorithm based on the principle of optimism in the face of uncertainty. We proved that our algorithm nearly-matches the presented lower bound.

13.2.1 Future directions

There are many future directions in the online learning problem in Stochastic Games. From our work in Non-cooperative Configurable MDPs, a direct follow-up will be extending the setting to the case of stochastic agent's policy and the derivation of specific confidence intervals for the reward function, based on IRL. Another future direction is to extend the approach to deal with continuous state and action spaces using, for example, function approximation. Currently, there is a need for a formal understanding of the online MARL problem to construct provably-efficient learning algorithms for this context. As our result suggested, the MARL setting poses novel challenges, especially in the well-known exploration-exploitation dilemma, i.e., the trade-off between gathering new information and exploiting it: in a multi-agent environment, the agent needs to explore not only to understand the underlying environment but also to learn the other agents' behaviors. Moreover, from our findings, it is clear that the algorithm design and the resulting performance guarantees heavily depend on any knowledge about the opponents, either known as a priori or obtainable during the learning process. Indeed there are many open problems in the agnostic setting, presented in [Tian et al., 2020], as it is possible to achieve better theoretical (regret) guarantees and construct algorithms with optimal sample complexity. This scenario, having no assumptions on the opponents, is widely applicable to capture real-world problems. On the other hand, assuming to have the possibility to observe other agents' interactions with the environment or having some previous knowledge about the other agents (as having access to a finite set of opponents [Balcan et al., 2015] or considering a larger set of opponents' classes with some regularity assumptions [Sessa et al., 2020]) we could hope to obtain better theoretical guarantees.

13.3 Optimization methods for Multi-Agent Reinforcement Learning

Part IV is devoted to the policy search approaches for MARL. We start by casting the MARL problem in a more general setting called *Continuous*

Games. Then we provide the necessary background to understand the optimization difficulties that multiple functions create, and we broadly revise the literature for this setting. We underline the importance of accounting for the dynamics of the system to construct provably efficient algorithms. In Chapter 12, we have shown how to apply Newton-based methods to optimize Continuous Games. We started by proposing an approach to adapt Newton's optimization to two classes of games: Potential and Hamiltonian games. We then presented a novel algorithm called NOHD, which applies a Newton-based optimization approach to general games. The algorithm considers that agents can also act against their own interests to achieve a balance as quickly as possible. Then we showed that NOHD avoids unstable fixed points and is attracted to stable ones. We performed an extensive experimental evaluation of the proposed method showing that the algorithm outperforms (when it was submitted at AAAI 2021) state-of-the-art baselines in all experiments.

13.3.1 Future directions

The landscape of continuous games and their application in the policy optimization methods for MARL is far from being completely understood. The local Nash Equilibrium seems to be a hard objective to be optimized efficiently, especially in general games. Moreover, there are some cases where we could be interested also in converging to other game-theoretic solutions as, for example, Correlated, Minimax, or Stackelberg equilibrium points. Recently some works go in this direction. In [Celli et al., 2020] the authors propose exciting results for extensive-form games to converge to Correlated equilibria. In [Fiez et al., 2020b] the authors propose a second-order optimization algorithm to converge to local differentiable Stackelberg equilibrium solution. These equilibria are of interest, especially in zero-sum games, where they coincide with the Nash ones. In [Kamalaruban et al., 2020] the authors take their previous results on GANs [Hsieh et al., 2019] and optimize over the set of probability distributions over pure strategies to solve the robust RL. Their results are interesting also for continuous competitive games in general. In [Mangoubi and Vishnoi, 2021] the authors propose min-max problems a new solution concept, called greedy adversarial equilibrium. Providing a connection between these equilibria, studying algorithms to converge to them is undoubtedly an important future direction. Moreover, there is a current research line that considers different solutions concepts going beyond equilibria. For example, in [Papadimitriou and Piliouras, 2019, Omidshafiei et al., 2019] the authors consider the learning dynamics itself as a solution concept. Instead, in [Duvocelle et al., 2018, Cardoso et al., 2019, Skoulakis et al., 2021] the authors study a setting where both the agents and the games they play evolve strategically over time.

The majority of the proposed approaches deal only with exact gradients and exact Jacobians. On the other hand, in RL, it is crucial to have convergence guarantees for the approximated (stochastic) setting since the gradient of the expected discounted return can be learned only by interactions. Only some recent works [Fiez et al., 2020a, Mazumdar et al., 2020b] propose theoretical guarantees in the stochastic setting.

Finally, an interesting future direction could be to adapt the theoretical finding in continuous games to construct actor-critic algorithms for MARL and investigate, as in [Zheng et al., 2021], the applications of optimization methods in different contexts. An example could be Non-cooperative Configurable MDPs, where we can cast the problem as a continuous game between the configurator and the RL agent.

Bibliography

[Abbasi-Yadkori et al., 2013] Abbasi-Yadkori, Y., Bartlett, P. L., Kanade, V., Seldin, Y., and Szepesvári, C. (2013). Online learning in markov decision processes with adversarially chosen transition probability distributions. In *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, pages 2508–2516.

[Abbeel and Ng, 2004] Abbeel, P. and Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1.

[Adolphs et al., 2019] Adolphs, L., Daneshmand, H., Lucchi, A., and Hofmann, T. (2019). Local saddle point optimization: A curvature exploitation approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 486–495. PMLR.

[Agrawal and Jia, 2017] Agrawal, S. and Jia, R. (2017). Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1184–1194.

[Almingol and Montesano, 2015] Almingol, J. and Montesano, L. (2015). Learning multiple behaviours using hierarchical clustering of rewards. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4608–4613. IEEE.

[Argall et al., 2009] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

[Arulkumaran et al., 2017] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

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

[Auer et al., 2009] Auer, P., Jaksch, T., and Ortner, R. (2009). Near-optimal regret bounds for reinforcement learning. In *Advances in neural information processing systems*, pages 89–96.

[Aumann, 1987] Aumann, R. J. (1987). Correlated equilibrium as an expression of bayesian rationality. *Econometrica: Journal of the Econometric Society*, pages 1–18.

[Azar et al., 2013] Azar, M. G., Munos, R., and Kappen, H. J. (2013). Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91(3):325–349.

[Babes et al., 2011] Babes, M., Marivate, V. N., Subramanian, K., and Littman, M. L. (2011). Apprenticeship learning about multiple intentions. In *ICML*.

[Bai and Jin, 2020] Bai, Y. and Jin, C. (2020). Provable self-play algorithms for competitive reinforcement learning. In *International Conference on Machine Learning*, pages 551–560. PMLR.

[Bai et al., 2021] Bai, Y., Jin, C., Wang, H., and Xiong, C. (2021). Sample-efficient learning of stackelberg equilibria in general-sum games. *arXiv preprint arXiv:2102.11494*.

[Bai et al., 2020] Bai, Y., Jin, C., and Yu, T. (2020). Near-optimal reinforcement learning with self-play. *Advances in Neural Information Processing Systems*, 33.

[Bain and Sammut, 1995] Bain, M. and Sammut, C. (1995). A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129.

[Balakrishna et al., 2020] Balakrishna, A., Thananjeyan, B., Lee, J., Li, F., Zahed, A., Gonzalez, J. E., and Goldberg, K. (2020). On-policy robot imitation learning from a converging supervisor. In *Conference on Robot Learning*, pages 24–41. PMLR.

[Balcan et al., 2015] Balcan, M.-F., Blum, A., Haghtalab, N., and Procaccia, A. D. (2015). Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the sixteenth ACM conference on economics and computation*, pages 61–78.

[Balduzzi et al., 2020] Balduzzi, D., Czarnecki, W. M., Anthony, T. W., Gemp, I. M., Hughes, E., Leibo, J. Z., Piliouras, G., and Graepel, T. (2020). Smooth markets: A basic mechanism for organizing gradient-based learners. *arXiv preprint arXiv:2001.04678*.

[Balduzzi et al., 2018] Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., and Graepel, T. (2018). The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pages 354–363. PMLR.

[Bartlett and Tewari, 2009a] Bartlett, P. and Tewari, A. (2009a). Regal: a regularization based algorithm for reinforcement learning in weakly communicating mdps. In *Uncertainty in Artificial Intelligence: Proceedings of the 25th Conference*, pages 35–42. AUAI Press.

[Bartlett and Tewari, 2009b] Bartlett, P. and Tewari, A. (2009b). Regal: a regularization based algorithm for reinforcement learning in weakly communicating mdps. In *Uncertainty in Artificial Intelligence: Proceedings of the 25th Conference*, pages 35–42. AUAI Press.

[Başar and Bernhard, 2008] Başar, T. and Bernhard, P. (2008). *H-infinity optimal control and related minimax design problems: a dynamic game approach*. Springer Science & Business Media.

[Baxter and Bartlett, 2001] Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

[Bellman, 1957] Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

[Bertsekas et al., 1995] Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

[Bilmes et al., 1998] Bilmes, J. A. et al. (1998). A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126.

[Börgers and Sarin, 1997] Börgers, T. and Sarin, R. (1997). Learning through reinforcement and replicator dynamics. *Journal of economic theory*, 77(1):1–14.

[Borkar, 2009] Borkar, V. S. (2009). *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer.

[Boularias et al., 2011] Boularias, A., Kober, J., and Peters, J. (2011). Relative entropy inverse reinforcement learning. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 182–189, Fort Lauderdale, FL, USA. JMLR Workshop and Conference Proceedings.

[Boutilier, 1996] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *TARK*, volume 96, pages 195–210. Citeseer.

[Bowling and Veloso, 2002] Bowling, M. and Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.

[Brafman and Tennenholtz, 2002] Brafman, R. I. and Tennenholtz, M. (2002). R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231.

[Bravo et al., 2018] Bravo, M., Leslie, D. S., and Mertikopoulos, P. (2018). Bandit learning in concave n-person games. In *NIPS 2018-Thirty-second Conference on Neural Information Processing Systems*, pages 1–24.

[Breton et al., 1988] Breton, M., Alj, A., and Haurie, A. (1988). Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59(1):71–97.

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *CoRR*, abs/1606.01540.

[Brown et al., 2019] Brown, D., Goo, W., Nagarajan, P., and Niekum, S. (2019). Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pages 783–792.

[Bu et al., 2019] Bu, J., Ratliff, L. J., and Mesbahi, M. (2019). Global convergence of policy gradient for sequential zero-sum linear quadratic dynamic games. *arXiv preprint arXiv:1911.04672*.

[Bubeck et al., 2013] Bubeck, S., Perchet, V., and Rigollet, P. (2013). Bounded regret in stochastic multi-armed bandits. In *Conference on Learning Theory*, pages 122–134.

[Buşoniu et al., 2010] Buşoniu, L., Babuška, R., and De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221.

[Candogan et al., 2011] Candogan, O., Menache, I., Ozdaglar, A., and Parrilo, P. A. (2011). Flows and decompositions of games: Harmonic and potential games. *Mathematics of Operations Research*, 36(3):474–503.

[Cardoso et al., 2019] Cardoso, A. R., Abernethy, J., Wang, H., and Xu, H. (2019). Competing against nash equilibria in adversarially changing zero-sum games. In *International Conference on Machine Learning*, pages 921–930. PMLR.

[Casella and Berger, 2002] Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

[Castro et al., 2019] Castro, P. S., Li, S., and Zhang, D. (2019). Inverse reinforcement learning with multiple ranked experts. *arXiv preprint arXiv:1907.13411*.

[Celli et al., 2020] Celli, A., Marchesi, A., Farina, G., and Gatti, N. (2020). No-regret learning dynamics for extensive-form correlated equilibrium. *Advances in Neural Information Processing Systems*, 33.

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

[Chasnov et al., 2019] Chasnov, B., Fiez, T., and Ratliff, L. J. (2019). Gradient conjectures for strategic multi-agent learning.

[Chasnov et al., 2020a] Chasnov, B., Ratliff, L., Mazumdar, E., and Burden, S. (2020a). Convergence analysis of gradient-based learning in continuous games. In *Uncertainty in Artificial Intelligence*, pages 935–944. PMLR.

[Chasnov et al., 2020b] Chasnov, B. J., Calderone, D., Açıkmeşe, B., Burden, S. A., and Ratliff, L. J. (2020b). Stability of gradient learning dynamics in continuous games: Scalar action spaces. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3543–3548. IEEE.

[Chen et al., 2009] Chen, X., Deng, X., and Teng, S.-H. (2009). Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57.

[Chen and Caramanis, 2013] Chen, Y. and Caramanis, C. (2013). Noisy and missing data regression: Distribution-oblivious support recovery. In *International Conference on Machine Learning*, pages 383–391.

[Choi and Kim, 2012] Choi, J. and Kim, K.-E. (2012). Nonparametric bayesian inverse reinforcement learning for multiple reward functions. *Advances in Neural Information Processing Systems*, 25:305–313.

[Chong and Zak, 2004] Chong, E. K. and Zak, S. H. (2004). *An introduction to optimization*. John Wiley & Sons.

[Christiano et al., 2017] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307.

[Chung et al., 2020] Chung, H.-M., Maharjan, S., Zhang, Y., and Eliassen, F. (2020). Distributed deep reinforcement learning for intelligent load scheduling in residential smart grid. *IEEE Transactions on Industrial Informatics*.

[Coniglio et al., 2020] Coniglio, S., Gatti, N., and Marchesi, A. (2020). Computing a pessimistic stackelberg equilibrium with multiple followers: The mixed-pure case. *Algorithmica*, 82(5):1189–1238.

[Conitzer and Sandholm, 2007] Conitzer, V. and Sandholm, T. (2007). Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43.

[Da Silva and Costa, 2019] Da Silva, F. L. and Costa, A. H. R. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703.

[Daskalakis et al., 2009] Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259.

[Daskalakis et al., 2020] Daskalakis, C., Skoulakis, S., and Zampetakis, M. (2020). The complexity of constrained min-max optimization. *arXiv preprint arXiv:2009.09623*.

[Deisenroth et al., 2013] Deisenroth, M. P., Neumann, G., Peters, J., et al. (2013). A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

[Diakonikolas et al., 2021] Diakonikolas, J., Daskalakis, C., and Jordan, M. (2021). Efficient methods for structured nonconvex-nonconcave min-max optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2746–2754. PMLR.

[Dick et al., 2014] Dick, T., Gyorgy, A., and Szepesvari, C. (2014). Online learning in markov decision processes with changing cost sequences. In *International Conference on Machine Learning*, pages 512–520. PMLR.

[Domingues et al., 2020] Domingues, O. D., Ménard, P., Kaufmann, E., and Valko, M. (2020). Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited.

[Dorato et al., 2000] Dorato, P., Cerone, V., and Abdallah, C. (2000). *Linear quadratic control: an introduction*. Krieger Publishing Co., Inc.

[Duvocelle et al., 2018] Duvocelle, B., Mertikopoulos, P., Staudigl, M., and Vermeulen, D. (2018). Learning in time-varying games. *arXiv preprint arXiv:1809.03066*.

[Even-Dar et al., 2009] Even-Dar, E., Kakade, S. M., and Mansour, Y. (2009). Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736.

[Facchinei and Kanzow, 2007] Facchinei, F. and Kanzow, C. (2007). Generalized nash equilibrium problems. *4or*, 5(3):173–210.

[Facchinei and Pang, 2007] Facchinei, F. and Pang, J.-S. (2007). *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media.

[Feller, 1957] Feller, W. (1957). An introduction to probability theory and its applications.

[Fiez et al., 2020a] Fiez, T., Chasnov, B., and Ratliff, L. (2020a). Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3133–3144. PMLR.

[Fiez et al., 2020b] Fiez, T., Chasnov, B., and Ratliff, L. (2020b). Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning*, pages 3133–3144. PMLR.

[Fiez and Ratliff, 2020] Fiez, T. and Ratliff, L. (2020). Gradient descent-ascent provably converges to strict local minmax equilibria with a finite timescale separation. *arXiv preprint arXiv:2009.14820*.

[Flet-Berliac et al., 2021] Flet-Berliac, Y., Ouhamma, R., Maillard, O.-A., and Preux, P. (2021). Learning value functions in deep policy gradients using residual variance. In *ICLR 2021-International Conference on Learning Representations*.

[Foerster et al., 2018] Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2018). Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems.

[François-Lavet et al., 2018] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., and Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354.

[Fruit et al., 2018] Fruit, R., Pirotta, M., and Lazaric, A. (2018). Near optimal exploration-exploitation in non-communicating markov decision processes. In *32nd Conference on Neural Information Processing Systems*.

[Fudenberg et al., 1998] Fudenberg, D., Drew, F., Levine, D. K., and Levine, D. K. (1998). *The theory of learning in games*, volume 2. MIT press.

[Fudenberg and Tirole, 1991] Fudenberg, D. and Tirole, J. (1991). Perfect bayesian equilibrium and sequential equilibrium. *journal of Economic Theory*, 53(2):236–260.

[Gallego et al., 2019] Gallego, V., Naveiro, R., and Insua, D. R. (2019). Reinforcement learning under threats. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9939–9940.

[Garivier and Cappé, 2011] Garivier, A. and Cappé, O. (2011). The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th annual conference on learning theory*, pages 359–376.

[Gemp and Mahadevan, 2018] Gemp, I. and Mahadevan, S. (2018). Global convergence to the equilibrium of gans using variational inequalities. *arXiv preprint arXiv:1808.01531*.

[Gergely Neu et al., 2010] Gergely Neu, A. G., Szepesvári, C., and Antos, A. (2010). Online markov decision processes under bandit feedback. In *Proceedings of the Twenty-Fourth Annual Conference on Neural Information Processing Systems*.

[González-Sánchez and Hernández-Lerma, 2013] González-Sánchez, D. and Hernández-Lerma, O. (2013). *Discrete–time stochastic control and dynamic potential games: the Euler–Equation approach*. Springer Science & Business Media.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27:2672–2680.

[Goyal and Grand-Clement, 2019] Goyal, V. and Grand-Clement, J. (2019). A first-order approach to accelerated value iteration. *arXiv preprint arXiv:1905.09963*.

[Gravell et al., 2020] Gravell, B., Ganapathy, K., and Summers, T. (2020). Policy iteration for linear quadratic games with stochastic parameters. *IEEE Control Systems Letters*, 5(1):307–312.

[Gruver et al., 2020] Gruver, N., Song, J., Kochenderfer, M. J., and Ermon, S. (2020). Multi-agent adversarial inverse reinforcement learning with latent variables. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1855–1857.

[Gupta and Nagar, 2018] Gupta, A. K. and Nagar, D. K. (2018). *Matrix variate distributions*. Chapman and Hall/CRC.

[Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR.

[Hadfield-Menell et al., 2016] Hadfield-Menell, D., Dragan, A., Abbeel, P., and Russell, S. (2016). Cooperative inverse reinforcement learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3916–3924.

[Hart and Mas-Colell, 2003] Hart, S. and Mas-Colell, A. (2003). Uncoupled dynamics do not lead to nash equilibrium. *American Economic Review*, 93(5):1830–1836.

[Héliou et al., 2020] Héliou, A., Mertikopoulos, P., and Zhou, Z. (2020). Gradient-free online learning in continuous games with delayed rewards. In *International Conference on Machine Learning*, pages 4172–4181. PMLR.

[Hernandez-Leal et al., 2019] Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797.

[Hespanha, 2018] Hespanha, J. P. (2018). *Linear systems theory*. Princeton university press.

[Hestenes et al., 1952] Hestenes, M. R., Stiefel, E., et al. (1952). *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC.

[Heusel et al., 2017] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637.

[Ho and Ermon, 2016] Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4572–4580.

[Hofbauer et al., 1998] Hofbauer, J., Sigmund, K., et al. (1998). *Evolutionary games and population dynamics*. Cambridge university press.

[Hoffman and Karp, 1966] Hoffman, A. J. and Karp, R. M. (1966). On nonterminating stochastic games. *Management Science*, 12(5):359–370.

[Hsieh et al., 2021] Hsieh, Y.-G., Antonakopoulos, K., and Mertikopoulos, P. (2021). Adaptive learning in continuous games: Optimal regret bounds and convergence to nash equilibrium. *arXiv preprint arXiv:2104.12761*.

[Hsieh et al.,] Hsieh, Y.-G., Iutzeler, F., Malick, J., and Mertikopoulos, P. Optimization in open networks via dual averaging.

[Hsieh et al., 2019] Hsieh, Y.-P., Liu, C., and Cevher, V. (2019). Finding mixed nash equilibria of generative adversarial networks. In *International Conference on Machine Learning*, pages 2810–2819. PMLR.

[Hu and Wellman, 2003] Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069.

[Hussein et al., 2017] Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):21.

[Ibarz et al., 2018] Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. (2018). Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, pages 8011–8023.

[Isola et al., 2017] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

[Iyengar, 2005] Iyengar, G. N. (2005). Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280.

[Jacobson, 1973] Jacobson, D. (1973). Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games. *IEEE Transactions on Automatic control*, 18(2):124–131.

[Jacq et al., 2019] Jacq, A., Geist, M., Paiva, A., and Pietquin, O. (2019). Learning from a learner. In *International Conference on Machine Learning*, pages 2990–2999. PMLR.

[Jaderberg et al., 2016] Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

[Jain et al., 2019] Jain, V., Doshi, P., and Banerjee, B. (2019). Model-free irl using maximum likelihood estimation.

[Jaksch et al., 2010] Jaksch, T., Ortner, R., and Auer, P. (2010). Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4).

[Jaynes, 1957] Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical review*, 106(4):620.

[Jin et al., 2018] Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018). Is q-learning provably efficient? *arXiv preprint arXiv:1807.03765*.

[Jin et al., 2020a] Jin, C., Jin, T., Luo, H., Sra, S., and Yu, T. (2020a). Learning adversarial markov decision processes with bandit feedback and unknown transition. In *International Conference on Machine Learning*, pages 4860–4869. PMLR.

[Jin et al., 2020b] Jin, C., Netrapalli, P., and Jordan, M. (2020b). What is local optimality in nonconvex-nonconcave minimax optimization? In *International Conference on Machine Learning*, pages 4880–4889. PMLR.

[Johnson and Zhang, 2013] Johnson, R. and Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323.

[Kakade, 2001] Kakade, S. M. (2001). A natural policy gradient. *Advances in neural information processing systems*, 14.

[Kamalaruban et al., 2020] Kamalaruban, P., Huang, Y.-T., Hsieh, Y.-P., Rolland, P., Shi, C., and Cevher, V. (2020). Robust reinforcement learning via adversarial training with langevin dynamics. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8127–8138. Curran Associates, Inc.

[Kartal et al., 2019] Kartal, B., Hernandez-Leal, P., and Taylor, M. E. (2019). Terminal prediction as an auxiliary task for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pages 38–44.

[Kearns et al., 2013] Kearns, M., Mansour, Y., and Singh, S. (2013). Fast planning in stochastic games. *arXiv preprint arXiv:1301.3867*.

[Kearns and Singh, 1999] Kearns, M. and Singh, S. (1999). Finite-sample convergence rates for q-learning and indirect algorithms. *Advances in neural information processing systems*, pages 996–1002.

[Klein et al., 2012] Klein, E., Geist, M., Piot, B., and Pietquin, O. (2012). Inverse reinforcement learning through structured classification. In *Advances in Neural Information Processing Systems*, pages 1007–1015.

[Klein et al., 2013] Klein, E., Piot, B., Geist, M., and Pietquin, O. (2013). A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 1–16. Springer.

[Knyazev et al., 2010] Knyazev, A., Jujunashvili, A., and Argentati, M. (2010). Angles between infinite dimensional subspaces with applications to the rayleigh–ritz and alternating projectors methods. *Journal of Functional Analysis*, 259(6):1323–1345.

[Ko and Lin, 1995] Ko, K.-I. and Lin, C.-L. (1995). On the complexity of min-max optimization problems and their approximation. In *Minimax and Applications*, pages 219–239. Springer.

[Konda and Tsitsiklis, 2000] Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014. Citeseer.

[Lã et al., 2016] Lã, Q. D., Chew, Y. H., and Soong, B.-H. (2016). Potential game theory. *pringer International Publishing*.

[La Salle, 1976] La Salle, J. P. (1976). *The stability of dynamical systems*. SIAM.

[Lagoudakis and Parr, 2002] Lagoudakis, M. G. and Parr, R. (2002). Value function approximation in zero-sum markov games. pages 283–292.

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

[Lambert Iii et al., 2005] Lambert Iii, T. J., Epelman, M. A., and Smith, R. L. (2005). A fictitious play approach to large-scale optimization. *Operations Research*, 53(3):477–489.

[Lattimore and Munos, 2014] Lattimore, T. and Munos, R. (2014). Bounded regret for finite-armed structured bandits. In *Advances in Neural Information Processing Systems*, pages 550–558.

[Lattimore and Szepesvári, 2020] Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

[Lauer and Riedmiller, 2000] Lauer, M. and Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690.

[Ledoit and Wolf, 2004] Ledoit, O. and Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411.

[Lee et al., 2017] Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. (2017). First-order methods almost always avoid saddle points. *arXiv preprint arXiv:1710.07406*.

[Lee et al., 2016] Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016). Gradient descent converges to minimizers. *University of California, Berkeley*, 1050:4.

[Letcher, 2020] Letcher, A. (2020). On the impossibility of global convergence in multi-loss optimization. *arXiv preprint arXiv:2005.12649*.

[Letcher et al., 2019] Letcher, A., Balduzzi, D., Racaniere, S., Martens, J., Foerster, J. N., Tuyls, K., and Graepel, T. (2019). Differentiable game mechanics. *Journal of Machine Learning Research*, 20(84):1–40.

[Letcher et al., 2018] Letcher, A., Foerster, J., Balduzzi, D., Rocktäschel, T., and Whiteson, S. (2018). Stable opponent shaping in differentiable games. In *International Conference on Learning Representations*.

[Levine et al., 2011] Levine, S., Popovic, Z., and Koltun, V. (2011). Nonlinear inverse reinforcement learning with gaussian processes. *Advances in neural information processing systems*, 24:19–27.

[Li et al., 2020] Li, J., Zhou, Y., Ren, T., and Zhu, J. (2020). Exploration analysis in finite-horizon turn-based stochastic games. In *Conference on Uncertainty in Artificial Intelligence*, pages 201–210. PMLR.

[Li, 2017] Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

[Likmeta et al., 2021] Likmeta, A., Metelli, A. M., Ramponi, G., Tirinzoni, A., Giuliani, M., and Restelli, M. (2021). Dealing with multiple experts and non-stationarity in inverse reinforcement learning: an application to real-life problems. *Machine Learning*, pages 1–36.

[Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

[Lim et al., 2013] Lim, S. H., Xu, H., and Mannor, S. (2013). Reinforcement learning in robust markov decision processes. *Advances in Neural Information Processing Systems*, 26:701–709.

[Lin et al., 2020] Lin, T., Jin, C., and Jordan, M. (2020). On gradient descent ascent for nonconvex-concave minimax problems. In *International Conference on Machine Learning*, pages 6083–6093. PMLR.

[Lin et al., 2019a] Lin, X., Adams, S. C., and Beling, P. A. (2019a). Multi-agent inverse reinforcement learning for certain general-sum stochastic games. *Journal of Artificial Intelligence Research*, 66:473–502.

[Lin et al., 2019b] Lin, X., Baweja, H. S., Kantor, G., and Held, D. (2019b). Adaptive auxiliary task weighting for reinforcement learning. *Advances in neural information processing systems*, 32.

[Littman, 1994] Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

[Littman, 2001] Littman, M. L. (2001). Value-function reinforcement learning in markov games. *Cognitive systems research*, 2(1):55–66.

[Liu et al., 2020] Liu, Q., Yu, T., Bai, Y., and Jin, C. (2020). A sharp analysis of model-based reinforcement learning with self-play. *arXiv preprint arXiv:2010.01604*.

[Lykouris et al., 2019] Lykouris, T., Simchowitz, M., Slivkins, A., and Sun, W. (2019). Corruption robust exploration in episodic reinforcement learning. *arXiv preprint arXiv:1911.08689*.

[Maclaurin et al., 2015] Maclaurin, D., Duvenaud, D., and Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR.

[Madry et al., 2018] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

[Mangoubi and Vishnoi, 2021] Mangoubi, O. and Vishnoi, N. K. (2021). Greedy adversarial equilibrium: An efficient alternative to nonconvex-nonconcave min-max optimization.

[Mannor et al., 2012] Mannor, S., Mebel, O., and Xu, H. (2012). Lightning does not strike twice: robust mdps with coupled uncertainty. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 451–458.

[Mannor et al., 2016] Mannor, S., Mebel, O., and Xu, H. (2016). Robust mdps with k-rectangular uncertainty. *Mathematics of Operations Research*, 41(4):1484–1509.

[Manton et al., 2003] Manton, J. H., Mahony, R., and Hua, Y. (2003). The geometry of weighted low-rank approximations. *IEEE Transactions on Signal Processing*, 51(2):500–514.

[Mazumdar et al., 2020a] Mazumdar, E., Ratliff, L. J., Jordan, M. I., and Sastry, S. S. (2020a). Policy-gradient algorithms have no guarantees of convergence in linear quadratic games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 860–868.

[Mazumdar et al., 2020b] Mazumdar, E., Ratliff, L. J., and Sastry, S. S. (2020b). On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2(1):103–131.

[Mazumdar et al., 2019] Mazumdar, E. V., Jordan, M. I., and Sastry, S. S. (2019). On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*.

[McWilliams et al., 2014] McWilliams, B., Krummenacher, G., Lucic, M., and Buhmann, J. M. (2014). Fast and robust least squares estimation in corrupted linear models. In *Advances in Neural Information Processing Systems*, pages 415–423.

[Melo et al., 2008] Melo, F. S., Meyn, S. P., and Ribeiro, M. I. (2008). An analysis of reinforcement learning with function approximation. In *Proceedings of the 25th international conference on Machine learning*, pages 664–671.

[Mertikopoulos, 2019] Mertikopoulos, P. (2019). *Online optimization and learning in games: Theory and applications*. PhD thesis, Grenoble 1 UGA-Université Grenoble Alpes.

[Mertikopoulos et al., 2018a] Mertikopoulos, P., Lecouat, B., Zenati, H., Foo, C.-S., Chandrasekhar, V., and Piliouras, G. (2018a). Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. *arXiv preprint arXiv:1807.02629*.

[Mertikopoulos et al., 2018b] Mertikopoulos, P., Papadimitriou, C., and Piliouras, G. (2018b). Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2703–2717. SIAM.

[Mescheder et al., 2017] Mescheder, L., Nowozin, S., and Geiger, A. (2017). The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835.

[Metelli et al., 2019a] Metelli, A. M., Ghelfi, E., and Restelli, M. (2019a). Reinforcement learning in configurable continuous environments. In *International Conference on Machine Learning*, pages 4546–4555.

[Metelli et al., 2019b] Metelli, A. M., Manneschi, G., and Restelli, M. (2019b). Policy space identification in configurable environments. *arXiv preprint arXiv:1909.03984*.

[Metelli et al., 2018] Metelli, A. M., Mutti, M., and Restelli, M. (2018). Configurable markov decision processes. In *International Conference on Machine Learning*, pages 3491–3500. PMLR.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[Monderer and Shapley, 1996] Monderer, D. and Shapley, L. S. (1996). Potential games. *Games and economic behavior*, 14(1):124–143.

[Morgenstern and Von Neumann, 1953] Morgenstern, O. and Von Neumann, J. (1953). *Theory of games and economic behavior*. Princeton university press.

[Morimoto and Doya, 2005] Morimoto, J. and Doya, K. (2005). Robust reinforcement learning. *Neural computation*, 17(2):335–359.

[Munk et al., 2016] Munk, J., Kober, J., and Babuška, R. (2016). Learning state representation for deep actor-critic control. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4667–4673. IEEE.

[Myerson, 2013] Myerson, R. B. (2013). *Game theory*. Harvard university press.

[Nagarajan and Kolter, 2017] Nagarajan, V. and Kolter, J. Z. (2017). Gradient descent gan optimization is locally stable. In *Advances in neural information processing systems*, pages 5585–5595.

[Namkoong and Duchi, 2017] Namkoong, H. and Duchi, J. C. (2017). Variance-based regularization with convex objectives. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2975–2984.

[Nash et al., 1950] Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

[Natarajan et al., 2010] Natarajan, S., Kunapuli, G., Judah, K., Tadepalli, P., Kersting, K., and Shavlik, J. (2010). Multi-agent inverse reinforcement learning. In *2010 Ninth International Conference on Machine Learning and Applications*, pages 395–400. IEEE.

[Neu et al., 2012] Neu, G., Gyorgy, A., and Szepesvári, C. (2012). The adversarial stochastic shortest path problem with unknown transition probabilities. In *Artificial Intelligence and Statistics*, pages 805–813. PMLR.

[Ng et al., 2000] Ng, A. Y., Russell, S. J., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2.

[Nguyen et al., 2020] Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839.

[Nie et al., 2018] Nie, W., Narodytska, N., and Patel, A. (2018). Relgan: Relational generative adversarial networks for text generation.

[Nilim and El Ghaoui, 2005] Nilim, A. and El Ghaoui, L. (2005). Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798.

[Nilim and Ghaoui, 2003] Nilim, A. and Ghaoui, L. E. (2003). Robustness in markov decision problems with uncertain transition matrices. In Thrun, S., Saul, L. K., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 839–846. MIT Press.

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

[Omidshafiei et al., 2019] Omidshafiei, S., Papadimitriou, C., Piliouras, G., Tuyls, K., Rowland, M., Lespiau, J.-B., Czarnecki, W. M., Lanctot, M., Perolat, J., and Munos, R. (2019). -rank: Multi-agent evaluation by evolution. *Scientific reports*, 9(1):9937.

[OroojlooyJadid and Hajinezhad, 2019] OroojlooyJadid, A. and Hajinezhad, D. (2019). A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*.

[Osa et al., 2018] Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. (2018). An algorithmic perspective on imitation learning. *CoRR*, abs/1811.06711.

[Osband and Van Roy, 2017] Osband, I. and Van Roy, B. (2017). Why is posterior sampling better than optimism for reinforcement learning? In *International Conference on Machine Learning*, pages 2701–2710. PMLR.

[Osband et al., 2013] Osband, I., Van Roy, B., and Russo, D. (2013). (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*.

[Ota et al., 2020] Ota, K., Oiki, T., Jha, D., Mariyama, T., and Nikovski, D. (2020). Can increasing input dimensionality improve deep reinforcement learning? In *International Conference on Machine Learning*, pages 7424–7433. PMLR.

[Papadimitriou and Piliouras, 2016] Papadimitriou, C. and Piliouras, G. (2016). From nash equilibria to chain recurrent sets: Solution concepts and topology. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 227–235.

[Papadimitriou and Piliouras, 2019] Papadimitriou, C. and Piliouras, G. (2019). Game dynamics as the meaning of a game. *ACM SIGecom Exchanges*, 16(2):53–63.

[Papadimitriou, 1992] Papadimitriou, C. H. (1992). On inefficient proofs of existence and complexity classes. In *Annals of Discrete Mathematics*, volume 51, pages 245–250. Elsevier.

[Papini et al., 2018] Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. (2018). Stochastic variance-reduced policy gradient. In *International Conference on Machine Learning*, pages 4026–4035. PMLR.

[Papoudakis et al., 2019] Papoudakis, G., Christianos, F., Rahman, A., and Albrecht, S. V. (2019). Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

[Paternain et al., 2019] Paternain, S., Mokhtari, A., and Ribeiro, A. (2019). A newton-based method for nonconvex optimization with fast evasion of saddle points. *SIAM Journal on Optimization*, 29(1):343–368.

[Peng et al., 2019] Peng, B., Shen, W., Tang, P., and Zuo, S. (2019). Learning optimal strategies to commit to. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2149–2156.

[Perkins and Precup, 2003] Perkins, T. J. and Precup, D. (2003). A convergent form of approximate policy iteration. In *Advances in neural information processing systems*, pages 1627–1634. Citeseer.

[Perolat et al., 2020] Perolat, J., Munos, R., Lespiau, J.-B., Omidshafiei, S., Rowland, M., Ortega, P., Burch, N., Anthony, T., Balduzzi, D., De Vylder, B., et al. (2020). From poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. *arXiv preprint arXiv:2002.08456*.

[Perolat et al., 2018] Perolat, J., Piot, B., and Pietquin, O. (2018). Actor-critic fictitious play in simultaneous move multistage games. In *International Conference on Artificial Intelligence and Statistics*, pages 919–928. PMLR.

[Perolat et al., 2015] Perolat, J., Scherrer, B., Piot, B., and Pietquin, O. (2015). Approximate dynamic programming for two-player zero-sum markov games. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1321–1329, Lille, France. PMLR.

[Pérolat et al., 2017] Pérolat, J., Strub, F., Piot, B., and Pietquin, O. (2017). Learning nash equilibrium for general-sum markov games from batch data. In *Artificial Intelligence and Statistics*, pages 232–241. PMLR.

[Peters et al., 2010] Peters, J., Mulling, K., and Altun, Y. (2010). Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24.

[Peters and Schaal, 2008a] Peters, J. and Schaal, S. (2008a). Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190.

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

[Pirotta, 2016] Pirotta, M. (2016). Reinforcement learning: from theory to algorithms.

[Pirotta and Restelli, 2016] Pirotta, M. and Restelli, M. (2016). Inverse reinforcement learning through policy gradient minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

[Pirotta et al., 2013] Pirotta, M., Restelli, M., and Bascetta, L. (2013). Adaptive step-size for policy gradient methods. In *Advances in Neural Information Processing Systems*, pages 1394–1402.

[Pomerleau, 1989] Pomerleau, D. A. (1989). Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313.

[Powell, 2007] Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

[Puterman, 2014] Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

[Rabinowitz et al., 2018] Rabinowitz, N., Perbet, F., Song, F., Zhang, C., Eslami, S. A., and Botvinick, M. (2018). Machine theory of mind. In *International Conference on Machine Learning*, pages 4218–4227.

[Rajasekaran et al., 2017] Rajasekaran, S., Zhang, J., and Fu, J. (2017). Inverse reinforce learning with nonparametric behavior clustering. *arXiv preprint arXiv:1712.05514*.

[Rajeswaran et al., 2017] Rajeswaran, A., Lowrey, K., Todorov, E., and Kakade, S. (2017). Towards generalization and simplicity in continuous control. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6553–6564.

[Ramachandran and Amir, 2007] Ramachandran, D. and Amir, E. (2007). Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591.

[Ramponi et al., 2020a] Ramponi, G., Drappo, G., and Restelli, M. (2020a). Inverse reinforcement learning from a gradient-based learner. 33:2458–2468.

[Ramponi et al., 2020b] Ramponi, G., Likmeta, A., Metelli, A. M., Tirinzoni, A., and Restelli, M. (2020b). Truly batch model-free inverse reinforcement learning about multiple intentions. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2369. PMLR.

[Ramponi et al., 2021] Ramponi, G., Metelli, A. M., Concetti, A., and Restelli, M. (2021). Online learning in non-cooperative configurable markov decision process. *AAAI Workshop on Reinforcement Learning in Games*.

[Ramponi and Restelli, 2021] Ramponi, G. and Restelli, M. (2021). Newton optimization on helmholtz decomposition for continuous games. *Thirty-Fifth AAAI conference on artificial intelligence*.

[Rao et al., 1973] Rao, S. S., Chandrasekaran, R., and Nair, K. (1973). Algorithms for discounted stochastic games. *Journal of Optimization Theory and Applications*, 11(6):627–637.

[Ratliff, 2021] Ratliff, L. (2021). An introduction to learning in games.

[Ratliff et al., 2013] Ratliff, L. J., Burden, S. A., and Sastry, S. S. (2013). Characterization and computation of local nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924. IEEE.

[Ratliff et al., 2016] Ratliff, L. J., Burden, S. A., and Sastry, S. S. (2016). On the characterization of local nash equilibria in continuous games. *IEEE Transactions on Automatic Control*, 61(8):2301–2307.

[Ratliff et al., 2007] Ratliff, N., Bradley, D., Bagnell, J. A., and Chestnutt, J. (2007). Boosting structured prediction for imitation learning.

[Ratliff et al., 2006] Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2006). Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736.

[Ravi and Larochelle, 2016] Ravi, S. and Larochelle, H. (2016). Optimization as a model for few-shot learning.

[Rigollet, 2015] Rigollet, P. (2015). High-dimensional statistics. spring 2015. *Massachusetts Institute of Technology: MIT OpenCourseWare*.

[Robinson, 1951] Robinson, J. (1951). An iterative method of solving a game. *Annals of mathematics*, pages 296–301.

[Rosen, 1965] Rosen, J. B. (1965). Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534.

[Rosenberg and Mansour, 2019] Rosenberg, A. and Mansour, Y. (2019). Online convex optimization in adversarial markov decision processes. In *International Conference on Machine Learning*, pages 5478–5486. PMLR.

[Rosenthal, 1973] Rosenthal, R. W. (1973). A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2(1):65–67.

[Rummery and Niranjan, 1994] Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

[Russell, 1998] Russell, S. (1998). Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103.

[Schäfer and Anandkumar, 2019] Schäfer, F. and Anandkumar, A. (2019). Competitive gradient descent. In *Advances in Neural Information Processing Systems*, pages 7623–7633.

[Schulman et al., 2017a] Schulman, J., Chen, X., and Abbeel, P. (2017a). Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*.

[Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

[Schulman et al., 2017b] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Sehnke et al., 2008] Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2008). Policy gradients with parameter-based exploration for control. In *International Conference on Artificial Neural Networks*, pages 387–396. Springer.

[Sessa et al., 2020] Sessa, P. G., Bogunovic, I., Kamgarpour, M., and Krause, A. (2020). Learning to play sequential games versus unknown opponents. *Advances in Neural Information Processing Systems 33*.

[Shalev-Shwartz et al., 2016] Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

[Shapley, 1964] Shapley, L. (1964). Some topics in two-person games. *Advances in game theory*, 52:1–29.

[Shapley, 1953] Shapley, L. S. (1953). Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100.

[Shen et al., 2019] Shen, Z., Ribeiro, A., Hassani, H., Qian, H., and Mi, C. (2019). Hessian aided policy gradient. In *International Conference on Machine Learning*, pages 5729–5738. PMLR.

[Shteingart and Loewenstein, 2014] Shteingart, H. and Loewenstein, Y. (2014). Reinforcement learning and human behavior. *Current Opinion in Neurobiology*, 25:93–98.

[Shum et al., 2019] Shum, M., Kleiman-Weiner, M., Littman, M. L., and Tenenbaum, J. B. (2019). Theory of minds: Understanding behavior in groups through inverse planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6163–6170.

[Sidford et al., 2020] Sidford, A., Wang, M., Yang, L., and Ye, Y. (2020). Solving discounted stochastic two-player games with near-optimal time and sample complexity. In *International Conference on Artificial Intelligence and Statistics*, pages 2992–3002. PMLR.

[Silver et al., 2014] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

[Simchowitz and Jamieson, 2019] Simchowitz, M. and Jamieson, K. (2019). Non-asymptotic gap-dependent regret bounds for tabular mdps. *arXiv preprint arXiv:1905.03814*.

[Singh et al., 2000] Singh, S., Jaakkola, T., Littman, M. L., and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38(3):287–308.

[Skoulakis et al., 2021] Skoulakis, S., Fiez, T., Sim, R., Piliouras, G., and Ratliff, L. (2021). Evolutionary game theory squared: Evolving agents in endogenously evolving zero-sum games.

[Song et al., 2019] Song, X., Wang, T., and Zhang, C. (2019). Convergence of multi-agent learning with a finite step size in general-sum games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 935–943. International Foundation for Autonomous Agents and Multiagent Systems.

[Sorin, 2020] Sorin, S. (2020). Replicator dynamics: Old and new. *Journal of Dynamics & Games*, 7(4):365.

[Spokoiny et al., 2012] Spokoiny, V. et al. (2012). Parametric estimation. finite sample theory. *The Annals of Statistics*, 40(6):2877–2909.

[Sutton et al., 1998] Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

[Sutton et al., 2000] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063.

[Sutton et al., 1999] Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y., et al. (1999). Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer.

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

[Szepesvári et al., 1997] Szepesvári, C. et al. (1997). The asymptotic convergence-rate of q-learning. In *NIPS*, volume 10, pages 1064–1070. Citeseer.

[Szepesvári and Littman, 1996] Szepesvári, C. and Littman, M. L. (1996). Generalized markov decision processes: Dynamic-programming and reinforcement-learning algorithms. In *Proceedings of International Conference of Machine Learning*, volume 96.

[Szepesvári and Littman, 1999] Szepesvári, C. and Littman, M. L. (1999). A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation*, 11(8):2017–2060.

[Taslaman, 2014] Taslaman, L. (2014). The principal angles and the gap.

[Tateo et al., 2017] Tateo, D., Pirotta, M., Restelli, M., and Bonarini, A. (2017). Gradient-based minimization for multi-expert inverse reinforcement learning. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE.

[Tessler et al., 2019] Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR.

[Tian et al., 2020] Tian, Y., Wang, Y., Yu, T., and Sra, S. (2020). Provably efficient online agnostic learning in markov games. *arXiv preprint arXiv:2010.15020*.

[Tirinzoni et al., 2020] Tirinzoni, A., Poiani, R., and Restelli, M. (2020). Sequential transfer in reinforcement learning with a generative model. In *International Conference on Machine Learning*, pages 9481–9492. PMLR.

[Tramèr et al., 2018] Tramèr, F., Boneh, D., Kurakin, A., Goodfellow, I., Papernot, N., and McDaniel, P. (2018). Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*.

[Tseng, 2001] Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494.

[Tsitsiklis, 1994] Tsitsiklis, J. N. (1994). Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202.

[Van Der Wal, 1978] Van Der Wal, J. (1978). Discounted markov games: Generalized policy iteration method. *Journal of Optimization Theory and Applications*, 25(1):125–138.

[Vanderbei et al., 2015] Vanderbei, R. J. et al. (2015). *Linear programming*, volume 3. Springer.

[Von Neumann and Morgenstern, 2007] Von Neumann, J. and Morgenstern, O. (2007). *Theory of games and economic behavior (commemorative edition)*. Princeton university press.

[Von Stackelberg, 2010] Von Stackelberg, H. (2010). *Market structure and equilibrium*. Springer Science & Business Media.

[Vroman, 2014] Vroman, M. C. (2014). *Maximum likelihood inverse reinforcement learning*. PhD thesis, Rutgers University-Graduate School-New Brunswick.

[Wan et al., 2021] Wan, Y., Tu, W.-W., and Zhang, L. (2021). Online strongly convex optimization with unknown delays. *arXiv preprint arXiv:2103.11354*.

[Wang and Klabjan, 2018] Wang, X. and Klabjan, D. (2018). Competitive multi-agent inverse reinforcement learning with sub-optimal demonstrations. In *International Conference on Machine Learning*, pages 5143–5151. PMLR.

[Wang and Sandholm, 2002] Wang, X. and Sandholm, T. (2002). Reinforcement learning to play an optimal nash equilibrium in team markov games. *Advances in neural information processing systems*, 15:1603–1610.

[Watkins and Dayan, 1992] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

[Wedin, 1973] Wedin, P. (1973). Perturbation theory for pseudo-inverses. *BIT Numerical Mathematics*, 13(2):217–232.

[Wei et al., 2017] Wei, C.-Y., Hong, Y.-T., and Lu, C.-J. (2017). Online reinforcement learning in stochastic games. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4994–5004.

[Whittle, 1981] Whittle, P. (1981). Risk-sensitive linear/quadratic/gaussian control. *Advances in Applied Probability*, pages 764–777.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

[Wills, 1958] Wills, A. P. (1958). Vector analysis with and introduction to tensor analysis. Technical report.

[Xie et al., 2020a] Xie, Q., Chen, Y., Wang, Z., and Yang, Z. (2020a). Learning zero-sum simultaneous-move markov games using function approximation and correlated equilibrium. In *Conference on Learning Theory*, pages 3674–3682. PMLR.

[Xie et al., 2020b] Xie, Q., Yang, Z., Wang, Z., and Minca, A. (2020b). Provable fictitious play for general mean-field games. *arXiv preprint arXiv:2010.04211*.

[Yang et al., 2020] Yang, J., Kiyavash, N., and He, N. (2020). Global convergence and variance-reduced optimization for a class of nonconvex-nonconcave minimax problems. *arXiv preprint arXiv:2002.09621*.

[Yoshikawa, 1978] Yoshikawa, T. (1978). Decomposition of dynamic team decision problems. *IEEE Transactions on Automatic Control*, 23(4):627–632.

[Yu and Mannor, 2009] Yu, J. Y. and Mannor, S. (2009). Arbitrarily modulated markov decision processes. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 2946–2953. IEEE.

[Yu et al., 2019] Yu, L., Song, J., and Ermon, S. (2019). Multi-agent adversarial inverse reinforcement learning. In *International Conference on Machine Learning*, pages 7194–7201. PMLR.

[Yu et al., 2017] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Yu et al., 2018] Yu, Y., Wang, T., and Liew, S. C. (2018). Deep-reinforcement learning multiple access for heterogeneous wireless networks. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE.

[Zanette et al., 2019] Zanette, A., Kochenderfer, M. J., and Brunskill, E. (2019). Almost horizon-free structure-aware best policy identification with a generative model. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 5625–5634. Curran Associates, Inc.

[Zhang et al., 2018] Zhang, A., Satija, H., and Pineau, J. (2018). Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*.

[Zhang and Lesser, 2010] Zhang, C. and Lesser, V. (2010). Multi-agent learning with policy prediction. In *Twenty-fourth AAAI conference on artificial intelligence*.

[Zhang et al., 2009] Zhang, H., Chen, Y., and Parkes, D. C. (2009). A general approach to environment design with one agent.

[Zhang et al., 2020a] Zhang, K., Hu, B., and Basar, T. (2020a). Policy optimization for $\mathcal{H}_2$ linear control with $\mathcal{H}_2$ ∞ robustness guarantee: Implicit regularization and global convergence. In *Learning for Dynamics and Control*, pages 179–190. PMLR.

[Zhang et al., 2020b] Zhang, K., Kakade, S., Basar, T., and Yang, L. (2020b). Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity. *Advances in Neural Information Processing Systems*, 33.

[Zhang et al., 2020c] Zhang, K., Sun, T., Tao, Y., Genc, S., Mallya, S., and Basar, T. (2020c). Robust multi-agent reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 33.

[Zhang et al., 2019a] Zhang, K., Yang, Z., and Başar, T. (2019a). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*.

[Zhang et al., 2019b] Zhang, K., Yang, Z., and Basar, T. (2019b). Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games. In *Advances in Neural Information Processing Systems*, pages 11602–11614.

[Zhang et al.,] Zhang, K., Zhang, X., Hu, B., and Başar, T. Derivative-free policy optimization for risk-sensitive and robust control design: Implicit regularization and sample complexity. *arXiv preprint arXiv:2101.01041*.

[Zhao et al., 2016] Zhao, T., Niu, G., Xie, N., Yang, J., and Sugiyama, M. (2016). Regularized policy gradients: direct variance reduction in policy gradient estimation. In *Asian Conference on Machine Learning*, pages 333–348. PMLR.

[Zheng et al., 2021] Zheng, L., Fiez, T., Alumbaugh, Z., Chasnov, B., and Ratliff, L. J. (2021). Stackelberg actor-critic: A game-theoretic perspective.

[Zhou et al., 2018] Zhou, Z., Mertikopoulos, P., Athey, S., Bambos, N., Glynn, P., and Ye, Y. (2018). Learning in games with lossy feedback. In *NIPS 2018-Thirty-second Conference on Neural Information Processing Systems*, pages 1–11.

[Ziebart et al., 2010] Ziebart, B. D., Bagnell, J. A., and Dey, A. K. (2010). Modeling interaction via the principle of maximum causal entropy. In *ICML*.

[Ziebart et al., 2008] Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA.

[Zimin and Neu, 2013] Zimin, A. and Neu, G. (2013). Online learning in episodic markovian decision processes by relative entropy policy search. In *Neural Information Processing Systems 26*.

[Zinkevich et al., 2006] Zinkevich, M., Greenwald, A., and Littman, M. (2006). Cyclic equilibria in markov games. *Advances in Neural Information Processing Systems*, 18:1641.

APPENDIX $\mathcal{A}$

Learning in Continuous Games: a (brief) introduction

In this appendix, we would like to revise the basic concepts of learning in games. In the main manuscript, we provide the necessary background to read and follow the proposed algorithms and theoretical results; however, we would like to give the interested reader a brief introduction to learning in continuous games. Some concepts are also explained in Chapter 11. Obviously, this appendix is not sufficient to completely learn this wide topic, and if the reader is excited about this argument, we would like to suggest reading [Fudenberg et al., 1998] and also the notes of [Ratliff, 2021] as well as the manuscript of [Mertikopoulos, 2019].

A.1 Continuous Games

In continuous games, the players act on continuous action spaces. However, every normal-form finite game can be cast as a continuous game if we consider the set of possible actions of the mixed strategies. We introduce the continuous games using the definition used in Chapter 11.

Definition A.1.1 (Continuous Games). *A n-agents continuous game is de-*

fined by $(\Theta, C_1, \dots, C_n)$ *where* $\Theta = (\Theta_1, \dots, \Theta_n)$*, with* $\Theta_i \in \mathbb{R}^d$*, are the parameters' space of each agent and* $C_i : \Theta \to \mathbb{R}$ *is the cost function of the i-th agent.*

Since the cost function might not be convex, it is necessary to introduce the concept of Local Nash Equilibrium.

Definition A.1.2 (Local Nash Equilibrium)**.** *A point* $\boldsymbol{\theta}^*$ *is a local Nash Equilibrium if, for each agent* $1 \leq i \leq n$*, there is a neighborhood* B_i *of* $\boldsymbol{\theta}_i^*$ *such that* $C_i(\boldsymbol{\theta}_i, \boldsymbol{\theta}_{-i}^*) \geq C_i(\boldsymbol{\theta}_i^*, \boldsymbol{\theta}_{-i}^*)$ *for any* $\boldsymbol{\theta}_i \in B_i$*.*

If the relation is strictly greater than then, the point is a strict local Nash Equilibrium. From an optimization viewpoint in [Ratliff et al., 2013] the authors introduced the concept of *differential Nash Equilibrium.*

Definition A.1.3 (Differential Nash Equilibrium)**.** *A point* $\boldsymbol{\theta}^*$ *is a differential Nash Equilibrium if, for each agent* $1 \leq i \leq n$*, if* $\nabla_{\boldsymbol{\theta}_i} C_i = 0$ *and* $\nabla^2_{\boldsymbol{\theta}_i} C_i > 0$*.*

Every differential Nash Equilibrium is also a strict local Nash Equilibria [Ratliff et al., 2013]. Moreover for every local Nash Equilibria it is true that $\nabla_{\boldsymbol{\theta}_i} C_i = 0$ and $\nabla^2_{\boldsymbol{\theta}_i} C_i \geq 0$ for every $1 \leq i \leq n$. It is important to notice that it is only a necessary condition.

In continuous games, there are many classes of games: potential games, hamiltonian games, convex games, and non-convex games. We refer the reader to Chapter 11 where we introduce the Potential Games and Hamiltonian Games.

Convex games Convex games are games where the cost functions are convex, and the strategy spaces are convex and compact. For these games, many interesting results were derived. The most important is the existence of pure Nash Equilibria for this class of games [Rosen, 1965]. Moreover, in [Rosen, 1965] it was showed that under suitable assumptions, the Nash equilibrium is unique.

Non-convex games Non-convex games are games that have non-convex cost functions. Nash equilibria in non-convex games are NP-hard to compute in general, even in zero-sum settings [Ko and Lin, 1995, Daskalakis et al., 2020]. In zero-sum setting the problem is connected to min-max games, i.e., games where there is a *minimizer* player and a *maximizer* player.

A.2 Classical learning dynamics

In this section, we briefly introduce some *classical* learning dynamics.

Best response and gradient dynamics The best response dynamics are given at each step considering as updating rule for each player its best response to the current strategies of the other players. The gradient dynamics are an approximation of the best response dynamics. The players follow the gradient of their cost functions. These learning dynamics converge to Nash Equilibrium solutions only in a certain class of games, as for example, Potential games (in this class of games, we are optimizing a single loss function, called potential function (see Chapter 11)). In other games, such as zero-sum games, following this dynamic exhibit cyclic behaviors [Hart and Mas-Colell, 2003, Mertikopoulos et al., 2018b].

Fictitious play Fictitious play is one of the first learning rules studied in literature [Robinson, 1951, Fudenberg et al., 1998, Lambert Iii et al., 2005, Perolat et al., 2018, Xie et al., 2020b]. In this learning dynamic, every player plays the best response to the historical average of the other player. Also, this learning dynamics converges only for some games, as for example, potential games. However, in [Shapley, 1964] it is showed that in some games, the learning dynamic will cycle indefinitely.

Replicator dynamics Another famous class of algorithms for learning in games is replicator dynamics, and in the discrete-time multiplicative weights [Börgers and Sarin, 1997, Hofbauer et al., 1998, Cesa-Bianchi and Lugosi, 2006]. This dynamic is mostly studied in evolutionary game theory [Hofbauer et al., 1998]. In the context of a game, it has been mostly studied for bimatrix games [Sorin, 2020]. In this case, the dynamic does not converge to equilibrium solutions for every game.

As we have seen, constructing dynamics that converge to the Nash Equilibrium solution is really challenging, and this is a currently active research area.

APPENDIX $\mathcal{B}$

Supporting lemmas for Chapter 6

Lemma B.0.1. *Let* $\mathbf{A} \in \mathbb{R}^{d\times dq}$ *and* $\mathbf{B} \in \mathbb{R}^{dq\times dq}$ *symmetric positive definite. Then, it holds that:*

$$\left\| \mathbf{A}^T \left(\mathbf{A}\mathbf{B}\mathbf{A}^T\right)^{-1} \mathbf{A} \right\|_F \leq \frac{\sqrt{d}}{s_{\min}(\mathbf{B})}.$$

Proof. First recall that for a symmetric positive definite matrix the following identity involving the square root holds:

$$\left(\mathbf{B}^{\frac{1}{2}}\right)^T = \left(\mathbf{B}^T\right)^{\frac{1}{2}} = \mathbf{B}^{\frac{1}{2}}.$$

Consider now the following derivation:

$$\begin{aligned}
\left\| \mathbf{A}^T \left(\mathbf{A}\mathbf{B}\mathbf{A}^T\right)^{-1} \mathbf{A} \right\|_F &= \left\| \mathbf{B}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}\mathbf{A}^T \left(\mathbf{A}\mathbf{B}\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{B}^{\frac{1}{2}}\mathbf{B}^{-\frac{1}{2}} \right\|_F \\
&\leq \left\| \mathbf{B}^{-\frac{1}{2}} \right\|_2^2 \left\| \mathbf{B}^{\frac{1}{2}}\mathbf{A}^T \left(\mathbf{A}\mathbf{B}\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{B}^{\frac{1}{2}} \right\|_F \\
&= \frac{1}{s_{\min}(\mathbf{B})} \left\| \mathbf{B}^{\frac{1}{2}}\mathbf{A}^T \left(\mathbf{A}\mathbf{B}\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{B}^{\frac{1}{2}} \right\|_F,
\end{aligned}$$

where we exploited the inequality $\|\mathbf{X}\mathbf{Y}\|_F \leq \|\mathbf{X}\|_2 \|\mathbf{Y}\|_F$ and fact that $\left\|\mathbf{B}^{-\frac{1}{2}}\right\|_2 \frac{1}{\sqrt{s_{\min}(\mathbf{B})}}$. Let us bound the second term.

$$\begin{aligned}\left\|\mathbf{B}^{\frac{1}{2}}\mathbf{A}^T\left(\mathbf{ABA}^T\right)^{-1}\mathbf{AB}^{\frac{1}{2}}\right\|_F^2 &= \operatorname{tr}\left(\mathbf{B}^{\frac{1}{2}}\mathbf{A}^T\left(\mathbf{ABA}^T\right)^{-1}\mathbf{AB}^{\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}\mathbf{A}^T\left(\mathbf{ABA}^T\right)^{-1}\mathbf{AB}^{\frac{1}{2}}\right)\\ &= \operatorname{tr}\left(\mathbf{ABA}^T\left(\mathbf{ABA}^T\right)^{-1}\mathbf{ABA}^T\left(\mathbf{ABA}^T\right)^{-1}\right)\\ &= \operatorname{tr}\left(\mathbf{I}_d\mathbf{I}_d\right) = d,\end{aligned}$$

where we exploited the identity $\|\mathbf{X}\|_F^2 = \operatorname{tr}\left(\mathbf{X}^T\mathbf{X}\right)$ and the cyclic property of the trace. $\square$

Lemma B.0.2. *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{dq \times dq}$ be two symmetric positive semidefinite matrices. Then, for any $\boldsymbol{\omega} \in \mathbb{R}^q_+$ it holds that:*

$$\begin{aligned}&\left|l_{\mathbf{A}}(\boldsymbol{\omega}) - l_{\ s_{\min}(\mathbf{Q})vB}(\boldsymbol{\omega})\right|\\ &\leq \left\|(\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\right\|_F\\ &\quad\times\left\|(\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\right\|_F\left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right\|_F^2\|\mathbf{B}-\mathbf{A}\|_F.\end{aligned}$$

Proof. We explicitly write down the expression of $l_{\mathbf{A}}(\boldsymbol{\omega})$ and $l_{\mathbf{B}}(\boldsymbol{\omega})$ and perform a sequence of algebric manipulations:

$$\begin{aligned}&l_{\mathbf{A}}(\boldsymbol{\omega}) - l_{\mathbf{B}}(\boldsymbol{\omega}) = \left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right\|^2_{\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}} - \left\|\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right\|^2_{\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}}\\ &= \boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega} - \boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\\ &= \boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left\{\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1} - \left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\right\}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\\ &= \boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\left\{\mathbf{I}_{dq} - \left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\right\}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\\ &= \boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\left\{(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d) - (\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right\}\\ &\quad\times\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\\ &= \boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\{\mathbf{B}-\mathbf{A}\}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\\ &= \operatorname{tr}\left(\boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\{\mathbf{B}-\mathbf{A}\}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\right)\\ &= \operatorname{tr}\left((\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\{\mathbf{B}-\mathbf{A}\}\right)\\ &= \operatorname{vec}\left((\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\right)\\ &\quad\times\operatorname{vec}(\mathbf{B}-\mathbf{A})\\ &\leq \left\|\operatorname{vec}\left((\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\right)\right\|_2\\ &\quad\times\|\operatorname{vec}(\mathbf{B}-\mathbf{A})\|_2\\ &= \left\|(\boldsymbol{\omega}\otimes\mathbf{I}_d)\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{B}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\boldsymbol{\omega}\boldsymbol{\omega}^T\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})^T\left[(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\mathbf{A}(\boldsymbol{\omega}\otimes\mathbf{I}_d)\right]^{-1}(\boldsymbol{\omega}\otimes\mathbf{I}_d)^T\right\|_F\|\mathbf{B}-\mathbf{A}\|_F,\end{aligned}$$

where we applied the trace since the quantity is scalar, we exploited the cyclic property of the trace, we used the inequality $\operatorname{tr}(\mathbf{X}^T\mathbf{Y}) = \operatorname{vec}(\mathbf{X})^T\operatorname{vec}(\mathbf{Y})$, Cauchy-Swartz inequality and finally observed that $\|\mathrm{vec}(\mathbf{X})\|_2 = \|\mathbf{X}\|_F$. To conclude consider:

$$\begin{aligned}
\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega} &= \mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega}\right) \\
&= \mathrm{vec}\left(\mathbf{I}_d \widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta}) \boldsymbol{\omega}\right) \\
&= \left(\boldsymbol{\omega}^T \otimes \mathbf{I}_d\right) \mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})\right) \\
&= \left(\boldsymbol{\omega} \otimes \mathbf{I}_d\right)^T \mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}} \boldsymbol{\psi}(\boldsymbol{\theta})\right).
\end{aligned}$$

Using the properties of the Frobenius norm, the result follows. □

Lemma B.0.3. *Let* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ *any pair of vectors, then it holds that:*

$$\left\| \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \right\|_2 \le \frac{2\|\mathbf{x}-\mathbf{y}\|_2}{\max\{\|\mathbf{x}\|_2, \|\mathbf{y}\|_2\}}.$$

Proof. The result follows from the following sequence of algebraic manipulations:

$$\begin{aligned}
\left\| \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \right\|_2 &= \left\| \frac{\mathbf{x}}{\|\mathbf{x}\|_2} - \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \pm \frac{\mathbf{y}}{\|\mathbf{x}\|_2} \right\|_2 \\
&\le \frac{\|\mathbf{x}-\mathbf{y}\|_2}{\|\mathbf{x}\|_2} + \frac{\left|\|\mathbf{x}\|_2 - \|\mathbf{y}\|_2\right|}{\|\mathbf{x}\|_2} \\
&\le 2\frac{\|\mathbf{x}-\mathbf{y}\|_2}{\|\mathbf{x}\|_2},
\end{aligned}$$

where we applied the triangular inequality in the second line and the reverse triangular inequality in the last one, i.e. $\left|\|\mathbf{x}\|_2 - \|\mathbf{y}\|_2\right| \le \|\mathbf{x}-\mathbf{y}\|_2$. By observing that, for symmetry reasons, the same derivation can be performed getting $\|\mathbf{y}\|_2$ at the denominator, we get the result. □

Lemma B.0.4. *Let* $\mathbf{A} = (\mathbf{a}_1|\dots|\mathbf{a}_q)$, $\mathbf{B} = (\mathbf{b}_1|\dots|\mathbf{b}_q) \in \mathbb{R}^{d \times q}$ *be two matrices of rank* $q-1$ *such that* $s_{q-1}(\mathbf{B}) > 0$, *where* s_{q-1} *denotes the* $(q-1)$*-th singular value. Let* $\mathcal{A} = \mathrm{span}(\{\mathbf{a}_1,\dots,\mathbf{a}_q\})$ *and* $\mathcal{B} = \mathrm{span}(\{\mathbf{b}_1,\dots,\mathbf{b}_q\})$ *be the vector spaces generated by the columns of* $\mathbf{A}$ *and* $\mathbf{B}$ *respectively. Then, the cosine of the (principal) angle* α *between the corresponding orthogonal complements* $\mathcal{A}^\perp$ *and* $\mathcal{B}^\perp$ *is lower bounded by:*

$$\cos\alpha = \cos\sphericalangle\left(\mathcal{A}^\perp, \mathcal{B}^\perp\right) \ge 1 - \frac{2}{s_{q-1}(\mathbf{A})^2} \min_{\mathbf{\Pi} \in \mathrm{Perm}_q} \|\mathbf{A} - \mathbf{B}\mathbf{\Pi}\|_F^2,$$

where Perm_q *is the set of all permutation matrices of order* q *and* $\|\cdot\|_F$ *denotes the Frobenius norm.*

Proof. Since both matrices $\mathbf{A}$ and $\mathbf{B}$ have rank $q-1$, the orthogonal complements $\mathcal{A}^\perp$ and $\mathcal{B}^\perp$ have dimension 1. Since the principal angles (which in this case is just one) of the orthogonal complements are essentially the same as those of the correponding spaces [Knyazev et al., 2010], we reduce the problem to the computation of $\sphericalangle(\mathcal{A}, \mathcal{B})$. In particular, we are interested in the maximum (and only non-zero) principal angle α, whose cosine can be conveniently defined as [Taslaman, 2014]:

$$\cos\alpha = \min_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1}} \max_{\substack{\mathbf{y}\in\mathbb{R}^q \\ \|\mathbf{B}\mathbf{y}\|_2=1}} (\mathbf{A}\mathbf{x})^T\mathbf{B}\mathbf{y} = 1 - \frac{1}{2} \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1}} \min_{\substack{\mathbf{y}\in\mathbb{R}^q \\ \|\mathbf{B}\mathbf{y}\|_2=1}} \|\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{y}\|_2^2,$$

where the identity follows from recalling that $\|\mathbf{a}-\mathbf{b}\|_2^2 = \|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2 - 2\mathbf{a}^T\mathbf{b}$. Consider now the set $\mathcal{X} = \{\mathbf{x}\in\mathbb{R}^q : \|\mathbf{A}\mathbf{x}\|_2 = 1\}$. Since $\mathbf{A}$ is not full rank, the set $\mathcal{X}$ will contain vectors with non-zero projection onto the null-space of $\mathbf{A}$. Thus, for any $\mathbf{x}\in\mathcal{X}$ we can write $\mathbf{x} = \mathbf{x}^{\perp} + \mathbf{x}^{\parallel}$, where $\mathbf{x}^{\perp}\in \text{null}(\mathbf{A})$ and $\mathbf{x}^{\parallel} \perp \text{null}(\mathbf{A})$. Furthermore, we have that $\mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{x}^{\perp} + \mathbf{x}^{\parallel}) = \mathbf{A}\mathbf{x}^{\parallel}$, by definition of null space. Therefore, for the computation of the min, we can limit our search of $\mathbf{x}$ to the set $\{\mathbf{x}\in\mathbb{R}^q : \|\mathbf{A}\mathbf{x}\|_2 = 1 \wedge \mathbf{x}\perp\text{null}(\mathbf{A})\}$. Let $\mathbf{\Pi}$ be a permutation matrix, we now consider the following sequence of inequalities:

$$\begin{aligned}
\max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \min_{\substack{\mathbf{y}\in\mathbb{R}^q \\ \|\mathbf{B}\mathbf{y}\|_2=1}} \|\mathbf{A}\mathbf{x}-\mathbf{B}\mathbf{y}\|_2 &\le \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \min_{\mathbf{\Pi}\in\text{Perm}_q} \left\|\mathbf{A}\mathbf{x} - \frac{\mathbf{B}\mathbf{\Pi}\mathbf{x}}{\|\mathbf{B}\mathbf{\Pi}\mathbf{x}\|_2}\right\|_2^2 && \text{(B.1)} \\
&\le 2 \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \min_{\mathbf{\Pi}\in\text{Perm}_q} \frac{\|\mathbf{A}\mathbf{x}-\mathbf{B}\mathbf{\Pi}\mathbf{x}\|_2}{\max\{1, \|\mathbf{B}\mathbf{\Pi}\mathbf{x}\|_2\}} && \text{(B.2)} \\
&\le 2 \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \min_{\mathbf{\Pi}\in\text{Perm}_q} \|\mathbf{A}\mathbf{x}-\mathbf{B}\mathbf{\Pi}\mathbf{x}\|_2 && \text{(B.3)} \\
&\le 2 \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \min_{\mathbf{\Pi}\in\text{Perm}_q} \left\|\sum_{i=1}^q x_i\left(\mathbf{a}_i - \mathbf{b}_{\pi(i)}\right)\right\|_2 && \text{(B.4)} \\
&\le 2 \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \min_{\mathbf{\Pi}\in\text{Perm}_q} \sum_{i=1}^q |x_i| \left\|\mathbf{a}_i - \mathbf{b}_{\pi(i)}\right\|_2 && \text{(B.5)} \\
&\le 2 \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \|\mathbf{x}\|_2 \min_{\mathbf{\Pi}\in\text{Perm}_q} \sqrt{\sum_{i=1}^q \left\|\mathbf{a}_i - \mathbf{b}_{\pi(i)}\right\|_2^2} && \text{(B.6)} \\
&\le 2 \max_{\substack{\mathbf{x}\in\mathbb{R}^q \\ \|\mathbf{A}\mathbf{x}\|_2=1 \\ \mathbf{x}\perp\text{null}(\mathbf{A})}} \|\mathbf{x}\|_2 \min_{\mathbf{\Pi}\in\text{Perm}_q} \|\mathbf{A}-\mathbf{B}\mathbf{\Pi}\|_F, && \text{(B.7)}
\end{aligned}$$

where line (B.1) follows from bounding the min over $\mathbf{y}$ with a specific choice of $\mathbf{y} = \mathbf{\Pi}\mathbf{x}$. Line (B.2) is obtained from Lemma B.0.3 and line (B.3) derives from bounding the maximum at the denominator with its first argument. Line (B.4) follows from the definition of permutation matrix, having denoted with $\pi : \{1,\dots,q\} \to \{1,\dots,q\}$ the permutation realized by $\mathbf{\Pi}$. Line (B.5) follows from expanding the expression at the previous line, while line (B.6) is an application of Cauchy-Swartz inequality. Finally, line (B.7) is obtained from the definition of Frobenius norm. To conclude, we bound the norm $\|\mathbf{x}\|_2$ under the constraints $\|\mathbf{A}\mathbf{x}\|_2 = 1$ and $\mathbf{x}\perp\text{null}(\mathbf{A})$. For this purpose, we consider the singular value decomposition of $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, where $\mathbf{S} = \text{diag}(s_1,\dots,s_{q-1},0)$ and $s_{q-1} > 0$ for the hypothesis. Moreover, let $\mathbf{V} = (\mathbf{v}_1|\dots|\mathbf{v}_q)$, we know that $\text{null}(\mathbf{A}) = \text{span}(\{\mathbf{v}_q\})$. Therefore, our chosen $\mathbf{x}$ is orthogonal to $\mathbf{x}^T\mathbf{v}_q = \mathbf{0}$. We now consider the matrix-vector product norm:

$$\begin{aligned}\|\mathbf{A}\mathbf{x}\|_2^2 &= \left\|\mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{x}\right\|_2^2 = \mathbf{x}^T\mathbf{V}\mathbf{S}\mathbf{U}^T\mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{x} = \mathbf{x}^T\mathbf{V}\mathbf{S}^2\mathbf{V}^T\mathbf{x} = \sum_{i=1}^{q-1} s_i^2(\mathbf{x}^T\mathbf{v}_i)^2 \\ &\geq s_{q-1}^2\sum_{i=1}^{q-1}(\mathbf{x}^T\mathbf{v}_i)^2 = s_{q-1}^2\|\mathbf{x}\|_2^2,\end{aligned}$$

where we exploited the fact that $\mathbf{U}$ is a unitary matrix and the fact that $\sum_{i=1}^{q-1}(\mathbf{x}^T\mathbf{v}_i)^2 = \|\mathbf{x}\|_2^2$, being the vectors of $\mathbf{V}$ an orthonormal basis. Using this result, and recalling that $\|\mathbf{A}\mathbf{x}\|_2 = 1$, we can upper bound the value of $\|\mathbf{x}\|_2$, to get the result. □

Lemma B.0.5. *Let* $\mathrm{M}(\widehat{\boldsymbol{\omega}})$ *be the approximate Jacobian recovered by* Σ-*GIRL run with the covariance matrix* $\boldsymbol{\Sigma}$*, starting from the sample Jacobian* $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$*. Let* $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ *be the true Jacobian. Then, it holds that:*

$$\|\mathrm{vec}\left(\mathbf{M}(\widehat{\boldsymbol{\omega}}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\|_2^2 \leq 4\|\boldsymbol{\Sigma}\|_2\left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\right\|_{\boldsymbol{\Sigma}^{-1}}^2. \tag{B.8}$$

Proof. Given a vector $\mathbf{x}$, we upper bound the norm $\|\mathbf{x}\|_2$ with $\|\mathbf{x}\|_{\boldsymbol{\Sigma}^{-1}}$:

$$\|\mathbf{x}\|_{\boldsymbol{\Sigma}^{-1}}^2 = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mathbf{x} \geq s_{\min}\left(\boldsymbol{\Sigma}^{-1}\right)\mathbf{x}^T\mathbf{x} = s_{\min}\left(\boldsymbol{\Sigma}^{-1}\right)\|\mathbf{x}\|_2^2,$$

where $s_{\min}(\cdot)$ is the minimum singular value of a matrix. Now, since $s_{\min}\left(\boldsymbol{\Sigma}^{-1}\right) = \frac{1}{s_{\max}(\boldsymbol{\Sigma})} = \frac{1}{\|\boldsymbol{\Sigma}\|_2}$, we have that $\|\mathbf{x}\|_2^2 \leq \|\boldsymbol{\Sigma}\|_2\|\mathbf{x}\|_{\boldsymbol{\Sigma}^{-1}}^2$. Additionally, if $\boldsymbol{\Sigma}$ is the covariance matrix that is used for recovering $\mathbf{M}(\widehat{\boldsymbol{\omega}})$ it follows that the distance between $\mathbf{M}(\widehat{\boldsymbol{\omega}})$ and $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ cannot be larger than twice the distance between $\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})$:

$$\|\mathrm{vec}\left(\mathbf{M}(\widehat{\boldsymbol{\omega}}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\|_{\boldsymbol{\Sigma}^{-1}} \leq 2\left\|\mathrm{vec}\left(\widehat{\nabla}_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}(\boldsymbol{\theta})\right)\right\|_{\boldsymbol{\Sigma}^{-1}}. \tag{B.9}$$

Putting these two inequalities together, we get the result. □

APPENDIX *C*

Supporting lemmas and additional results for Chapter 7

C.1 Matrix perturbation and linear least square problems

Definition C.1.1. *The condition number of a matrix* $\mathbf{A} \in R^{m\times q}$ $\mathbf{A} \neq 0$ *is:*

$$\kappa = \|\mathbf{A}\|_2 \left\|\mathbf{A}^+\right\|_2 = \frac{\sigma_1}{\sigma_r},$$

where 0 < r = rank($\mathbf{A}$*)* $\leq \min(m, q)$*, and* $\sigma_1 \geq \cdots \geq \sigma_r > 0$ *are the nonzero singular values of* $\mathbf{A}$*.*

A least-squares problem is defined as:

$$\min_x \|\mathbf{A}x - b\|_2, \tag{C.1}$$

where the solution is $x = \mathbf{A}^+ b$. We denote with $\mathbf{A}^+$ the pseudoinverse of $\mathbf{A}$, the perturbed $\mathbf{A}$ as $\widehat{\mathbf{A}} = \mathbf{A} + \delta\mathbf{A}$ and the pertubed $\hat{b} = b + \delta b$ and the perturbed solution $\widehat{x} = \widehat{\mathbf{A}}^+ \hat{b} = x + \delta x$. Finally, we denote with $\mathbf{A}^H$ the adjoint of the matrix $\mathbf{A}$.

We define as $\chi = \frac{\|\delta\mathbf{A}\|_2}{\|\mathbf{A}\|_2}$ and $y = \mathbf{A}^{+H} x$.

Lemma C.1.1 (Perturbation on Least Square Problems [Wedin, 1973]). *Assume that rank($\mathbf{A} + \delta\mathbf{A}$) = rank($\mathbf{A}$) and $\chi\kappa < 1$ then:*

$$\|x - \hat{x}\|_2 \leq \frac{\kappa}{(1-\chi\kappa)\|\mathbf{A}\|_2}(\chi\|x\|_2\|\mathbf{A}\|_2 + \chi\kappa\|r\|_2 + \|\delta b\|_2) + \chi\|y\|_2\|\mathbf{A}\|_2 .$$

Proof. The proof can be find in [Wedin, 1973]. □

We adapt the lemma 6 in [Chen and Caramanis, 2013] to our context where $\widehat{\boldsymbol{\omega}}$ are the reward weights recovered with lemma 7.2.1.

Lemma C.1.2 (From lemma 6 in [Chen and Caramanis, 2013]). *Let* $\Sigma = (\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}^T \widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}})$ *and suppose the following strong convexity condition holds:* $\lambda_{\min}(\Sigma) \geq \lambda > 0$. *Then the estimation error satisfies:*

$$\left\|\widehat{\boldsymbol{\omega}} - \boldsymbol{\omega}^L\right\|_2 \leq O\left(\frac{1}{\lambda}\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^T\Delta - \Sigma\boldsymbol{\omega}^L\right\|_2\right).$$

Lemma C.1.3 (Revised from lemma 11 in [McWilliams et al., 2014]). *Suppose $X \in \mathbb{R}^{m\times q}$ and $W \in \mathbb{R}^{n\times M}$ are zero-mean sub-gaussian matrices with parameters $(\frac{1}{n}\Sigma_x, \frac{1}{n}\sigma_x^2)$, $(\frac{1}{n}\Sigma_w, \frac{1}{n}\sigma_w^2)$ respectively. Then for any fixed vectors v_1, v_2, we have:*

$$P[|v_1^T(W^TX - \mathbb{E}[W^TW])v_2| \geq t\|v_1\|_2\|v_2\|_2] \leq 3exp\left(-cn\min\left\{\frac{t^2}{\sigma_x^2\sigma_w^2}, \frac{t}{\sigma_x\sigma_w}\right\}\right),$$

in particular if $n \gtrsim \log p$ we have that:

$$|v_1^T(W^TX - \mathbb{E}[W^TW])v_2| \leq \sigma_x\sigma_w\|v_1\|_2\|v_2\|_2\sqrt{\frac{\log p}{n}}.$$

Setting v_1 to be the first standard basis vector and using a union bound over $j = 1, \cdots, p$ we have:

$$\left\|(W^TX - \mathbb{E}[W^TX])v\right\|_\infty \leq \sigma_x\sigma_w\|v\|_2\sqrt{\frac{\log p}{n}},$$

with probability $1 - c_1exp(-c_2\log p)$ where c_1, c_2 are positive constants which are independent from σ_x, σ_w, n, p.

Theorem C.1.1 (from Chapter 2 [Rigollet, 2015]). *Assume that the least-squares model:*

$$\min_x \|\mathbf{A}x - b + \epsilon\|$$

holds where $\epsilon \sim$ subGn(σ^2). Then, for any $\delta > 0$, with probability $1-\delta$ it holds:

$$\|x - \hat{x}\|_2 \leq \sigma\sqrt{\frac{r + \log(\frac{1}{\delta})}{n\sigma_{\min}}},$$

where $\sigma_{\min} = \frac{\mathbf{A}^T\mathbf{A}}{n}$ *is the minimum singular value of* $\mathbf{A}^T\mathbf{A}$ *and* r *is the rank(*$\mathbf{A}^T\mathbf{A}$*).*

C.2 Additional results

In this section we give give additional results for Chapter 7.

First, we will provide a finite sample analysis on the difference in norm between the reward vector of the learner $\boldsymbol{\omega}^L$ and the reward vector recoverd using (7.3), with a single learning step. This result was omitted in the Chapter as we can see this as a special case of Theorem 7.2.1, but with a different technique. We add it here as it provides a first insight on how, having enough demonstrations, we can recover the correct weights. In the demonstration, without loss of generality, we assume that the learning rate is 1.

Lemma C.2.1. *Let* $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}$ *be the real Jacobian and* $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ *the estimated Jacobian from* n *trajectories* $\{\tau_1, \cdots, \tau_n\}$*. Assume that* $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ *is uniformly bounded by* M*. Then with probability* $1-\delta$

$$\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2 \le M\sqrt{qd}\sqrt{\frac{\log(\frac{2}{\delta})}{2n}}.$$

Proof. We use Hoeffding's inequality:

$$\mathrm{P}\left[\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2 \ge t\right] \le \mathrm{P}\left[\sqrt{qd}\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_\infty \ge t\right] \le 2\exp\left(\frac{-2t^2n}{dqM^2}\right)$$

The result follows by setting $\delta = 2\exp\left(\frac{-2t^2n}{dqM^2}\right)$. □

Theorem C.2.1. *Let* $\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}$ *be the real Jacobian and* $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ *the estimated Jacobian from* n *trajectories* $\{\tau_1, \cdots, \tau_n\}$*. Assume that* $\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}$ *is uniformly bounded by* M*,* $rank(\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}) = rank(\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi})$ *and* $\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2 \cdot \kappa_{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} < \|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2$*. Then with probability* $1-\delta$*:*

$$\left\|\boldsymbol{\omega}^L - \hat{\boldsymbol{\omega}}\right\|_2 \le M\sqrt{qd}\sqrt{\frac{\log(\frac{2}{\delta})}{2n}}\left(\frac{\kappa_{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}}\left\|\boldsymbol{\omega}^L\right\|_2}{c\left\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2} + \|y\|_2\right),$$

where ω^L *are the real reward parameters and* $\hat{\omega}$ *are the parameters recovered with Equation* (7.3)*,* $c = 1 - \frac{\left\|\widehat{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} - \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\right\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\kappa_{\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}} > 0$*, and* $y = \nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}^{+H}\boldsymbol{\omega}$*.*

Proof. We need to bound the difference in norm between $\boldsymbol{\omega}^L$ and $\widehat{\boldsymbol{\omega}}$ that are the true parameters and the parameters that we recovered solving the minimization problem (7.2).

$$\begin{aligned}
&\left\|\boldsymbol{\omega}^L - \widehat{\boldsymbol{\omega}}\right\|_2 \\
&\leq \frac{\kappa}{\left(1-\kappa\frac{\|\delta\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\right)\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\left(\frac{\|\delta\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\left\|\boldsymbol{\omega}^L\right\|_2\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2\right)+\frac{\|\delta\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\|y\|_2\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2 \qquad \text{(C.2)}\\
&\leq \frac{\kappa}{c\,\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\left(\frac{\|\delta\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\left\|\boldsymbol{\omega}^L\right\|_2\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2\right)+\frac{\|\delta\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}{\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}\|y\|_2\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2 \qquad \text{(C.3)}\\
&= \|\delta\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2\left(\frac{\kappa\left\|\boldsymbol{\omega}^L\right\|_2}{c\,\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}+\|y\|_2\right) \qquad \text{(C.4)}\\
&\leq M\sqrt{qd}\sqrt{\frac{\log(\frac{2}{\delta})}{2n}}\left(\frac{\kappa\left\|\boldsymbol{\omega}^L\right\|_2}{c\,\|\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}\|_2}+\|y\|_2\right), \qquad \text{(C.5)}
\end{aligned}$$

where line C.2 is obtained by using Lemma C.1.1, lines C.2-C.4 by rearranging the terms, and line C.5 by using Lemma C.2.1. We can observe that the last term vanishes when the rank($\nabla_{\boldsymbol{\theta}}\boldsymbol{\psi}$)= q (see [Wedin, 1973]). □