McGill University

Temporal Representation Learning

by

Pierre-Luc Bacon

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Science
School of Computer Science, McGill University, Montreal

June 2018

McGill University, Montreal

Abstract

Faculty of Science
School of Computer Science, McGill University, Montreal

Doctor of Philosophy

by Pierre-Luc Bacon

Throughout this thesis, I develop the idea that the problem of learning good temporal abstractions in reinforcement learning is intimately tied to a kind of representation learning problem that I call *temporal representation learning*. This choice of terminology is meant to highlight the fact that *time* is an indissociable aspect of learning systems experiencing their environment. Reconciling the notion of time with learning leads us to solution methods that better fit into a continual learning paradigm (Ring, 1991). Based on this principle, I propose the *option-critic* architecture for constructing temporal abstractions end-to-end, in the *options* framework (Sutton et al., 1999a). Furthermore, acknowledging the importance of time renders inevitable the fact that learning should be as efficient as possible when facing limited resources. I express this idea in the bounded rationality framework of Simon (1957) and extend the option-critic architecture accordingly to learn better options. I also establish a connection between the bounded rationality perspective on options discovery and the problem of finding good matrix preconditioners in numerical linear algebra. I show this point formally by exposing a common mathematical structure between multi-step RL methods and options using matrix splitting theory (Varga, 1962).

Université McGill

Abrégé

Faculté des Sciences
School of Computer Science, Université McGill, Montréal

Doctor of Philosophy

par Pierre-Luc Bacon

Dans cette thèse, je propose l'idée que le problème d'apprentissage de bonnes abstractions temporelles est intimement lié à un problème d'apprentissage de représentation que j'appelle *apprentissage temporel de représentation*. Ce choix de terminologie vise à rappeler le fait que le *temps* est un aspect indissociable d'un système d'apprentissage plongé dans son expérience d'apprentissage avec son environnement. La réconciliation de la notion du temps avec celle de l'apprentissage mène à des solutions qui s'intègrent mieux dans un paradigme d'apprentissage continuel (Ring, 1991). Basé sur ce principe, je propose l'architecture *option-critic* pour construire des abstractions temporelles bout-en-bout (*end-to-end*) dans le cadre théorique des *options* (Sutton et al., 1999a). De plus, reconnaître l'importance du temps rend inévitable le fait que l'apprentissage doit être le plus efficace possible lorsque soumis à des limites en resources. Je développe cette idée dans le cadre de la notion de rationalité limitée (Simon, 1957) et montre comment elle peut être intégrée dans l'architecture *option-critic* pour apprendre de meilleures options. Je montre également qu'apprendre des options du point de vue de la rationalité limitée est lié au problème de construction de bons préconditionneurs de matrices en algèbre numérique linéaire. Je démontre cette idée formellement en établissant que les méthodes à pas multiples (*multi-step*) en apprentissage par renforcement ainsi que les options ont en commun une structure mathématique dont les origines proviennent du concept de *matrix splitting* de Varga (1962).

Contribution to Original Knowledge

This thesis contributes to the understanding of the notion of temporal abstraction in reinforcement learning. More specifically, it addresses the problem of options discovery in reinforcement learning by proposing:

1. An approach for learning options in an end-to-end fashion using policy gradient methods, including:
    - a novel *augmented state space* perspective on Markov options
    - a set of mathematical results showing the exact form of the gradients required to implement this system.
    - a learning architecture where options are jointly learned along with their value functions: the option-critic architecture.
2. An interpretation of the role played by options in the bounded rationality framework, leading to:
    - a practical regularization strategy to learn *longer* options within the option-critic architecture based on the idea of *deliberation cost*
    - a family of algorithms that can *introspect* about their computational effort at a potentially different time scale than their environment
3. A unified framework for multi-step reinforcement learning and options based on the concept of *matrix-splitting* providing:
    - a new interpretation of options discovery in connection with matrix preconditioning
    - a new characterization of the polling execution model for options and its relation to the call-and-return model

Contribution of Authors

- Chapter 1 on the history of temporal abstraction and the constructivist influence in the options framework is based on Bacon and Precup (2018).
- Chapter 2 and chapter 3 are based on new material written specifically for this thesis.
- Chapter 4 originates from two workshop papers: (Bacon and Precup, 2016, 2017). The material has been completely re-written in this thesis to provide a more unified presentation.
- Chapter 5 is based on a manuscript in preparation (Bacon et al., 2018). It evolved from two workshop publications (Bacon and Precup, 2015a,b) that were then extended into conference papers: Bacon et al. (2017); Harb et al. (2018). My co-author Jean Harb wrote the code for the DQN and A2OC variants of option-critic and conducted the experiments in the Atari domain. More insights and results regarding the DQN variant can be found in Jean's master's thesis (Harb, 2016).

Acknowledgements

Thank you to my wife Gayane, my parents and my sister for their encouragement. Thank you to my advisor Doina Precup for her constant optimism (even in the face of uncertainty) and for making me feel welcomed in this community. Thank you to Jean Harb whose experiments with the option-critic architecture were key to its success. Thank you to Joelle Pineau, David Meger and Ross Otto in my thesis committee. Thank you to my colleagues, collaborators and friends for their help in various forms: Ahmed Touati, Angus Leigh, Anna Harutyunyan, Borja Balle, Chenghui Zhou, Craig Sherstan, Daniel Mankowitz, Danny Tarlow, Emma Brunskill, Emmanuel Bengio, Genevieve Fried, Gheorghe Comanici, Giovanni Pezzulo, Guillaume Rabusseau, Hado van Hasselt, Harm van Seijen, Harsh Satija, Hugo Larochelle, Huizhen Yu, Marc Bellemare, Marlos Machado, Martin Klissarov, Matthew Smith, Nadia Ady, Neil Girdhard, Peter Henderson, Pedro Ortega, Philip Bachman, Philip Thomas, Pierre Thodoroff, Roshan Shariff, Ryan Lowe, Xiao-Wen Chang. Thank you to Martha White, Michael Bowling, Patrick Pilarski, and Rich Sutton for the many inspiring and fruitful discussions during my visits at UofA.

Contents

To my son Armand

Chapter 1

Introduction

Intelligent systems have the ability to adapt and generalize quickly in the presence of change and uncertainty in their environments. Fundamentally, the success of their adaptation and learning strategies hinges on the quality of their representations. Simon (1969) in fact argued that : "[...] solving a problem simply means representing it so as to make the solution transparent."

Building good representations is a longstanding challenge of artificial intelligence. In this thesis, we examine this problem in the context of reinforcement learning, the learning paradigm in which an agent interacts with its environment by making observations, choosing actions, and receiving feedback in the form of a numerical reward. The goal of the agent is to maximize an expected cumulative measure over the rewards. Since the environment might be enormous (as in the case of the game of Go, for example), and the reward may be sparse, a good representation needs to generalize well over observations (or perceptions) as well as over multiple timescales.

Over time, notable progress has been made in the realm of perceptual generalization. For example, the celebrated TD Gammon (Tesauro, 1995) program achieved unprecedented performance against the human backgammon champion by using a combination of reinforcement learning techniques and a two-layer neural network. Recent advances in deep neural networks have led to even more impressive demonstrations of this ability in tasks such as *daily-double* wagering (Tesauro et al., 2013), Atari game playing (Mnih et al., 2015), and playing the game of Go (Silver et al., 2016). In all these cases, the system is given the responsibility of building its own representation by leveraging data.

The twin problem of building good generalizations of actions over multiple timescales, otherwise known as temporal abstraction, has also received a steady influx of attention across different sub-fields of artificial intelligence (Minsky, 1961; Fikes et al., 1972; Kuipers, 1979; Korf, 1983; Iba, 1989; Drescher, 1991; Dayan and Hinton, 1992; Kaelbling, 1993; Dean and Lin, 1995; Sutton et al., 1999a). Even in the early stages of AI, Minsky (1961) recognized how "[...] we rarely solve a tricky problem by a steady climb toward success." and that a hierarchical approach to problem-solving is more likely to subtend our intellectual abilities. This organization of knowledge gives a system the ability to choose the right level of abstraction for a problem. As a consequence, progress made at one level may appear as a stroke of "insight" (Minsky, 1961) from the level above.

Systems with insight – the capacity to gain an accurate and deep intuitive understanding of a problem – (Oxford English Dictionary) have the flexibility to reason and learn beyond the confines of the knowledge provided to them *a priori*; these systems are "habile" (Nilsson, 1995). In contrast, "performance systems" (Nilsson, 1995) are designed for specific problems. While they may achieve super-human performance, they lack general autonomy and competency.

We have been pursuing the goal of temporal abstraction for building habile systems, which means that a learner should not just represent its knowledge at given timescales, but also automatically figure out which timescales are interesting for both prediction and control. The problem of knowledge representation at multiple scales can be handled in reinforcement learning systems through the framework of *options* (Sutton et al., 1999a). Generally speaking, options encapsulate behaviors that can be initiated and terminated, akin to subroutines in a programming language. Planning with given options, as well as learning options that achieve pre-specified subgoals, is well understood. However, the problem of "option discovery" – figuring out a good set of options fully automatically – has proven very hard to handle so far. A possible reason behind this impasse is the attention put on allowing program designers to specify what is *interesting* problem structure. The resulting programs are akin to Nilsson's "performance systems": they do well on certain tasks, but are often too brittle to deploy widely or scale up.

In chapter 5, we describe the option-critic architecture, which is our attempt to make the step towards more habile systems. The idea that a learning system should be in charge of finding the options that are suitable for itself, given its environment, is a core principle of the approach put forward in this thesis. We wish to allow the agent to continually learn and adapt its representation at all abstraction levels and based solely on the data stream it observes, without requiring biasing information from a human designer. Our approach builds on the actor-critic architecture (Sutton, 1984), which provides an incremental, online, and model-free approach to learning from a continual stream of experience. Unlike other previous or existing methods, the option-critic architecture requires no subgoals, pseudo-reward, decomposition, or demonstrations, and constructs options fully autonomously while embedded in a control task that has to be solved at the same time

While the option-critic architecture offers new optimization tools, it does not provide a full answer as to what good options ought to be. We argue that the main benefit of learning options – thereby *representations* – should be to learn and reason fast in the face of uncertain and novel experience. In making efficiency the mainstay of representation learning, the need for considering the reality of time and its *fleeting nature* (Sutton, 2015a) becomes unavoidable. The arguments put forward in our temporal approach to representation learning resemble in many ways those of the *dynamical* approaches to cognition (van Gelder and Port, 1995). In this framework, cognition is seen as a dynamical system which unfolds through time and in the environment. Because of this unfolding, cognitive processes must make an efficient use of their experience given their limited capabilities because:

> "[...] they cannot take too little time or too much time. The system must spend an appropriate amount of time in the vicinity of any given state. The timing of any particular operation must respect the rate at which other cognitive, bodily, and environmental processes are taking place." (van Gelder and Port, 1995, p. 19)

A natural framework to formalize this notion of efficiency under limited resources is with bounded rationality (Simon, 1957). When departing from perfect rationality, boundedly rational systems are naturally pressured into making use of the regularities of their environment. When such systems are learning representations, only the essential elements can be captured because the resources – time, energy, computation, favorable opportunities – are scarce.

In section 5.2.1, we formalize this intuition through what we call a *deliberation cost*, which is a regularizer that entices the construction of *longer* temporally extended actions through the option-critic architecture. Furthermore, we develop a mathematical theory in chapter 4 that puts multi-step reinforcement learning methods and temporal abstraction on the same substrate: that of matrix splitting methods (Varga, 1962). This allows us to discuss about the properties and design tradeoffs of good representations using a vocabulary borrowed from matrix preconditioning theory (Golub and Van Loan, 1996; Watkins, 2004). Matrix preconditioning techniques, like representation learning methods, also aim to find a transformation of a problem into one that is easier to solve. In the following chapter, we present a brief overview of reinforcement learning and the development of the notion of temporal abstraction in AI. This material will be supplemented with a more technical presentation in chapter 2. In describing the options framework, we will take a detour along the way to appreciate its roots in *constructivism* (Piaget, 1937; Drescher, 1991). This perspective will be useful to understand the properties sought in our system.

1.1 Reinforcement Learning

Reinforcement Learning (RL) refers to learning from the experience generated by an agent interacting with its *environment*. Conceptually, the field has been inspired by the work on trial-and-error learning from psychology, but methods were formalized and analyzed using the theory of Markov Decision Processes (MDPs) (Bellman, 1954). An MDP consists of a set of states (modelling the agent's perceptions) and a set of actions. For each state-action pair, there is a well-defined transition probability distribution from which the next state will be drawn. The *reward function* specifies, for each state-action pair, an immediate numerical reward that will be received by the agent. While the transition

and reward functions are assumed to exist, the agent does not have access to them. Instead, it interacts with the environment by observing states, choosing actions, and observing the resulting rewards and next states. A sequence of states, actions, and rewards generated in this manner is called a trajectory.

The agent will typically seek a way of choosing actions, conditioned on states, that is rewarding in the long run. Such a stochastic decision procedure is called a *policy* (denoted by π). Rather than simply maximizing the total reward, which may not be bounded in general, the agent usually attempts to maximize *discounted returns*. The discount factor γ can be conceptualized as an inflation rate that deprecates rewards at every time step. In the policy evaluation problem, the goal is to compute the expected discounted sum of rewards for a *given, fixed* policy over the distribution of possible trajectories. This information is summarized in a so-called *value function*. In the control problem, the goal is to find a policy that maximizes the expected return.

A natural approach to these problems for agents that are continually acting and learning is Temporal Difference (TD) learning, introduced by Sutton (1984, 1988) in the context of policy evaluation, and later adapted for control through the Q-learning (Watkins, 1989) and Sarsa (Rummery and Niranjan, 1994) algorithms. For the policy evaluation case, the core idea is that, after learning has completed, the value of a state should be equal, in expectation, to the reward plus the discounted value of the next state. The temporal difference error quantifies how different the estimated value of a state is at the current time step, compared to one time step later (when a new sample transition and reward have been observed). The algorithm uses this error to train an approximation of the value function associated with that policy. In the case of control, this idea is supplemented with a simple strategy for changing the policy over time: actions that lead to better-than-expected outcomes should be taken more often (i.e. reinforced).

1.2 Actions with Variable Duration

After having established its modern foundations at the end of the 1980s, a number of proposals were made to extend the scope of reinforcement learning methods from actions of fixed duration to temporally extended actions

(Watkins, 1989; Singh, 1992a; Dayan and Hinton, 1992; Kaelbling, 1993; Thrun and Schwartz, 1995; Sutton, 1995), culminating at the end of the nineties (Parr and Russell, 1998; Hauskrecht et al., 1998; Dietterich, 1998; Sutton et al., 1999a) with several formulations based on Semi-Markov Decision Processes (SMDP) (Howard, 1963).

The MDP model makes the assumption that the environment transitions to a new state in a single time step (or, equivalently, in a constant amount of time), while in an SMDPs, the transition duration is a random variable. This provides a natural fit for representing temporally abstract actions, which can persist over time. More precisely, in an SMDP, the choice of action at a given state induces a joint probability distribution over the next state and the duration of the transition. Hence, a trajectory in an SMDP includes, in addition to states, actions and rewards, the duration of each transition. The name "semi-Markov" stems from the fact that the process is only assumed to be Markov from decision point to decision point, which conveniently allows for existing dynamic programming results to apply at the level of decisions (or action choices). However, the evolution of the system between two decisions may not even be Markov, and it is also allowed to unfold over continuous time. In fact, when the transition duration is exponentially distributed, this leads to a decision process called *continuous time* Markov Decision Process (CTMDP) (Puterman, 1994).

1.3 Seeing through the Black Box with Options

Despite adopting the same SMDP formalism, the *options* framework (Sutton et al., 1999a) differs from its contemporaries (Parr and Russell, 1998; Dietterich, 1998) in its emphasis on exposing and leveraging both the structure *within* and *over* temporally extended actions. The evolution of the process between two decision points is no longer a *black-box* and can be both controlled and observed. The ability to seamlessly learn and plan at different levels of abstraction stems from the assumption that there exists a base MDP that is overlaid with temporally extended actions called *options*: the combination is shown to induce an SMDP. With the expression "between MDPs and semi-MDPs" in the title of their paper, the authors of (Sutton et al., 1999a) tried to convey the idea that options provide a *lens* of variable resolution.

An option is a combination of three components: an initiation set, a policy (sometimes called *internal*), and a termination condition. The initiation set for an option is a subset of the states in which the given option could be chosen. In the most common execution model – *call-and-return* – the policy of an option acts until the probabilistic termination condition associated with it is met. More precisely, upon entering the next state, the system has to toss a biased coin, whose outcome dictates *continuation* or *termination* of the current option. Once an option has terminated, a policy over options μ chooses a new option, and the process is repeated.

1.3.1 Constructivist Influence

To get some perspective on what the options framework *is* and what it *ought* to be [1], it is useful to follow its lineage into the *Schema mechanism* of (Drescher, 1991). Inspired by Piaget's *constructivism* (Piaget, 1937), Drescher (1991) puts forward the idea that all knowledge acquired by an agent is represented in terms of its own experience with the sensation of its actions in the environment. A schema, whose semantics has much in common with options, is a symbolic structure that describes the *result* of an *action* given a *context*. The role of a schema goes beyond the simple specification of actions and can be used to express general knowledge about the environment. For example, a robot might choose to represent the fact that the charger is in front of it based on its own prediction of what would happen if it were to dock (Sutton, 2012). It might also choose to represent the presence of humans based on the predicted sensory inputs that would typically follow the playback of its "bebeep boop" sounds and dance sequence in the morning.

This idea of building representations of the world grounded in predictions about the outcome of temporally abstract actions has informed the development of the options framework. Building on an earlier line of work (Sutton, 1995; Precup and Sutton, 1997; Precup et al., 1998), Sutton et al. (1999a) showed that predictions about the expected return and future state and duration upon termination of an option could be used for planning. Like simple actions, options have associated *reward* and *transition* models, which can be used in a set of Bellman equations at the SMDP level, whose solution can be found by

[1] A personal interpretation from my conversations with Rich Sutton and Doina Precup

dynamic programming methods. Options and their models then play a representational role in the sense of Drescher (1991): they are agent-centric and encode meaning over action-conditional predictions.

1.3.2 A Multifaceted Framework

Looking beyond the purely constructivist perspective, thinking about options in isolation from their models has also been of practical interest. Rather than choosing actions after reasoning in a predictive representation, options can interact directly with the environment. This leads to what we call the *execution* perspective on options. Here, an option is more *procedural* in nature and acts as a data structure for expressing action choices [2]. Using computer program execution as an analogy, an option is akin to a function executed within a program in a *call-and-return* fashion: its instructions are read (policy of an option), moved to the CPU (environment), and upon terminating, the next function is loaded (initiated) from the call stack along with its arguments. Complex control flows can be generated in this manner and a generalization to *deeper* hierarchies follows naturally.

Fundamentally, it is the need for remembering which option is currently executing that leads to the concept of a stack. The content of the stack is also where we draw a line between *Markov options* and *semi-Markov options*. In the simplest case, the stack for Markov options is of constant size because it holds exclusively the identity of the current option: a single integer variable is sufficient for implementation. For example, we can imagine a robot navigation task for which a good Markov option might be: "if there is no obstacle" (initiation set), "move forward" (the policy of that option) "until the charger is reached" (termination condition). However, if we were to also specify that the robot should stop searching for the charger after some time, the corresponding option would be semi-Markov. In fact, the need to actively keep track of time creates a dependence on the history since initiation (unless timing information is included in the state space). An option is therefore Markov if its behavior depends only on the current state and not on any measurements of the history since its initiation. The restriction to Markov options leads to the powerful idea of *intra-option* learning (Sutton et al., 1998), which has no analogue in the

[2] Sutton rejects this view (Sutton, 2009).

semi-Markov case. The Markov option property, as well as the intra-option formulation, are central to our approach for learning options.

1.4 The Bottleneck Concept

Learning and planning with options has been well understood since Sutton et al. (1999a). If the options are pre-specified, dynamic programming or temporal difference learning methods can be used to learn about option values and models. However, the problem of discovering useful options automatically is still difficult to tackle. The challenge comes on two fronts: defining what *useful* or *good* options mean and designing algorithms for finding those options.

An important contribution to the discovery problem in the context of classical macro-actions came from Iba (1989) and his *peak-to-peak* heuristic, inspired by the concept of *chunking* (Mayzner and Gabriel, 1963) from psychology. The premise of this work is that pairs of *peaks* in the evaluation function should provide useful demarcations for where temporally extended actions should start and end. Based on the same intuition, (Konidaris and Barto, 2009; Niekum et al., 2012) later framed option discovery as a change-point detection problem from expert demonstrations.

This idea of *peaks* is also related to *bottleneck* states (McGovern and Barto, 2001; Stolle and Precup, 2002), which are states that occur more frequently on successful trajectories through the environment. Bottleneck states, like peaks, are intuitively associated with *breakthroughs* in the solution. Consider a goal-directed navigation task in an environment containing rooms and doorways. If the agent is starting in a separate room from the goal, doorways would necessarily be crossed on successful trajectories and should be useful subgoals.

Many graph-theoretic formulations of the bottleneck concept have been proposed over the years. For example, Özgür and Barto (2009) chose the notion of *betweenness centrality* (Freeman, 1977) – the relative number of shortest paths passing through a vertex – as a measure of importance. Alternatively, graph partitioning ideas have often been used to define options around the bottleneck states at the boundary of each partition (Dean and Lin, 1995; Menache et al.,

2002; Özgür et al., 2005; Botvinick et al., 2009; Chaganty et al., 2012; Bouvrie and Maggioni, 2012; Bacon, 2013; Krishnamurthy et al., 2016; Machado et al., 2017).

The bottleneck concept can be challenging to turn into practical and scalable algorithms. One important reason is the need for vast quantities of data (sometimes expert data, which is hard to obtain). Moreover, in the graph-theoretic formulation, the underlying state connectivity graph of the MDP must be approximated first, before the search for bottlenecks. While some progress has been made recently by Machado et al. (2017), the approximation step often renders the graph perspective incompatible with online implementations and over continuous spaces.

1.5 Desiderata

In our work on the discovery problem, a liberating decision has been to momentarily give the word "discovery" a break in order to refocus our attention on "learning". After all, as reinforcement learning researchers, learning is what gets us up in the morning after coffee... Yet, that seemingly innocuous change from "discovering options" to "learning options" had a significant impact on our understanding of the problem and the kind of properties that our algorithms should have.

The terminology of "learning options" had also been used in the past, but mostly in the context of subgoal (Sutton et al., 1999a) or pseudo-reward (Dietterich, 1998) methods, which allow leveraging such external information in order to learn the policies and termination conditions of options, by treating each option as an MDP on its own. Given that the environment allows for arbitrary resets, the policy of an option would be initialized within its initiation set and executed until its termination condition was met. This leads to a process that separates learning the internal information for the options from learning the policy over options, as well as from learning useful subgoals or pseudo-reward functions.

Our desire to avoid this kind of "partitioned" learning is led us to embrace a more integrative and continual perspective. We asked ourselves: wouldn't

it be possible to learn at all times, the elements of all options and the policy over them? This way of thinking also allows us to consider the question of *optimality* of a set of options. This issue is especially apparent in the subgoal and reward settings and pertains to the fact that *locally* optimal options may not lead to optimality of the overall system (Minsky, 1961; Watkins, 1989; Dietterich, 2000). Thus, we wanted to address this mismatch by learning options that were aligned with a well-defined objective for the system as a whole.

The end-to-end perspective not only provides this *alignment* with a given objective but also puts learning *in the hands* of the system. We are thereby strengthening the meaning of options as *internal* abstractions belonging exclusively to an agent, and not to the environment, as *latent* variables; they play a *subjective* role (Tanner et al., 2007; Sutton, 2012). In this sense, the study of options is intrinsically *phenomenological* and as once expressed by Stanislaw Ulam (Rota, 1986, p.2):

> [..] what you are describing is not an object, but a function, a role that is inextricably tied to some context. Take away the context, and the meaning also disappears.

Pushing the responsibility of learning good options to the agent was also a way to prevent ourselves from biasing towards the kind of options that we thought the system should have. Specifically, we wanted to explore beyond bottleneck options and to let them emerge from learning only if deemed useful by the agent. Araújo and Davids (2011) argues that ascribing behavior externally in terms of personal features rather than within the agent-environment relation causes an "organismic asymmetry" and a lost sense of *private* (Sutton, 2012) directed purpose within the agent. In order to bring the balance back, a switch must be made from viewing options as external symbolic objects to one which emphasizes their functional relationship within a system and its environment. Options discovery then becomes more a process of "attenuation" and adaptation (Araújo and Davids, 2011) to the key properties of the environment.

1.6 Objectives and Outline

The main goal of this thesis is to develop a new approach for learning options, which is online, model-free and capable of handling continuous states and actions spaces. Furthermore, we want this method to provide us with a guarantee that the options it finds will help to solve the task at hand.

In order to get there, we first review some technical aspects of Markov Decision Processes (MDP) in chapter 2, with an emphasis on the notion of *discounted weighting of states* which inevitably appears in policy gradient methods (presented in section 2.1). The proposed approach for learning option, *option-critic*, belongs to this class of methods and will later be shown in section 3.3 to also involve a discounted weighting of state-option pairs. The development of this notion will then set the stage for the derivation of the policy gradient theorem in section 2.4 which will be shown in matrix form. A novel view on the policy gradient equations will also be offered, exploiting their recursive (*Bellman*-like) structure and showing that they specify a General Value Function (GVF). This new approach will ease the derivation of option-critic later in section 5.1.

A second objective of this thesis is to provide an answer as to what options ought to be. This is because option-critic provides an answer to the *how*, but does not answer the *what*: it simply gives us an optimization framework for learning parameterized options by gradient ascent on any objective expressed as a sum of *rewards*. In section 5.2.1, we put forward the idea that good options should be those which make learning and planning easier. We propose a specific regularizer based on a notion of *deliberation cost* which implements this idea while promoting longer options. We develop this notion within the *bounded rationality* framework of Simon (1957).

A third research objective is to understand precisely how both multi-step methods and options may contribute to this goal of facilitating learning and planning. In chapter 4, we show that these two notions can be explained with the concept of *matrix splitting* (Varga, 1962) from numerical linear algebra. This connection then allows us to relate the qualities of good options and representations with those of good matrix preconditioners. The inherent tension between cost of computation and convergence rate in matrix preconditioning reflects the same tradeoff that bounded agents are also facing.

Chapter 2

Technical Aspects

A Markov Decision Process (MDP) (Bellman, 1954) consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition probability function $P : \mathcal{S} \times \mathcal{A} \to Dist(\mathcal{S})$ and a bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For convenience, we develop our ideas assuming discrete state and action sets unless otherwise noted.

Assumption 1. *The reward function is bounded:* $\forall s \in \mathcal{S}, a \in \mathcal{A} : \max_s \max_a |r(s,a)| = \kappa < \infty$ *for some scalar* κ.

The MDP formulation also allows for the reward function to be randomized, in which case r becomes a distribution over possible rewards. A randomized reward function can be converted back into an equivalent deterministic form by simply taking its expectation: a transformation which we use in section 5.2.

A Markovian randomized stationary *policy* is a probability distribution over actions conditioned on states, $\pi : \mathcal{S} \to Dist(\mathcal{A})$. In discounted problems, the value function of a policy π (also called *return function* by Denardo (1967) or *cost-to-go function* in Bertsekas (2012)) is defined as the expected sum of discounted rewards called *return*:

$$v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s\right] \tag{2.1}$$

where $\gamma \in [0, 1)$ is the *discount factor*. The idea of discounting can be useful in modeling problems where it may be necessary to express the fact that rewards accrued far in the future may be worth less than in the immediate time. When $\gamma = 0$ for example, the value of a state is simply the expected immediate

reward under the given policy: $v_\pi(s) = r(s, \pi(s))$ for a deterministic policy, or $v_\pi(s) = \sum_a \pi(a \mid s)\, r(s, a)$ in the randomized case. Discounting also plays a practical role of bounding the returns. To see this, assume for simplicity that the rewards are within $[0, R_{\max}]$. In this case, it follows that the return is at most $\sum_{t=0}^{\infty} \gamma^t R_{\max} = \frac{R_{\max}}{1-\gamma}$ since $\sum_{t=0}^{\infty} \gamma^t$ is a geometric series. In an MDP where the transition matrix has no absorbing state, setting $\gamma = 1$ would then lead to unbounded returns.

While the infinite summation in (2.1) indicates that we are working under the *infinite horizon* (Puterman, 1994) case, we can also think of the discounted setting as a *finite horizon* problem where the horizon length is distributed according to a geometric distribution (Derman, 1970; Puterman, 1994; Shwartz, 2001).

Lemma 2.1. *The expected sum of discounted rewards is equal to the following undiscounted formulation:*

$$v_\pi(s) = \mathbb{E}\left[\mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t r(S_t, A_t)\right] \middle| S_0 = s\right] \ ,$$

where the length of the horizon T is a random variable, drawn from a geometric distribution with parameter γ.

Proof. This proof is adapted from (Puterman, 1994, proposition 5.3.1).
Working under assumption (1) and that the discount factor is strictly less than 1, we can interchange the order of summation in the following expression:

$$\begin{aligned}
\mathbb{E}\left[\mathbb{E}\left[\sum_{t=0}^{T-1} r(S_t, A_t)\right] \middle| S_0 = s\right] &= \mathbb{E}\left[\sum_{n=0}^{\infty} (1-\gamma)\gamma^n \sum_{t=0}^{n} r(S_t, A_t) \middle| S_0 = s\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} r(S_t, A_t) \sum_{n=t}^{\infty} (1-\gamma)\gamma^n \middle| S_0 = s\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \sum_{n=0}^{\infty} (1-\gamma)\gamma^n \middle| S_0 = s\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \middle| S_0 = s\right] \\
&= v_\pi(s) \ ,
\end{aligned}$$

where we proceeded to the interchange by re-writing the indices so as to maintain the ordering $0 \leq t \leq n-1 < \infty$. The penultimate step follows from the fact that the rightmost series is a geometric series converging to $\frac{1}{1-\gamma}$. □

2.1 Policy Evaluation Equations

By unrolling the infinite sum of rewards in (2.1), we can obtain a recursive expression for the value of a state:

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[r(S_0, A_0) + \sum_{t=1}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[r(S_0, A_0) + \sum_{t=0}^{\infty} \gamma^{t+1} r(S_{t+1}, A_{t+1}) \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[r(S_0, A_0) + \gamma \sum_{t=0}^{\infty} \gamma^t r(S_{t+1}, A_{t+1}) \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[r(S_0, A_0) + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_1\right] \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[r(S_0, A_0) + \gamma v_\pi(S_1) \,|\, S_0 = s\right] \quad ,
\end{aligned}$$

where the last step follows from the law of total expectation. We can then expand this expectation by marginalizing over A_0 and S_1 conditioned on the some initial state $S_0 = s$ and obtain:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi\left(a \,|\, s\right) \left(r(s, a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) v_\pi(s')\right) \quad . \tag{2.2}$$

We use the notation $Q_\pi(s, a)$ for the expected sum of discounted rewards from a designated state and action, which also admits a recursive form:

$$Q_\pi(s, a) \doteq \mathbb{E}\left[\sum_{t=0}^{\infty} r(S_t, A_t) \,\middle|\, S_0 = s, A_0 = a\right] \quad = r(s, a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) v_\pi(s') \quad .$$

In Sutton and Barto (2018), and for a large body of literature in reinforcement learning from the AI community, the equations (2.2) are commonly called *Bellman equations*. However, for some authors (Puterman, 1994; Bertsekas, 2012), the expression *Bellman equations* refers exclusively to the nonlinear optimality equations which we will encounter in section 2.2. To avoid any confusion, we adopt the terminology of Denardo (1981) and denote (2.2) as the *policy evaluation equations*. Given a policy and an MDP, the *policy evaluation* problem consists in finding the associated value function v_π by solving equations (2.2). This can be achieved either by *direct* (Householder, 1964) methods or via successive approximations (Varga, 1962; Householder, 1964) in an iterative fashion.

Let us first rewrite (2.2) in a more convenient matrix form by defining:

$$\mathbf{r}_\pi(s) \doteq \sum_a \pi\left(a \,|\, s\right) r(s,a) \qquad \mathbf{P}_\pi(s,s') \doteq \sum_a \pi\left(a \,|\, s\right) P\left(s' \,\middle|\, s,a\right) \quad ,$$

where $\mathbf{r}_\pi \in \mathbb{R}^{|\mathcal{S}|}$ and $\mathbf{P}_\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$. These equations do not involve actions explicitly because the policy π has been coupled inside $\mathbf{r}_\pi$ and $\mathbf{P}_\pi$. From this perspective, $\mathbf{P}_\pi$ is now simply defining the dynamics of a Markov chain where the actions have been abstracted away. The combination of $\mathbf{r}_\pi$ and $\mathbf{P}_\pi$ defines a *Markov Reward Process* (MRP) (Puterman, 1994): a Markov chain with rewards. In vector form, the policy evaluation equations amount to the statement:

$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi \quad ,$$

where the value function for a policy is now seen as a vector $\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$. By grouping the terms involving $\mathbf{v}_\pi$, we can obtain an equivalent expression:

$$\begin{aligned} \mathbf{v}_\pi &= \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi \iff \\ \mathbf{v}_\pi - \gamma \mathbf{P}_\pi \mathbf{v}_\pi &= \mathbf{r}_\pi \\ (\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{v}_\pi &= \mathbf{r}_\pi \quad . \end{aligned}$$

The last line is the familiar form describing a linear system of equations "$\mathbf{A}\mathbf{x} = \mathbf{b}$". When solving the policy evaluation problem, $\mathbf{v}_\pi$ is seen as our unknown "$\mathbf{x}$" which we solve for by forming:

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} \mathbf{r}_\pi \quad .$$

The existence of a solution to the policy evaluation problem thus hinges on the matrix $\mathbf{I} - \gamma \mathbf{P}_\pi$ being nonsingular. One way to address this question is with the so-called spectral radius of a linear operator (Householder, 1964; Varga, 1962; Young and Rheinboldt, 1971; Puterman, 1994).

Definition 2.2 (Spectral radius)**.** Let $\mathbf{A} \in \mathbb{R}^{n \times n}$, then the spectral radius of that matrix is $\rho(\mathbf{A}) \doteq \max\{|\lambda_1|, \ldots, |\lambda_n|\}$ where λ_i is the ith eigenvalue of $\mathbf{A}$, or equivalently, $\rho(\mathbf{A}) \doteq \lim_{k \to \infty} \|\mathbf{A}^k\|^{1/k}$ using Gelfand's formula (Gelfand, 1941). Note that throughout this thesis, we will assume the infinity norm when writing $\|\mathbf{A}\| \doteq \max_i \sum_j \mathbf{A}(i,j)$.

The spectral radius of a matrix is always less than or equal to its norm. This follows from the fact that for any submultiplicative matrix norm (Watkins, 2004), $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\|$ for any $\mathbf{A} \in \mathbb{R}^{n \times n}, \mathbf{B} \in \mathbb{R}^{n \times n}$. Using Gelfrand's formulation for the spectral radius, we then have $\rho(\mathbf{A}) = \lim_{k \to \infty} \|\mathbf{A}^k\|^{1/k} \leq \|\mathbf{A}\|$. Another way (Watkins, 2004) to obtain the same inequality is through the definition of the spectral radius of a matrix in terms of its eigenvalues. If λ is an eigenvalue of $\mathbf{A}$, then it must be that $\mathbf{Ax} = \lambda \mathbf{x}$ for some eigenvector $\mathbf{x}$. Taking the norm on both sides, we then establish that:

$$\|\lambda \mathbf{x}\| = |\lambda| \, \|\mathbf{x}\| = \|\mathbf{Ax}\| \leq \|\mathbf{A}\| \, \|\mathbf{x}\| \ .$$

This inequality holds for any induced matrix norm (also called *operator norm*) (Watkins, 2004), such as the infinity norm assumed here. Hence, $|\lambda| \leq \|\mathbf{A}\|$ for any eigenvalue λ of $\mathbf{A}$ – including the one of maximum modulus. Because $\rho(\mathbf{A}) = \max\{|\lambda_i|\}_{i=1}^n$, we then need to have $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$.

Lemma 2.3. *When the discount factor satisfies* $0 \leq \gamma < 0$*, the inverse of* $\mathbf{I} - \gamma \mathbf{P}_\pi$ *of exists and:*

$$(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} = \sum_{t=0}^{\infty} (\gamma \mathbf{P}_\pi)^t \ . \tag{2.3}$$

Proof. This follows directly from (Puterman, 1994, corrolary C.4) which shows that $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ exists only when $\rho(\gamma \mathbf{P}_\pi) < 1$. Because $\mathbf{P}_\pi$ is a stochastic matrix (Cinlar, 1975), $\sum_{s'} \mathbf{P}_\pi(s, s') = 1$ for any state s so $\rho(\gamma \mathbf{P}_\pi) \leq \|\gamma \mathbf{P}_\pi\| < 1$. □

The series on the right-hand side of (2.3) is a *Neumann series* (Puterman, 1994) and can be though of as a generalization of the scalar geometric series to operators. In this expression, each row of $\mathbf{P}_\pi^t$ (the t-th power of $\mathbf{P}_\pi$) contains the distribution over the next states t steps in the future. In other words, if we choose a row s and column s', then $\mathbf{P}_\pi^t(s, s') \doteq P\left(S_t = s' \mid S_0 = s\right)$: the probability that the process is in state s' t steps into the future if it started from state s. Hence, when taking the sum to infinity in (2.3), we are effectively marginalizing over all possible paths between any two states. The entries of $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$ then have the following meaning:

$$(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}(s, s') = \sum_{t=0}^{\infty} \gamma^t P_\pi\left(S_t = s' \,\middle|\, S_0 = s\right) \quad .$$

2.1.1 Iterative Solution to the Policy Evaluation Problem

Our analysis in the previous section showed than when the discount factor is strictly less than 1, the value function of a policy is given by $\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}\mathbf{r}_\pi$. A direct approach for computing this expression may consist in first inverting the matrix explicitly and then computing the matrix-vector product with $\mathbf{r}_\pi$. This naive approach is however almost always avoided in practice because of numerical instabilities (Watkins, 2004). A better solution would be to solve for $\mathbf{v}_\pi$ through a QR factorization (Householder, 1964; Watkins, 2004) of $\mathbf{I} - \gamma \mathbf{P}_\pi$ without ever computing the inverse explicitly. While gaining in numerical stability, going through the QR factorization step does not lead to substantive computational saves since it would require $\mathcal{O}(|\mathcal{S}|^3)$ in time, as is also the case for matrix inversion.

An alternative to the direct approach is to iteratively improve an approximation to the solution based on an operator-theoretic point of view on the policy evaluation equations (2.2). In fact, we can define a linear operator $\mathcal{T}_\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ whose effect on any $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ is the following:

$$\mathcal{T}_\pi \mathbf{v} = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v} \quad .$$

The value function $\mathbf{v}_\pi$ corresponding to the policy π then satisfies $\mathcal{T}_\pi \mathbf{v}_\pi = \mathbf{v}_\pi$: ie. $\mathbf{v}_\pi$ is the fixed-point of $\mathcal{T}_\pi$. Our previous discussion on the existence of a solution to the policy evaluation equations was based on the eigenspectrum of

$\gamma \mathbf{P}_\pi$. Specifically, we have seen that a solution exists when $\rho(\gamma \mathbf{P}_\pi) < 1$. Under the operator point of view, this discussion now translates into an inquiry on the convergence of $\mathcal{T}_\pi$ to a fixed-point, which can be shown using Banach fixed-point theorem (Banach, 1922).

Definition 2.4 (Contraction mapping)**.** Let $\mathcal{U}$ be a Banach space. The operator $\mathcal{T} : \mathcal{U} \to \mathcal{U}$ is a *contraction mapping* if for any $u, v \in \mathcal{U}$ $\|\mathcal{T}v - \mathcal{T}u\| \leq \alpha \|v - u\|$ for some $0 \leq \alpha < 1$.

Definition 2.5 (Uniform Convergence)**.** A sequence $\{\mathbf{v}_k\}$ is said to converge to $\mathbf{v}$ if:

$$\lim_{k \to \infty} \|\mathbf{v}_k - \mathbf{v}\| = 0 \ .$$

Theorem 2.6 (Banach Fixed-Point Theorem)**.** *Let $\mathcal{U}$ be a Banach space. If $\mathcal{T} : \mathcal{U} \to \mathcal{U}$ is a contraction mapping on $\mathcal{U}$, then:*

1. *There exists a unique fixed point $v^\star \in \mathcal{U}$ such that $\mathcal{T}v^\star = v^\star$.*
2. *$v^\star$ can be found in the limit of the sequence defined by $v_{k+1} = \mathcal{T}v_k$, where v_0 is arbitrary.*

Proof. The proof can be found in Banach (1922), and (Puterman, 1994, theorem 6.2.3). The general outline is the following: The second statement regarding the iteration sequence is shown by establishing that $\{v_k\}$ is a Cauchy sequence. The fact that $\mathcal{T}v^\star = v^\star$ can then be established by first applying the triangle inequality:

$$\|\mathcal{T}v^\star - v^\star\| = \| (\mathcal{T}v^\star - v_k) + (v_k - v^\star) \| \leq \|\mathcal{T}v^\star - v_k\| + \|v_k - v^\star\| \ .$$

Because $\mathcal{T}$ is a contraction:

$$\|\mathcal{T}v^\star - \mathcal{T}v_{k-1}\| + \|v_k - v^\star\| \leq \|v^\star - v_{k-1}\| + \|v_k - v^\star\| \ ,$$

which means that:

$$\|\mathcal{T}v^\star - v^\star\| \leq \|v^\star - v_{k-1}\| + \|v_k - v^\star\| \ .$$

By virtue of $\{v_k\}$ being a Cauchy sequence, we know that we can make the distance between $\|v_k - v^\star\|$ arbitrarily small with large values of k: ie. v_k

converges to $v^\star$. This means that the right-hand side of the inequality can be annihilated and we have $\mathcal{T}v^\star = v^\star$. □

Casting the policy evaluation problem as an instance of fixed-point methods has the obvious advantage of providing us with a template for designing new iterative policy evaluation algorithms. This follows directly from the second statement of theorem 2.6 which suggests that the repeated application of the policy evaluation operator $\mathcal{T}_\pi$ converges to the value function v_π for any initial guess $v_\pi^{(0)}$. But in order for all of this hold, we need to establish that $\mathcal{T}_\pi$ is indeed a contraction mapping.

Lemma 2.7. *$\mathcal{T}_\pi$ is a contraction mapping.*

Proof. Applying the definition (2.4) of a contraction and because $\mathbf{P}_\pi$ is stochastic:

$$\|\mathcal{T}_\pi \mathbf{v} - \mathcal{T}_\pi \mathbf{v}'\| = \|\gamma \mathbf{P}_\pi (\mathbf{v} - \mathbf{v}')\| \leq \gamma \|\mathbf{P}_\pi\| \, \|\mathbf{v} - \mathbf{v}'\| = \gamma \|\mathbf{v} - \mathbf{v}'\|,$$

Hence $\mathcal{T}_\pi$ is a contraction if the discount factor is $0 \leq \gamma < 1$. □

Algorithm 1: Iterative Policy Evaluation

$\mathbf{v} = 0$
Input: A stationary policy π (deterministic or randomized), and discounted MDP.
Pre-compute:
$\mathbf{r}_\pi \in \mathbb{R}^{|\mathcal{S}|}$, where $\mathbf{r}(s) \doteq \sum_a \pi(a \mid s)\, r(s,a)$
$\mathbf{P}_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, where $\mathbf{P}_\pi(s,s') = \sum_a P(s' \mid s,a)\, \pi(a \mid s)$
Initialize:
$\mathbf{v} \leftarrow \mathbf{0}, \mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$
repeat
  $\mathbf{v} \leftarrow \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}$
until *a fixed number of steps is reached, or other exit condition (eg. using a bound)*

The fixed-point iteration scheme based on $\mathcal{T}_\pi$ can be implemented efficiently by directly using the vector form of the policy evaluation equations. This is shown in algorithm 1. On modern computer architectures and GPU hardware, such dense operations are especially well-suited (Dongarra et al., 1988; Dubois et al., 1996; Volkov and Demmel, 2008) and tend to yield better performance than their scalar counterparts.

However, it may still be the case that non-vectorized implementations perform favorably when the matrix $\mathbf{P}_\pi$ is sparse. This would be the case for example when the average branching factor is small. This could come from the fact that the set of available actions varies from state to state, where in certain states only a few actions are possible, or when certain actions have a near-deterministic effect over a few possible next states.

2.1.2 Temporal Difference Learning

Temporal difference learning is a Stochastic Approximation (SA) (Robbins and Monro, 1951; Benveniste et al., 1990; Kushner and Yin, 2003) algorithm for finding the parameters of a parameterized value function for a given policy. In the linear case, we represent the value function as $\hat{v}_\pi(s;\mathbf{w}) \doteq \boldsymbol{\phi}_s^\top \mathbf{w}$ where $\boldsymbol{\phi}_s$ is a vector of features provided by the system designer. The *tabular* case can be seen as a special case of this representation for the choice $\boldsymbol{\phi}_s = \mathbf{e}_s \in \mathbb{R}^{|\mathcal{S}|}$: the vector of zeros except in position s where it is 1. The nonlinear case is also addressed by TD – although with a loss of theoretical guarantees (Tsitsiklis and Roy, 1997a) – for any differentiable representation of the value function. When function approximation is introduced, the *true* value function underlying a given policy may no longer be representable in that space and the resulting solution will be approximate.

In the TD(0) algorithm, the parameters $\mathbf{w}$ of the value function are updated along the stationary distribution induced by π in the MDP according to the update rule:

$$\begin{aligned} \delta_t &\doteq R_{t+1} + \gamma \hat{v}_\pi(S_{t+1};\mathbf{w}_t) - \hat{v}_\pi(S_t;\mathbf{w}_t) \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \epsilon_t \delta_t \nabla_{\mathbf{w}} \hat{v}_\pi(S_t;\mathbf{w}_t) \ , \end{aligned}$$

where ϵ_t is a learning rate satisfying the Robbins Monro conditions $\sum_{t=0}^{\infty} \epsilon_t = \infty, \sum_{t=0}^{\infty} \epsilon_t^2 < \infty$ (Robbins and Monro, 1951). A parameter $\lambda \in [0,1]$ can also be introduced, leading to a variation called TD(λ):

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \epsilon_t \delta_t \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \nabla_{\mathbf{w}} \hat{v}_\pi(S_k;\mathbf{w}_t) \ . \tag{2.4}$$

The summation from $0 \leq k \leq t$ in this update can be computed online using an eligibility trace that is updated after each step:

$$\begin{aligned} \mathbf{z}_{\mathbf{w}}^{(t)} &\doteq \gamma\lambda\mathbf{z}_{\mathbf{w}}^{(t-1)} + \nabla_{\mathbf{w}}\hat{v}_{\pi}(S_t;\mathbf{w}_t) \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \epsilon_t\delta_t\mathbf{z}_{\mathbf{w}}^{(t)} \quad . \end{aligned} \tag{2.5}$$

The meaning of *eligible* here stems from the fact that the eligibility trace vector $\mathbf{z}_{\mathbf{w}}$ *marks* the weights that are *eligible* for reinforcement: an expression that originated (Sutton and Barto, 2018) from early work on trial-and-error learning (Klopf, 1982).

2.1.2.1 Lambda Return

We see from (2.4) that setting $\lambda = 0$ yields the TD(0) update. But what is the meaning of TD(λ) for $\lambda > 0$? Watkins (1989) showed that the effect of TD(λ) can be understood as if each TD update step involves taking an infinite convex combination of n-steps returns of the form:

$$G_t^{(\lambda)} \doteq (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}\left(\sum_{k=t}^{t+n-1}\gamma^{k-t}r(S_k,A_k) + \gamma^n v_{\pi}(S_{t+n})\right) \quad .$$

From an operator-theoretic point of view, the λ-return leads to the so-called λ-operator whose effect (in matrix form) on $\mathbf{v}_{\pi}$ is:

$$\mathcal{T}_{\pi}^{(\lambda)}\mathbf{v}_{\pi} \doteq (1-\lambda)\sum_{n=0}^{\infty}\lambda^n\left(\sum_{k=0}^{n}\left(\gamma\mathbf{P}_{\pi}\right)^k\mathbf{r}_{\pi} + \left(\gamma\mathbf{P}_{\pi}\right)^{n+1}\mathbf{v}_{\pi}\right) = \mathbf{v}_{\pi} \quad .$$

A useful connection can be made between the λ-operator in this form and the TD error.

Proposition 2.8. *The effect of the λ-operator on $\mathbf{v}_{\pi}$ can also be described by:*

$$\mathcal{T}_{\pi}^{(\lambda)}\mathbf{v}_{\pi} = \left(\mathbf{I} - \gamma\lambda\mathbf{P}_{\pi}\right)^{-1}\left(\mathbf{r}_{\pi} + \gamma\mathbf{P}_{\pi}\mathbf{v}_{\pi} - \mathbf{v}_{\pi}\right) + \mathbf{v}_{\pi} \quad .$$

Proof.

$$\begin{aligned}
\mathcal{T}_\pi^{(\lambda)}\mathbf{v}_\pi \doteq & (1-\lambda)\sum_{n=0}^{\infty}\lambda^n\left(\left(\gamma\mathbf{P}_\pi\right)^{n+1}\mathbf{v}_\pi+\sum_{k=0}^{n}\left(\gamma\mathbf{P}_\pi\right)^k\mathbf{r}_\pi\right)\\
&=(1-\lambda)\sum_{n=0}^{\infty}\left(\lambda\gamma\mathbf{P}_\pi\right)^n\left(\gamma\mathbf{P}_\pi\right)\mathbf{v}_\pi+(1-\lambda)\sum_{n=0}^{\infty}\sum_{k=0}^{n}\lambda^n\left(\gamma\mathbf{P}_\pi\right)^k\mathbf{r}_\pi\\
&=(1-\lambda)\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\left(\gamma\mathbf{P}_\pi\mathbf{v}_\pi\right)+\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\mathbf{r}_\pi\\
&=\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\left(\mathbf{r}_\pi+(1-\lambda)\gamma\mathbf{P}_\pi\mathbf{v}_\pi\right)\\
&=\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\left(\mathbf{r}_\pi+\gamma\mathbf{P}_\pi\mathbf{v}_\pi-\mathbf{v}_\pi+\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)\mathbf{v}_\pi\right)\\
&=\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\left(\mathbf{r}_\pi+\gamma\mathbf{P}_\pi\mathbf{v}_\pi-\mathbf{v}_\pi\right)+\mathbf{v}_\pi
\end{aligned}$$

This result can be found without proof in equation 3 of Geist and Scherrer (2014). It can also be verified more easily via the matrix splitting approach of chapter 4. In fact, (Chen, 2005, section 3.2) refers to this form (in the broader context of matrix preconditioning) as a *generalized Richardson iteration* and $(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi)$ as a *residual correction operator*. □

As usual, the inverse $\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}$ in proposition 2.8 exists when $\gamma\lambda<1$ and can be written in terms of its Neumann series expansion:

$$\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\left(\mathbf{r}_\pi+\gamma\mathbf{P}_\pi\mathbf{v}_\pi-\mathbf{v}_\pi\right)=\sum_{t=0}^{\infty}\left(\gamma\lambda\mathbf{P}_\pi\right)^t\left(\mathbf{r}_\pi+\gamma\mathbf{P}_\pi\mathbf{v}_\pi-\mathbf{v}_\pi\right)$$

The vector $\mathbf{r}_\pi+\gamma\mathbf{P}_\pi\mathbf{v}_\pi-\mathbf{v}_\pi$ appearing in the above equation corresponds to the expected immediate TD error. It follows that an element "s" of this vector is:

$$\mathbf{e}_s^\top\left(\mathbf{I}-\gamma\lambda\mathbf{P}_\pi\right)^{-1}\left(\mathbf{r}_\pi+\gamma\mathbf{P}_\pi\mathbf{v}_\pi-\mathbf{v}_\pi\right)=\mathbb{E}\left[\sum_{t=0}^{\infty}\left(\gamma\lambda\right)^t\delta_t\,\middle|\,S_0=s\right]\quad.$$

The policy evaluation equations under the λ-operator in proposition 2.8 can then be written in component form as:

$$\mathbf{v}_\pi(s_t)=\mathbf{v}_\pi(s_t)+\mathbb{E}\left[\sum_{k=t}^{\infty}\left(\gamma\lambda\right)^{k-t}\delta_t\,\middle|\,S_t=s_t\right]\quad.$$

Subtracting the value term $v_\pi(s_t)$ from this expression would leaves us with an expected sum of TD error, which means that:

$$\mathbb{E}\left[G_t^\lambda - v_\pi(s_t) \,\middle|\, S_t = s_t\right] = \mathbb{E}\left[\sum_{k=t}^{\infty} (\gamma\lambda)^{k-t}\,\delta_t \,\middle|\, S_t = s_t\right] \quad .$$

An interpretation for this expression is that the difference of $G_t^{(\lambda)}$ (which can only be measured in the future) and $\mathbf{v}_\pi(s_t)$ is equal to sum of differences in predictions: this is the main insight developed by Sutton (1988). As shown in the previous section, the online implementation of TD(λ) (2.4) can be obtained by maintaining the sum of TD errors on the right-hand side using an eligibility trace.

2.1.2.2 Projected Equations

Tsitsiklis and Roy (1997b) established the convergence of TD(λ) in the online and linear setting by studying the ordinary differential equation (ODE) associated with the sequences of iterates shown in (2.4). In this kind of analysis (Benveniste et al., 1990; Kushner and Yin, 2003), we study the dynamics of a *deterministic* counterpart to the original algorithm by *averaging* the random iterates under the stationary distribution of the current policy.

In the linear case, the solution found by TD(λ) is shown to satisfy the so-called *projected form of the Bellman equations* (Bertsekas, 2012), also more succinctly called *projected Bellman equations*. Let $\mathbf{\Phi} \in \mathbb{R}^{|\mathcal{S}|\times k}$ be a matrix of k-dimensional features for the parameter vector $\mathbf{w} \in \mathbb{R}^k$. The linearly parameterized value function $\hat{\mathbf{v}}_\pi = \mathbf{\Phi}\mathbf{w}$ satisfies:

$$\mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}\hat{\mathbf{v}}_\pi = \hat{\mathbf{v}}_\pi \quad ,$$

where $\mathbf{\Pi} \doteq \mathbf{\Phi}\left(\mathbf{\Phi}^\top \mathbf{\Xi}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^\top\mathbf{\Xi}$, $\mathbf{\Pi} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ is the *hat* matrix (Christensen, 2011) and $\mathbf{\Xi}$ is a diagonal matrix containing the stationary distribution of π in the MDP. The matrix $\mathbf{\Pi}$ therefore corresponds to a projection with respect to a weighted norm under the stationary distribution $\mathbf{d}_\pi$. This norm is defined as: $\|\mathbf{v}\|^2_{\mathbf{d}_\pi} \doteq \mathbf{v}^\top\mathbf{\Xi}\mathbf{v}$, for any $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$. This means that the application of $\mathbf{\Pi}$ to any $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ yields its closest vector in the subspace spanned by the columns

of the feature matrix:

$$\begin{aligned}\mathbf{\Pi v} &\doteq \mathbf{\Phi w}^\star \\ \mathbf{w}^\star &\doteq \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^k} \|\mathbf{v} - \mathbf{\Phi w}\|^2_{\mathbf{d}_\pi} \ .\end{aligned}$$

The projected Bellman equations therefore arise from the composition of two operators which we call the *projected Bellman operator*: the λ-operator followed by a projection onto $\mathcal{V} \doteq \{\mathbf{\Phi w} : \mathbf{w} \in \mathbb{R}^k\}$. Except for the case where $\mathbf{\Phi}$ is the matrix of one-hot features (the identity matrix), an application of the λ-operator may not result in a value function representable by the chosen features: hence the projection *down* into the representable space. Using proposition 2.8, we find that the vector of parameters $\mathbf{w}$ that best approximates the value function of policy π can be found as the solution to the following equations Tsitsiklis and Roy (1997a); Boyan (2002); Bertsekas (2012):

$$\left(\mathbf{\Phi}^\top \mathbf{\Xi} \left(\mathbf{I} - \gamma\lambda \mathbf{P}_\pi\right)^{-1} \left(\mathbf{I} - \gamma \mathbf{P}_\pi\right)^{-1} \mathbf{\Phi}\right) \mathbf{w} = \mathbf{\Phi}^\top \mathbf{\Xi} \left(\mathbf{I} - \gamma\lambda \mathbf{P}_\pi\right)^{-1} \mathbf{r}_\pi \ .$$

The existence of $\mathbf{w}$ in the projected Bellman equations can be established by showing that the projected Bellman operator is a contraction with respect to the norm $\|\cdot\|_{\mathbf{d}_\pi}$. Our choice of weighted norm plays a crucial role in establishing this property, and more specifically, the fact that the weighting is under the stationary distribution of the current policy: the *on-policy* distribution. This condition allows us to establish the following important fact.

Lemma 2.9. *For any* $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$,

$$\|\mathbf{P}_\pi \mathbf{v}\|_{\mathbf{d}_\pi} \leq \|\mathbf{v}\|_{\mathbf{d}_\pi} \ .$$

Proof. From (Bertsekas, 2012, lemma 6.3.1):

$$\begin{aligned}\|\mathbf{P}_\pi \mathbf{v}\|^2_{\mathbf{d}_\pi} &= \sum_s \mathbf{d}_\pi(s) \left(\sum_{s'} \mathbf{P}_\pi(s,s')\mathbf{v}(s')\right)^2 \\ &\leq \sum_s \mathbf{d}_\pi(s) \sum_{s'} \mathbf{P}_\pi(s,s')\mathbf{v}(s')^2 \\ &= \sum_s \mathbf{d}_\pi(s)\mathbf{v}(s)^2 = \|\mathbf{v}\|^2_{\mathbf{d}_\pi} \ ,\end{aligned}$$

because $\mathbf{d}_\pi$ is a stationary distribution and $\mathbf{d}_\pi^\top \mathbf{P}_\pi = \mathbf{d}_\pi^\top$. □

Equipped with lemma 2.9, we can now show that the λ-operator is a contraction mapping under this choice of weighted norm.

Lemma 2.10. *$\mathcal{T}_\pi^{(\lambda)}$ is a contraction mapping with respect to $\|\cdot\|_{\mathbf{d}_\pi}$.*

Proof. In matrix form, the λ-return can be written as:

$$\mathcal{T}_\pi^{(\lambda)}\mathbf{v} = (\mathbf{I} - \gamma\lambda\mathbf{P}_\pi)^{-1}\,\mathbf{r}_\pi + \gamma\,(\mathbf{I} - \gamma\lambda\mathbf{P}_\pi)^{-1}\,(1-\lambda)\mathbf{P}_\pi\mathbf{v}_\pi$$

It then follows that for any $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^{|\mathcal{S}|}$:

$$\begin{aligned}
\|\mathcal{T}_\pi^{(\lambda)}\mathbf{v} - \mathcal{T}_\pi^{(\lambda)}\mathbf{v}'\,\|_{\mathbf{d}_\pi} &= \|\gamma(1-\lambda)\,(\mathbf{I} - \gamma\lambda\mathbf{P}_\pi)^{-1}\,\mathbf{P}_\pi\left(\mathbf{v} - \mathbf{v}'\right)\|_{\mathbf{d}_\pi} \\
&\leq \gamma(1-\lambda)\|\sum_{t=0}^{\infty}(\gamma\lambda)^t\,\mathbf{P}_\pi^{t+1}\left(\mathbf{v} - \mathbf{v}'\right)\|_{\mathbf{d}_\pi} \\
&\leq \gamma(1-\lambda)\sum_{t=0}^{\infty}(\gamma\lambda)^t\,\|\mathbf{P}_\pi^{t+1}\left(\mathbf{v} - \mathbf{v}'\right)\|_{\mathbf{d}_\pi} \\
&\leq \gamma(1-\lambda)\sum_{t=0}^{\infty}(\gamma\lambda)^t\,\|\mathbf{v} - \mathbf{v}'\|_{\mathbf{d}_\pi} \\
&= \frac{\gamma(1-\lambda)}{1-\gamma\lambda}\|\mathbf{v} - \mathbf{v}'\|_{\mathbf{d}_\pi}\ .
\end{aligned}$$

This result is adapted from (Bertsekas, 2012, proposition 6.3.2). □

Having established that the λ-operator is a contraction, we need to ask whether its composition with the projection operator remains a contraction. As it turns out, $\mathbf{\Pi}$ can be shown to corresponds to *nonexpansive* operator rather than a contraction. An operator is nonexpansive when its associated Lipschitz constant is 1 instead of being strictly smaller than 1, as in the contractive case. Nevertheless, the composition of the λ-operator with the projection operator maintains the contractive property.

Lemma 2.11. *For any $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^{|\mathcal{S}|}$,*

$$\|\mathbf{\Pi}\mathbf{v} - \mathbf{\Pi}\mathbf{v}'\|_{\mathbf{d}_\pi} \leq \|\mathbf{v} - \mathbf{v}'\|_{\mathbf{d}_\pi}\ .$$

Proof. Adapted from (Bertsekas, 2012, proposition 6.3.1). From a geometrical perspective, the projection $\mathbf{\Pi}\mathbf{v}$ gives us the closest vector in the subspace $\mathcal{V}$

spanned by the columns of $\mathbf{\Phi}$. Hence, the vector $\mathbf{v} - \mathbf{\Pi v} = (\mathbf{I} - \mathbf{\Pi})\mathbf{v}$ is orthogonal to the subspace $\mathcal{V}$ and by the Pythagorean theorem:

$$\|\mathbf{v}\|_{\mathbf{d}_\pi}^2 = \|\mathbf{\Pi v}\|_{\mathbf{d}_\pi}^2 + \|\left(\mathbf{I} - \mathbf{\Pi}\right)\mathbf{v}\|_{\mathbf{d}_\pi}^2 \ .$$

Using this fact, we can establish that for $\mathbf{u} \doteq \mathbf{v} - \mathbf{v}'$:

$$\|\mathbf{\Pi u}\|_{\mathbf{d}_\pi}^2 \leq \|\mathbf{\Pi u}\|_{\mathbf{d}_\pi}^2 + \|(\mathbf{I} - \mathbf{\Pi})\mathbf{u}\|_{\mathbf{d}_\pi}^2 = \|\mathbf{u}\|_{\mathbf{d}_\pi}^2 \ ,$$

where the inequality simply follows from seeing $\mathbf{\Pi u}$ as one *side of the triangle*. □

Because $\mathbf{\Pi}$ is a nonexpansion, and $\mathcal{T}_\pi^{(\lambda)}$ is a contraction, it follows that their the composition $\mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}$ (the projected Bellman operator) is also a contraction:

$$\|\mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}\mathbf{v} - \mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}\mathbf{v}'\|_{\mathbf{d}_\pi} \leq \|\mathcal{T}_\pi^{(\lambda)}\mathbf{v} - \mathcal{T}_\pi^{(\lambda)}\mathbf{v}'\|_{\mathbf{d}_\pi} \leq \frac{\gamma(1-\lambda)}{1-\gamma\lambda}\|\mathbf{v} - \mathbf{v}'\|_{\mathbf{d}_\pi} \ .$$

Having established that the projected Bellman operator is a contraction, we can derive the following bound that characterizes the error to the true value function in proportion to the irreducible error inherent to the choice of features.

Proposition 2.12. *Let* $\mathbf{v}_\pi$ *be the true value function and* $\hat{\mathbf{v}}_\pi = \mathbf{\Phi w}$ *the unique solution to the projected Bellman equation:*

$$\|\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi} \leq \frac{1}{\sqrt{(1-\kappa^2)}}\|\mathbf{v}_\pi - \mathbf{\Pi v}_\pi\|_{\mathbf{d}_\pi} \ ,$$

where $\kappa \doteq \left(\frac{\gamma(1-\lambda)}{1-\gamma\lambda}\right)^2$.

Proof. This result is adapted from (Bertsekas, 2012, proposition 6.3.2). Using the Pythagorean theorem once again, and noting that $\mathbf{\Pi}\hat{\mathbf{v}}_\pi = \hat{\mathbf{v}}_\pi$ (because $\hat{\mathbf{v}}_\pi$ already lies in $\mathcal{V}$), we have:

$$\begin{aligned}\|\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi}^2 &= \|\mathbf{\Pi}\left(\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\right)\|_{\mathbf{d}_\pi}^2 + \|\left(\mathbf{I} - \mathbf{\Pi}\right)\left(\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\right)\|_{\mathbf{d}_\pi}^2 \\ &= \|\mathbf{\Pi v}_\pi - \hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi}^2 + \|\mathbf{v}_\pi - \mathbf{\Pi v}_\pi\|_{\mathbf{d}_\pi}^2 \ .\end{aligned}$$

Furthermore, because $\mathbf{v}_\pi$ is the fixed point of $\mathcal{T}_\pi^{(\lambda)}$ and is $\hat{\mathbf{v}}_\pi$ the one associated with $\mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}$, we can write the right-hand side of the last equation as follows:

$$\begin{aligned}\|\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi}^2 &= \|\mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}\mathbf{v}_\pi - \mathbf{\Pi}\mathcal{T}_\pi^{(\lambda)}\hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi}^2 + \|\mathbf{v}_\pi - \mathbf{\Pi}\mathbf{v}_\pi\|_{\mathbf{d}_\pi}^2 \\ &\leq \left(\frac{\gamma(1-\lambda)}{1-\gamma\lambda}\right)^2 \|\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi}^2 + \|\mathbf{v}_\pi - \mathbf{\Pi}\mathbf{v}_\pi\|_{\mathbf{d}_\pi}^2 \ ,\end{aligned}$$

and where we use the fact that the projected Bellman operator is a contraction on the last line. Re-arranging the terms, and letting $\kappa \doteq \left(\frac{\gamma(1-\lambda)}{1-\gamma\lambda}\right)^2$ we finally obtain:

$$(1-\kappa^2)\|\mathbf{v}_\pi - \hat{\mathbf{v}}_\pi\|_{\mathbf{d}_\pi}^2 \leq \|\mathbf{v}_\pi - \mathbf{\Pi}\mathbf{v}_\pi\|_{\mathbf{d}_\pi}^2 \ .$$

□

Proposition 2.12 provides some insights on the nature of the bias-variance tradeoff involved in the choice of the λ parameter. In fact, we see that as $\lambda \to 1$ the contraction factor κ becomes smaller. Note also that the right-hand side of the inequality represents the error inherent to the choice of features. In the tabular case, we would have $\mathbf{\Phi} = \mathbf{I}$ and the error $\|\mathbf{v}_\pi - \mathbf{\Pi}\mathbf{v}_\pi\|_{\mathbf{d}_\pi}$ is zero. In the more general case, $\mathbf{\Pi}\mathbf{v}_\pi$ is the *best approximation* (Bertsekas, 2012) under $\mathbf{\Phi}$ and we are left with only λ to control the bias $\hat{\mathbf{v}}_\pi - \mathbf{\Pi}v_\pi$. While larger values of λ reduce this error, it also leads to more variance in the resulting value estimates: Bertsekas (2012) refers to this as the *simulation noise*. Surprisingly, very few results are currently available regarding the nature of this tradeoff as well as methods for selecting λ optimally: Kearns and Singh (2000); Konda (2002); White and White (2016); Mann et al. (2016).

2.2 Bellman Optimality Equations

Definition 2.13 (Optimal Policy)**.** A policy $\pi^\star$ is optimal when its value function is such that for any other $\pi \neq \pi^\star$ and for any state s, $\mathbf{v}_{\pi^\star}(s) \geq \mathbf{v}_\pi(s)$.

The optimal value function $\mathbf{v}^\star$ (the value function associated with an optimal policy) satisfies a nonlinear system of equations that we call the *Bellman optimality equations*:

$$\mathbf{v}^\star = \max_{\pi \in \Pi^{\text{MD}}} \left(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}^\star \right) \quad . \tag{2.6}$$

where Π^{MD} denotes the set of deterministic Markov policies. We can also write these equations more simply in component form:

$$\begin{aligned} \mathbf{v}^\star(s) &= \max_{\pi \in \Pi^{\text{MD}}} \left(r(s, \pi(s)) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, \pi(s)\right) \mathbf{v}^\star(s) \right) \\ &= \max_a \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \mathbf{v}^\star(s') \right) \quad . \end{aligned}$$

In a finite discounted MDP, there exists at least one optimal stationary deterministic policy that satisfies the Bellman optimality equations (Puterman, 1994). The restriction to deterministic policies in (2.6) does not affect the question of optimality and simply eases the development of *control algorithms*: algorithms for finding optimal policies.

Proposition 2.14. *The maximum on the right-hand size of the Bellman optimality equations is also attained in the class of stationary randomized Markov policies. Searching in the space of deterministic policies does not involve any loss of optimality.*

$$\max_{\pi \in \Pi^{MD}} \left(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v} \right) = \max_{\pi \in \Pi^{MR}} \left(r_\pi + \gamma \mathbf{P}_\pi \mathbf{v} \right) \quad ,$$

where Π^{MD} and Π^{MR} denote the class of stationary deterministic and randomized policies respectively.

Proof: *Adapted from (Puterman, 1994, proposition 6.2.1).*

Pick any randomized policy $\pi \in \Pi^{MR}$. Because $\pi(\cdot \,|\, s)$ is a conditional distribution over actions:

$$\begin{aligned} &\max_a \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \mathbf{v}(s') \right) \\ &= \sum_a \pi\left(a \,|\, s\right) \max_a \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \mathbf{v}(s') \right) \\ &\geq \sum_a \pi\left(a \,|\, s\right) \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \mathbf{v}(s') \right) \quad . \end{aligned}$$

It means that:

$$\max_{\pi \in \Pi^{MD}} \left(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v} \right) \geq \max_{\pi \in \Pi^{MR}} \left(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v} \right)$$

The inequality in the other direction follows from the fact that the class of randomized policies is strictly larger than the class of deterministic ones. □

The stationary deterministic policy $\pi^\star$ that selects actions according to:

$$\pi^\star(s) \doteq \operatorname*{argmax}_a \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \mathbf{v}^\star(s') \right) \ ,$$

is called the *greedy* policy and can be obtained after having found $\mathbf{v}^\star$ as the *fixed point* of the *Bellman optimality operator* $\mathcal{L}$:

$$\mathcal{L}\mathbf{v} \doteq \max_{\pi \in \Pi^{\mathrm{MD}}} \left(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v} \right) \ .$$

The optimal value function satisfies $\mathcal{L}\mathbf{v}^\star = \mathbf{v}^\star$. After having established that $\mathcal{L}$ is a contraction mapping, we will be able to leverage theorem 2.6 to find $\mathbf{v}^\star$ as the fixed point in $\lim_{t \to \infty} \mathcal{L}^t \mathbf{v} = \mathbf{v}^\star$.

The resulting fixed-point iteration algorithm based on $\mathcal{L}$ is called *value iteration* (Bellman, 1957; Howard, 1960) or the *method of successive approximation* (Blackwell, 1965; Denardo, 1967) (mostly in older literature).

Lemma 2.15 (The Bellman operator is a contraction mapping)**.** *For any two value functions* $\mathbf{v}$ *and* $\mathbf{v}'$*:*

$$\| \mathcal{L}\mathbf{v} - \mathcal{L}\mathbf{v}' \| \leq \gamma \| \mathbf{v} - \mathbf{v}' \| \ .$$

where $\mathcal{L}$ *is the Bellman optimality operator* (2.6) *and* γ *is a discount factor strictly smaller than* 1*.*

Proof. We note that for any $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ associated with a value function $v : \mathcal{S} \to \mathbb{R}$ and for any $Q' : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ corresponding to a $v' : \mathcal{S} \to \mathbb{R}$:

$$\begin{aligned} |\max_a Q(s,a) - \max_a Q'(s,a)| &\leq \max_a |Q(s,a) - Q'(s,a)| \\ &= \gamma \max_a \sum_{s'} P\left(s' \,\middle|\, s,a\right) |\mathbf{v}(s') - \mathbf{v}'(s')| \\ &\leq \gamma \max_s |\mathbf{v}(s) - \mathbf{v}'(s)| \end{aligned}$$

By the definition of a contraction (2.4), it follows under the infinity norm that:

$$\|\mathcal{L}\mathbf{v} - \mathcal{L}\mathbf{v}'\| \doteq \max_s |\max_a Q(s,a) - \max_a Q'(s,a)| \leq \gamma \|\mathbf{v} - \mathbf{v}'\| \ .$$

A more general proof can also be found in (Puterman, 1994, theorem 6.2.3). □

The value iteration algorithm follows the same template as the iterative policy evaluation procedure previously shown in algorithm 1. However, this time we are only given an MDP as input since as our goal is to output an optimal policy.

Algorithm 2: Value Iteration

Input: A discounted MDP
Output: The optimal state-action value function $Q^\star$ for this MDP.
Initialize:
$Q^\star \leftarrow 0, Q^\star \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$
repeat
  forall $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
    $Q^\star(s,a) \leftarrow r(s,a) + \gamma \sum_{s'} P\left(s' \,|\, s,a\right) \max_a Q^\star(s',a)$
  end
until *a fixed number of steps is reached, or other exit condition (Puterman, 1994)*

2.2.1 Policy Iteration

The value iteration algorithm follows directly from the operator-theoretic point of view on the Bellman optimality equations (2.6). Every application of the Bellman optimality operator thus involves a one-step look-ahead followed by a maximization operation across actions. Rather than propagating the value only for the next states, the *policy iteration* algorithm (Howard, 1960) solves the

policy evaluation equations for the current candidate optimal policy before proceeding to a maximization step – also called *greedification* step (Sutton and Barto, 2018). In policy iteration, the role of the value function is to support the search of an optimal policy. This is why policy iteration was originally referred to as *approximation in policy space* (Howard, 1960; Bellman, 1954, 1957): to highlight the distinction with value iteration in which an approximation of the optimal value function is successively refined.

Algorithm 3: Policy Iteration

Input: A discounted MDP
Output: An optimal deterministic stationary policy
Initialize: π_0 arbitrarily
$k \leftarrow 0$
repeat
  Policy Evaluation:
  $\mathbf{v}_{\pi_k} \leftarrow (\mathbf{I} - \gamma \mathbf{P}_{\pi_k})^{-1} \mathbf{r}_{\pi_k}$
  Policy Improvement:
  $\pi_{k+1} \leftarrow \operatorname{argmax}_{\pi \in \Pi^{\text{MD}}} (\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_{\pi_k})$
  $k \leftarrow k + 1$
until $\pi_{k+1} = \pi_k$

In this algorithm, the policy evaluation step need not be computed by a direct method, as implied by the matrix inverse, but can also be obtained with the iterative policy evaluation procedure 1. In practice, the policy improvement step is always computed component-wise and not directly over the space of deterministic policies Π^{MD} (Puterman, 1994). The improved policy π_{k+1} is obtained as follows:

$$\pi_{k+1}(s) = \operatorname*{argmax}_a Q_{\pi_k}(s,a) = \operatorname*{argmax}_a \left(r(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) \mathbf{v}_{\pi_k}(s') \right) \quad .$$

Remarkably, improving the policy in a single component is sufficient to guarantee convergence of policy iteration. This property follows from a more general result that Sutton and Barto (2018) call the *policy improvement theorem*. The idea is intuitively simple: let's assume that a policy currently achieves a certain value $\mathbf{v}_\pi(s)$ in a given state. In order to obtain more expected return, we contemplate the effect of choosing a different action in state s (vaguely speaking a *local perturbation*) and following π onwards. Based on the outcome or this process, we then update our current policy to select the action that provided the

highest return. More precisely, the improved policy π_{k+1} should now satisfy:

$$\mathbf{r}_{\pi_{k+1}} + \gamma \mathbf{P}_{\pi_{k+1}} \mathbf{v}_{\pi_k} \geq \mathbf{v}_{\pi_k} = \mathbf{r}_{\pi_k} + \mathbf{P}_{\pi_k} \mathbf{v}_{\pi_k} \ ,$$

with a strict inequality for at least one state. Puterman (1994) uses the expression $\mathbf{v}$-*improving* to refer to a policy that satisfies the above inequality, for a given $\mathbf{v}$: π_{k+1} is $\mathbf{v}_{\pi_k}$-improving for example.

Proposition 2.16. *Let π_k and π_{k+1} be two successive policies generated during the course of policy iteration:*

$$\mathbf{v}_{\pi_{k+1}} \geq \mathbf{v}_{\pi_k} \ .$$

Proof. When selecting a new policy π_{k+1} among all $\mathbf{v}_{\pi_k}$-improving policies, it means that:

$$\mathbf{r}_{\pi_{k+1}} + \gamma \mathbf{P}_{\pi_{k+1}} \mathbf{v}_{\pi_k} \geq \mathbf{v}_{\pi_k} \iff \mathbf{r}_{\pi_{k+1}} \geq \left(\mathbf{I} - \gamma \mathbf{P}_{\pi_{k+1}}\right) \mathbf{v}_{\pi_k} \ .$$

Because $\left(\mathbf{I} - \gamma \mathbf{P}_{\pi_{k+1}}\right) \mathbf{v}_{\pi_{k+1}} = \mathbf{r}_{\pi_{k+1}}$, by multiplying on both sides we obtain:

$$\left(\mathbf{I} - \gamma \mathbf{P}_{\pi_{k+1}}\right)^{-1} \mathbf{r}_{\pi_{k+1}} = \mathbf{v}_{\pi_{k+1}} \geq \mathbf{v}_{\pi_k} \ .$$

This fact can also be found in (Puterman, 1994, proposition 6.4.1). □

Theorem 2.17 (Convergence of Policy Iteration)**.** *In finite MDPs, policy iteration converges to an optimal policy in a finite number of steps.*

Proof. Proposition 2.16 shows that the value functions associated to a pair of successive iterates of policy iteration are nondecreasing. This gives us a principle by which we can enumerate deterministic stationary policies. In a finite MDP, we know that there can only be $|\mathcal{A}|^{|\mathcal{S}|}$ of them. It follows that eventually, we would either have enumerated them all or terminated earlier under the condition $\pi_{k+1} = \pi_k$. This condition will necessarily be met because we are breaking any possible ties in a consistent manner when taking the argmax. Policy iteration must then terminate in a finite number of steps under the termination condition $\pi_{k+1} = \pi_k$, at which point $\mathbf{v}_{\pi_{k+1}} = \mathbf{v}_{\pi_k}$ and:

$$\mathbf{v}_{\pi_{k+1}} = \mathbf{r}_{\pi_{k+1}} + \gamma \mathbf{P}_{\pi_{k+1}} \mathbf{v}_{\pi_{k+1}} = \mathbf{r}_{\pi_{k+1}} + \gamma \mathbf{P}_{\pi_{k+1}} \mathbf{v}_{\pi_k} = \max_{\pi_{k+1} \in \Pi^{\text{MD}}} \mathbf{r}_{\pi_{k+1}} + \gamma \mathbf{P}_{\pi_{k+1}} \mathbf{v}_{\pi_k} \ .$$

Therefore $\mathbf{v}_{\pi_{k+1}}$ satisfies the Bellman optimality equations and π_{k+1} is an optimal policy. See (Puterman, 1994, theorem 6.4.2) for further reference. □

2.3 Discounted Weighting of States

Before delving into policy gradient methods in section 2.4, it is useful to spend some time understanding the nature of the *discounting weighting of states*, a quantity which frequently appears in their statement.

Definition 2.18. Let $\boldsymbol{\alpha} \in \mathbb{R}^{|\mathcal{S}|}, \boldsymbol{\alpha}^\top \mathbf{1} = 1$ be a distribution over initial states, the *discounted weighting of states* $\mathbf{d}_{\boldsymbol{\alpha},\gamma,\pi}$ is:

$$\mathbf{d}_{\alpha,\gamma,\pi}^\top \doteq \boldsymbol{\alpha}^\top \left(\mathbf{I} - \gamma \mathbf{P}_\pi\right)^{-1} \quad .$$

The notation chosen here is meant to emphasize the dependence on both the initial distributions of states $\boldsymbol{\alpha}$ and discount factor γ. In the average reward setting, we will use a similar notation but this time to denote the stationary distribution of a Markov chain. However, reaching the stationary distribution also implies that a process is independent of its initial states, hence we will simply write $\mathbf{d}_\pi$ without $\boldsymbol{\alpha}$ as a subscript.

While it is tempting to think of $\mathbf{d}_{\boldsymbol{\alpha},\gamma,\pi}$ as a distribution, a simple calculation for the case $\boldsymbol{\alpha} = \mathbf{e}_s$ shows that the rows of $\left(\mathbf{I} - \gamma \mathbf{P}_\pi\right)^{-1}$ do not sum up to 1:

$$\sum_{s'} \sum_{t=0}^{\infty} \gamma^t \mathbf{P}_\pi \left(S_t = s' \,\middle|\, s\right) = \sum_{t=0}^{\infty} \gamma^t \sum_{s'} P_\pi \left(S_t = s' \,\middle|\, s\right) = \frac{1}{1-\gamma} \quad .$$

Hence, $\mathbf{d}_{\alpha,\gamma,\pi}$ neither is a distribution, or for that matter a stationary distribution, but simply is a *discounted weighting of states* – an expression used by Sutton et al. (1999b). In order to work with a distribution, some authors (Altman, 1999; Kakade, 2003; Thomas, 2014) prefer to use a normalized variant of $\mathbf{d}_{\boldsymbol{\alpha},\gamma,\pi}$ defined as follows:

$$\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\pi} \doteq (1-\gamma)\mathbf{d}_{\alpha,\gamma,\pi}$$

Kakade (2003) calls this quantity the *discounted future state distribution* while Thomas (2014) refers to it as the *discounted state distribution*. In this case, taking

the inner product of $\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\pi}$ with the reward vector gives us:

$$\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\pi}^{\top}\mathbf{r}_{\pi} = \boldsymbol{\alpha}^{\top}\sum_{t=0}^{\infty}\gamma^{t}\mathbf{P}_{\pi}^{t}\left(1-\gamma\right)\mathbf{r}_{\pi} \ ,$$

which would be obtained in an MDP where all the rewards are scaled by $1-\gamma$. If the reward function is known to be within a certain interval $r : \mathcal{S}\times\mathcal{A} \to [0, R_{\max}]$ then using normalized rewards implies that $\mathbf{v}_{\pi} : \mathcal{S} \to [0, R_{\max}]$ and not $\frac{R_{\max}}{1-\gamma}$ as in the usual case. Furthermore, if the rewards are within $[0,1]$, it also means that the normalized value function is at most 1.

The expected discounted sum of unnormalized discounted rewards (our usual discounted return) is written in terms of $\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\pi}$ as follows:

$$\frac{1}{1-\gamma}\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\pi}^{\top}\mathbf{r}_{\pi} = \mathbb{E}_{\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\pi}}\left[r(S_t, A_t)\right] = \boldsymbol{\alpha}^{\top}\mathbf{v}_{\pi} \ ,$$

and setting $\boldsymbol{\alpha} = \mathbf{e}_s$ in the above gives us $\mathbf{v}_{\pi}(s)$. Instead of working with a discounted weighting over states only, it is also possible to use a representation over a discounted weighting of state-action pairs, as often found in the dual of linear programming solutions for MDPs (Puterman, 1994; Derman, 1970) or in constrained formulations (Altman, 1999) (who uses *occupation measure* to refer to the discounted weighting/distribution). In chapter 5, we use such a similar occupation measure but over state-option pairs.

2.3.1 Recursive Expression

As we did for the policy evaluation equations, we can unroll the Neumman series expansion of the discounted weighting of states to obtain a recursive expression:

$$\begin{aligned}
\boldsymbol{d}_{\boldsymbol{\alpha},\gamma,\pi}^{\top} &\doteq \boldsymbol{\alpha}^{\top}\sum_{t=0}^{\infty}\left(\gamma\mathbf{P}_{\pi}\right)^{t} \\
&= \boldsymbol{\alpha}^{\top} + \boldsymbol{\alpha}^{\top}\sum_{t=1}^{\infty}\left(\gamma\mathbf{P}_{\pi}\right)^{t} \\
&= \boldsymbol{\alpha}^{\top} + \gamma\left(\boldsymbol{\alpha}^{\top}\sum_{t=0}^{\infty}\left(\gamma\mathbf{P}_{\pi}\right)^{t}\right)\mathbf{P}_{\pi} \\
&= \boldsymbol{\alpha}^{\top} + \gamma\boldsymbol{d}_{\alpha,\gamma,\pi}^{\top}\mathbf{P}_{\pi} \ . \qquad (2.7)
\end{aligned}$$

These equations look similar to the policy evaluation equations but where $\boldsymbol{d}_{\alpha,\gamma,\pi}^{\top}$ plays the role of the "value function" and $\boldsymbol{\alpha}$ is its associated *reward function*. Hence, for any given initial distribution $\boldsymbol{\alpha}$, there exists a corresponding linear system of equations whose solution is the discounted weighting of states. In Sutton and Barto (2018), a similar expression for $\boldsymbol{d}_{\alpha,\gamma,\pi}$ is provided for the so-called *on-policy distribution* in the undiscounted episodic case. In the same way that the usual policy evaluation equations describe the expected sum of discounted rewards, the recursive expression for $\mathbf{d}_{\boldsymbol{\alpha},\gamma,\pi}$ corresponds to an expected sum of discounted indicator variables:

$$\begin{aligned}\mathbf{d}_{\boldsymbol{\alpha},\gamma,\pi}(s) &\doteq \sum_{s_0} \boldsymbol{\alpha}(s_0) \sum_{t=0}^{\infty} \gamma^t P_\pi \left(S_t = s \mid S_0 = s_0\right) \\ &= \sum_{s_0} \boldsymbol{\alpha}(s_0) \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\left[\mathbb{1}_{S_t=s} \mid S_0 = s_0\right] \\ &= \mathbb{E}_{\boldsymbol{\alpha}}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{S_t=s}\right] \quad .\end{aligned}$$

This interpretation of the discounted weighting of states as sum of indicator variables can be leveraged in the design of learning algorithms as originally shown by Dayan (1993).

Instead of having a set of evaluation equations for each s, we can also use a more compact recursive expression involving a matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ whose rows are initial state distributions:

$$\mathbf{D}_{\mathbf{A},\gamma,\pi} = \mathbf{A} \sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_\pi\right)^t = \mathbf{A} + \gamma \mathbf{D}_{\mathbf{A},\gamma,\pi} \mathbf{P}_\pi = \mathbf{A}\left(\mathbf{I} - \gamma \mathbf{P}_\pi\right)^{-1} \quad .$$

The choice $\mathbf{A} = \mathbf{I}$ describes the evaluation equations associated with the so-called *Successor Representation* (SR) (Dayan, 1993). Introduced in the context of reinforcement learning, the SR approach leverages the fact that the value function $\mathbf{v}_\pi$ is linear in the discounted weighting of states: $\mathbf{v}_\pi = \mathbf{D}_{\mathbf{A},\gamma,\pi}\mathbf{r}_\pi$. Because $\mathbf{D}_{\mathbf{A},\gamma,\pi}$ does not depend on the reward function, once it has been obtained, policy evaluation only involves $\mathcal{O}(|\mathcal{S}|^2)$ operations to compute a matrix-vector product rather than the $\mathcal{O}(|\mathcal{S}|^3)$ which would otherwise be required for matrix inversion when solving the policy evaluation problem from scratch. This property is beneficial in problems where the reward function (the *task*) may change but the transition matrix and policy remain the same. Of course, when the SR is not given apriori, we must also incur a cost $\mathcal{O}(|\mathcal{S}|^3)$ initially to solve for $\mathbf{D}$

in $(\mathbf{I} - \gamma \mathbf{P}_\pi)\,\mathbf{D} = \mathbf{I}$ since $\mathbf{D}$ is nothing more but the inverse $(\mathbf{I} - \gamma \mathbf{P}_\pi)$. In that sense, solving for the discounted weighting is as costly as solving for the value function directly. We will see in chapter 4.4 that $\mathbf{D}_{\mathbf{A},\gamma,\pi}$ can be interpreted as a *matrix preconditioner* arising from a more general multi-step formulation for the policy evaluation equations.

2.3.2 Average Reward and Discounting

Assumption 2 (Existence of a stationary distribution)**.** *The Markov chain induced by any policy π in a given MDP has a unique limiting distribution $\mathbf{d}_\pi$ satisfying the system of equations:*

$$\mathbf{d}_\pi^\top \mathbf{P}_\pi = \mathbf{d}_\pi^\top \ .$$

We call $\mathbf{d}_\pi \in \mathbb{R}^{|\mathcal{S}|}$ the stationary distribution of $\mathbf{P}_\pi$ and it satisfies:

$$\mathbf{P}_\pi^\star = \mathbf{1}\mathbf{d}_\pi^\top \ ,$$

where $\mathbf{P}_\pi^\star \doteq \lim_{N\to\infty} \sum_{t=0}^{N-1} \mathbf{P}_\pi^t$ (Cinlar, 1975).

The average reward of a policy (from a given state) is defined generally (Puterman, 1994; Bertsekas, 2012) as:

$$\mathbf{g}_\pi(s) \doteq \lim_{N\to\infty} \frac{1}{N} \mathbb{E}\left[\sum_{t=0}^{N-1} r(S_t, A_t) \,\middle|\, S_0 = s \right] \ .$$

and $\mathbf{g}_\pi$ is also referred to as the *gain* of policy π (Puterman, 1994). Under assumption 2, the average reward is independent of the starting state and can be written as:

$$\mathbf{g}_\pi = \lim_{N\to\infty} \frac{1}{N} \sum_{t=0}^{N-1} \mathbf{P}_\pi^t \mathbf{r}_\pi = \mathbf{P}_\pi^\star \mathbf{r}_\pi = \left(\mathbf{1}\mathbf{d}_\pi^\top\right) \mathbf{r}_\pi \ ,$$

and where all the components of $\mathbf{g}_\pi$ are the same. Hence, some authors (Sutton and Barto, 2018; Kakade, 2001; Baxter and Bartlett, 2001; Singh et al., 1994) refer to the average reward as the scalar resulting from taking the inner product of

the stationary distribution with the reward vector:

$$\eta(\pi) \doteq \mathbf{d}_\pi^\top \mathbf{r}_\pi \ .$$

Interestingly, the average reward $\eta(\pi)$ can be related to the discounted return when taking the expectation over starting states according to the stationary distribution $\mathbf{d}_\pi$.

Proposition 2.19. *The $\mathbf{d}_\pi$-weighted average of the discounted values is proportional to the average reward:*

$$\mathbf{d}_\pi^\top \mathbf{v}_{\gamma,\pi} = \frac{\eta(\pi)}{1-\gamma} \ .$$

where $\mathbf{v}_{\gamma,\pi} = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} \mathbf{r}_\pi$ is the value function for the expected discounted sum of rewards and $\mathbf{d}_\pi$ is the stationary distribution under $\mathbf{P}_\pi$.

Proof. Expanding $\mathbf{v}_\pi$ using the policy evaluation equations (2.2):

$$\mathbf{d}_\pi^\top \mathbf{v}_\pi = \mathbf{d}_\pi^\top (\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi) \ .$$

Because $\mathbf{d}_\pi$ is a stationary distribution, we have:

$$\mathbf{d}_\pi^\top \mathbf{v}_\pi = \mathbf{d}_\pi^\top \mathbf{r}_\pi + \gamma \mathbf{d}_\pi^\top \mathbf{v}_\pi \iff \mathbf{d}_\pi^\top \mathbf{v}_\pi = \frac{\mathbf{d}_\pi^\top \mathbf{r}_\pi}{1-\gamma} \ .$$

□

Proposition 2.19 provides a relationship between the discounted return and the average return when taking the expectation of the discounted values under the stationary distribution. Using the Laurent series expansion (Puterman, 1994), we can derive a complementary result connecting directly the average reward with the discounted values in the limit of discount factor going to 1.

Proposition 2.20. *Let $\mathbf{v}_{\gamma,\pi}$ be the expected sum of discounted returns, and $\mathbf{g}_\pi$ the average reward vector (bias):*

$$\lim_{\gamma \to 1} (1-\gamma) \mathbf{v}_{\gamma,\pi} = \mathbf{g}_\pi \ .$$

Proof. The following proof is based on (Puterman, 1994, corollary 8.2.5) using Laurent series. Note that the same result is also derived in Bertsekas (2012) but using the properties of the determinant solely.

By expressing the discount factor in terms of the *interest rate* $\varrho \doteq (1-\gamma)\gamma^{-1} \Rightarrow \gamma = (1+\varrho)^{-1}$ in $(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1}$, we get:

$$\begin{aligned}(\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} &= \left(\mathbf{I} - (1+\varrho)^{-1}\mathbf{P}_\pi\right)^{-1} \\ &= \left((1+\varrho)^{-1}\left((\varrho+1)\,\mathbf{I} - \mathbf{P}_\pi\right)\right)^{-1} \\ &= (1+\varrho)\left(\varrho\mathbf{I} + (\mathbf{I} - \mathbf{P}_\pi)\right)^{-1} \quad .\end{aligned}$$

This substitution exposes the matrix $\left(\varrho\mathbf{I} + (\mathbf{I} - \mathbf{P}_\pi)\right)^{-1}$ which is called the *resolvent* of $\mathbf{I} - \mathbf{P}_\pi$. The resolvent admits the following Laurent series expansion (Puterman, 1994, Theorem A.8, section A.6):

$$\left(\varrho\mathbf{I} + (\mathbf{I} - \mathbf{P}_\pi)\right)^{-1} = \varrho^{-1}\mathbf{P}_\pi^\star + \sum_{t=0}^{\infty} (-\varrho)^t \left((\mathbf{I} - \mathbf{P}_\pi + \mathbf{P}_\pi^\star)^{-1}(\mathbf{I} - \mathbf{P}_\pi^\star)\right)^{t+1}$$

Therefore, the expected sum of discounted normalized rewards can also be written as:

$$\begin{aligned}(1-\gamma)\mathbf{v}_{\gamma,\pi} &= (\mathbf{I} - \gamma\mathbf{P}_\pi)^{-1}\mathbf{r}_\pi \\ &= \mathbf{P}_\pi^\star\mathbf{r}_\pi + \frac{1-\gamma}{\gamma}\sum_{t=0}^{\infty}\left(-\frac{(1-\gamma)}{\gamma}\right)^t \left((\mathbf{I} - \mathbf{P}_\pi + \mathbf{P}_\pi^\star)^{-1}(\mathbf{I} - \mathbf{P}_\pi^\star)\right)^{t+1}\mathbf{r}_\pi \ ,\end{aligned}$$

where the first term is the average reward. As $\gamma \to 1$, the second term vanishes and we are left with $\lim_{\gamma\to 1}(1-\gamma)\mathbf{v}_{\gamma,\pi} = \mathbf{g}_\pi$. □

2.4 Policy Gradient Methods

Policy gradient methods are analogous to policy iteration in that in both cases the search for an optimal policy is guided by a policy evaluation procedure. Rather than representing the optimal greedy policy implicitly through the optimal value function, policy gradient methods operate within a designated parametrized family of stationary randomized policies. Given the striking resemblance with policy iteration, some authors (Sutton et al., 1999b; Bertsekas,

2012) also describe policy gradient methods as performing *approximation in policy space*: the terminology originally used by Bellman (1957) to describe policy iteration.

The advantage of this approach is that it provides more freedom in the modeling effort, allowing the practitioner to express potential prior knowledge regarding the structure of the optimal policy. The price to pay is of course that the model may happen to be mis-specified: ie. when the chosen parametrized family does not contain a policy whose value function is optimal. Under a suitable choice of parameterized family of policies, policy gradient methods may benefit from the regularities of the policy space, even when the underlying space of value function space is much more complex (Bertsekas, 2012). Furthermore, the emphasis on randomized policy should not be thought so much as a *restriction* but more as a *feature*, a *blessing*. As shown by Singh et al. (1994); Sutton and Barto (2018), randomized policies may be the optimal kind of policies to consider when facing imperfect knowledge about the state (partial observability): a situation which would inevitably occur when introducing function approximation. Note that this is contrary to proposition 2.14 that allowed us to concentrate only on deterministic policies for the control problem. In this setting, deterministic policies are sufficient because the state is fully known: the tabular case. In the more general case, randomized policies may come in handy.

Although both policy gradient methods and policy iteration use a value function to improve their approximation to the optimal policy, policy gradient methods perform their improvement step using the gradient of some objective with respect to the parameters of the policy; in policy iteration, the improved policy is obtained from the greedy policy. Given an initial state distribution $\boldsymbol{\alpha}$, the discounted objective for policy gradient methods is:

$$J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \doteq \boldsymbol{\alpha}^\top \mathbf{v}_{\boldsymbol{\theta}} = \boldsymbol{\alpha}^\top \left(\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\theta}}\right)^{-1} \mathbf{r}_{\boldsymbol{\theta}} \ ,$$

where we used the subscript $\boldsymbol{\theta}$ instead of $\pi_{\boldsymbol{\theta}}$ to ease notation. With the discounted weighting of states 2.3 appearing in this expression, an equivalent definition for this objective is:

$$J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \doteq \mathbf{d}_{\boldsymbol{\alpha},\gamma,\boldsymbol{\theta}}^\top \mathbf{r}_{\boldsymbol{\theta}} \ .$$

By defining our objective with respect to an initial distribution over states, the resulting set of maximizing policies is not *optimal* in the same sense as in section 2.2. In fact, the kind of policies satisfying the Bellman optimality equations 2.6 are said to be *uniformly optimal* (Altman, 1999): they are optimal no matter where the system starts. In contrast, the policies obtained by policy gradient methods in the discounted case may be optimal for one distribution over initial states but not for a different one. This limitation also arises in constrained MDPs (Altman, 1999) or in partial observable settings (Singh et al., 1994; Bertsekas, 2012). In the average reward case, the assumption on the existence of a stationary distribution dispenses us with the dependence on the initial distribution in the resulting policies.

In order to proceed with the derivation of the policy gradient theorem (Sutton et al., 1999b), we need to restrict our attention to the class of randomized policies. This requirement is meant first and foremost to provide us with smooth gradients (Konda, 2002): a condition which would not be satisfied readily with deterministic policies over discrete action spaces. Note that even if the set of actions is continuous, the objective $J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ may not be a smooth function of $\boldsymbol{\theta}$ (Konda, 2002). In the case of the deterministic policy gradient (Silver et al., 2014) for example, smoothness is assumed rather than being satisfied by construction.

Assumption 3. *$\pi_{\boldsymbol{\theta}}$ is a stationary randomized policy and for any $a \in \mathcal{A}, s \in \mathcal{S}$, $\pi_{\boldsymbol{\theta}}(a \mid s)$ is differentiable in $\boldsymbol{\theta}$.*

Another benefit of randomized policies is that for problems with partial observability they might in fact be the optimal class of policies to consider (Singh et al., 2004a; Sutton and Barto, 2018): ie. optimal deterministic policies may not even exist. Partial observability arises naturally whenever value function approximation is used (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 2018) and having policies that are robust to imperfect information is certainly an asset.

Theorem 2.21 (Policy Gradient Theorem)**.** *Under assumptions 1,3, and with $0 \leq \gamma < 1$:*

$$\frac{\partial J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{\partial \theta_i} = \sum_s \mathbf{d}_{\boldsymbol{\alpha},\gamma,\boldsymbol{\theta}}(s) \sum_a \frac{\partial \pi_{\boldsymbol{\theta}}(a \mid s)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s,a) \ ,$$

where θ_i is the ith component of $\boldsymbol{\theta}$ and Q_θ is the state-action value function of π_θ.

Proof.

$$\begin{aligned}\frac{\partial v_{\boldsymbol{\theta}}(s)}{\partial \theta_i} &= \sum_a \left(\frac{\partial \pi_{\boldsymbol{\theta}}\left(a \,|\, s\right)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s,a) + \pi_{\boldsymbol{\theta}}\left(a \,|\, s\right) \frac{\partial Q_{\boldsymbol{\theta}}(s,a)}{\partial \theta_i} \right) \\ &= \sum_a \pi_{\boldsymbol{\theta}}\left(a \,|\, s\right) \left(\left(\sum_a \frac{\partial \pi_{\boldsymbol{\theta}}\left(a \,|\, s\right)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s,a) \right) + \gamma \sum_{s'} P\left(s' \,|\, s,a\right) \frac{\partial v_{\boldsymbol{\theta}}(s')}{\partial \theta_i} \right)\end{aligned}$$

The last line was written so as to highlight the common structure with the policy evaluation equations. This is also where we diverge from the proof presented in the appendix of Sutton et al. (1999b). Let us then define:

$$\mathbf{h}_{\boldsymbol{\theta}}^{(i)}(s) \doteq \sum_a \frac{\partial \pi\left(a \,|\, s\,;\, \theta\right)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s,a) \ ,$$

where $\mathbf{h}_{\boldsymbol{\theta}}^{(i)} \in \mathbb{R}^{|\mathcal{S}|}$. Furthermore, let $\mathbf{f}_{\boldsymbol{\theta}}^{(i)} \doteq \frac{\partial v_{\boldsymbol{\theta}}}{\partial \theta_i}$, so that the recursive expression for the derivative of the value function with respect to θ_i can now be written in vector form as:

$$\mathbf{f}_{\boldsymbol{\theta}}^{(i)} = \mathbf{h}_{\boldsymbol{\theta}}^{(i)} + \gamma \mathbf{P}_{\boldsymbol{\theta}} \mathbf{f}_{\boldsymbol{\theta}}^{(i)} \ ,$$

and where $\mathbf{P}_{\boldsymbol{\theta}}$ is a shorthand notation for $\mathbf{P}_{\pi_{\boldsymbol{\theta}}}$. Because $\rho(\gamma \mathbf{P}_{\boldsymbol{\theta}}) < 1$, $\mathbf{f}_{\boldsymbol{\theta}}^{(i)}$ is the solution to the linear system:

$$\left(\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\theta}}\right) \mathbf{f}_{\boldsymbol{\theta}}^{(i)} = \mathbf{h}_{\boldsymbol{\theta}}^{(i)} \ .$$

The derivative with respect to the objective $J_{\boldsymbol{\alpha}}$ is then:

$$\frac{\partial J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{\partial \theta_i} = \boldsymbol{\alpha}^\top \left(\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\theta}}\right)^{-1} \mathbf{h}_{\boldsymbol{\theta}}^{(i)} = \mathbf{d}_{\boldsymbol{\alpha},\gamma,\boldsymbol{\theta}}^\top \mathbf{h}_{\boldsymbol{\theta}}^{(i)} \ .$$

□

Corollary 2.22 (Evaluation Equations for the Policy Gradient)**.** *Let $\nabla_{\boldsymbol{\theta}} \mathbf{v}_{\boldsymbol{\theta}}(s) \in \mathbb{R}^k$ and $\mathbf{F}_{\boldsymbol{\theta}} \in \mathbb{R}^{|\mathcal{S}| \times k}$ be the matrix whose rows contain the gradient of $\mathbf{v}_{\boldsymbol{\theta}}$ evaluated at every state: $\mathbf{F}_{\boldsymbol{\theta}}(s,\cdot) = \nabla_{\boldsymbol{\theta}} \mathbf{v}_{\boldsymbol{\theta}}(s)$. Under the assumptions of theorem 2.21, the Jacobian $\mathbf{F}_{\boldsymbol{\theta}}$ satisfies:*

$$\left(\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\theta}}\right) \mathbf{F}_{\boldsymbol{\theta}} = \mathbf{H}_{\boldsymbol{\theta}} \ , \tag{2.8}$$

where $\mathbf{H}_{\boldsymbol{\theta}} \in \mathbb{R}^{|\mathcal{S}| \times k}$ *and* $\mathbf{H}_{\boldsymbol{\theta}}(s, i) = \sum_a \frac{\partial \pi(a \mid s; \theta)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s, a)$.

Proof. The result follows immediately from theorem 2.21 which gives an expression for the partial derivative with respect to a component θ_i of $\boldsymbol{\theta}$. □

Corollary 2.22 establishes that the policy gradient admits the same form as the policy evaluation equations (2.2) but where $\mathbf{H}_{\boldsymbol{\theta}}$ replaces what would otherwise be the "reward" vector $\mathbf{r}_{\boldsymbol{\theta}}$. Remarkably, going from scalar "rewards" to vector-valued "rewards" does not involve any special theoretical or algorithmic considerations. This stems from the fact that the existence of $\mathbf{F}_{\boldsymbol{\theta}}$ hinges entirely on the matrix $\mathbf{I} - \gamma \mathbf{P}_{\boldsymbol{\theta}}$ being invertible, without regards to the nature of the "reward" term on the right-hand side of the equation. This means that to obtain the policy gradient, one could either apply a direct method or solve for $\mathbf{F}_{\boldsymbol{\theta}}$ using exactly the same iterative policy evaluation procedure as in algorithm 1. In reinforcement learning, the idea of applying policy evaluation to other reward-like quantities than the reward function from the MDP itself has been described under the General Value Function (GVF) framework (Sutton et al., 2011). To avoid overloading the word "reward", it is preferable to call the term $\mathbf{H}_{\boldsymbol{\theta}}$ a *cumulant* (White, 2015; Sutton, 2015b; Sutton and Barto, 2018) per the GVF framework. To our knowledge, the idea of solving the policy gradient in a dynamic programming fashion or through TD has never been attempted in practice and may be an interesting future research avenue.

2.4.1 Estimation

Rather than forming $\mathbf{F}_{\boldsymbol{\theta}}$ explicitly, all policy gradient methods (Williams, 1992; Marbach and Tsitsiklis, 1998; Sutton et al., 1999b; Bartlett and Baxter, 2000; Konda, 2002; Kakade, 2003) sample the gradient in a model-free fashion. While many seemingly different policy gradient estimators have been proposed in the literature, they all have in common equations (2.8).

As a starting point for deriving a gradient estimator, we want to express the policy gradient theorem as an expectation that we can estimate by sampling. However, the fact that $\mathbf{d}_{\alpha,\gamma,\theta}$ is not a distribution over states is our first obstacle in this project. But as we have seen in section 2.3, a possible remedy is to work with a normalized counterpart of the discounted weighting of states and later

correct by a factor $1/(1-\gamma)$ (Kakade, 2003) to recover the intended expected values:

$$\frac{\partial J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{\partial \theta_i} = \frac{1}{(1-\gamma)} \mathbb{E}_{\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\boldsymbol{\theta}}} \left[\sum_a \frac{\partial \pi_{\boldsymbol{\theta}}(a \mid S_t)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(S_t, a) \right] \quad . \tag{2.9}$$

Getting unbiased estimates of this expectation requires special care here because the discount factor has been coupled inside the Markov chain. The normalized discounted weighting of states is in fact of the following form:

$$\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\boldsymbol{\theta}} = \boldsymbol{\alpha}^\top \sum_{t=0}^{\infty} (1-\gamma) \left(\gamma \mathbf{P}_{\boldsymbol{\theta}}\right)^t \quad .$$

The term $(1-\gamma)\gamma^t$ in the above summation is from a geometric distribution representing the probability that the trial at $t+1$ is a first "success". Using the random horizon perspective on the discounted setting from lemma 2.1, $1-\gamma$ is the parameter of a geometric distribution where "success" is the event of reaching an absorbing state with probability $1-\gamma$ and "failure" is continuing for one more step. Therefore, writing $(1-\gamma)\gamma^t$ expresses the fact that the process has continued for t steps, and on the last step, it transitioned to an absorbing state with probability $1-\gamma$. In order to sample from the distribution $\bar{\mathbf{d}}_{\boldsymbol{\alpha},\gamma,\boldsymbol{\theta}}$, we would have to apply the policy in the MDP for one step, and upon entering the next state sample from the geometric distribution to decide whether to terminate (perhaps prematurely) the trajectory. In practice, this approach is never used because of the loss of samples due to truncation; instead, the policy gradient is estimated from samples along the on-policy undiscounted stationary.

This mismatch between the distribution used to estimate the policy gradient and the one specified by the policy gradient theorem is an issue that was brought back to light by Thomas (2014), which then led to a revision in Sutton and Barto (2018). Moreover, Thomas (2014) showed that the undiscounted estimator used for the discounted policy gradient was related to the gradient of the average reward setting. This fact can also be found in earlier work on policy gradient methods (Baxter and Bartlett, 2001; Konda, 2002; Kakade, 2003) focusing on estimating the policy gradient in the average reward case. In this body of work, discounting is seen as a knob on the bias-variance trade-off: discounting is *mean*, not an *end*.

Lemma 2.23. *Let $\mathbf{d}_{\boldsymbol{\theta}}$ be the stationary distribution induced by policy $\pi_{\boldsymbol{\theta}}$ in the given MDP:*

$$\mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{F}_{\boldsymbol{\theta}} = \frac{\mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{H}_{\boldsymbol{\theta}}}{1-\gamma} \ .$$

Proof. This is the same proof as in 2.19 but applied to $\mathbf{F}_{\boldsymbol{\theta}}$ instead of the value function. This result can also be found in (Thomas, 2014, lemma 6.1), (Kakade, 2001, theorem 2), and implicitly in (Konda, 2002, section 2.4). Because $\mathbf{d}_{\boldsymbol{\theta}}$ is a stationary distribution:

$$\mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{F}_{\boldsymbol{\theta}} = \mathbf{d}_{\boldsymbol{\theta}}^{\top}\left(\mathbf{H}_{\boldsymbol{\theta}} + \gamma\mathbf{P}_{\boldsymbol{\theta}}\mathbf{F}_{\boldsymbol{\theta}}\right) = \mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{H}_{\boldsymbol{\theta}} + \gamma\mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{F}_{\boldsymbol{\theta}} \iff (1-\gamma)\mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{F}_{\boldsymbol{\theta}} = \mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{H}_{\boldsymbol{\theta}} \ ,$$

and where we leveraged the evaluation equations for the policy gradient from corollary 2.22. □

Because $\mathbf{H}_{\boldsymbol{\theta}}$ defined in corollary 2.22 contains the discount factor, the right-hand side of the equation in lemma 2.23 is not equal to the gradient of the average reward formulation. However, lemma 2.23 suggests that using the term $\mathbf{d}_{\boldsymbol{\theta}}^{\top}\mathbf{H}_{\boldsymbol{\theta}}$ as an approximation of the average reward gradient is equivalent to taking the discounted gradient from an initial state distribution $\boldsymbol{\alpha} = \mathbf{d}_{\boldsymbol{\theta}}$. Rephrasing this statement the other way: using $\boldsymbol{\alpha} = \mathbf{d}_{\boldsymbol{\theta}}$ as an initial state distribution in the policy gradient theorem 2.21 leads to an approximation of the average reward gradient. This is the essence of what (Sutton and Barto, 2018, chapter 10) are trying to convey in their commentary on the "futility of discounting":

> *"Perhaps discounting can be saved by choosing an objective that sums discounted values over the distribution with which states occur under the policy [...]"*

Given this connection with the average reward setting, (Konda, 2002, theorem 2.13) suggests using $\mathbf{d}_{\theta}$ as an initial state distribution for implementing the discounted gradient. In practice, this would however entail a construction in which we simulate the undiscounted process up to a geometric stopping time (lemma 2.1) and then switch to the stationary distribution – a probabilistic analogue to proposition 2.20 which can also be found in (Puterman, 1994, corollary 8.2.5). The resulting estimator would then be seen as an estimator

of the normalized policy gradient 2.9 taken under the stationary distribution. This approach is by no means more practical than just sampling from the normalized policy gradient by truncation of the trajectories, which we already deemed wasteful.

This led Thomas (2014) to propose a policy gradient estimator that is estimated under the undiscounted distribution, but where discounting is integrated-back explicitly in a multiplicative manner rather than via sampling. A convenient way to see this is to write the recursive expression for the discounted policy gradient as a sum of *immediate gradients*:

$$\mathbf{F}_{\boldsymbol{\theta}}(s,\cdot) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t \sum_{a}\nabla_{\boldsymbol{\theta}}\left(\pi_{\boldsymbol{\theta}}\left(a \mid S_t\right)\right) Q_{\boldsymbol{\theta}}(S_t,a) \,\middle|\, S_0 = s\right]$$

For simplicity assume that $Q_{\boldsymbol{\theta}}$ is given, an estimator of the policy gradient in the episodic setting is then:

$$J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\gamma^t \sum_{a}\nabla_{\boldsymbol{\theta}}\left(\pi_{\boldsymbol{\theta}}\left(a \,\middle|\, s_t^{(i)}\right)\right) Q_{\boldsymbol{\theta}}(s_t^{(i)},a) \ ,$$

where $s_t^{(i)}$ denotes the state sampled at time t in the ith trajectory. In this case, the samples are obtained under the undiscounted process and the discount factor enters the estimator in a multiplicative fashion. The γ^t correction proposed by Thomas (2014), and later shown in Sutton and Barto (2018), revolves exactly around in this idea. Just as discounting of the rewards, the γ-corrected assigns less importance to the later steps than the more recent ones. Despite being mathematically correct, Thomas (2014) offers empirical evidence that the unbiased estimator tends to perform worse than its biased counterpart. An intuitive explanation for this phenomenon may be that discounting results in a *loss of information* of the same nature as for the approach based on truncated trajectories: discounting attenuates gradients, while geometric sampling discards samples. Despite being mathematically equivalent, Thomas (2014) reports that the explicit multiplicative discounting approach compares favorably in practice to the truncation method.

2.4.2 Monte-Carlo Estimators

Under the perspective of the policy gradient as a sum of discounted gradients, we now express the inner summation over actions as an expectation over actions:

$$\begin{aligned}\mathbf{F}_{\boldsymbol{\theta}}(s,\cdot) &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\mathbb{E}\left[\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)}{\pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)}Q_{\boldsymbol{\theta}}(S_t,A_t)\,\middle|\,S_t\right]\,\middle|\,S_0=s\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\nabla_{\boldsymbol{\theta}}\left(\log\pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right)Q_{\boldsymbol{\theta}}(S_t,A_t)\,\middle|\,S_0=s\right] \quad . \end{aligned} \tag{2.10}$$

This transformation is valid only when the ratio $\frac{\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(A_t \mid S_t)}{\pi_{\boldsymbol{\theta}}(A_t \mid S_t)}$ is bounded, which can be guaranteed whenever the probability of selecting an action always has at least some small probability of being selected. This ratio can also be written in another form using the fact that $\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right) = \frac{1}{\pi_{\boldsymbol{\theta}}(A_t \mid S_t)}\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)$. This form involving the gradient of the likelihood function can be found under different names depending on the field: *characteristic eligibility* (of $\boldsymbol{\theta}$) (Williams, 1992) or *eligibility vector* (Sutton and Barto, 2018) in reinforcement learning, *likelihood ratio derivative* in the simulation literature (L'Ecuyer, 1990) or *score function* in statistics (Cox and V., 1974).

Equation 2.10 can also be stated even more explicitly by expressing $Q_{\boldsymbol{\theta}}$ as a sum of discounted rewards starting from the current state:

$$\mathbf{F}_{\boldsymbol{\theta}}(s,\cdot) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\nabla_{\boldsymbol{\theta}}\left(\log\pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right)\sum_{k=t}^{\infty}\gamma^{k-t}r(S_k,A_k)\,\middle|\,S_0=s\right] \quad .$$

We can estimate this expectation by taking Monte-Carlo samples along the undiscounted process and computing the following quantity at the end of a trajectory:

$$\Delta_{\boldsymbol{\theta}} \doteq \sum_{t=0}^{T-1}\gamma^t\nabla_{\boldsymbol{\theta}}\left(\log\pi_{\boldsymbol{\theta}}\left(a_t \mid s_t\right)\right)\sum_{k=t}^{T-1}\gamma^{k-t}r(s_k,a_k) \quad .$$

Instead of collecting all samples in a batch, it is also possible to construct iteratively $\Delta_{\boldsymbol{\theta}}$ without having to store previous sampled transitions. To see this, we need to apply the same kind of summation interchange as in the proof of theorem 2.1, making sure that the indices are ordered according to

$0 \leq t \leq k < \infty$:

$$\begin{aligned}\mathbf{F}_{\boldsymbol{\theta}}(s, \cdot) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_{\boldsymbol{\theta}} \left(\log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right) \sum_{k=t}^{\infty} \gamma^{k-t} r(S_k, A_k) \,\middle|\, S_0 = s\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty} r(S_k, A_k) \sum_{t=0}^{k} \gamma^{k-t} \left(\gamma^t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right) \,\middle|\, S_0 = s\right] \quad .\end{aligned}$$

The term $\gamma^t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)$ in the inner summation is parenthesized to highlight the factor γ^{k-t} which unavoidably appears in the derivation of a *backward view* (Watkins, 1989; Kimura and Kobayashi, 1998; Tsitsiklis and Roy, 1997a; Sutton and Barto, 1998). We obtain a recursive update rule for the inner sum by pulling out the last term and adjusting the indices accordingly. For the *biased* form of the policy gradient, we have:

$$\begin{aligned}\mathbf{z}_{\boldsymbol{\theta}}^{(k)} &\doteq \sum_{t=0}^{k} \gamma^{k-t} \gamma^t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right) \\ &= \gamma \left(\sum_{t=0}^{k-1} \gamma^{k-1-t} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right) + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_k \mid S_k\right) \\ &= \gamma \mathbf{z}_{\boldsymbol{\theta}}^{(k-1)} + \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(A_k \mid S_k\right) \quad .\end{aligned}$$

The vector $\mathbf{z}_k$ is called an *eligibility trace*: a memory or *trace* of the past gradients decayed based on their recency. By incorporating the eligibility trace back into 2.10, we obtain a Monte-Carlo estimator for the episodic setting of the form:

$$\Delta_{\boldsymbol{\theta}} = \sum_{t=0}^{T-1} r(S_t, A_t) \mathbf{z}_{\boldsymbol{\theta}}^{(t)} \quad .$$

Interestingly, this form has the immediate rewards weighted by an accumulation of past gradients whereas 2.10 uses immediate eligibility vectors $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}$ weighted by an accumulation of future rewards. The use of an eligibility trace for the parameters $\boldsymbol{\theta}$ gives us a *causal* (Sutton and Barto, 1998) estimator for the policy gradient: an estimator which uses only information available up to that point. Note that the above estimator using an eligibility trace on $\boldsymbol{\theta}$ also appeared in the Adaptive Heuristic Critic algorithm (Barto et al., 1983; Sutton, 1984) in the context of *stochastic learning automata* (Barto et al., 1981). It was only later with Williams (1992); Kimura and Kobayashi (1998); Marbach and Tsitsiklis (1998); Sutton et al. (1999b); Konda and Tsitsiklis (2000); Baxter

and Bartlett (2001) that it was re-discovered in the context of policy gradient methods.

2.4.3 Advantage and TD Errors

Williams (1992) proposed a Monte-Carlo estimator using a *baseline* $b : \mathcal{S} \to \mathbb{R}$ to reduce variance. Any such state-dependent baseline can be added to the policy gradient without impacting the unbiasedness of the resulting estimator because:

$$\begin{aligned}
&\mathbb{E}\left[\sum_a \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}\left(a \mid S_t\right)\left(Q_{\boldsymbol{\theta}}(S_t, a) - b(s)\right) \,\middle|\, S_0 = s\right] \\
&= \mathbf{F}_{\boldsymbol{\theta}}(s, \cdot) - \mathbb{E}\left[b(s) \nabla_{\boldsymbol{\theta}} \sum_a \pi_{\boldsymbol{\theta}}\left(a \mid S_t\right) \,\middle|\, S_0 = s\right] \\
&= \mathbf{F}_{\boldsymbol{\theta}}(s, \cdot) \ ,
\end{aligned}$$

where the last term disappeared because $\pi_{\boldsymbol{\theta}}$ is a distribution, and the gradient of a constant is 0. While not initially recognized by Williams (1992), baselines correspond to the well-understood notion of *control variates* (Hammersley and Handscomb, 1964; Rubinstein, 1981) in statistics. This connection appears to have been first made by Greensmith et al. (2004) and has become a well-accepted fact nowadays – despite Sutton and Barto (2018) avoiding it.

A common choice of baseline is the value function $v_{\boldsymbol{\theta}}$ [1] , resulting in a quantity called the *advantage function* (Baird, 1993): $A_{\boldsymbol{\theta}}(s, a) \doteq Q_{\boldsymbol{\theta}}(s, a) - v_{\boldsymbol{\theta}}(s)$. This form suggests that we could also form an estimator of the policy gradient that does not rely on knowing $Q_{\boldsymbol{\theta}}$, a function over $\mathcal{S} \times \mathcal{A}$, and that learning $v_{\boldsymbol{\theta}}$ over $\mathcal{S}$ might suffice. To see this, note that:

$$Q_{\boldsymbol{\theta}}(s, a) - v_\theta(s) = r(s, a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) v_\theta(s') - v_\theta(s') \ ,$$

[1] The subscript $\boldsymbol{\theta}$ in $v_{\boldsymbol{\theta}}$ stands for the value function associated to the policy parameterized by $\boldsymbol{\theta}$. We write $\hat{v}_{\boldsymbol{\theta}}(s; \mathbf{w})$ when the value function is parameterized by a vector $\mathbf{w}$.

so that an equivalent form of the policy gradient would be:

$$\begin{aligned}\mathbf{F}_{\boldsymbol{\theta}}(s,\cdot) &= \mathbb{E}\left[\sum_a \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}\left(a \mid S_t\right)\left(Q_{\boldsymbol{\theta}}(S_t,a) - v_{\boldsymbol{\theta}}(s)\right) \,\middle|\, S_0 = s\right] \\ &= \mathbb{E}\left[\nabla_{\boldsymbol{\theta}}\left(\log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right)\left(r(S_t,A_t) + \gamma v_{\boldsymbol{\theta}}(S_{t+1}) - v_{\boldsymbol{\theta}}(S_t)\right) \mid S_0 = s\right] \quad .\end{aligned} \tag{2.11}$$

The quantity multiplying the eligibility vector on the right-hand side is the TD error from section 2.1.2:

$$\delta_t \doteq R_{t+1} + \gamma v_{\boldsymbol{\theta}}(S_{t+1}) - v_{\boldsymbol{\theta}}(S_t) \quad .$$

Therefore, we can write (Kimura and Kobayashi, 1998) the biased policy gradient from (2.10) as:

$$\begin{aligned}\mathbf{F}_{\boldsymbol{\theta}}(s,\cdot) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_{\boldsymbol{\theta}}\left(\log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right)\left(\left(\sum_{k=t}^{\infty}\gamma^{k-t} r(S_k,A_k)\right) - v_{\boldsymbol{\theta}}(S_t)\right) \,\middle|\, S_0 = s\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_{\boldsymbol{\theta}}\left(\log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t\right)\right)\sum_{k=t}^{\infty}\gamma^{k-t}\delta_k \,\middle|\, S_0 = s\right] \quad .\end{aligned} \tag{2.12}$$

This result follows from recognizing that the inner sum involves a telescoping series where every two consecutive terms $v_{\boldsymbol{\theta}}(S_k)$ cancelling each other:

$$\sum_{k=t}^{\infty}\gamma^{k-t}\delta_k = \left(\sum_{k=t}^{\infty}\gamma^{k-t} r(S_k,A_k)\right) + \left(\sum_{k=t}^{\infty}\gamma^{k-t}\left(\gamma v_{\boldsymbol{\theta}}(S_{k+1}) - v_{\boldsymbol{\theta}}(S_k)\right)\right) \quad ,$$

and the second term involving the differences converges to $-v_{\boldsymbol{\theta}}(S_t)$. Schulman et al. (2016) re-discovered this form and introduced an additional "λ" parameter in the inner summation: $\sum_{k=t}^{\infty}\left(\lambda\gamma\right)^{k-t}\delta_k$. The effect of this parameter is to control the bias-variance tradeoff, with $\lambda = 0$ resulting in 2.11 and single immediate TD term while $\lambda = 1$ leads to the advantage function. The bias-variance interpretation and the mechanism at play are the same as for $TD(\lambda)$ (Sutton, 1984, 1988) shown in chapter 2.1.2.

2.4.4 Actor-Critic Architecture

Up to now, we assumed that the value term $Q_{\boldsymbol{\theta}}$ in the policy gradient theorem was either given apriori (presumably using a direct or iterative policy evaluation methods of section 2.1) or estimated in a Monte-Carlo fashion using full rollouts. A third option consists in learning the value function from samples using Temporal Difference (TD) learning (Sutton, 1984, 1988): a *model-free* approach to the policy evaluation problem. The idea of learning a parametrized policy by gradient ascent jointly with its associated value function was pioneered by (Barto et al., 1983; Sutton, 1984) in their Adaptive Heuristic Critic algorithm. The resulting architecture, which nowadays is simply called *actor-critic* (Sutton and Barto, 2018), was later analyzed theoretically by Marbach and Tsitsiklis (1998); Konda and Tsitsiklis (2000); Konda (2002).

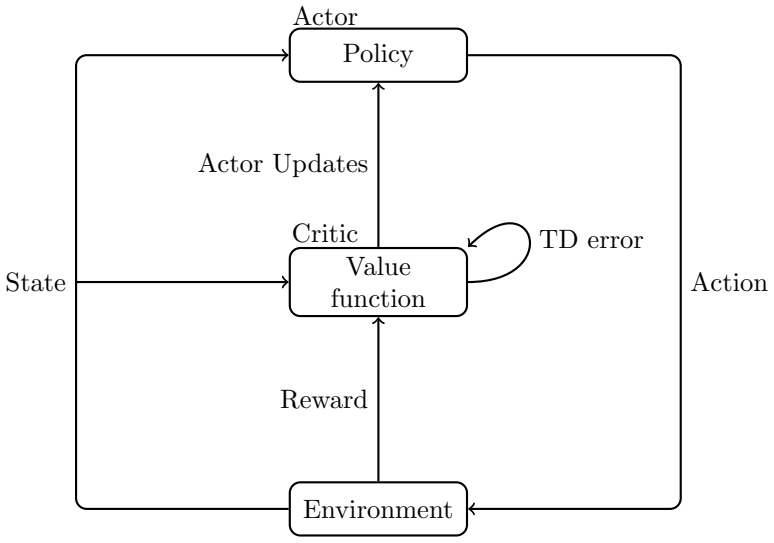


FIGURE 2.1 – The Actor-Critic Architecture

In the original actor-critic architecture, the temporal difference error provides a reinforcement signal flowing not only through the *critic* but also through the *actor* which executes the parameterized policy in the environment. The critic is more of an *ally* than an adversary to the actor as it seeks to improve its performance using value estimates. The combination of a TD(0) learning algorithm for the critic and the bias form of the policy gradient is shown in algorithm 4. In this algorithm, the TD error term in the actor update comes from the advantage estimator shown in section 2.4.3 where $Q_{\boldsymbol{\theta}}$ is estimated with the one-step return. Many other variants on the actor-critic architecture

Algorithm 4: Actor-Critic with One-Step TD updates

Initialize (or other initialization method):
Input: Critic learning rate $\epsilon^{(\mathbf{w})}$ and actor learning rate $\epsilon^{(\boldsymbol{\theta})}$
$\mathbf{w} \leftarrow \mathbf{0}$
$\boldsymbol{\theta} \leftarrow \mathbf{0}$
repeat
  $\delta_t \doteq r(S_t, A_t) - \hat{v}_{\boldsymbol{\theta}}(S_t; \mathbf{w})$
  if S_{t+1} *is not terminal* **then**
    $\delta_t \leftarrow \delta_t + \gamma \hat{v}_{\boldsymbol{\theta}}(S_{t+1}; \mathbf{w})$
  end
  $\mathbf{w} = \mathbf{w} + \epsilon_t^{(\mathbf{w})} \delta_t \nabla_{\mathbf{w}} \hat{v}_{\boldsymbol{\theta}}(S_t; \mathbf{w})$
  $\boldsymbol{\theta} = \boldsymbol{\theta} + \epsilon_t^{(\boldsymbol{\theta})} \delta_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} (A_t \mid S_t)$
until S_{t+1} *is terminal*

can be obtained using multi-step targets for the actor and critic, which can either be implemented offline or using eligibility traces.

Chapter 3

Temporal Abstraction

Options (Sutton et al., 1999a) provide a framework for representing, planning and learning with temporally abstraction actions. The option framework assumes the existence of a base MDP on which are overlaid temporally abstract actions called *options*. An option is a triple $(\mathcal{I}_o, \pi_o, \beta_o)$ where $\mathcal{I} \subseteq \mathcal{S}$ is an initiation set[1], $\pi_o : \mathcal{S} \to Dist(\mathcal{A})$ is the policy of an option (which can also be deterministic) and $\beta_o : \mathcal{S} \to [0, 1]$ is a termination condition.

In the *call-and-return* model of option execution, a policy over options $\mu : \mathcal{S} \to Dist(\mathcal{O})$ (deterministic if wanted) chooses an option among those which can be initiated in a given state and executes the policy of that option until termination. Once the chosen option has terminated, the policy over options chooses a new option and the process is repeated until the end of the episode.

For a more unified view, we now write $\pi(a \mid s, o) \doteq \pi_o(a \mid s, o)$ and $\beta(s, o) \doteq \beta_o(s)$. This change of notation is meant to highlight the fact that the choice of primitive actions by the option policies as well as the termination conditions are functions of the state-option pairs. Indeed, we show formally in section 3.3 that the dynamics of the process induced by a set of options and a policy over them can be viewed as an MDP over such state-option pairs. Another consequence of writing π_o and β_o is that it leads the reader into thinking that an agent must represent $|\mathcal{O}|$ entities separately. Using the notion of parameterized options, we see in section 5.1 that this needs not be the case and that sharing of parameters is in fact possible. This means for example that one could choose

[1] Initiation sets were part of the original definition in Sutton et al. (1999a) but have since been removed in Sutton and Barto (2018).

to represent the options policies under a single function taking as input a state and an option and outputting a probability distribution over actions. Viewing options through this lens opens the way towards better generalization using more sample efficient representations of options.

Throughout this thesis, initiation sets are purposefully ignored by assuming that every option can be initialized everywhere, that is: $\forall o \in \mathcal{O}, \forall s \in \mathcal{S} : s \in \mathcal{I}_o$. The reason for this omission pertains to the difficulty in adapting the policy gradient methodology to initiation *sets* rather than smooth functions. We require in section 5.1 that all components of the system be randomized so as to guarantee smoothness of the optimization objective. Thus going from initiation sets to randomized entities would require rethinking the semantics of the resulting system with respect to the original options framework. We prefer to leave such conceptual changes for future work. However, we discuss in section 5.5 how a generalization of initiation sets can readily be incorporated within our existing framework when using a randomized policy over options.

3.1 Option Models

The combination of a set of options and base MDP leads to a semi-Markov decision process (SMDP) (Howard, 1963; Puterman, 1994) in which the transition time between two decision points is a random variable. When considering the induced process only at the level of state-option pairs, usual dynamic programming results can be reused after a transformation to an equivalent MDP (Puterman, 1994). For *Markov Options* – options which do not depend on the history since initiation – this transformation can conveniently be expressed either in closed-form or as the solution to evaluation equations over their *models*. The reward model for options $b : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ is a mapping from state-option pairs to the expected discounted return until termination and satisfies the equations:

$$b(s,o) = \sum_a \pi\left(a \,|\, s,o\right) \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s,a\right) \left(1 - \beta(s',o)\right) b(s',o) \right) \quad . \quad (3.1)$$

Similarly, the transition model for options $F : \mathcal{S} \times \mathcal{O} \times \mathcal{S} \to \mathbb{R}$ is an expectation over the next state upon termination and is the solution to:

$$F(s,o,s') = \gamma \sum_{a} \pi\left(a \,|\, s,o\right) \sum_{\bar{s}} P\left(\bar{s} \,|\, s,a\right) \left((1-\beta(\bar{s},o))F(\bar{s},o,s') + \beta(\bar{s},o)\mathbb{1}_{s'=\bar{s}}\right) \quad . \tag{3.2}$$

Note that the reward and transition models are recursive expressions of the same nature as the policy evaluation equations from section 2.1: they possess a *Bellman-like* (Sutton et al., 1999a) structure. This means that any direct or iterative approach of the kind shown in section 2.1 could be used, in addition to temporal difference learning methods. This perspective has been fully adopted in the latest draft of the RL textbook by Sutton and Barto (2018) where option models are presented as a special case (White, 2017) of the GVF framework (Sutton et al., 2011; White, 2015). We also show in chapter 4 how these equations naturally emerge from a broader notion of multi-step models that we call λ-models (section 4.3).

3.2 Evaluation Equations

The expected discounted return associated with a set of options $\mathcal{O}$ and a policy over them can be expressed as:

$$\begin{aligned} Q_{\mathcal{O}}(s,o) \doteq & \mathbb{E}\left[\sum_{t=0} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s, O_0 = o\right] \\ & = b(s,o) + \sum_{s'} F(s,o,s') \sum_{o'} \mu\left(o' \,|\, s'\right) Q_{\mathcal{O}}(s',o') \quad . \end{aligned} \tag{3.3}$$

Here b is treated identically as the instantaneous reward term in the usual MDP formulation, effectively hiding the fact that it is the result of a temporally extended action. Because (3.3) describes the dynamics of the decision process at the level of the policy over options, we refer to these equations as the evaluation equations for options at the SMDP level.

A benefit of the options framework is that the stream of experience between two decision steps of the SMDP level is accessible and can be utilized for learning. Expanding (3.1) and (3.2) inside (3.3), we obtain a new expression for the

evaluation equations which we call the *intra-option Bellman equations*:

$$Q_{\mathcal{O}}(s,o) = \sum_a \pi\left(a \mid s,o\right)\left(r(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right) U_{\mathcal{O}}(s',o)\right) \quad . \tag{3.4}$$

The term $U_{\mathcal{O}}(s',o) \doteq (1-\beta(s',o))Q_{\mathcal{O}}(s',o) + \beta(s',o)v_{\mathcal{O}}(s')$ represents the *utility* of persisting with the same option versus changing to a better one. Note that the utility term can also be written as $U_{\mathcal{O}}(s',o) = Q_{\mathcal{O}}(s',o) - \beta(s',o)A_{\mathcal{O}}(s',o)$ which leads to:

$$Q_{\mathcal{O}}(s,o) = \sum_a \pi\left(a \mid s,o\right)\Big(r(s,a) + \gamma \sum_{s'} P\left(s' \mid s,a\right)\left(Q_{\mathcal{O}}(s',o) - \beta(s',o)A_{\mathcal{O}}(s',o)\right)\Big), \tag{3.5}$$

where $A_{\mathcal{O}}(s',o) \doteq Q_{\mathcal{O}}(s',o) - v_{\mathcal{O}}(s')$ is the *advantage function* (Baird, 1993). This form suggests an interpretation in which continuing with an option incurs a *cost* $\beta(s',o)A_{\mathcal{O}}(s',o)$. If an option is advantageous but likely to terminate (higher value of β), a higher cost is incurred; if advantageous but persistent (smaller value of β), the cost is less.

3.3 Augmented MDP

The intra-option Bellman equations can also be obtained when considering the evolution a *flat* process over an augmented[2] state space $\widetilde{\mathcal{S}} \doteq \mathcal{S} \cup \mathcal{O}$. This augmentation is necessary to restore the Markov property which would otherwise be lost – even with Markov options – when considering trajectories only over states or state-action pairs (Sutton et al., 1999a). The transition probability function $\widetilde{P}$ of the Markov process over the augmented state space $\widetilde{\mathcal{S}}$ is then:

$$\widetilde{P}\left(\tilde{s}' \mid \tilde{s},a\right) = P\left(s' \mid s,a\right)\left((1-\beta(s',o))\mathbb{1}_{o'=o} + \beta(s',o)\mu\left(o' \mid s'\right)\right) \quad . \tag{3.6}$$

In this formulation, the MDP transition probability function $P\left(\cdot \mid s,a\right)$ is coupled with the termination conditions and policy over options. Hence, if $o = o'$ it can either be that the current option has terminated and μ happened to have picked the same option again, or that the current option has continued with the same option. However, if $o \neq o'$ it can only be, by the call-and-return

[2]The augmented MDP considered here is unrelated to the one from Roy (2003).

execution model, that the current option has terminated and that μ chose a different option. Another way to interpret this expression is to view the factor $P\left(o' \mid s', o\right) \doteq (1-\beta(s', o))\mathbb{1}_{o'=o} + \beta(s', o)\mu\left(o' \mid s'\right)$ as a mixture policy (section 3.5) over options using $\mu\left(o' \mid s'\right)$ in proportion $\beta(s', o)$ but persisting with the same option when $\mathbb{1}_{o'=o}$ with probability $1-\beta(s', o)$.

Equipped with a reward function $\widetilde{r} : \widetilde{\mathcal{S}} \times \mathcal{A} \to \mathbb{R}$ where $\widetilde{r}(\tilde{s}, a) \doteq r(s, a)$, we can define a discounted MDP over the augmented states $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{P}, \widetilde{r}, \gamma)$. Accordingly, the combination of a policy $\widetilde{\pi} : \widetilde{\mathcal{S}} \to Dist(\mathcal{A})$ with $\widetilde{M}$ induces a Markov process over augmented states, actions and rewards. The expected discounted return from an augmented state $\tilde{s}$ then follows the usual structure of the evaluation equations in an MDP:

$$\widetilde{v}_{\widetilde{\pi}}(\tilde{s}) = \sum_{a} \widetilde{\pi}\left(a \mid \tilde{s}\right) \left(\widetilde{r}(\tilde{s}, a) + \gamma \sum_{\tilde{s}'} \widetilde{P}\left(\tilde{s}' \,\middle|\, \tilde{s}, a\right) \widetilde{v}_{\widetilde{\pi}}(\tilde{s}') \right) \tag{3.7}$$

As shown below, the intra-option Bellman equations (3.5) are recovered by expanding the augmented transition probability function $\widetilde{P}$, reward $\widetilde{r}$ function and policy $\widetilde{\pi}$ inside (3.7):

$$\begin{aligned} \widetilde{v}_{\widetilde{\pi}}(\tilde{s}) &= \sum_{a} \pi\left(a \mid s, o\right) \Big(r(s, a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \left(Q_{\mathcal{O}}(s', o) + \beta(s', o) A_{\mathcal{O}}(s', o) \right) \Big) \\ &\doteq Q_{\mathcal{O}}(s, o) \ . \end{aligned}$$

The solution to the evaluation equations over the augmented state space can be written in matrix form with:

$$\widetilde{\mathbf{P}}_{\widetilde{\pi}}(\tilde{s}, \tilde{s}') = \sum_{a} \widetilde{\pi}\left(a \mid \tilde{s}\right) \widetilde{P}\left(\tilde{s}' \,\middle|\, \tilde{s}, a\right), \qquad \text{and} \qquad \widetilde{\mathbf{r}}_{\widetilde{\pi}}(\tilde{s}) = \sum_{a} \widetilde{\pi}\left(a \mid \tilde{s}\right) \widetilde{r}(\tilde{s}, a) \ ,$$

which then allows us to write the option-value function $\widetilde{\mathbf{v}}$ as:

$$\widetilde{\mathbf{v}} \doteq (\mathbf{I} - \gamma \widetilde{\mathbf{P}}_{\widetilde{\pi}})^{-1} \widetilde{\mathbf{r}}_{\widetilde{\pi}} = \sum_{t=0}^{\infty} \left(\gamma \widetilde{\mathbf{P}}_{\widetilde{\pi}} \right)^{t} \widetilde{\mathbf{r}}_{\widetilde{\pi}} \ . \tag{3.8}$$

Similar to its MDP counterpart of section 2.3, a *discounted weighting* over augmented states can also be defined as a function of a corresponding initial distribution $\widetilde{\boldsymbol{\alpha}}$:

$$\mathbf{d}_{\widetilde{\boldsymbol{\alpha}}}^{\top} = \widetilde{\boldsymbol{\alpha}}^{\top} \sum_{t=0}^{\infty} \gamma \widetilde{\mathbf{P}}_{\widetilde{\pi}} = \widetilde{\boldsymbol{\alpha}}^{\top} (\mathbf{I} - \gamma \widetilde{\mathbf{P}}_{\widetilde{\pi}})^{-1} \ .$$

The expected sum of discounted rewards under the initial distribution is then given by $\mathbf{d}_{\widetilde{\alpha}}^{\top}\widetilde{\mathbf{r}}_{\pi}$.

3.4 Optimality Equations

When it comes to learning or planning with temporal abstraction, multiple notions of *optimality* exist depending on which *level* of the model one is referring to and whether it is a *local* or *global* property. For example, the MAXQ (Dietterich, 2000) framework is built around the idea of *recursive optimality*, which is asking more than the simple notion of optimality at the SMDP level assumed in the options framework. For a given set of options in an MDP, the corresponding Bellman optimality equations for options are:

$$v_{\mathcal{O}}^{\star}(s) \doteq \max_{\mu \in \Pi^{\mathrm{MD}}} v_{\mu}(s) \ .$$

Just as it is the case with the usual optimality equations for MDPs (section 2.2), the maximization on the right-hand side of the optimality equations for options need not be carried directly in the space of deterministic policies. Instead, we can leverage the componentwise partial order and write:

$$\begin{aligned} v_{\mathcal{O}}^{\star}(s) &= \max_{o} Q_{\mathcal{O}}^{\star}(s,o) \\ &= \max_{o} \left(b(s,o) + \sum_{s'} F(s,o,s') v_{\mathcal{O}}^{\star}(s') \right) \\ &= \max_{o} \left(\sum_{a} \pi\left(a \mid s, o\right) \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) U_{\mathcal{O}}^{\star}(s', o) \right) \right) \ . \end{aligned}$$

The solution to the optimality equations for options can be guaranteed to be same as the one from the underlying MDP when the set of options is augmented with primitive options (primitive actions in disguise). This also entails that we should not expect to attain more expected return using options than with primitive actions. The use of options should be to learn and plan faster: not to gain more return. Although this fact can be understood intuitively, a proof seems to be missing from the original options paper (Sutton et al., 1999a). Proposition 3.1 offers a more detailed argument to support this fact.

Proposition 3.1 (Planning with Primitive Options)**.** *Let $v^\star$ be the optimal value function in an MDP, and $v^\star_{\mathcal{O}\cup\mathcal{A}}$ the optimal value function given any set of options $\mathcal{O}$ augmented with primitive options (one for each primitive actions):*

$$v^\star = v^\star_{\mathcal{O}\cup\mathcal{A}} \ .$$

Proof. The optimal value function at the SMDP level is such that for any stationary Markov policy over options μ:

$$v^\star_{\mathcal{O}\cup\mathcal{A}} \geq v_\mu \ .$$

Note that any primitive policy (a policy over primitive actions only) can be embedded within a policy over options. Hence the class of all primitive policies is also contained within the set of all possible policies over options augmented with primitives. So if the above statement holds for any μ in this larger class, it must also hold for all primitive policies and $v^\star_{\mathcal{O}\cup\mathcal{A}} \geq v^\star$.

In the other direction, we need to also establish that $v^\star \geq v^\star_{\mathcal{O}\cup\mathcal{A}}$. If it were not the case, then it means that $v^\star_{\mathcal{O}\cup\mathcal{A}} > v^\star$ in one or more states. This can only be the case if μ either chooses a primitive action whose value is different from the one in $v^\star$, or it could also mean that μ takes a temporally extended option leading to more return. Both situations are impossible as it would imply loosing the principle of optimality which $v^\star$ depends upon. □

If the MDP is known and can fit in memory, we could solve for the optimality equations for options using either policy iteration or value iteration. This would require the options models to be obtained either by solving their corresponding evaluation problem in closed-form or using iterative methods. Value iteration can then be applied over option models by adopting the operator perspective on equations (3.3) at the SMDP level. Alternatively, we could use the evaluation equations (3.5) at the *intra* level without having to form models apriori (given that the options are Markov). At this level, the Bellman optimality equations are:

$$\begin{aligned} v^\star_{\mathcal{O}}(s) &= \max_o Q^\star_{\mathcal{O}}(s,o) \\ &= \max_o \sum_a \pi\left(a \,|\, s,o\right) \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s,a\right) U^\star_{\mathcal{O}}(s',o) \right). \end{aligned}$$

where $U^\star_{\mathcal{O}}(s',o) \doteq (1-\beta(s',o))Q^\star_{\mathcal{O}}(s',o) + \beta(s',o)v^\star_{\mathcal{O}}(s')$. The *intra-option Q-learning* algorithm (Sutton et al., 1999a), a model-free method, follows directly from these equations and uses the following sequence of iterates:

$$G_t^{(1)} = R_{t+1} + \gamma\left((1-\beta(S_{t+1},O_t))Q_t(S_{t+1},O_t) - \beta(S_{t+1},O_t)\max_o Q_t(S_{t+1},o)\right)$$
$$Q_{t+1}(S_t,O_t) = Q_t(S_t,O_t) + \epsilon_t\left(G_t^{(1)} - Q_t(S_t,O_t)\right) \quad .$$

In online and incremental RL algorithms (Sutton, 1988; Sutton and Barto, 2018), experience needs to be assimilated as fast as possible, ideally without having to wait longer than the next step. Intra-Option Q-learning embodies this philosophy in the same way that TD(λ) with eligibility traces allows us to implement the *acausal* forward view of the λ-return. Intra-option Q-learning achieves this online goal while its Q-learning counterpart at the SMDP level must wait until termination before making an update. The *SMDP Q-learning* (Bradtke and Duff, 1995; Sutton et al., 1999a) originates from the optimality equation at the SMDP level, but instead of computing option models, it samples through them:

$$G_t^{(N)} = \left(\sum_{n=t}^{N-1}\gamma^{n-t}\right) + \gamma^N \max_o Q_t(S_{t+K},o)$$
$$Q_{t+1}(S_t,O_t) = Q_t(S_t,O_t) + \epsilon_t\left(G_t^{(N)} - Q_t(S_t,O_t)\right) \quad .$$

Another way to think of SMDP Q-learning is as a form of n-steps Q-learning where the number of steps is a random variable corresponding to the number of steps that the option has taken until termination.

3.5 Mixture Distribution

The evaluation equations for Markov options admits a third form that involves only a single $Q_{\mathcal{O}}$ term on the right-hand side rather than a pair $Q_{\mathcal{O}}$ and $v_{\mathcal{O}}$:

$$Q_{\mathcal{O}}(s,o) = \sum_a \pi\left(a \mid s,o\right)\left(r(s,a) + \gamma\sum_{s'} P\left(s' \mid s,a\right)\sum_{o'} P\left(o' \mid s,o\right) Q_{\mathcal{O}}(s',o)\right) \quad , \tag{3.9}$$

where $P\left(o' \mid s,o\right) = \left(1-\beta(s',o)\right)\mathbb{1}_{s'=s} + \beta(s',o)\mu\left(o' \mid s'\right)$. Written in this way, we see that the evaluation equations for Markov options involve a mixture

distribution over two mixture components: a *degenerate* distribution $\mathbb{1}_{o'=o}$ and the policy over options μ itself which is a categorical distribution.

Two approaches are possible when it comes to sampling from the distribution $P(o' \mid s, o)$: either we compute the distribution explicitly, then use the inverse transform sampling (Devroye, 1986) method, or we use a two-steps process where we first pick a mixture component, then sample from it. The second approach is how the call-and-return model is typically implemented: at every step, we sample from the termination condition, keeping the same option if the outcome is "continue" or sampling a new option according to μ if the outcome is "terminate". The alternative where the probability distribution over next options is computed explicitly bears a slightly different semantics: termination is now *implicit* because the termination events are no longer sampled explicitly (two implement the two-steps sampling approach). It is possible that this *implicit* variant may provide some practical variance reduction benefits in practice. In fact, we may think of this approach as an *expected* form of the two-steps counterpart, the same way that we have *expected SARSA* (John, 1994; Sutton and Barto, 1998; van Seijen et al., 2009; Sutton and Barto, 2018) as a variance-reduced counterpart to *SARSA* (Rummery and Niranjan, 1994). The statistical mechanism at play here is the *conditioning* (Rubinstein, 1981) strategy for variance reduction.

3.6 Distributed Representations

Interesting extensions to the options framework may be obtained by using richer distributions than the categorical case. In fact, learning with discrete options the way we currently do is amenable to learning with *local* representations (Hinton et al., 1986). That is, every discrete option is treated as a separate entity representing a fixed behavior and its duration. While this kind of representation tends to be easier to interpret, it does a poor job at representing knowledge compactly and may require more samples than a *distributed* representations.

In chapter 5.3, we propose a learning architecture where options have parameters that are learned in a policy gradient setting. The formulation is general enough to account for possible parameter sharing between various components of the system: policy over options, option policies and termination conditions. This form of parameter sharing is one mechanism by which generalization abilities and sample efficiency can be improved. The proposed shift to a more distributed representation for options would be another mechanism, distinct from parameter sharing, to improve generalization.

To better imagine what such distributed representation for options may look like, it is useful to view the mixture distribution $P(o' \mid s', o)$ as a distribution over *one-hot/one-of-k* (Bishop, 2006) vectors instead of integer *categories*. For example, the identity of option i out of $|\mathcal{O}|$ options can be represented by the vector $\mathbf{e}_i \in \mathbb{R}^{|\mathcal{O}|}, \mathbf{e}_i(s) = \mathbb{1}_{s=i}$, a vector of zeros except for its i-th element set to 1. In this case, a natural distributed extension to the one-hot representation would be to use binary strings $\{0,1\}^k$ with the policy over options μ as a multivariate Bernoulli distribution.

Under this representation, the presence (or absence) of a single bit may not fully signal the *identity* of a chosen option; it would rather be recovered by the overall *pattern of activation* (the particular configuration of ones and zeros). The meaning of individual bits in this representation remains to be properly defined, but the general idea is that they would capture different *aspects* of the behavior that the learning system is currently trying to produce. The simultaneous presence of certain bits, for example, may correspond to different properties that the system ought to *see* (predict) as a consequence of acting in the environment. This interpretation has much in common with the recent proposal put forward by Sutton (2016) and his *subgoal keyboard*: the representation of options-like abstractions by a weighted combination of subgoals or intentions. While distributions over binary strings are natural extensions, a distributed representation over real-valued vectors is also conceivable and would perhaps makes graded responses easier to express. We could for example choose to represent the policy over options as a normal distribution over real-valued vectors $\mathbf{o} \in \mathbb{R}^k$ with mean and standard deviation as a function of the state.

In order to maintain the semantics of the call-and-return execution model, it would be crucial to replicate the same *biasing* mechanism provided by the

degenerate mixture component $\mathbb{1}_{o'=o}$ in (3.9). That is, we would need the distribution over next option vectors to be more likely to samples similar option vectors as the previous ones if the probability of continuing is high. This view implies that *commitment* to an option could be a function not only of the termination condition, but also of the width of some kernel (a *bump*) centered on the previous option vector. The indicator function of the degenerate component in (3.9) would therefore be replaced by a smooth counterpart, where similarity would not be established as a hard equality of the form $o' = o$ but rather as a function of *closeness* in Euclidean distance to the center of that kernel. If the distribution over option vectors is binary, it would be more suitable to use a distance metric such as the Hamming distance (MacKay, 2002).

3.7 Related Contemporary Frameworks

The options framework was designed to be as simple as possible without compromising on its generality. Hence, it makes no assumption on the kind of state abstraction, policy representation, learning algorithm or network architectures. Yet, we have argued in section 3.5 that the framework may gain even more generality and expressivity through a *distributional shift* : viewing the identity of an option as property of the *pattern of activation* in its vector-valued representation.

In the following, we view two recent proposals through the lens of options and show how they leverage a similar idea: Feudal Network by Vezhnevets et al. (2017) and Adaptive Skills Adaptive Partitions by Mankowitz et al. (2016). We also review earlier work by Levy and Shimkin (2012) proposing a gradient-based approach similar to Mankowitz et al. (2016) and the one presented in this thesis (option-critic).

3.7.1 Augmented Hierarchical Policy (AHP)

Our previous construction of the MDP $\widetilde{M}$ in section 3.3 is based on the idea of augmenting the state with the current option while preserving the original set of actions. Under this perspective, the choice of option with μ and the termination events with β are hidden within the transition probability function $\widetilde{P}$.

The alternative put forward in Levy and Shimkin (2012) consists in specifying the choice of primitive action, whether to terminate, and which option to pick next as a single policy which they call an *Augmented Hierarchical Policy (AHP)*.

An *action* in this case is a triple $\tilde{A}_t \doteq (A_t, K_t, O_{t+1})$, $\tilde{A}_t \in \widetilde{\mathcal{A}} \doteq \mathcal{A} \cup \mathcal{K} \cup \mathcal{O}$ where $K_t \in \mathcal{K} \doteq \{\sharp, \flat\}$ is a random variable for the event of continuing ($\sharp$) or terminating ($\flat$) at time t. It can then be shown (Levy and Shimkin, 2012, after some simplification to their equation 14) that the policy $\bar{\pi}$ over $\widetilde{\mathcal{S}}$ and $\widetilde{\mathcal{A}}$ is:

$$\bar{\pi}\left(\widetilde{a}_t \,|\, \tilde{s}_t\right) = \pi\left(a_t \,|\, s_t, o_t\right)\left((1-\beta(s_t,o_t))\mathbb{1}_{k_t=\sharp}\mathbb{1}_{o_{t+1}=o_t} + \beta(s_t,o_t)\mu\left(o_{t+1} \,|\, s_t\right)\mathbb{1}_{k_t=\flat}\right).$$

Here the termination conditions and policy over option policy are coupled inside the policy $\bar{\pi}$ as opposed to section 3.3 where they were absorbed within transition probability function $\widetilde{P}$. With the transition probability function:

$$\bar{P}\left(\tilde{s}_{t+1} \,|\, \tilde{s}_t, \widetilde{a}_t\right) \doteq P\left(s_{t+1} \,|\, , s_t, a_t\right)\mathbb{1}_{o_{t+1}=o_t} \ ,$$

and reward function:

$$\bar{r}(\tilde{s}_t, \tilde{a}_t) = r(s_t, a_t) \ ,$$

we can also show that this formulation leads to the intra-option Bellman equations:

$$\begin{aligned}
\bar{v}_{\bar{\pi}}(\tilde{s}) &= \sum_{\tilde{a}'} \bar{\pi}\left(\tilde{a} \,|\, \tilde{s}\right)\left(\bar{r}(\tilde{s},\tilde{a}) + \sum_{\tilde{s}'} \bar{P}\left(\tilde{s}' \,\middle|\, \tilde{s}, \tilde{a}\right)\bar{v}_{\bar{\pi}}(\tilde{s}')\right) \\
&= \sum_{a} \pi\left(a \,|\, s, o\right)\Big(r(s,a) + \\
&\sum_{s'} P\left(s' \,\middle|\, , s, a\right)\sum_{k,o'}\left((1-\beta(s,o))\mathbb{1}_{k=\sharp}\mathbb{1}_{o'=o} + \beta(s,o)\mu\left(o' \,\middle|\, s\right)\mathbb{1}_{k=\flat}\right)\bar{v}_{\bar{\pi}}(\tilde{s}')\Big) \\
&= Q_{\mathcal{O}}(s,o)
\end{aligned}$$

This connection to the intra-option Bellman equations had not been recognized by Levy and Shimkin (2012). Hence, their proposed learning method had to rely to the explicit AHP representation of $\bar{\pi}$ which made the corresponding gradients more opaque. The approach that we put forward in chapter 5 is based instead on the construction of section 3.3 and has the advantage of exposing the underlying gradients directly. The raw form of our equations allows

for various algorithmic improvements regarding variance reduction methods and decoupling of the updates.

3.7.2 Adaptive Skills Adaptive Partitions (ASAP)

The ASAP framework of Mankowitz et al. (2016) is also related to the idea of representing options as binary vectors described in section 3.5. Despite not being cast explicitly in the options framework, the notion of *skill* in ASAP closely matches that of an option. The policy over such skills (the policy over options) is of a specific kind, namely that of a multivariate Bernoulli distribution where each component is independent. The probability of selecting a particular option is then $\mu\left(\mathbf{o} \mid s\right) = \prod_{i=0}^{k} P\left(\mathbf{o}(i) \mid s\right)$ where $\mathbf{o}(i)$ is the i-th component of the option vector $\mathbf{o}$, and $P\left(\mathbf{o}(i) \mid s\right)$ may be expressed in tabular form, or using function approximation (as the authors did). The authors view this binary option vector as a partitioning of the state space arising from the intersection of hyperplanes that each vector component specifies.

As in Feudal Networks Vezhnevets et al. (2017), ASAP has no explicit termination conditions and interprets a transition out of a partition as the termination of the option associated with it. Because it does not implement the call-and-return model, the boundaries between the partitions are smooth. It is as if the process would terminate, initiate and sample a new skills at every step, irrespective of the identity of the previous option. ASAP raises some interesting questions and potential research avenues regarding the interplay of state abstraction with temporal abstraction. Under which circumstances is it preferable to think of temporal abstraction as an induced property of specific state abstraction ? Are there specific kinds of state abstractions giving rise to temporal abstraction under the exact call-and-return semantics ?

3.7.3 Feudal Networks (FuN)

FuN (Vezhnevets et al., 2017) is a neural network architecture inspired by earlier work on *Feudal Reinforcement Learning* from Dayan and Hinton (1992). As in the options framework, FuN distinguishes between a high level *manager* and a low-level *worker* mirroring the structure of the options framework with

its policy over options and the options *below*. Due to architectural and algorithmic choices, the manager implicitly implements a Von Mises–Fisher distribution, a distribution over periodic random variables (MacKay, 2002). Viewed through the lens of the proposal put forward in section 3.5, this is as if option vectors would now be unit vectors representing directions on the unit hypersphere. The Von Mises-Fisher distribution being a *wrapped* analogue to the usual Gaussian distribution, it also has a *dispersion* parameter controlling the spread around a mean direction vector, the distance to which is measured by the cosine distance. In practice, this dispersion constant is not represented explicitly but rather appears implicitly depending on the ability of the workers to achieve the *goal* specified by the manager.

Temporal commitment to a certain behavior is also implicit in FuN because termination conditions are not part of the model. However, a similar effect is achieved indirectly through dilated LSTMs (Yu and Koltun, 2016), whose net effect is to decouple the update schedules for the manager and the worker. This is similar to the options framework and the distinctions between the decision stages at the SMDP levels, and those taking place at the *natural* level (Puterman, 1994) within the options themselves. Note that in section 5.1, we show that the policy gradient for the policy over options sees the termination function appearing as a multiplication factor which attenuates the gradient updates when the probability of continuing is high. The net effect is also to spread out (to *tick at a slower rate*) the updates for the policy over options from those of the options. In our problem setting, the timescale of those updates is learned, by virtue of learning termination conditions, instead of being fixed in advance.

Chapter 4

Unifying Multi-Step Methods

In section 3.4, we suggested that options may have something in common with multi-step temporal difference learning. Indeed, we have seen that the SMDP Q-learning algorithm can be understood as a special case of TD learning with n-step returns targets, where n is randomly distributed according to the number of steps that an option lasts. Furthermore, the form of the λ-return developed in section 2.1.2.1 shows a striking resemblance with the expression derived in lemma 2.1 linking the expected discount sum of rewards to an equivalent undiscounted random-horizon formulation. In this chapter, we show that these similarities are not a coincidence but rather the manifestation of a common structure which we explain using *matrix splitting* theory (Varga, 1962; Young and Rheinboldt, 1971; Puterman, 1994).

The starting point of our endeavour is a generalization of the evaluation equations which separates the sum of discounted rewards up to a certain stopping time from the value onward. This form has much in common with the idea of *bootstrapping* in reinforcement learning where we make an estimate of the return using a sum of sampled rewards, plus an approximation of the value on the last sampled state. We develop this connection by first studying the dynamic programming setting, and then later consider the *projected* variant in the context of linear function approximation.

4.1 n-step models

In section 2.1, we have seen that the policy evaluation equations can be written in matrix form as:

$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi \ .$$

This linear system is a statement regarding what the value function of a policy should satisfy: it should be such that applying the linear transformation on the right-hand side once more to $\mathbf{v}_\pi$ give us back $\mathbf{v}_\pi$. A natural question to then ask is whether there exist generalizations of that same statement which can also characterize $\mathbf{v}_\pi$. An obvious answer to that question simply comes from expanding $\mathbf{v}_\pi$ on the right-hand side of the evaluations equations to get:

$$\mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{r}_\pi + \gamma^2 \mathbf{P}_\pi^2 \mathbf{v}_\pi \ .$$

Hence, if $\mathbf{v}_\pi$ satisfies the usual policy evaluation equations, it also satisfies the above *two-steps* equations. That is, if we project the value function two steps ahead ($\gamma^2 \mathbf{P}_\pi^2 \mathbf{v}_\pi$) and add the expected reward over two steps, we get back $\mathbf{v}_\pi$. At this point, it is convenient to gather the terms arising from the expansion of the evaluation equations on the right-hand side and call them *models*: a two-steps reward model $\mathbf{b}_\pi^{(2)} \doteq \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{r}_\pi$ and a two-steps transition model $\mathbf{F}_\pi^{(2)} \doteq \gamma^2 \mathbf{P}_\pi^2$. More generally, expanding the right-hand side of the policy evaluation equations n times gives us n-step models.

Definition 4.1 (n-step models)**.** An n-step reward model for a policy π in an MDP is:

$$\mathbf{b}_\pi^{(n)} \doteq \sum_{t=0}^{n-1} (\gamma \mathbf{P}_\pi)^t \mathbf{r}_\pi \ ,$$

and its n-step transition model is:

$$\mathbf{F}_\pi^{(n)} \doteq (\gamma \mathbf{P}_\pi)^n \ .$$

An important property of n-step models is that they can be substituted into the policy evaluation equations without impacting the existence or uniqueness of the underlying value function $\mathbf{v}_\pi$. That is, $\mathbf{v}_\pi$ still holds under the n-step

policy evaluation equations:

$$\mathbf{v}_\pi = \sum_{t=0}^{n-1} (\gamma \mathbf{P}_\pi)^t \mathbf{r}_\pi + (\gamma \mathbf{P}_\pi)^n \mathbf{v}_\pi \ . \tag{4.1}$$

Lemma 4.2. *The solution $\mathbf{v}_\pi$ to the one-step policy evaluation equations is also the solution to the n-step equations:*

$$\mathbf{v}_\pi = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} \mathbf{r}_\pi = \left(\mathbf{I} - \mathbf{F}_\pi^{(n)}\right)^{-1} \mathbf{b}_\pi^{(n)} \ .$$

Proof. If $\mathbf{v}_\pi$ is the solution to the one-step policy evaluation equations, then we know that $(\mathbf{I} - \gamma \mathbf{P}_\pi)\mathbf{v}_\pi = \mathbf{r}_\pi$ and we can write:

$$\begin{aligned}
\left(\mathbf{I} - \mathbf{F}_\pi^{(n)}\right) \mathbf{v} &= \mathbf{b}_\pi^{(n)} \\
&= \left(\sum_{k=0}^{n-1} (\gamma \mathbf{P}_\pi)^k\right) \mathbf{r}_\pi \\
&= \left(\sum_{k=0}^{n-1} (\gamma \mathbf{P}_\pi)^k\right) (\mathbf{I} - \gamma \mathbf{P}_\pi)\, \mathbf{v}_\pi \ .
\end{aligned}$$

We establish that $\mathbf{v}_\pi = \mathbf{v}$ after having recognized that:

$$\begin{aligned}
\left(\sum_{k=0}^{n-1} (\gamma \mathbf{P}_\pi)^k\right) (\mathbf{I} - \gamma \mathbf{P}_\pi) &= \left(\sum_{k=0}^{n-1} (\gamma \mathbf{P}_\pi)^k\right) - \left(\sum_{k=1}^{n} (\gamma \mathbf{P}_\pi)^k\right) \\
&= \mathbf{I} - \gamma^n \mathbf{P}_\pi^n \\
&= \mathbf{I} - \mathbf{F}_\pi^{(n)} \ .
\end{aligned}$$

□

Instead of writing the n-step evaluation equations in matrix form, we can also write them more explicitly as an expectation:

$$\mathbf{v}_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{n-1} \gamma^t r(S_t, A_t) + \gamma^n \mathbf{v}_\pi(S_n) \,\middle|\, S_0 = s\right] \ .$$

In this expression, S_t, A_t and S_{t+1} denote random variables while n is a given constant. This raises the question: would there be evaluation equations where n is random variable ? The answer to that question was developed extensively

by van Nunen and Wessels (1976a); van Nunen (1976); Wessels (1977) through their notion of *stopping time*.

4.2 Generalized Policy Evaluation Equations

The form of the policy evaluation equations for the general case where the number of steps is random is:

$$\mathbf{v}_\pi(s) \doteq \mathbb{E}\left[\sum_{t=0}^{N-1} \gamma^t r(S_t, A_t) + \gamma^N \mathbf{v}_\pi(S_N) | S_0 = s\right] \quad , \tag{4.2}$$

and N is now a random variable denoting the number of steps up to a certain stopping time. van Nunen and Wessels (1976a) developed the theory for the general case where n can be history dependent (in which case stationary policies may not exist). In this chapter, we restrict our attention to the Markov case where N can only be a function of a Markov stopping time function.

To understand the process by which N is realized, it is useful (Shwartz, 2001) to extend the state space to indicate whether the unrolling of the sum $\sum_{t=0}^{N-1}$ in (4.2) continues or not. Therefore, we consider trajectories of the form $(\mathcal{S} \times \{\sharp, \flat\})^\infty$ where $\sharp$ denotes the event *continue* and $\flat$ means *terminate*. The probability associated to the event $\sharp$ (continue) is given by a function $\lambda : \mathcal{S} \to [0, 1]$ and similarly, $P(\flat \mid S_t) = 1 - \lambda(S_t)$ is the probability of terminating in state S_t. Our choice of notation is deliberate: the use of the symbol λ is meant to unify a variety of multi-step methods, including those based on the λ-return. In the following, we refer to the function λ as a *termination function* to streamline our terminology with that of van Nunen and Wessels (1976a); Sutton et al. (1999a) – although *continuation function* would be a more appropriate term.

In order to appropriately describe the augmented Markov process, we need to express the probability of transitioning to the next state and *continuing* upon entering it. In an MDP with policy π (without options, for the moment), we define the matrix $\mathbf{P}_{\pi,\sharp}$ containing that information as follows:

$$\mathbf{P}_{\pi,\sharp}(s, s') \doteq \sum_a \pi(a|s)\, P\left(s'|s, a\right) \lambda(s, s') = \mathbf{P}_\pi(s, s')\lambda(s, s') \quad .$$

The generalized policy evaluation equations (4.2) can then be written in matrix form as:

$$\mathbf{v}_\pi = \left(\sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_{\pi,\sharp}\right)^t \mathbf{r}_\pi \right) + \sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_{\pi,\sharp}\right)^t \left(\gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp}\right) \mathbf{v}_\pi \ . \tag{4.3}$$

Note that in this expression, the difference $\gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp}$ represents the probability of transitioning and terminating upon entering the next state. The first term involving the reward vector $\mathbf{r}_\pi$ is the expected sum of discounted rewards before termination, while the second term is the expectation of the value vector $\mathbf{v}_\pi$ upon termination.

4.3 λ-models

The equations described in (4.3) look very similar to the n-step evaluation equations obtained earlier in (4.1). This is because equations (4.3) are simply a *smooth* version of their n-step counterparts: smooth in the sense that they do not involve a *hard* cutoff on the number of steps. Such models were studied by Sutton (1995) who called them *beta*-models. For more cohesion with the rest of this chapter, we refer to them instead as "λ-models."

Definition 4.3 (λ-models)**.** Given a discounted MDP and stationary policy π, a λ-reward model is defined as:

$$\mathbf{b}_\pi^{(\lambda)} \doteq \sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_{\pi,\sharp}\right)^t \mathbf{r}_\pi \ .$$

Furthermore, we define a λ-transition model as:

$$\mathbf{F}_\pi^{(\lambda)} \doteq \sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_{\pi,\sharp}\right)^t \left(\gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp}\right) \ .$$

The generalized policy evaluation equations over the λ-models are then:

$$\mathbf{v}_\pi = \mathbf{b}_\pi^{(\lambda)} + \mathbf{F}_\pi^{(\lambda)} \mathbf{v}_\pi \ . \tag{4.4}$$

Given λ-models, the generalized policy evaluation equations can be solved using the two usual main approaches : in a direct fashion via matrix inversion,

or in an iterative fashion by switching to the operator viewpoint. This now leaves us with the problem of obtaining the models in first place.

Recognizing the familiar form of the Neumann series in definition 4.3, we now establish *evaluation equations* for the models themselves.

Proposition 4.4. *Given any termination function $\lambda : \mathcal{S} \times \mathcal{S} \to [0,1]$, if*

$$\rho(\gamma \mathbf{P}_{\pi,\sharp}) < 1 \ ,$$

the λ-models exist and satisfy:

$$\mathbf{b}_\pi^{(\lambda)} = \sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_{\pi,\sharp}\right)^t \mathbf{r}_\pi = \left(\mathbf{I} - \gamma \mathbf{P}_{\pi,\sharp}\right)^{-1} \mathbf{r}_\pi$$
$$\mathbf{F}_\pi^{(\lambda)} = \sum_{t=0}^{\infty} \left(\gamma \mathbf{P}_{\pi,\sharp}\right)^t \left(\gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp}\right) = \left(\mathbf{I} - \gamma \mathbf{P}_{\pi,\sharp}\right)^{-1} \left(\gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp}\right) \ .$$

Proof. This result follows directly from the fact that we have a Neumman series, whose convergence is determined by the spectral radius of the corresponding matrix. See (Puterman, 1994, corollary C.4) for a proof, as a well as section 2.1 for the usual policy evaluation equations. □

Proposition 4.4 suggests that given an MDP and a policy π, the corresponding λ-models can be computed directly via matrix inversion. As usual, it is preferable (for numerical stability) to solve for the vector $\mathbf{b}$ in

$$\left(\mathbf{I} - \gamma \mathbf{P}_{\pi,\sharp}\right) \mathbf{b} = \mathbf{r}_\pi \ ,$$

to get the λ-reward model $\mathbf{b}_\pi^{(\lambda)}$ and similarly, to solve for $\mathbf{F}$ in

$$\left(\mathbf{I} - \gamma \mathbf{P}_{\pi,\sharp}\right) \mathbf{F} = \gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp} \ ,$$

to get the λ-transition model $\mathbf{F}_\pi^{(\lambda)}$.

An iterative approach for computing the λ-models can also be obtained by writing them in a recursive form (as *Bellman equations* as we would often say informally). For the λ-reward model, this would correspond to :

$$\mathbf{b}_\pi^{(\lambda)} = \mathbf{r}_\pi + \gamma \mathbf{P}_{\pi,\sharp} \mathbf{b}_\pi^{(\lambda)} \ ,$$

or in component form :

$$\mathbf{b}_\pi^{(\lambda)}(s) = \sum_a \pi\left(a \mid s\right)\left(r(s,a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) \lambda(s,s') \mathbf{b}_\pi^{(\lambda)}(s')\right) \quad . \tag{4.5}$$

As for the λ-transition model, its corresponding recursive expression is:

$$\mathbf{F}_\pi^{(\lambda)} = \left(\gamma \mathbf{P}_\pi - \gamma \mathbf{P}_{\pi,\sharp}\right) + \gamma \mathbf{P}_{\pi,\sharp} \mathbf{F}_\pi^{(\lambda)} \quad .$$

The structure of this equation is perhaps clearer when writing it in component form for a pair (s, s') of $\mathbf{F}_\pi$:

$$\mathbf{F}_\pi^{(\lambda)}(s,s') \tag{4.6}$$

$$= \gamma \sum_a \pi\left(a \mid s\right)\left(P\left(s' \mid s, a\right)\left(1 - \lambda(s,s')\right) + \sum_{\bar{s}} P\left(\bar{s} \mid s, a\right) \lambda(s,\bar{s}) \mathbf{F}_\pi^{(\lambda)}(\bar{s}, s')\right) \tag{4.7}$$

$$= \gamma \sum_a \pi\left(a \mid s\right) \sum_{\bar{s}} P\left(\bar{s} \mid s, a\right)\left(\left(1 - \lambda(s,\bar{s})\right) \mathbb{1}_{\bar{s}=s'} + \lambda(s,\bar{s}) \mathbf{F}_\pi^{(\lambda)}(\bar{s}, s')\right) \quad . \tag{4.8}$$

With these recursive equations in place, we finish our tour of the evaluation equations by recognizing that λ-models are *just* General Value Functions (GVFs) (Sutton et al., 2011). In fact, another way to write a λ-reward model is:

$$\begin{aligned}
\mathbf{b}_\pi^{(\lambda)}(s) &= \mathbb{E}\left[r(S_0, A_0) + \gamma\lambda(S_0, S_1)\mathbf{b}_\pi^{(\lambda)}(S_1) \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[r(S_0, A_0) + \gamma\lambda(S_0, S_1)\left(r(S_1, A_1) + \gamma\lambda(S_1, S_2)\mathbf{b}_\pi^{(\lambda)}(S_2)\right) \,\middle|\, S_0 = s\right] \\
&= \mathbb{E}\left[\sum_{t=0}^{\infty} r(S_t, A_t) \prod_{k=0}^{t-1} \gamma\lambda(S_k, S_{k+1}) \,\middle|\, S_0 = s\right] \quad ,
\end{aligned}$$

with the convention that $\prod_i^j = 1$ for $j < i$. Using the GVF terminology, the "reward" term is called a *cumulant* and $\gamma\lambda$ is its corresponding termination function. Note that $\mathbf{b}_\pi^{(\lambda)}$ is nothing more than the value function for the policy π in an MDP where the *discount factor* (or its random horizon) is given by $\gamma\lambda$: a generalization of the result shown in lemma 2.1. Using the same reasoning, $\mathbf{F}_\pi^{(\lambda)}$ simply corresponds to the Successor Representation (SR) (section 2.3.1) under an MDP whose discount factor is $\gamma\lambda$. This means that the λ-transition

model can also be written as :

$$\begin{aligned}\mathbf{F}_\pi^{(\lambda)}(s,s') &= \mathbb{E}\left[\gamma(1-\lambda(S_0,S_1))\mathbb{1}_{S_1=s'} + \gamma\lambda(S_0,S_1)\mathbf{F}_\pi^{(\lambda)}(S_1,s') \,\middle|\, S_0=s\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma(1-\lambda(S_t,S_{t+1}))\mathbb{1}_{S_{t+1}=s'}\prod_{k=0}^{t-1}\gamma\lambda(S_k,S_{k+1})|S_0=s\right] \quad . \quad (4.9)\end{aligned}$$

The term $\gamma(1-\lambda(S_t,S_{t+1}))\mathbb{1}_{S_{t+1}=s'}$ here plays the role of what would be the *reward function* in the usual policy evaluation equations. It would also be called the *cumulant* of the corresponding GVF with the same termination function $\gamma\lambda$ as in the GVF representation of the λ-reward model. While we choose to write (4.9) in component form for each (s,s') pairs, it does not mean that we have to allocate one GVF *deamon* per s' (column). In fact, the same comment that we made for the SR in section 2.3.1 applies also here: nothing prevents us from using vector-valued *rewards/cumulant*. This is because the existence of a solution to the evaluation equations depends only on the properties of the matrix $\gamma\mathbf{P}_{\pi,\sharp}$, and as long as the "rewards" are bounded: whether we have a vector or matrix "reward" term on the right-hand side of proposition 4.4 is irrelevant.

4.4 Matrix Splitting

We now go back to the generalized policy evaluation equations (4.3) but write them in terms of the closed-form solutions for the λ-models:

$$\mathbf{v}_\pi = \left(\mathbf{I}-\gamma\mathbf{P}_{\pi,\sharp}\right)^{-1}\mathbf{r}_\pi + \left(\mathbf{I}-\gamma\mathbf{P}_{\pi,\sharp}\right)^{-1}\left(\gamma\mathbf{P}_\pi - \gamma\mathbf{P}_{\pi,\sharp}\right)\mathbf{v}_\pi \quad .$$

Let us also rename some matrices in this equations for more clarity, namely :

$$\mathbf{M}_{\pi,\sharp} \doteq \mathbf{I}-\gamma\mathbf{P}_{\pi,\sharp} \qquad \text{and} \qquad \mathbf{N}_{\pi,\flat} \doteq \gamma\mathbf{P}_\pi - \gamma\mathbf{P}_{\pi,\sharp} \quad .$$

The generalized policy evaluations (4.3) then become:

$$\mathbf{v}_\pi = \mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_\pi + \mathbf{M}_{\pi,\sharp}^{-1}\mathbf{N}_{\pi,\flat}\mathbf{v}_\pi \quad .$$

A key observation to make is that $\mathbf{M}_{\pi,\sharp}$ and $\mathbf{N}_{\pi,\flat}$ are related in a specific way:

$$\mathbf{M}_{\pi,\sharp} - \mathbf{N}_{\pi,\flat} = \mathbf{I} - \gamma \mathbf{P}_\pi \ .$$

It turns out that this representation of $\mathbf{I} - \gamma \mathbf{P}_\pi$ as a difference of two other matrices corresponds to the notion of *matrix splitting* extensively developed by Varga (1960, 1962) but originating from Keller (1958) [1]. Matrix splitting methods were initially devised as a way to speed up iterative solvers for partial differential equations. The scope of this approach was then extended to general linear systems of equations in the seminal textbook on iterative methods by Varga (1962). Porteus (1975) then used Varga's results on the convergence rate of matrix splitting methods to motivate the usefulness of his *pre-inverse transform* for solutions to MDPs.

Let us take a moment to contemplate how we got here. Going back to the beginning of this chapter, we started with a generalization of the policy evaluation equations based on the notion of stopping time. We then derived a matrix representation of those equations, and showed that they can be described using what we called λ-models. These λ-models were then shown to have evaluation equations (*Bellman equations*) of their own and it is when we wrote their closed-form solution in the generalized policy evaluations equations that we found a matrix splitting. But why should we care about these generalized equations in the first place ? The matrix splitting perspective is clear on that account : to speed up our solution methods. At this point, the answer as to why this should be the case may not be surprising since the generalized policy evaluation equations are of the form:

$$\mathbf{v}_\pi = \mathbf{b}_\pi^{(\lambda)} + \mathbf{F}_\pi^{(\lambda)} \mathbf{v}_\pi \ ,$$

and we know that the rate of convergence of the corresponding sequence of iterates:

$$\mathbf{v}_{k+1} = \mathbf{b}_\pi^{(\lambda)} + \mathbf{F}_\pi^{(\lambda)} \mathbf{v}_k \ , \tag{4.10}$$

depends on the spectral radius $\rho(\mathbf{F}_\pi^{(\lambda)}) = \rho(\mathbf{M}_{\pi,\sharp}^{-1} \mathbf{N}_{\pi,\flat})$. For example, consider one extreme with $\lambda(s, s') = 1 \, \forall s, s' \in \mathcal{S}$ such that the augmented process (over states and termination events) of section 4.2 *continues* everywhere. The

[1] According to Varga (1960)

sequence (4.10) corresponding to $\lambda(s, s') = 1$ is then:

$$\mathbf{v}_{k+1} = (\mathbf{I} - \gamma \mathbf{P}_\pi)^{-1} \mathbf{r}_\pi = \mathbf{v}_\pi \ . \tag{4.11}$$

This means that in a single step, the corresponding *iterative* solver would have already computed the solution $\mathbf{v}_\pi$ (note $\mathbf{v}_k$ does not even appear on the right-hand side of (4.11)). From a practical point of view, it may not be advantageous to go this far along the spectrum induced by λ because forming the λ-models would be just as *hard* as solving the original problem in the first place. On the other hand, if $\lambda(s, s') = 0 \, \forall s, s' \in \mathcal{S}$ the resulting iterations are:

$$\mathbf{v}_{k+1} = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_k$$

and we are back to the usual one-step policy evaluation equations (2.2) without having gained any speedup.

It is clear from the previous examples that the extremes points (where λ *continues* everywhere or when it *terminates* everywhere) preserve $\mathbf{v}_\pi$ as the final answer of our policy evaluation algorithms: a *consistency* property (Young and Rheinboldt, 1971). In the following, we provide a λ counterpart to lemma 4.2, previously introduced for the n-step case.

Lemma 4.5 (Consistency)**.** *The value function $\mathbf{v}_\pi$ of policy π in the given MDP is the unique solution to the generalized policy evaluation equations for any termination function $\lambda : \mathcal{S} \times \mathcal{S} \to [0, 1]$ and $\gamma \in [0, 1)$. In other words, $\mathbf{v} = \mathbf{v}_\pi$ is the unique solution to the following linear system :*

$$\left(\mathbf{I} - \mathbf{F}_\pi^{(\lambda)}\right) \mathbf{v} = \mathbf{b}_\pi^{(\lambda)} \ .$$

Proof. Using the matrix splitting notation:

$$\left(\mathbf{I} - \mathbf{M}_{\pi,\sharp}^{-1} \mathbf{N}_{\pi,\flat}\right) \mathbf{v} = \mathbf{M}_{\pi,\sharp}^{-1} \mathbf{r}_\pi \ .$$

Multiplying by $\mathbf{M}_{\pi,\sharp}$ on both sides:

$$\left(\mathbf{M}_{\pi,\sharp} - \mathbf{N}_{\pi,\flat}\right) \mathbf{v} = \mathbf{r}_\pi \ .$$

Because $\mathbf{M}_{\pi,\sharp} - \mathbf{N}_{\pi,\flat}$ is matrix splitting for $\mathbf{I} - \gamma\mathbf{P}_\pi$, the last equation simplifies to:

$$(\mathbf{I} - \gamma\mathbf{P}_\pi)\,\mathbf{v} = \mathbf{r}_\pi \ ,$$

whose solution is $\mathbf{v}_\pi$. □

Using a similar matrix splitting argument, we can also establish the consistency properties of our generalized Bellman operator in the control case.

Lemma 4.6 (Consistency in the control case)**.** *The generalized Bellman optimality equations are consistent with the optimal value function* $\mathbf{v}^\star$*:*

$$\mathbf{v}^\star = \max_{\pi \in \Pi^{MD}} \left(\mathbf{b}_\pi^{(\lambda)} + \mathbf{F}_\pi^{(\lambda)} \mathbf{v}_\pi \right) \ .$$

Proof. The optimal value function is such that for any policy π, $\mathbf{v}^\star \geq \mathbf{v}_\pi$. Using the closed-form expression for generalized policy evaluation equations, it follows that:

$$\begin{aligned}
\mathbf{v}^\star &\geq \left(\mathbf{I} - \gamma\mathbf{F}_\pi^{(\lambda)} \right)^{-1} \mathbf{b}_\pi^{(\lambda)} \\
&= \left(\mathbf{M}_{\pi,\sharp}^{-1} \left(\mathbf{M}_{\pi,\sharp} - \mathbf{N}_{\pi,\flat} \right) \right)^{-1} \mathbf{M}_{\pi,\sharp}^{-1} \mathbf{r}_\pi \\
&= \left(\mathbf{I} - \gamma\mathbf{P}_\pi \right)^{-1} \mathbf{r}_\pi \\
&= \mathbf{v}_\pi \ .
\end{aligned}$$

□

Historically, the development of matrix splitting methods for iterative solvers came jointly with the idea of matrix preconditioning. In fact, any matrix splitting specifies a choice of preconditioner (Chen, 2005), namely the matrix $\mathbf{M}_{\pi,\sharp}$ itself. Instead of solving linear systems by an iterative successive approximation approach, matrix preconditioning methods are often based on more *direct* methods. The matrix preconditioner is then be applied (most of the time *pre-multiplied*) apriori to the coefficient matrix to obtain a transformed linear system of equations which hopefully is easier to solve directly. The preconditioning form corresponding to generalized policy evaluation equations and

their matrix splitting is :

$$\begin{aligned} \left(\mathbf{I} - \mathbf{M}_{\pi,\sharp}^{-1}\mathbf{N}_{\pi,\flat}\right)\mathbf{v}_\pi &= \mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_\pi \\ \iff \mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{I} - \gamma\mathbf{P}_\pi\right)\mathbf{v}_\pi &= \mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_\pi \end{aligned} \tag{4.12}$$

We see here that $\mathbf{M}_{\pi,\sharp}^{-1}$ pre-multiplies $\mathbf{I} - \gamma\mathbf{P}_\pi$ which is the *coefficient matrix* appearing in the basic policy evaluation equations (2.2). To understand how the resulting linear system of equations can be easier to solve, consider the extreme case where $\lambda(s, s') = 1\,\forall s, s' \in \mathcal{S}$ thereby *continuing* everywhere. The left-hand side of (4.12) is then $\mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{I} - \gamma\mathbf{P}_\pi\right) = \mathbf{I}$ and the solution to the linear system of equation can be read from the right-hand side of the equation $\mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_\pi = \left(\mathbf{I} - \gamma\mathbf{P}_\pi\right)\mathbf{r}_\pi = \mathbf{v}_\pi$. This choice of preconditioner is not very interesting because forming it is at least as expensive as solving the original problem in the first place. In general, the design of a good matrix preconditioner (or equivalent of a matrix splitting) must find a balance between the computational cost for obtaining it versus the effort involved in solving the original problem (Golub and Van Loan, 1996). An effective way to reduce the computational expense associated with the construction of a preconditioner is to *amortize* its cost over many different problems (Golub and Van Loan, 1996; Hackbusch, 2016): ie. for many different right-hand sides (reward vector in our case) but for the same coefficient matrix (the same transition matrix). We made a similar point regarding the usefulness of the successor representation in chapter 2.3, which as we will see in the next section, corresponds to the extreme case $\lambda = 1$. A similar case can also be made in favor of constructing good options, which also admit a matrix splitting/preconditioning form.

4.5 Generalized Projected Evaluation Equations

Generalized policy evaluation equations can also be defined in the context of linear function approximation. As in section 2.1.2.2, we assume that we are given a feature matrix $\mathbf{\Phi} \in \mathbb{R}^{\mathcal{S}\times k}$ with the goal of finding a parameter vector $\mathbf{w} \in \mathbb{R}^k$ such that $\widehat{\mathbf{v}}_\pi \doteq \mathbf{\Phi}\mathbf{w}$ is as close as possible to the true $\mathbf{v}_\pi$. Using our previous notation, we have a projection operator $\mathbf{\Pi} \doteq \mathbf{\Phi}\left(\mathbf{\Phi}^\top\mathbf{\Xi}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^\top\mathbf{\Xi}$, where $\mathbf{\Xi} \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ is a diagonal matrix containing the stationary distribution $\mathbf{d}_\pi$.

Definition 4.7. The Generalized Projected policy evaluation equations are :

$$\mathbf{\Phi w} = \mathbf{\Pi}\left(\mathbf{b}_{\pi}^{(\lambda)} + \mathbf{F}_{\pi}^{(\lambda)}\mathbf{\Phi w}\right) \quad . \tag{4.13}$$

The projected equations given in the above definition can be re-arranged as an equivalent linear system of equations whose *unknown variables* are the parameters $\mathbf{w}$.

Lemma 4.8. *The solution* $\mathbf{w}$ *to the generalized projected policy evaluation equations satisfies the following linear system:*

$$\left(\mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{I}-\gamma\mathbf{P}_{\pi}\right)\mathbf{\Phi}\right)\mathbf{w} = \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_{\pi} \quad . \tag{4.14}$$

Proof. Multiplying by $\mathbf{\Phi}^{\top}\mathbf{\Xi}$ on both sides of (4.13), we get:

$$\begin{aligned}
\mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{\Phi w} &= \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{b}_{\pi}^{(\lambda)} + \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{F}_{\pi}^{(\lambda)}\mathbf{\Phi w} \\
\left(\mathbf{\Phi}^{\top}\mathbf{\Xi}\left(\mathbf{I}-\mathbf{F}_{\pi}^{(\lambda)}\right)\mathbf{\Phi}\right)\mathbf{w} &= \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{b}_{\pi}^{(\lambda)} \\
\left(\mathbf{\Phi}^{\top}\mathbf{\Xi}\left(\mathbf{I}-\mathbf{M}_{\pi,\sharp}^{-1}\mathbf{N}_{\pi,\flat}\right)\mathbf{\Phi}\right)\mathbf{w} &= \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_{\pi}
\end{aligned}$$

We then use the fact that we have a matrix splitting:

$$\mathbf{I}-\mathbf{M}_{\pi,\sharp}^{-1}\mathbf{N}_{\pi,\flat} = \mathbf{I}-\mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{M}_{\pi,\sharp}-\left(\mathbf{I}-\gamma\mathbf{P}_{\pi}\right)\right) = \mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{I}-\gamma\mathbf{P}_{\pi}\right) \quad ,$$

which leads us to the desired result:

$$\left(\mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{I}-\gamma\mathbf{P}_{\pi}\right)\mathbf{\Phi}\right)\mathbf{w} = \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{M}_{\pi,\sharp}^{-1}\mathbf{r}_{\pi} \quad .$$

□

To show that the generalized projected equations have a solution, we could use the same strategy as in section 2.1.2.2 by first establishing the contraction property for the generalized policy evaluation operator in combination with the fact that $\mathbf{\Phi}$ is a nonexpansion. Another common approach (Sutton, 1988; Tsitsiklis and Roy, 1997a; Sutton, 2015b; Sutton and Barto, 2018) based on the ODE perspective (Benveniste et al., 1990; Kushner and Yin, 2003) is to establish that the *key matrix* (Sutton, 1988) $\mathbf{A} \doteq \mathbf{\Phi}^{\top}\mathbf{\Xi}\mathbf{M}_{\pi,\sharp}^{-1}\left(\mathbf{I}-\gamma\mathbf{P}_{\pi}\right)\mathbf{\Phi}$ in (4.14) is positive definite. However, in general $\mathbf{A}$ need not be positive definite for arbitrary

matrix splitting. For example, we have shown in Touati et al. (2018) that the matrix splitting corresponding to the multi-step TD method algorithm called *tree-backup* (Precup et al., 2000) does not satisfy this constraint.

4.6 Matrix Splitting for Options

In light of section 4.3, option models can simply be seen as a specific kind of λ-models. In fact, we saw in section 3.1 that option models can be written as:

$$\mathbf{b}_{\pi,o}^{(\beta)}(s) = \sum_a \pi\left(a \,|\, s, o\right) \left(r(s,a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) \left(1 - \beta(s', o)\right) \mathbf{b}_{\pi,o}^{(\beta)}(s') \right)$$

for the reward model $\mathbf{b}_{\pi,o}^{(\beta)}$ of option o and:

$$\mathbf{F}_{\pi,o}^{(\beta)}(s,s') = \gamma \sum_a \pi\left(a \,|\, s, o\right) \sum_{\bar{s}} P\left(\bar{s} \,|\, s, a\right) \left(\beta(\bar{s}, o) \mathbb{1}_{s'=\bar{s}} + \left(1 - \beta(\bar{s}, o)\right) \mathbf{F}_{\pi,o}^{(\beta)}(\bar{s}, s') \right) \quad ,$$

for its transition model. These equations are exactly the ones found in section (4.5) and (4.8) but where we replace π by π_o, and $(1 - \lambda(s,s'))$ by $\beta(s', o)$. It then follows through proposition 4.4 that the closed-form expression for the reward model of an option is:

$$\mathbf{b}_{\pi,o}^{(\beta)} = \left(\mathbf{I} - \gamma \mathbf{P}_{\pi,o,\sharp}\right)^{-1} \mathbf{r}_{\pi,o} \quad ,$$

where we define $\mathbf{r}_{\pi,o} \in \mathbb{R}^{|\mathcal{S}|}$ and $\mathbf{P}_{\pi,o,\sharp} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ as:

$$\mathbf{r}_{\pi,o}(s) \doteq \sum_a \pi\left(a \,|\, s, o\right) r(s,a), \;\; \mathbf{P}_{\pi,o,\sharp} \doteq \sum_a \pi\left(a \,|\, s, o\right) P\left(s' \,\middle|\, s, a\right) \left(1 - \beta(s', o)\right) \quad .$$

Similarly, the transition model of an option is given by:

$$\mathbf{F}_{\pi,o}^{(\beta)} = \left(\mathbf{I} - \gamma \mathbf{P}_{\pi,o,\sharp}\right)^{-1} \left(\gamma \mathbf{P}_{\pi,o} - \gamma \mathbf{P}_{\pi,o,\sharp}\right) \quad ,$$

where $\mathbf{P}_{\pi,o} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ is:

$$\mathbf{P}_{\pi,o}(s,s') \doteq \sum_a \pi\left(a \,|\, s, o\right) P\left(s' \,\middle|\, s, a\right) \quad .$$

Borrowing the matrix splitting notation, the reward model $\mathbf{b}_{\pi,o}^{(\beta)}$ and the transition model $\mathbf{F}_{\pi,o}^{(\beta)}$ can be written as follows:

$$\mathbf{b}_{\pi,o}^{(\beta)} = \mathbf{M}_{\pi,o,\sharp}^{-1}\mathbf{r}_{\pi,o} \qquad \text{and} \qquad \mathbf{F}_{\pi,o}^{(\beta)} = \mathbf{M}_{\pi,o,\sharp}^{-1}\mathbf{N}_{\pi,o,\flat} \ ,$$

where $\mathbf{M}_{\pi,o,\sharp}^{-1} \doteq \mathbf{I} - \gamma\mathbf{P}_{\pi,o,\sharp}$ and $\mathbf{N}_{\pi,o,\flat} \doteq \gamma\mathbf{P}_{\pi,o} - \gamma\mathbf{P}_{\pi,o,\sharp}$. Having found this matrix splitting, the burning question is: what are the policy evaluation equations associated with them ? According to the template of the generalized policy evaluation equations in (4.4), we have:

$$\begin{aligned} \mathbf{v}_{\pi,o} &= \mathbf{b}_{\pi,o}^{(\beta)} + \mathbf{F}_{\pi,o}^{(\beta)}\mathbf{v}_{\pi,o} \\ &= \mathbf{M}_{\pi,o,\sharp}^{-1}\mathbf{r}_{\pi,o} + \mathbf{M}_{\pi,o,\sharp}^{-1}\mathbf{N}_{\pi,o,\flat}\mathbf{v}_{\pi,o} \\ &= \left(\mathbf{I} - \gamma\mathbf{P}_{\pi,o,\sharp}\right)^{-1}\mathbf{r}_{\pi,o} + \left(\mathbf{I} - \gamma\mathbf{P}_{\pi,o,\sharp}\right)^{-1}\left(\gamma\mathbf{P}_{\pi,o} - \gamma\mathbf{P}_{\pi,o,\sharp}\right)\mathbf{v}_{\pi,o} \ . \end{aligned} \tag{4.15}$$

It is crucial to realize that these equations only describe the value function for executing the policy of an option in a discounted MDP, hence : $\mathbf{v}_{\pi,o}(s) \neq \mathbf{Q}_{\mathcal{O}}(s,o)$. In fact, each value function $\mathbf{v}_{\pi,o}$ represents the cumulative discounted rewards of the associated option policy in isolation from the rest of the system. They do not involve the call-and-return execution model, nor any other continuation strategy for picking the *next option*. Furthermore, by the consistency property, the solution $\mathbf{v}_{\pi,o}$ is left with no trace of the matrix splitting used to find it, including the termination condition β itself. In other words, if we were to solve for an unknown value vector $\mathbf{v} \in \mathbb{R}^{|\mathcal{S}|}$ in the following generalized policy evaluation, we would get:

$$\begin{aligned} \left(\mathbf{I} - \mathbf{M}_{\pi,o,\sharp}^{-1}\mathbf{N}_{\pi,o,\flat}\right)\mathbf{v} &= \mathbf{M}_{\pi,o,\sharp}^{-1}\mathbf{r}_{\pi,o} \\ \left(\mathbf{M}_{\pi,o,\sharp} - \mathbf{N}_{\pi,o,\flat}\right)\mathbf{v} &= \mathbf{r}_{\pi,o} \\ \iff \left(\mathbf{I} - \gamma\mathbf{P}_{\pi,o}\right)\mathbf{v} &= \mathbf{r}_{\pi,o} \ . \end{aligned}$$

This means that $\mathbf{v} = \left(\mathbf{I} - \gamma\mathbf{P}_{\pi,o}\right)^{-1}\mathbf{r}_{\pi,o} = \mathbf{v}_{\pi,o}$.

So is there any relation between the option-level matrix splitting and the value function over options $\mathbf{v}_{\mathcal{O}}$? One way to answer this question is by defining a

block matrix $\mathbf{F}_{\mathcal{O}} \in \mathbb{R}^{(|\mathcal{S}|\times|\mathcal{O}|)\times|\mathcal{S}|}$ and block vector $\mathbf{b}_{\mathcal{O}} \in \mathbb{R}^{(|\mathcal{S}||\mathcal{O}|)\times 1}$ by *concatenating* the option models vertically. Visually, these matrices look like:

$$\mathbf{b}_{\mathcal{O}} = \begin{bmatrix} \mathbf{b}^{(\beta)}_{\pi,o_1} \\ \vdots \\ \mathbf{b}^{(\beta)}_{\pi,|\mathcal{O}|} \end{bmatrix}, \qquad \text{and} \qquad \mathbf{F}_{\mathcal{O}} = \begin{bmatrix} \mathbf{F}^{(\beta)}_{\pi,o_1} \\ \vdots \\ \mathbf{F}^{(\beta)}_{\pi,|\mathcal{O}|} \end{bmatrix}$$

The value function over options, taking into account the call-and-return execution model is then:

$$\begin{aligned} \mathbf{v}_{\mathcal{O}}(s) &= \sum_o \mu\left(o \,|\, s\right) \left(\mathbf{b}(s,o) + \sum_{s'} \mathbf{F}_{\mathcal{O}}(s,o,s') \mathbf{v}_{\mathcal{O}}(s') \right) \\ &= \sum_o \mu\left(o \,|\, s\right) \left(\mathbf{b}^{(\beta)}_{\pi,o}(s) + \sum_{s'} \mathbf{F}^{(\beta)}_{\pi,o}(s,s') \mathbf{v}_{\mathcal{O}}(s') \right) \\ &= \sum_o \mu\left(o \,|\, s\right) \left(\mathbf{b}^{(\beta)}_{\pi,o} + \mathbf{F}^{(\beta)}_{\pi,o} \mathbf{v}_{\mathcal{O}} \right)(s) \ . \end{aligned} \tag{4.16}$$

The main difference with the previous expression for $\mathbf{v}_{\pi,o}$ in (4.15) is that the linear map $\mathbf{b}_{\pi,o} + \mathbf{F}^{(\beta)}_{\pi,o}$ inside the parenthesis is backing up values according to the call-and-return execution model using $\mathbf{v}_{\mathcal{O}}$ rather than $\mathbf{v}_{\pi,o}$. This subtle difference hints at interesting variants on the Bellman equations for options involving different levels of coupling. At one end of the spectrum, we have equations of the form (4.15) which are completely decoupled from each other, and (4.16) at the opposite end where all options are coupled via the policy over options. The decoupling brought by (4.15) would certainly help the implementation of asynchronous updates of the kind described by Bertsekas (2012) in the dynamic programming setting. This could lead to new execution models or more *implicit* forms of planning – methods where the result of all local updates (potentially asynchronous) mimics or subsumes the explicit planning model for options from chapter 3.

4.6.1 Polling Execution

A simple form for the value function over options is obtained when using *polling execution*. This mode of execution seems to have originated from Kaelbling (1993), and was later incorporated into Dietterich's MAXQ framework (Dietterich, 2000) as well as in Ryan (2004). The options paper (Sutton et al.,

1999a) also mentions this alternative in the context of *interruption execution*, which allows for switching to another option at every step. The following results were first developed in Bacon and Precup (2016) while having mistakenly forgotten the commitment constraint coming from the call-and-return model. After having recognized this discrepancy, the updated results were used in Harutyunyan et al. (2018).

By virtue of the fact that we do not have to commit to the same option, it is possible under polling execution to flatten the distribution over primitive actions. That is, we can now define a new policy $\sigma : \mathcal{S} \to Dist(\mathcal{A})$ coupling μ and π by averaging out the choice of options:

$$\sigma\left(a \mid s\right) \doteq \sum_{o} \mu\left(o \mid s\right) \pi\left(a \mid s, o\right)$$

Hence, the transition matrix under σ is now:

$$\mathbf{P}_{\sigma}(s, s') = \sum_{a} \sigma\left(a \mid s\right) P\left(s' \,\middle|\, s, a\right) \quad ,$$

and the immediate average reward vector is:

$$\mathbf{r}_{\sigma} = \sum_{a} \sigma\left(a \mid s\right) r(s, a) \quad .$$

It follows that the evaluation equations for the policy σ are:

$$\mathbf{v}_{\sigma} = \mathbf{r}_{\sigma} + \mathbf{P}_{\sigma}\mathbf{v}_{\sigma} \quad .$$

At this point, we have expressed the policy over options μ and the option policy π in our equations but the temporal aspect due to the termination condition is still missing. What we are now after is some splitting matrix $\mathbf{M}_{\sigma,\sharp}$ that implements a notion of stopping time using termination conditions. The natural candidate is:

$$\mathbf{M}_{\sigma,\sharp} \doteq \mathbf{I} - \gamma \mathbf{P}_{\sigma,\sharp} \quad ,$$

where $\mathbf{P}_{\sigma,\sharp} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}, \mathbf{P}_{\sigma,\sharp}(s, s') = \sum_{o} \mu\left(o \mid s\right) \sum_{a} \pi\left(a \mid s, o\right) \left(1 - \beta(s', o)\right)$. The generalized policy evaluation equations corresponding to $\mathbf{v}_{\sigma}$ are then:

$$\mathbf{v}_{\sigma} = \mathbf{M}_{\sigma,\sharp}^{-1}\mathbf{r}_{\sigma} + \mathbf{M}_{\sigma,\sharp}^{-1}\mathbf{N}_{\sigma,\flat}\mathbf{v}_{\sigma} \quad , \tag{4.17}$$

and $\mathbf{N}_{\sigma,\flat}$ is the usual splitting matrix which contains the termination probabilities:

$$\mathbf{N}_{\sigma,\flat} \doteq \gamma \mathbf{P}_{\sigma} - \gamma \mathbf{P}_{\sigma,\sharp} \quad .$$

In component form, the generalized equations for σ for (4.17) are:

$$\mathbf{v}_{\sigma}(s) = \sum_{o} \mu\left(o \,|\, s\right) \sum_{a} \pi\left(a \,|\, s, o\right) \Big(r(s,a) + \tag{4.18}$$

$$\gamma \sum_{s'} P\left(s' \,\big|\, s, a\right) \sum_{o'} \left((1 - \beta(s', o)) \mu\left(o' \,\big|\, s'\right) + \beta(s', o) \mu\left(o' \,\big|\, s'\right) \right) \mathbf{Q}_{\sigma}(s', o') \Big) \quad , \tag{4.19}$$

and where the *mixture term* simplifies to:

$$(1 - \beta(s', o)) \mu\left(o' \,\big|\, s'\right) + \beta(s', o) \mu\left(o' \,\big|\, s'\right) = \mu\left(o' \,\big|\, s'\right) \quad . \tag{4.20}$$

We made a deliberate choice to write (4.19) as a mixture of two identical components for ease of comparison with the policy evaluation equations for Markov options in the call-and-return execution model. As shown in (3.9) for the call-and-return execution, the inner sum over next options is instead:

$$\left(1 - \beta(s', o)\right) \mathbb{1}_{o'=o} + \beta(s', o) \mu\left(o' \,\big|\, s'\right) = P\left(o' \,\big|\, s, o\right) \quad .$$

It is the inclusion of the indicator $\mathbb{1}_{o'=o}$ as opposed to $\mu\left(o' \,|\, s'\right)$ which differentiates the two models of execution. In the polling execution model, the "mixture" (see section 3.5) is over the exact same two components $\mu\left(o' \,|\, s'\right)$, while in call-and-return, the policy over options is mixed with the degenerate distribution located on the previous options. Hence, despite the fact that the trivial "mixture" (4.20) has a dependence on the previous option o on the left-hand side, this dependence eventually goes away and we are simply left with $\mu\left(o' \,|\, s'\right)$; in the call-and-return model, o needs to be kept around to evaluate $P\left(o' \,|\, s, o\right)$. This is why an equivalent flat policy over primitive actions is possible in the polling model, but not for call-and-return.

The equivalence with the marginal policy σ can be leveraged to evaluate an arbitrary stationary policy over actions, but assisted with options and a policy over them. The idea here is that if we can find a way to *embed* a target policy π that we wish to evaluate inside σ so that $\pi\left(a \,|\, s\right) = \sigma\left(a \,|\, s\right) \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$,

then the consistency property of matrix splitting methods guarantees that $\mathbf{v}_\pi$ can be recovered. As we showed in lemma 4.5 for λ-models, we now establish the consistency of the matrix splitting method associated with $\mathbf{M}_{\sigma,\sharp}$.

Lemma 4.9 (Consistency of Matrix Splitting for Polling Execution)**.** *The value function* $\mathbf{v}_\sigma = (\mathbf{I} - \gamma \mathbf{P}_\sigma)^{-1} \mathbf{r}_\sigma$ *also satisfies the following evaluation equations:*

$$\mathbf{v}_\sigma = \mathbf{M}_{\sigma,\sharp}^{-1} \mathbf{r}_\sigma + \mathbf{M}_{\sigma,\sharp}^{-1} \mathbf{N}_{\sigma,\flat} \mathbf{v}_\sigma \ .$$

Proof. Because $\mathbf{M}_{\sigma,\sharp} - \mathbf{N}_{\sigma,\flat} = \mathbf{I} - \gamma \mathbf{P}_\sigma$, solving for $\mathbf{v}$ in:

$$\left(\mathbf{I} - \mathbf{M}_{\sigma,\sharp}^{-1} \mathbf{N}_{\sigma,\flat}\right) \mathbf{v} = \mathbf{M}_{\sigma,\sharp}^{-1} \mathbf{r}_\sigma \ ,$$

is equivalent to:

$$\left(\mathbf{I} - \gamma \mathbf{P}_\sigma\right) \mathbf{v} = \mathbf{r}_\sigma \ .$$

Hence $\mathbf{v} = \mathbf{v}_\sigma$. □

By showing that the polling execution model for options gives rise to a specific matrix splitting, we can now better understand theoretically why following options for only one time step may still help (as long as we maintain termination conditions around) thereby reinforcing a similar claim made by (Sutton et al., 1999a, page 199):

> "Thus, even if multi-step options are never actually followed for more than one step they can still provide substantial advantages in computation and in our theoretical understanding."

Using matrix splitting theory, we know that this advantages in computation (its rate of convergence) is precisely a function of the spectral radius of the corresponding splitting: $\rho(\mathbf{M}_{\sigma,\sharp}^{-1} \mathbf{N}_{\sigma,\flat})$ to be exact.

4.7 A Family of Algorithms

While developing their theory of contracting MDPs, van Nunen (1976) also showed that many classical iterative algorithms can be obtained for specific

choices of stopping time functions: Gauss-Seidel, Jacobi, Successive Overrelaxation, or Richardson's iteration for example (Varga, 1962; Watkins, 2004; Chen, 2005). Note that a similar generalization of those algorithms had already been made in Varga's work using the matrix splitting formalism (Varga, 1962), but the uniqueness of van Nunen (1976) here is that it comes from a *temporal* notion : stopping time. It is this realization which subtends the title of the technical report by van Nunen and Wessels (1976b) proposing "the generation of successive approximation methods" using stopping times. Hence, a choice of stopping time function becomes a choice of algorithm.

Termination Function	$\mathbf{M}_{\pi,\sharp}$	Method
$\lambda(S_t, S_{t+1}) = 0$	$\mathbf{I}$	Policy evaluation
$\lambda(S_t, S_{t+1}) = \mathbb{1}_{S_t<S_{t+1}}$	$\mathbf{D} - \mathbf{E}$	Gauss-Seidel iteration
$\lambda(S_t, S_{t+1}) = \mathbb{1}_{S_t=S_{t+1}}$	$\mathbf{D}$	Jacobi iteration
$\lambda(S_t, S_{t+1}) = 1$	$\mathbf{I} - \gamma\mathbf{P}_\pi$	SR, Dyna, ER
$\lambda(S_t, S_{t+1}) = \lambda$	$\mathbf{I} - \gamma\lambda\mathbf{P}_\pi$	TD(λ)
$\lambda(S_t, S_{t+1}) = \beta(S_{t+1})$	$\mathbf{I} - \gamma\mathbf{P}_\pi \operatorname{diag}(\beta(\cdot))$	β-models
$\lambda(s', o) = 1 - \beta(S_{t+1}, O_t)$	$\mathbf{I} - \gamma\mathbf{P}_{\pi,O_t} \odot \mathbf{B}$	Options

Table 4.1 – Matrix splitting for some multi-step methods RL methods. SR stands for Successor Representation Dayan (1993). The matrix $\mathbf{D}$ here stands for the diagonal matrix extracted from $\mathbf{I} - \gamma\mathbf{P}_\pi$, while $\mathbf{E}$ is its strictly lower triangular part.

The generalized policy evaluation equations (4.2) also encompass classical methods (Varga, 1962), which we summarize in table 4.1. But more importantly, theoretical foundations laid in this chapter also help us understand our multi-step RL methods, starting with TD(λ). While the λ-operator in the TD(λ) algorithm is often conceptualized as an infinite convex combination of n-step returns (as we have seen in section 2.1.2.1), the generalized policy evaluation equations and the notion of stopping time provide another interpretation. Just as for the duality "discounting/random horizon" of the discount factor (lemma 2.1), the λ parameter of λ of TD(λ) plays the role of our termination function λ from this chapter. More specifically, we can think of the λ-operator as a choice of termination function where the probability of continuation is a constant "λ" and whose induced matrix splitting matrix is $\mathbf{M}_{\pi,\sharp} = \mathbf{I} - \gamma\lambda\mathbf{P}_\pi$. Inserting this choice of $\mathbf{M}_{\pi,\sharp}$ into the generalized projected policy evaluation equations from lemma 4.8, we get:

$$\left(\mathbf{\Phi}^\top \mathbf{\Xi} \left(\mathbf{I} - \gamma\lambda\mathbf{P}_\pi\right)^{-1} \left(\mathbf{I} - \gamma\mathbf{P}_\pi\right) \mathbf{\Phi}\right) \mathbf{w} = \mathbf{\Phi}^\top \mathbf{\Xi} \left(\mathbf{I} - \gamma\lambda\mathbf{P}_\pi\right)^{-1} \mathbf{r}_\pi \ ,$$

an expression which can also be found in Tsitsiklis and Roy (1997a); Boyan (2002); Bertsekas (2012); Sutton (2015b); Gehring et al. (2016). Yu et al. (2017) independently proposed a generalized form of the Bellman equations based on the same stopping time perspective shown in this chapter. Note that Yu et al. (2017) considers the more general case of history-dependent stopping time functions, purposefully ignored in this chapter, but studied in van Nunen (1976). Bertsekas (2012) also develops a different interpretation of the λ-return for linear function approximation based on the *random horizon* perspective. The essence of the proposed *geometric sampling* (Bertsekas, 2012, chapter 6) is to *sample through* the λ-models from section 4.3, up to termination (where termination events are sampled explicitly).

While the successor representation (SR) is usually not conceptualized as a multi-step RL method, we have seen in this chapter that it can be obtained for the choice of termination function $\lambda = 1$. By listing it in table 4.1 along other methods, we are not claiming that they are *equivalent* in the practical sense but simply that they belong to the same family. Furthermore, the fact that TD(λ) is listed along SR does not preclude one from combining the eligibility traces mechanism of TD(λ) to *learn* the SR by temporal difference learning (see recursive equations in section 2.3).

In addition, SR serves as a useful example that despite being a *bad* preconditioner (in the sense that it is directly the inverse of the original coefficient matrix), what matters really is that it can be *transferred*/reused in other tasks. What seems to be an unreasonable expense makes a lot more sense when amortized over a lifetime of tasks. And this amortization pays off because our world has regularities which afford generalization. We come back to this point in chapter 5.2.1 with a bounded rationality Simon (1957) argument. Related ideas can also be found in Solway et al. (2014) who goes through an accounting exercise (in the Bayesian setting) to justify the usefulness of bottleneck options in transfer learning. Hackbusch (2016) also proposes a measure of *efficacy* to compare the usefulness of two given preconditioners, taking into account their set up time, increase in convergence rate, as well as their amortization over different problems.

The matrix preconditioning perspective presented in this chapter may give us a new vocabulary to talk about representation learning in the context of multi-step RL. In fact, the qualities of good preconditioners (or matrix splittings) have

a lot in common those of good representations. For example, when talking about the problem of finding good representations, Smolensky (1990) wrote:

> "[...] a poor representation will often doom the model to failure, and an excessively generous representation may essentially solve the problem in advance. " ,

a statement which would equally hold for matrix preconditioning. In general, we should strive for a balance of those two extremes by a having a representation which is *good enough* (Goldstein and Gigerenzer, 2002).

4.8 Bibliographical Remarks

The dynamic programming foundations for the material presented in this chapter was laid out in the 70s by Jaap Wessel and Jo van Nunen (his PhD student at the time) at the Eindhoven University of Technology in a series of papers (van Nunen and Wessels, 1976a; Wessels, 1977; van Nunen and Wessels, 1981; van Nunen and Stidham, 1981) and in van Nunen's thesis (van Nunen, 1976). The generalization of the Bellman equations put forward by van Nunen and Wessel led to a formalization of what we now call *modified policy iteration*, which had been proposed earlier by Morton (1971). A similar algorithm was also developed independently by Puterman and Shin (1978) but using a different theoretical analysis based on the connection between policy iteration and Newton-Kantorovich iteration from Puterman and Brumelle (1979). The uniqueness of van Nunen's approach lies in its use of the notion of *stopping time* to characterize the contraction properties of the resulting generalized operator. As we show below, this concept turns out to be a natural fit to talk about multi-step methods in RL.

A closely related idea was proposed independently by Bertsekas and Ioffe (1996) under the name λ-iteration. However, the connection to the earlier work by van Nunen must have been unknown to the authors due to the lack of reference in Bertsekas and Ioffe (1996); Bertsekas and Tsitsiklis (1996); Bertsekas (2013). Around the same time, Sutton (1995) introduced a notion of *generalized Bellman equations* in the context of policy evaluation which turns out to be a subcase of the general framework put forward by van Nunen (1976) (which

encompasses both policy evaluation and control). The idea of *beta-models* introduced in Sutton (1995) laid the groundwork for subsequent developments with the options framework in Sutton et al. (1999a), predictive state representations (PSR) (Littman et al., 2001) and Horde (Sutton et al., 2011; White, 2015). What we call λ-models in this chapter are essentially beta-models with β defined over successive states (S_t, S_{t+1}).

An indirect connection to the matrix splitting perspective developed in this chapter can also be found in van Nunen (1976), citing earlier work by Porteus (1975). Instead of studying the contraction properties of a generalized operator, Porteus (1975) considered different algorithmic improvements to speed up successive approximation methods using a matrix-theoretic approach. It is in this work that the concept of matrix splitting was introduced, although indirectly through a reference to Varga (1962). In fact, Porteus (1975) used the results from matrix splitting theory in Varga (1962) to prove the properties of his *pre-inverse transform*, which nowadays would simply be called *matrix preconditioning* (Watkins, 2004; Golub and Van Loan, 1996; Chen, 2005).

The recursive form of the λ-models can be found in (Sutton et al., 1999a, section 5) for *intra-option model learning* of options, with the change of notation $\beta = 1 - \lambda$. The GVF interpretation provided above has also recently been added to the latest draft of the 2nd edition of the RL textbook (Sutton and Barto, 2018). The terminology λ-models is new to this thesis but is closely related to the concept of β-models from Sutton (1995). In fact, the β term in Sutton's paper corresponds in our notation to a "lambda" depending on S_t only. Equations (8) and (9) of Sutton (1995) can also be interpreted as GVFs for the transition and reward models respectively. In the tabular case, the "x_t" on the right-hand side of (8) in Sutton (1995) is the indicator function appearing in our equation (4.9). Bertsekas (2012) also considers a similar idea in section 6.3 equations 6.67 and 6.68 of his textbook. Bertsekas' notation "$C_k^{(\lambda)}$" and "$d_k^{(\lambda)}$" can be interpreted as empirical estimates of what we call here a λ-transition model and a λ-reward model respectively.

Chapter 5

Learning Options End-to-End

Option discovery has been challenging partly because it is hard to define *what* constitute good options. Over the years, many heuristics (which we review in the section below) have been proposed in an attempt to characterize common properties sought for in *good options*. However, it has remained difficult in general to understand which objective any of these heuristics is trying to accomplish (speed of learning and planning, achieving fast environment coverage or transfer learning). The premise of the work presented in this chapter is that we should decouple the choice of objective from the optimization tools used to solve it. Specifically, we focus on the question of *how* to learn and construct a set of options that achieve a given objective.

By casting the problem of options discovery under the control setting, we consider the problem of finding options that achieve a task as well as possible, as measured by the expected discounted return. Based on the actor-critic architecture (Sutton, 1984), we developed the option-critic architecture that uses stochastic gradient ascent to learn parameterized options based on this objective. But option-critic is also not simply a policy gradient method to learn the policy of an option, one at time. Instead, we consider the problem of learning option policies, termination functions and the policy over options in a joint manner. Hence, every sample of experience contributes to learning about the expected value associated with the performance of the agent and how to update its components accordingly. The benefits are twofold: first learning is fully integrated and does not involve restarting the agent to learn each option

separately, and second we are guaranteed that any update applied to an option is taking into consideration its effect on other options and the policy over them. The second property is crucial when it comes to achieving our stated goal of aligning our solution with the overall control objective; otherwise, *locally* optimal options may not lead to *global* optimality of the agent (Minsky, 1961; Watkins, 1989; Dietterich, 2000).

The option-critic architecture is capable of learning both options and the policy over them by leveraging the structure of the intra-option Bellman equations (Sutton et al., 1999a), which we review in section 2. We then show in section 3.3 how those equations can be obtained by considering the evolution of a special Markov process state-option pairs – an augmented state space. The construction of this augmented MDP then helps the derivation of new gradient theorem for options in section 5.1. Having access to the exact form of these gradients, we show in section 5.2 how to design a regularizer that favors long options and penalizes for frequent switches. The option-critic architecture, introduced in section 5.3, combines the gradient results into a learning architecture that learn value functions in tandem with the options and policy over them. We present three possible implementations for our architecture in section 5.4: a tabular version in a grid domain, one using the DQN algorithm (Mnih et al., 2015) in ALE (Bellemare et al., 2013) and a fast asynchronous extension inspired by (Mnih et al., 2016).

5.1 Option Gradient Theorem

In the same way that policy gradient methods offer approximation in policy space, we propose to learn options within a given parametric family.

Definition 5.1. A parametric family of randomized options and randomized policy over options is such that for each $\boldsymbol{\theta} \in \mathbb{R}^n$ and for all $s \in \mathcal{S}, o \in \mathcal{O}$ the termination functions are specified by $\beta_{\boldsymbol{\theta}}(s, o) \in [0, 1]$, the option policies by $\sum_a \pi_{\boldsymbol{\theta}}(a \mid s, o) = 1$ and the policy over option $\sum_o \mu_{\boldsymbol{\theta}}(o \mid s) = 1$.

Because of the generality of definition 5.1, parameters can be shared across different components. However, in order to guarantee smooth gradients (Konda and Tsitsiklis, 2000), we make the usual assumption in (4) that the termination

conditions, the option policies and the policy over them are chosen within a randomized family.

Assumption 4. *The termination functions, option policies and policy over options are randomized.*

For example, a randomized policy for discrete action spaces could be the *softmax* policy, defined as :

$$\pi_{\boldsymbol{\theta}}\left(a \mid s, o\right) = \frac{\exp^{(1/T)\boldsymbol{\phi}_{s,a,o}^{\top}\boldsymbol{\theta}}}{\sum_a \exp^{(1/T)\boldsymbol{\phi}_{s,a,o}^{\top}\boldsymbol{\theta}}} \ ,$$

where $\boldsymbol{\phi}_{s,a,o}^{\top}$ is some feature vector and T is a temperature parameter (Sutton and Barto, 1998). As the temperature parameter goes to zero, it is possible to recover a near-deterministic policy from the randomized softmax parameterization (Konda and Tsitsiklis, 2000). In a deep network, the feature vector ϕ might correspond to the activation of the penultimate layer of a network with $\pi_{\boldsymbol{\theta}}$ as one of its output, as shown in section 5.4. As for the parameterized termination conditions, the logistic function is a natural choice because $\beta_{\boldsymbol{\theta}}(s, o)$ must specify the mean of a Bernoulli distribution dependent on a state and option:

$$\beta_{\boldsymbol{\theta}}(s, o) = \frac{1}{1 + \exp^{-\boldsymbol{\phi}_{s,o}^{\top}\boldsymbol{\theta}}} \ .$$

When using parameterized options and a parameterized policy over options, we use the following notation to express the intra-option Bellman equations (3.5) :

$$\begin{aligned} Q_{\boldsymbol{\theta}}(s, o) &= \sum_a \pi_{\boldsymbol{\theta}}\left(a \mid s, o\right) Q_{U_{\boldsymbol{\theta}}}(s, o, a) \\ &= \sum_a \pi_{\boldsymbol{\theta}}\left(a \mid s, o\right) \left(r(s, a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) \left(Q_{\boldsymbol{\theta}}(s', o) - \beta_{\boldsymbol{\theta}}(s', o) A_{\boldsymbol{\theta}}(s', o) \right) \right), \end{aligned} \tag{5.1}$$

where $A_{\boldsymbol{\theta}}(s', o) = Q_{\boldsymbol{\theta}}(s', o) - \sum_{o'} \mu_{\boldsymbol{\theta}}(o' \mid s') Q_{\boldsymbol{\theta}}(s', o')$ is the parameterized advantage function and Q_{U_θ} is the expected discounted return given a state, option, and primitive action taken under that option:

$$\begin{aligned} Q_{U_\theta}(s, o, a) \doteq & \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) \,\middle|\, S_0 = s, O_0 = o, A_0 = a\right] \\ & = r(s, a) + \gamma \sum_{s'} P\left(s' \,\middle|\, s, a\right) U_{\boldsymbol{\theta}}(s', o) \ . \end{aligned}$$

5.1.1 Objective

In a discounted MDP, there always exists a *uniformly* optimal (Altman, 1999) deterministic stationary policy: this policy is optimal regardless of where the agent starts in the environment. However, when moving to parameterized families of policies, uniform optimality can no longer be maintained (Konda and Tsitsiklis, 2000; Singh et al., 1994) and the concept of optimality as a whole needs to be defined with respect to an initial distribution. The performance objective considered throughout the rest of this chapter therefore has to be of the form:

$$J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \sum_{s,o} \boldsymbol{\alpha}(s, o) Q_{\boldsymbol{\theta}}(s, o) = \mathbb{E}_{\boldsymbol{\alpha}, \boldsymbol{\theta}}\left[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t)\right] \ , \tag{5.2}$$

where $\boldsymbol{\alpha} : Dist(\mathcal{S} \times \mathcal{O})$ is an initial distribution over state and options. In order to optimize this objective by gradient ascent, we need to make the complementary assumption to (4) according to which the option policies, termination functions and the policy over options are differentiable.

Assumption 5. *For all $\boldsymbol{\theta}$ and any $a \in \mathcal{A}, o \in \mathcal{O}$: $\frac{\partial \mu_{\boldsymbol{\theta}}(o \mid s)}{\partial \theta_i}$, $\frac{\partial \pi_{\boldsymbol{\theta}}(a \mid s, o)}{\partial \theta_i}$ and $\frac{\partial \beta_{\boldsymbol{\theta}}(s, o)}{\partial \theta_i}$ exist.*

Using the structure of the augmented transition probability function $\widetilde{P}$, we are now able to apply the blueprints of the policy gradient theorem (2.21) and derive the gradient of (5.2) with respect to the parameters underlying the options and policy over them.

Theorem 5.2 (Joint Gradient)**.** *Let $J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ be the expected discounted return from an initial distribution $\boldsymbol{\alpha}$ over state-option pairs. Under the assumptions* (4) *and* (5)*, the*

gradient of $J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ is :

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\alpha},\boldsymbol{\theta}}\Big[&\beta_{\boldsymbol{\theta}}(S_{t+1}, O_t) \sum_{o'} \nabla_{\boldsymbol{\theta}} \left(\mu_{\boldsymbol{\theta}}\left(o' \mid S_{t+1}\right)\right) Q_{\boldsymbol{\theta}}(S_{t+1}, o') \\ &+ \sum_{a} \nabla_{\boldsymbol{\theta}} \left(\pi_{\boldsymbol{\theta}}\left(a \mid S_t, O_t\right)\right) Q_{U_{\boldsymbol{\theta}}}(S_t, O_t, a) \\ &- \nabla_{\boldsymbol{\theta}} \left(\beta_{\boldsymbol{\theta}}(S_{t+1}, O_t)\right) A_{\boldsymbol{\theta}}(S_{t+1}, O_t)\Big] \ ,\end{aligned}$$

where the expectation is taken under the discounted future state-option distribution.

Proof. Starting from (5.2), we have:

$$\frac{\partial J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{\partial \theta_i} = \sum_{s,o} \boldsymbol{\alpha}(s,o) \frac{\partial Q_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i} \ ,$$

which we expand using the parameterized intra-option Bellman equations (5.1):

$$\frac{\partial Q_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i} = \sum_{a} \left(\frac{\partial \pi_{\boldsymbol{\theta}}\left(a \mid s,o\right)}{\partial \theta_i} Q_{U_{\boldsymbol{\theta}}}(s,o,a) + \gamma \pi_{\boldsymbol{\theta}}\left(a \mid s,o\right) \sum_{s'} P\left(s' \mid s,a\right) \frac{\partial U_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} \right). \tag{5.3}$$

Taking the derivative of the utility function $U_{\boldsymbol{\theta}}$ in (5.3) yields:

$$\frac{\partial U_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} = \frac{\partial Q_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} - \frac{\partial \beta_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} A_{\boldsymbol{\theta}}(s',o) - \beta_{\boldsymbol{\theta}}(s',o) \frac{\partial A_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} \ . \tag{5.4}$$

Expanding the derivative of the advantage function then reveals the policy over options $\mu_{\boldsymbol{\theta}}$:

$$\frac{\partial A_{\boldsymbol{\theta}}(s'\, o)}{\partial \theta_i} = \frac{\partial Q_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} - \sum_{o'} \frac{\partial \mu_{\boldsymbol{\theta}}\left(o' \mid s'\right)}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s',o') - \sum_{o'} \mu_{\boldsymbol{\theta}}\left(o' \mid s'\right) \frac{\partial Q_{\boldsymbol{\theta}}(s',o')}{\partial \theta_i} \ . \tag{5.5}$$

Bringing (5.5), (5.4) and (5.3) back together, we finally obtain :

$$\begin{aligned}\frac{\partial Q_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i} = \sum_{a} \pi_{\boldsymbol{\theta}}\left(a \mid s,o\right) \Bigg(H_{\boldsymbol{\theta}}^{(i)}(s,o) + \gamma \sum_{s'} P\left(s' \mid s,a\right) \Bigg(\\ \frac{\partial Q_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} - \beta_{\boldsymbol{\theta}}(s',o) \left(\frac{\partial Q_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} - \sum_{o'} \mu_{\boldsymbol{\theta}}\left(o' \mid s'\right) \frac{\partial Q_{\boldsymbol{\theta}}(s',o')}{\partial \theta_i} \right) \Bigg) \Bigg),\end{aligned} \tag{5.6}$$

where :

$$H_{\boldsymbol{\theta}}^{(i)}(s,o) \doteq \left(\sum_a \frac{\partial \pi_{\boldsymbol{\theta}}(a \mid s,o)}{\partial \theta_i} Q_{U_{\boldsymbol{\theta}}}(s,o,a) \right) + \gamma \sum_a \pi_{\boldsymbol{\theta}}(a \mid s,o) \sum_{s'} P\left(s' \mid s,a\right) \Bigg(- \frac{\partial \beta_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} A_{\boldsymbol{\theta}}(s',o) + \beta_{\boldsymbol{\theta}}(s',o) \sum_{o'} \frac{\partial \mu_{\boldsymbol{\theta}}(o' \mid s')}{\partial \theta_i} Q_{\boldsymbol{\theta}}(s',o') \Bigg) \ .$$

The expression for $\frac{\partial Q_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i}$ bears the same form as the intra-option Bellman equations (3.5) but with $H_{\boldsymbol{\theta}}^{(i)}(s,o)$ playing the role of "reward" term. We leverage this intuition to solve the recursive form by drawing a connection to (3.7) in which we used the structure of the augmented transition probability function:

$$\frac{\partial Q_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i} = \sum_a \pi_{\boldsymbol{\theta}}(a \mid s,o) \Bigg(H_{\boldsymbol{\theta}}^{(i)}(s,o) + \gamma \sum_{s',o'} P\left(s' \mid s,a\right) \Big((1 - \beta_{\boldsymbol{\theta}}(s',o)) \mathbb{1}_{o'=o} + \beta_{\boldsymbol{\theta}}(s',o) \mu_{\boldsymbol{\theta}}\left(o' \mid s'\right) \Big) \frac{\partial Q_{\boldsymbol{\theta}}(s',o')}{\partial \theta_i} \Bigg) \ ,$$

where the term between square brackets is $\widetilde{P}$. We can then write the solution to the recursive expression for $\frac{\partial J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{\partial \theta_i}$ in terms of the discounted weighting of state-option pairs $\mathbf{d}_{\alpha,\boldsymbol{\theta}}$:

$$\frac{\partial J_{\boldsymbol{\alpha}}(\boldsymbol{\theta})}{\partial \theta_i} = \sum_{s,o} \boldsymbol{\alpha}(s,o) \frac{\partial Q_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i} = \sum_{s,o} \mathbf{d}_{\alpha,\boldsymbol{\theta}}(s,o) H_{\boldsymbol{\theta}}^{(i)}(s,o) \ .$$

□

Proposition 5.3. *Let* $\left\{\epsilon_t^{(\boldsymbol{\theta})}\right\}_{t=0}^{\infty}$ *be a nonnegative stepsize sequence such that:*

$$\sum_{t=0}^{\infty} \epsilon_t^{(\boldsymbol{\theta})} = \infty, \qquad \sum_{t=0}^{\infty} \epsilon_t^{(\boldsymbol{\theta})} < \infty$$

If the rewards are bounded and that the Hessian for $\beta_{\boldsymbol{\theta}}$, $\pi_{\boldsymbol{\theta}}$ *and* $\mu_{\boldsymbol{\theta}}$ *is also bounded, then the sequence* $\left\{\boldsymbol{\theta}_t\right\}_{t=0}^{\infty}$ *defined by:*

$$h_t \doteq \beta_{\boldsymbol{\theta}}(S_{t+1},O_t) \sum_{o'} \frac{\partial \mu_{\boldsymbol{\theta}}(o' \mid S_{t+1})}{\partial \theta_i} Q_{\boldsymbol{\theta}}(S_{t+1},o) + \sum_a \frac{\partial \pi_{\boldsymbol{\theta}}(a \mid S_t,O_t)}{\partial \theta_i} Q_{U_{\boldsymbol{\theta}}}(S_t,O_t,a) - \frac{\partial \beta_{\boldsymbol{\theta}}(S_{t+1},O_t)}{\partial \theta_i} A_{\boldsymbol{\theta}}(S_{t+1},O)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \epsilon_t h_t \ ,$$

converges with probability 1 and $\lim_{t\to\infty} \nabla_{\boldsymbol{\theta}_t} J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}_t) = 0$.

Proof: The boundedness of the rewards, policies and termination conditions lead to the Hessian of $J_{\boldsymbol{\alpha}}$ also being bounded. This provides the Lipschitz continuity condition on $\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\alpha}}$ necessary for standard convergence results on stochastic gradient descent to apply (Bertsekas and Tsitsiklis, 1996, proposition 4.1). □

As a consequence of the option gradient theorem 5.2, we can also obtain expressions for the gradients of the termination conditions (5.4) and policies inside options (5.5) when these components are optimized separately. This can be useful when a certain structure of a set of options needs to be retained but other parameters can be freely optimized. For example, one could choose to manually specify the policies inside options but only optimize the termination conditions. This decoupling could also be for planning by deriving a policy over options using options models but while learning the option policies and termination conditions by gradient ascent.

Lemma 5.4 (Termination Gradient)**.** *If the parameter vector is of the form* $\boldsymbol{\theta} = [\boldsymbol{\theta}_\pi, \boldsymbol{\theta}_\beta, \boldsymbol{\theta}_\mu]$*, with disjoint parameters* $\boldsymbol{\theta}_\beta$ *for the termination conditions, then under the assumptions* (4) *and* (5)*, the termination gradient is :*

$$\nabla_{\boldsymbol{\theta}_\beta} J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{\alpha},\boldsymbol{\theta}} \left[\nabla_{\boldsymbol{\theta}_\beta} \left(\beta_{\boldsymbol{\theta}}(S_{t+1}, O_t) \right) A_{\boldsymbol{\theta}}(S_{t+1}, O_t) \right] \quad .$$

The termination gradient in both lemma 5.4 and theorem 5.2 can be understood intuitively. When the option choice is suboptimal with respect to the expected value over all options, the advantage function $A_{\boldsymbol{\theta}}$ is negative, thus driving the gradient corrections up and increasing the odds of terminating in a given state. Upon termination, the agent then has the opportunity to pick a better option using its policy over option $\mu_{\boldsymbol{\theta}}$. Conversely, if it is advantageous to maintain the same option, the termination gradient will decrease the value of $\beta_{\boldsymbol{\theta}}$ in that state thereby making the option lasts longer. In contrast to the way that policy gradient for MDPs is often formulated, the advantage function $\mathbf{A}_{\boldsymbol{\theta}}$ does not stem from a deliberate choice on our behalf to use a control variate Rubinstein (1981) or *baseline* Williams (1992) for variance reduction. Our result is stated directly from the derivation of the true gradient, before even thinking about the algorithmic and statistical aspects pertaining the choice of a gradient estimator.

The termination gradient also shares the same semantics behind the *interrupting* execution model of options (Sutton et al., 1999a). While executing an option, the interruption mechanism monitors at every state the option-value function $Q_{\mathcal{O}}$ and compares it to value function $v_{\mathcal{O}}$. Whenever $Q_{\mathcal{O}}(s', o) < v_{\mathcal{O}}(s')$ the current option is terminated, $\beta(s', o) = 1$ and a new option is chosen by μ. To see the parallel with the termination gradient, it suffices to note that the interruption condition can also be stated as : $Q_{\mathcal{O}}(s', o) < v_{\mathcal{O}}(s') \Leftrightarrow Q_{\mathcal{O}}(s', o) - v_{\mathcal{O}}(s') = A_{\mathcal{O}}(s', o) < 0$. Under this interpretation, the termination gradient can therefore be understood as a gradient-based interruption execution mechanism.

Lemma 5.5 (Option Policy Gradient)**.** *If the parameter vector is of the form* $\boldsymbol{\theta} = [\boldsymbol{\theta}_\pi, \boldsymbol{\theta}_\beta, \boldsymbol{\theta}_\mu]$*, with disjoint parameters* $\boldsymbol{\theta}_\pi$ *for the policies inside options, then under the assumptions* (4) *and* (5)*, the option policy gradient is:*

$$\nabla_{\boldsymbol{\theta}_\pi} J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\alpha},\boldsymbol{\theta}} \left[\sum_a \nabla_{\boldsymbol{\theta}_\pi} \left(\pi_{\boldsymbol{\theta}} \left(a \mid S_t, O_t \right) \right) Q_{U_{\boldsymbol{\theta}}}(S_t, O_t, a) \right] \quad .$$

The meaning of the option policy gradient remains similar to that of the original policy gradient theorem. If a choice of action is deemed good according to $Q_{U_{\boldsymbol{\theta}}}$, the option policy gradient will point more in the direction that makes the policy $\pi_{\boldsymbol{\theta}}$ more likely to pick that action again. The fact that the option policy gradient contains the term $Q_{U_{\boldsymbol{\theta}}}$ is what allow the updates to move in the direction that makes the overall system perform better. The value of a primitive action expressed by $Q_{U_{\boldsymbol{\theta}}}$ is taken with respect to all possible future trajectories stemming from this choice. As can be seen from the intra-option Bellman equations (3.5), $Q_{U_{\boldsymbol{\theta}}}$ is not confined *locally* to the return up to termination and crosses that boundary through the policy over options $\mu_{\boldsymbol{\theta}}$ into the entire system. Hence, the option gradient theorem does not amount to a pseudo-reward Dietterich (2000) setting in which the policy of an option would be optimized, using the usual policy gradient methods for example, independently from the other options and the policy over them. As for $A_{\boldsymbol{\theta}}$ and $Q_{\boldsymbol{\theta}}$, $Q_{U_{\boldsymbol{\theta}}}$ is *holistic* and properly reflects the contribution of every component – terminations, option policies, policy over options – into the overall performance.

5.2 Regularization

While proposition 5.3 shows convergence to a locally optimal solution, it does not provide any guarantee on the structure of the learned options, which in fact need not be temporally extended. To see this, let us assume that the primitive actions are discrete so that we can generate a corresponding set of *primitive options* $\mathcal{O}_{\mathcal{A}}$ of size $|\mathcal{A}|$ as follows:

$$\mathcal{O}_{\mathcal{A}} \doteq \{(\mathcal{I}_a, \pi_a, \beta_a)\}_{a \in \mathcal{A}} \text{ where } \mathcal{I}_a = \{s \in \mathcal{S} \mid a \in \mathcal{A}(s)\},\ \pi_a(s) = a,\ \beta_a(s) = 1 \ .$$

The optimality equations associated with $\mathcal{O}_{\mathcal{A}}$ are then:

$$\begin{aligned} v^\star_{\mathcal{O}_{\mathcal{A}}}(s) &= \max_{o \in \mathcal{O}_{\mathcal{A}}} Q^\star_{\mathcal{O}_{\mathcal{A}}}(s, o) \\ Q^\star_{\mathcal{O}_{\mathcal{A}}}(s, o) &= \max_a \left(r(s, a) + \gamma \sum_{s'} P\left(s' \mid s, a\right) v^\star_{\mathcal{O}_{\mathcal{A}}}(s') \right) \\ &= v^\star(s) \ , \end{aligned}$$

where $v^\star$ is the optimal value function in the base discounted MDP. We can see that $\mathcal{O}_{\mathcal{A}}$ then also belongs to the set of maximizers for:

$$\max_{\mathcal{O} \in \Omega^{\mathrm{D}}} \sum_{s,o} \boldsymbol{\alpha}(s, o) Q_{\mathcal{O}}(s, o) \ ,$$

which is the same objective as (5.2), except that we are not searching in a parametric family of randomized options as in section 5.1 but over a finite set of options with deterministic policies and termination functions Ω^{D}.

But if the degenerate solution $\mathcal{O}_{\mathcal{A}}$ is not directly representable, it also does not preclude the possibility of approaching it arbitrarily close. For example with the softmax parameterization, the temperature could be decreased to zero (Konda and Tsitsiklis, 2000) to obtain near-deterministic option policies in the limit. Hence, maintaining a higher temperature is desirable to prevent premature convergence to the solution $\mathcal{O}_{\mathcal{A}}$. In that sense, the inherent noise from the parameterized options acts as an implicit regularizer for finding meaningful options. A more direct approach for encoding this propensity for diverse action selection is the *entropy regularizer* of Williams and Peng (1991); Mnih et al. (2016), which we found to be highly effective in our *deep* variants of the option-critic architecture (section 5.3).

One problem remains : how can ensure that the options will not just terminate everywhere ? To gain some intuition, consider again the class of deterministic termination conditions and assume that the option policies and policy over options are fixed. Then the deterministic analogue to the termination gradient (5.4) is the *interrupting operator* (Sutton et al., 1999b; Mann et al., 2014) that sets $\beta_o(s') = 1$ whenever $Q_{\mathcal{O}}(s', o) < v_{\mathcal{O}}(s')$. With a greedy policy over options $\mu_{\mathcal{O}}(s) = \max_o Q_{\mathcal{O}}(s', o)$ and because $\forall s \in \mathcal{S}, \forall o \in \mathcal{O}$, $Q(s', o) \leq v_{\mathcal{O}}(s') = \max_o Q_{\mathcal{O}}(s', o)$, the termination conditions would be set to one immediately under the interruption operator. The termination gradient being a function of the negative of the advantage function, the resulting updates would drive the parameterized termination conditions to 1 when optimizing by gradient ascent. How can we then bias the termination gradient to favor long options ? The answer is simple : we need to make temporally extended options more *advantageous* than short ones. This means that the advantage function needs to be more positive than it would usually be in order to tilt the balance in favor of stronger *commitment* as opposed to *switching*.

The solution that we develop in this section boils down to adding a constant scalar to the advantage function, thereby increasing the baseline of *advantegeousness* for long options. A similar idea had also been explored by Mann et al. (2014) in a dynamic programming setting but we extend it here to the policy gradient framework. Furthermore, we show that it can be recovered as a special case of a more general problem formulation that considers the computational cost associated with a solution in addition to its expected performance over the reward function of the base MDP.

5.2.1 Cost Model

The idea that the degree of optimality should be modulated to take into account the finiteness of its computational resource is a mainstay of *bounded rationality* (Simon, 1957). Under those constraints, a decision maker must learn to represent information as efficiently as possible (Simon, 1969). Thinking about options from a representation learning perspective then suggests that good options are those that make a problem easier to solve (Minsky, 1961; Simon, 1969). With such knowledge, a planner or learning algorithm decision could be able to derive a useful solution in fewer steps than if it had access only

to primitive actions (Sutton et al., 1999a; Hauskrecht et al., 1998; Mann and Mannor, 2013). It this section, we formalize what learning and planning faster with options might mean by incorporating a *cost* into our original objective (5.2). We then show that the interpretation regarding the role of a baseline (or *margin*) term in the advantage function can be recovered within this problem formulation.

Using the notation for the augmented state space of section 3.3, we define a cost $\widetilde{c} : \widetilde{\mathcal{S}} \times \mathcal{A} \times \widetilde{\mathcal{S}} \to \mathbb{R}$ which is a function of the current augmented state, the current action, and next augmented state : $\widetilde{c}(\tilde{s}_t, a_t, \tilde{s}_{t+1})$. A benefit of defining our notion of cost in this form is that it is now possible to use the same optimization tools developed in section 5.1 for the expected discounted sum of reward.

As for reward function of the base MDP, the interaction of $\widetilde{\pi}$ with the augmented transition probability function $\widetilde{P}$ (section 3.6) and cost $\widetilde{c}$ underlies the definition of the following expected discounted cost :

$$\widetilde{D}_{\mathcal{O}}(\tilde{s}) \doteq \mathbb{E}\left[\sum_{t=0} \gamma^t \widetilde{c}(\tilde{S}_t, A_t, \tilde{S}_{t+1}) \,\middle|\, \tilde{S}_0 = \tilde{s}\right] \quad .$$

We then express the fact that we are interested in not only options that achieve a good return, but also those which are *cheap* in a constrained formulation of our original objective (5.2). Switching to the notation $\mathbf{D}_{\boldsymbol{\theta}}(s, o) \doteq \widetilde{\mathbf{D}}_{\boldsymbol{\theta}}(\tilde{s})$, our problem is formulated with respect to a randomized family of options (definition 5.1) as follows :

$$\begin{aligned} &\max_{\boldsymbol{\theta}} \sum_{s,o} \boldsymbol{\alpha}(s, o) Q_{\boldsymbol{\theta}}(s, o) \\ &\text{subject to: } \sum_{s,o} \boldsymbol{\alpha}(s, o) D_{\boldsymbol{\theta}}(s, o) \leq k \ , \end{aligned} \tag{5.7}$$

where $k \in \mathbb{R}$ is a set constant. While the topic of constrained optimization for dynamic programming is well-understood (Altman, 1999), the Linear Programming (LP) algorithmic solutions required to solve such problems are incompatible with model-free methods and computationally very demanding. To avoid these complications, we cast our problem differently by taking the

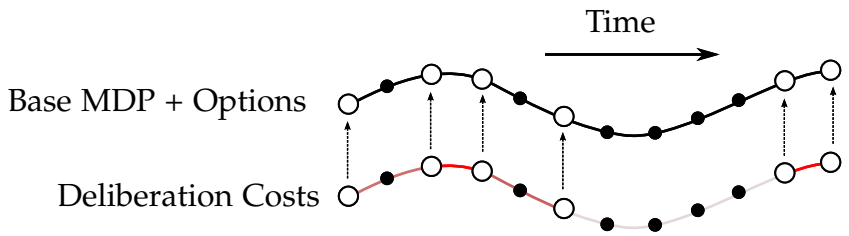


FIGURE 5.1 – The switching cost is incurred upon entering SMDP decision points, represented by open circles. The average decision cost per primitive step (filled circle) is represented by the intensity of the subtrajectory.

following Lagrangian relaxation (Sennott, 1991) :

$$\max_{\boldsymbol{\theta}} J^c_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \qquad \text{where} \qquad J^c_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) \doteq \sum_{s,o} \alpha(s,o)\left(Q_{\boldsymbol{\theta}}(s,o) - \eta D_{\boldsymbol{\theta}}(s,o)\right) \quad , \tag{5.8}$$

and η is scalar that reflects the importance of maximizing the reward in the external environment relative to the intrinsic cost that a decision maker has to incur.

5.2.2 Switching Cost

Long options can be promoted by imposing a cost for deciding too often about what option to choose next. To see this, imagine that we must pay a cost η whenever an option terminates. A back-of-the-envelope calculation tells us that if the probability of continuing with the same option (one minus the termination condition) is a constant κ, then the expected discounted duration of that option is $\frac{1}{1-\kappa\gamma}$ and the average decision cost per step is $\eta(1-\kappa\gamma)$. If the probability κ of continuing increases, we see that the average cost rate decreases. On the other hand, if an option terminates at every step then $k = 0$ and the cost rate reaches its maximum with η units per steps. Long options therefore lead to a better amortization of the decision cost.

Consider the case where the cost function is of the form $c_{\boldsymbol{\theta}}(s', o)$. Since $D_{\boldsymbol{\theta}}$ shares the same underlying transition probability function as $Q_{\boldsymbol{\theta}}$, we can also write (5.8) as the sum of the *base* MDP reward function r with the cost function $c_{\boldsymbol{\theta}}$. This additive transformation of the reward function can be thought of as defining a new MDP over an augmented state space. The following intra-option Bellman expresses the expected discounted (*mixed*) return over

the transformed reward function:

$$Q^c_{\boldsymbol{\theta}}(s,o) = \sum_a \pi\left(a \,|\, s,o\right) \sum_{s'} P\left(s' \,\middle|\, s,a\right) \Big(r(s,a) \\ - \eta c_{\boldsymbol{\theta}}(s',o) + \gamma Q^c_{\boldsymbol{\theta}}(s',o) - \gamma \beta_{\boldsymbol{\theta}}(s',o) A^c_{\boldsymbol{\theta}}(s',o)\Big).$$

Furthermore, if $c_{\boldsymbol{\theta}}(s',o) = \gamma \beta_{\boldsymbol{\theta}}(s',o)$ (which we call a switching cost function) we have :

$$Q^c_{\boldsymbol{\theta}}(s,o) = \sum_a \pi\left(a \,|\, s,o\right) \Big(r(s,a) \tag{5.9}$$

$$+ \gamma \sum_{s'} P\left(s' \,\middle|\, s,a\right) \left(Q_{\boldsymbol{\theta}}(s',o) - \beta_{\boldsymbol{\theta}}(s',o) \left(A^c_{\boldsymbol{\theta}}(s',o) + \eta\right)\right) \Big), \tag{5.10}$$

where $A^c_{\boldsymbol{\theta}}(s',o) \doteq Q^c_{\boldsymbol{\theta}}(s',o) - v^c_{\boldsymbol{\theta}}(s')$. The introduction of the switching cost to the base MDP reward therefore leads to a different form for the intra-option Bellman equations (3.5) where a scalar η is now added to the advantage function. This suggests that the effect of using a switching cost η is to set a *baseline* on how good an option is believed to be compared to $\mathbf{v}_{\boldsymbol{\theta}}$. By increasing η, we effectively express that persisting with an option might be preferable to reconsidering the current course of actions immediately. This preference for committing to the same option might be motivated by computational or metabolic limitations (Simon, 1957), by the inherent approximation error (due to finite predictive capacity) or due to uncertainty in the value estimates (Lloyd and Dayan, 2018).

5.2.3 Different Horizons for Cost and Reward

The generality of the regularized objective (5.8) allows a decoupling of the internal horizon on the expected discounted cost with the discount factor of the external environment. In this case, the unconstrained objective becomes:

$$J_\alpha^{\gamma,\tau}(\boldsymbol{\theta}) \doteq \sum_{s,o} \boldsymbol{\alpha}(s,o) \left(Q^\gamma_{\boldsymbol{\theta}}(s,o) - D^\tau_{\boldsymbol{\theta}}(s,o)\right) \quad . \tag{5.11}$$

where $D^\tau_{\boldsymbol{\theta}}$ is the expected τ-discounted cost and $Q^\gamma_{\boldsymbol{\theta}}$ the expected discount sum of rewards in the base MDP. The intra-option Bellman equations over the

switching cost being:

$$D^{\tau}_{\boldsymbol{\theta}}(s,o) = \sum_{a} \pi\left(a \,|\, s,o\right) \sum_{s'} P\left(s' \,\middle|\, s,a\right) \left(c_{\boldsymbol{\theta}}(s',o) + \tau Q_{\boldsymbol{\theta}}(s',o) - \tau \beta_{\boldsymbol{\theta}}(s',o) A_{\boldsymbol{\theta}}(s',o)\right),$$

setting $\tau = 0$ with $c_{\boldsymbol{\theta}}(s',o) = \gamma \beta_{\boldsymbol{\theta}}(s',o)$ leads to :

$$D^{\tau=0}_{\boldsymbol{\theta}}(s,o) = \sum_{a} \pi\left(a \,|\, s,o\right) \sum_{s'} P\left(s' \,\middle|\, s,a\right) c_{\boldsymbol{\theta}}(s,o,s') \ .$$

Taking the derivative with respect to the termination-specific parameters $\boldsymbol{\theta}_{\beta}$, we have :

$$\frac{\partial D^{\tau=0}_{\boldsymbol{\theta}}(s,o)}{\partial \theta_i} = \sum_{a} \pi\left(a \,|\, s,o\right) \sum_{s'} P\left(s' \,\middle|\, s,a\right) \gamma \frac{\partial \beta_{\boldsymbol{\theta}}(s',o)}{\partial \theta_i} \ .$$

Finally by linearity of (5.11), the termination gradient (5.4) for the mixed-discounted case $\gamma, \tau = 0$ becomes:

$$\nabla_{\boldsymbol{\theta}_{\beta}} J^{\gamma,\tau=0}_{\alpha}(\boldsymbol{\theta}) = -\mathbb{E}_{\alpha,\boldsymbol{\theta}} \left[\nabla_{\boldsymbol{\theta}_{\beta}} \beta_{\boldsymbol{\theta}}(S_{t+1}, O_t) \left(A_{\boldsymbol{\theta}}(S_{t+1}, O_t) + \eta\right)\right] \ . \tag{5.12}$$

Once again, η appears along with the advantage function. But there is here a subtle (but structurally important) difference with the case considered in the previous section where the two horizons are aligned and $\gamma = \tau$. Indeed, the termination gradient based on (5.10) turns out to be instead :

$$\nabla_{\boldsymbol{\theta}_{\beta}} J^{\lambda=\tau}_{\alpha}(\boldsymbol{\theta}) = -\mathbb{E}_{\alpha,\boldsymbol{\theta}} \left[\frac{\partial \beta_{\boldsymbol{\theta}}(S_{t+1}, O_t)}{\partial \theta_i} \left(A^{c}_{\boldsymbol{\theta}}(S_{t+1}, O_t) + \eta\right)\right] \ , \tag{5.13}$$

where the key difference is that we have $A^{c}_{\boldsymbol{\theta}}$ rather than $A_{\boldsymbol{\theta}}$. In the first case where $\tau = 0$, the advantage function $A_{\boldsymbol{\theta}}$ in the termination gradient does not contain any information about the expected cost corresponding to a set of options and policy over them and η. On the other hand, when $\gamma = \tau$ the term $A^{c}_{\boldsymbol{\theta}}$ is the advantage function over a transformed reward function which is not the difference between the base (original) MDP reward and the cost function.

The gradient (5.12) corresponding to $\tau = 0$ therefore does not take into account the full extent of a change in the termination probabilities on the overall expected cost beyond the next step. On the other hand, $\tau = \gamma$ allows an agent to properly recognize that it might be preferable to sometimes incur a higher cost immediately if the expected cost of the overall solution can be improved

in the future. The choice of τ is therefore closely related to an agent's ability or willingness to predict the future cost of its own internal computational processes.

5.3 The Option-Critic Architecture

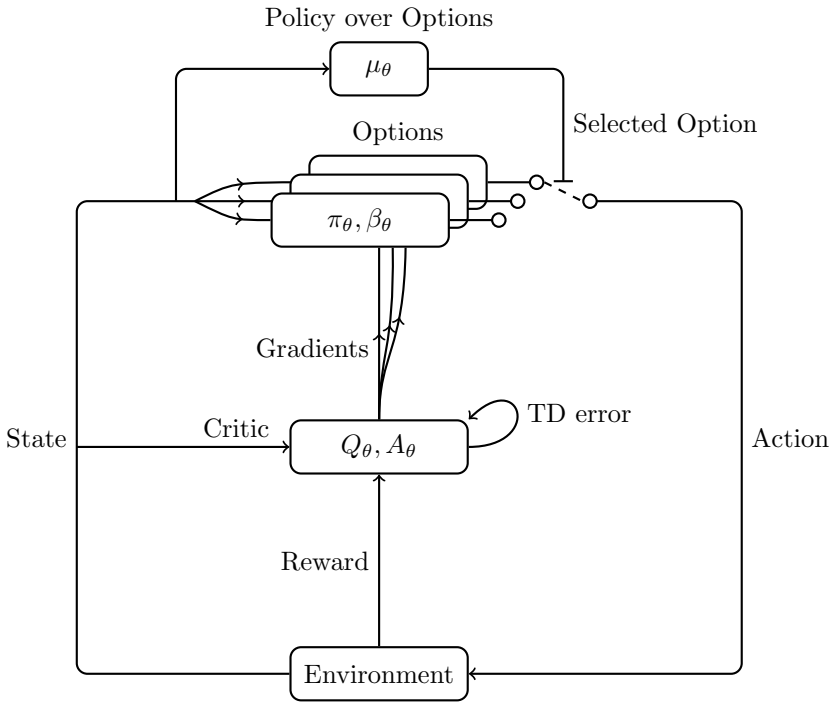


FIGURE 5.2 – Option-critic is an actor-critic architecture for learning options end-to-end. The *actor* contains a set of options parameterized by $\boldsymbol{\theta}$ and a policy over them $\mu_{\boldsymbol{\theta}}$ that are updated based on the value predictions made by a *critic*.

The results of the previous section gave us the form of the gradient of expected discounted return with respect to parameterized options. Within the same gradient-based framework, we also proposed a regularizer that favors longer options and penalizes frequent interruptions. But in order to use these results for learning options, we have yet to specify how to obtain gradient estimates from experience. As in actor-critic architectures (Sutton, 1984), we propose to learn the value terms in a critic while simultaneously applying the resulting gradient updates to the parameterized options and policy over them in the actor. The representation of options within an actor together with the generation of feedback using a value-based critic result in what we call the *option-critic* architecture (figure 5.2).

In this section, we develop temporal difference learning algorithms for evaluating the performance of a solution based on the structure of the intra-option Bellman equations (5.1). We then show how gradient estimators can be built from those value estimates using the likelihood-ratio sampling trick (L'Ecuyer, 1990; Williams, 1992) and how to reduce the variance using a control variate (baseline) or by *conditioning* (Rubinstein, 1981)

5.3.1 Intra-Option Learning Algorithms for Value Prediction

Using the perspective on Markov options from the augmented state space, we can easily derive a temporal difference (TD) learning algorithms for learning state-option or state-option-action values needed in the critic component of our system. For simplicity, let us assume the linear function approximation setting in which a parameter vector $\mathbf{w}$ must be found such that $Q_{\boldsymbol{\theta}}(S_t, O_t) \approx \boldsymbol{\phi}_t^\top \mathbf{w} \doteq \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \mathbf{w})$ and where $\boldsymbol{\phi}_t \doteq \boldsymbol{\phi}(S_t, O_t)$ is a feature vector over state-option pairs. The TD(0) learning rule Sutton (1988) over the augmented state space is then:

$$\begin{aligned} & \mathbf{w}_{t+1} = \mathbf{w}_t + \epsilon_t^{(\mathbf{w})} \left(R_{t+1} + \gamma \widetilde{v}_{\boldsymbol{\theta}}(S_{t+1}; \mathbf{w}_t) - \widetilde{v}_{\boldsymbol{\theta}}(S_t; \mathbf{w}_t) \right) \boldsymbol{\phi}_t \\ \Leftrightarrow \; & \mathbf{w}_{t+1} = \mathbf{w}_t + \epsilon_t^{(\mathbf{w})} \left(R_{t+1} + \gamma \widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, O_{t+1}; \mathbf{w}_t) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \mathbf{w}_t) \right) \boldsymbol{\phi}_t \;, \end{aligned} \tag{5.14}$$

where $\epsilon_t^{(\mathbf{w})}$ is the learning rate at time t. We call the resulting learning algorithm *augmented TD(0)* as it directly uses samples from the augmented transition probability function $\widetilde{P}_{\boldsymbol{\theta}}$ without leveraging its structure (3.6). The fixed point of this algorithm can be characterized through the ODE method (Benveniste et al., 1990) as first shown by Tsitsiklis and Roy (1997a) for the case of TD(λ). Using the augmented MDP constructed in section 3.3, we can easily adapt the main result from Tsitsiklis and Roy (1997a) to the options framework.

Assumption 6. *The augmented Markov chain induced by the option policies, termination functions and policy over options is ergodic and admits a unique stationary distribution* $\xi \in \mathbb{R}^{|\mathcal{S}\times\mathcal{O}|}$, $\boldsymbol{\Xi} \doteq diag(\xi)$.

Proposition 5.6. *Under assumption 6, the deterministic counterpart to the stochastic augmented TD(0) algorithm* (5.14) *is described by:*

$$\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t + \epsilon_t^{(\mathbf{w})} \boldsymbol{\Phi}^\top \boldsymbol{\Xi} \left(\widetilde{\mathbf{r}}_{\widetilde{\pi}} - (\mathbf{I} - \gamma \widetilde{\mathbf{P}}_{\widetilde{\pi}}) \boldsymbol{\Phi} \bar{\mathbf{w}}_t \right) \quad .$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{|\mathcal{S} \times \mathcal{O}| \times k}$ *contains the feature vectors of dimension k on its rows.*

Proof: The *average* behavior of the stochastic updates (5.14) is taken under the stationary distribution $\boldsymbol{\Xi}$:

$$\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t + \epsilon_t^{(\mathbf{w})} \mathbb{E}_{\xi} \left[\left(r(S_t, A_t) + \gamma \widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, O_{t+1}; \bar{\mathbf{w}}_t) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \bar{\mathbf{w}}_t) \right) \boldsymbol{\phi}_t \right]$$

Breaking down this expectation, we have:

$$\begin{aligned} \mathbb{E}_{\xi} \left[r(S_t, A_t) \boldsymbol{\phi}_t \right] &= \boldsymbol{\Phi}^\top \boldsymbol{\Xi} \widetilde{\mathbf{r}}_{\widetilde{\pi}} \\ \mathbb{E}_{\xi} \left[\widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, O_{t+1}; \bar{\mathbf{w}}_t) \boldsymbol{\phi}_t \right] &= \boldsymbol{\Phi}^\top \boldsymbol{\Xi} \widetilde{P}_{\widetilde{\pi}} \boldsymbol{\Phi} \mathbf{w}_t \\ \mathbb{E}_{\xi} \left[\widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \bar{\mathbf{w}}_t) \boldsymbol{\phi}_t \right] &= \boldsymbol{\Phi}^\top \boldsymbol{\Xi} \boldsymbol{\Phi} \mathbf{w}_t \quad , \end{aligned}$$

which once combined back together give us the desired form. See (Tsitsiklis and Roy, 1997a, lemma 7) for more details . □

Under the usual assumptions on the sequence of step sizes: $\sum_{t=0}^{\infty} \epsilon_t^{(\mathbf{w})} = \infty, \sum_{t=0}^{\infty} \left(\epsilon_t^{(\mathbf{w})} \right)^2 < \infty$ and other mild regularity conditions (Tsitsiklis and Roy, 1997a), we can then show that the augmented TD(0) converges to the solution of the deterministic system in propositionf 5.6.

An extension to *n*-steps targets for augmented TD can also be easily devised under the same framework. To see this, it suffices to note that the intra-option Bellman equations for the true parameterized option-value function $Q_{\boldsymbol{\theta}}$ can be written as:

$$Q_{\boldsymbol{\theta}}(s, o) = \mathbb{E} \left[\sum_{t=0}^{n-1} \gamma^t r(S_t, A_t) + \gamma^n Q_{\boldsymbol{\theta}}(S_n, O_n) \,\middle|\, S_0 = s, O_0 = o \right] \quad .$$

Hence, to derive an n-steps augmented TD algorithm for learning $\widehat{Q}_{\boldsymbol{\theta}}$ we need to use the following sequence of updates:

$$\begin{aligned}\delta_t &= \left(\sum_{k=0}^{n-1}\gamma^k r(S_{t+k}, A_{t+k}) + \gamma^n \widehat{Q}_{\boldsymbol{\theta}}(S_{t+n}, O_{t+n}; \mathbf{w}_t) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \mathbf{w}_t)\right) \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \epsilon_t^{(\mathbf{w})}\delta_t \boldsymbol{\phi}_t \ . \end{aligned} \tag{5.15}$$

The use of n-steps targets has recently received a lot of interest due to Mnih et al. (2016) who showed increased learning performance in the ALE environment (Bellemare et al., 2013). Hence, we also incorporate this idea in our A2OC algorithm presented in section 5.4.3.

Knowing the structure of the transition probability function $\widetilde{\mathbf{P}}_{\boldsymbol{\theta}}$, we can improve augmented TD (3.6) by the method of *conditioning* (Rubinstein, 1981) for variance reduction. The basic idea here is that we can reduce the variance in our estimators by explicitly computing conditional expectation terms, assuming that the all necessary quantities are known and tractable. The utility term $U_{\boldsymbol{\theta}}$ in the intra-option Bellman equation (5.1) is one such case where $U_{\mathcal{O}}(S_{t+1}, O_t)$ can be computed directly given a sampled state and option. The resulting algorithm called *intra-option TD(0)* is described as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \epsilon_t^{(\mathbf{w})}\left(R_{t+1} + \gamma \widehat{U}_{\boldsymbol{\theta}}(S_{t+1}, O_t; \mathbf{w}_t) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \mathbf{w}_t)\right)\boldsymbol{\phi}_t \ , \tag{5.16}$$

where $\widehat{U}_{\boldsymbol{\theta}}$ is the utility function defined as :

$$\begin{aligned}&\widehat{U}_{\boldsymbol{\theta}}(S_{t+1}, O; \mathbf{w}_t) \doteq \\ &(1 - \beta_{\boldsymbol{\theta}}(S_{t+1}, O_t))\widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, O_t; \mathbf{w}_t) + \beta_{\boldsymbol{\theta}}(S_{t+1}, O_t)\sum_o \mu_{\boldsymbol{\theta}}(o \mid S_{t+1})\,\widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, o; \mathbf{w}_t) \ .\end{aligned}$$

The distinction between (5.14) and (5.16) is analogous to SARSA (Rummery and Niranjan, 1994) vs *expected SARSA* (Sutton and Barto, 1998) which was shown to reduce the variance (van Seijen et al., 2009). Furthermore, when the policy over options $\mu_{\boldsymbol{\theta}}$ is the greedy policy over options, (5.16) yields the control algorithm called intra-option Q-learning introduced in Sutton et al. (1999a).

5.3.2 Variance Reduction and Avoiding Primitive Actions

When estimating the gradients along the discounted weighting of state-option pairs, it is common to reduce the variance of the policy gradient estimator using a *baseline* (Williams, 1992; Sutton et al., 1999b), which also corresponds to the notion of control variates in Monte-Carlo methods (Rubinstein, 1981).

In the usual policy gradient setting over primitive actions (Sutton et al., 1999b), the value function is often used as a baseline for the state-action value function. Correspondingly, a natural choice of baseline for the gradient for the option policies (5.5) involving $Q_{U_\theta} : \mathcal{S} \times \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ is the option-value function $Q_\theta : \mathcal{S} \times \mathcal{O} \to \mathbb{R}$:

$$\begin{aligned}\nabla_{\theta_\pi} J_\alpha(\theta) &= \mathbb{E}_{\alpha,\theta}\left[\sum_a \nabla_{\theta_\pi}\left(\pi_\theta\left(a \mid S_t, O_t\right)\right)\left(Q_{U_\theta}(S_t, O_t, a) - Q_\theta(S_t, O_t)\right)\right] \\ &= \mathbb{E}_{\alpha,\theta}\left[\sum_a \nabla_{\theta_\pi}\left(\pi_\theta\left(a \mid S_t, O_t\right)\right) Q_{U_\theta}(S_t, O_t, a)\right] \quad .\end{aligned}$$

The difference $A_{U_\theta}(S_t, O_t, a) \doteq Q_{U_\theta}(S_t, O_t, a) - Q_\theta(S_t, O_t)$ can also be conceptualized as an advantage function (Sutton et al., 1999b), but one that compares the value of taking a certain primitive action to the expected performance of the system over all actions. As usual (Sutton et al., 1999b; Peters and Schaal, 2006), the baseline Q_θ – or in fact any function which does not depend on primitive actions – does not add bias to the overall gradient:

$$\begin{aligned}&\mathbb{E}_{\alpha,\theta}\left[\sum_a \nabla_{\theta_\pi}\left(\pi_\theta\left(a \mid S_t, O_t\right)\right) Q_\theta(S_t, O_t)\right] \\ &\quad = \mathbb{E}_{\alpha,\theta}\left[Q_\theta(S_t, O_t) \nabla_{\theta_\pi} \sum_a \pi_\theta\left(a \mid S_t, O_t\right)\right] \\ &\quad = 0 \quad .\end{aligned}$$

The presence of primitive actions in the gradient for option policies can be troublesome when $\mathcal{A}$ is large. A well-known approach that weakens this dependence is the likelihood ratio estimator (L'Ecuyer, 1990; Williams, 1992). Using the fact that $\nabla_{\theta_\pi} \log \pi_\theta\left(a \mid s\right) = \frac{1}{\pi_\theta(a \mid s)} \nabla_{\theta_\pi} \pi_\theta\left(a \mid s\right)$, the inner summation over primitive actions can then be written as an expectation through a change

of measure:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_\pi} J_{\boldsymbol{\alpha}}(\boldsymbol{\theta}) &= \mathbb{E}_{\alpha,\boldsymbol{\theta}}\left[\sum_a \pi_{\boldsymbol{\theta}}\left(a \mid S_t, O_t\right) \nabla_{\boldsymbol{\theta}_\pi}\left(\log \pi_{\boldsymbol{\theta}}\left(a \mid S_t, O_t\right)\right) A_{U_{\boldsymbol{\theta}}}(S_t, O_t, a)\right] \\ &= \mathbb{E}_{\alpha,\boldsymbol{\theta}}\left[\nabla_{\boldsymbol{\theta}_\pi} \log \pi_{\boldsymbol{\theta}}\left(A_t \mid S_t, O_t\right) A_{U_{\boldsymbol{\theta}}}(S_t, O_t, A_t)\right] \quad ,\end{aligned}$$

an idea which is also at the core of importance sampling (Rubinstein, 1981) methods.

Having gotten rid of the summation over primitive actions, we have to address the problem that A_{U_θ} is defined over $\mathcal{S} \times \mathcal{O} \times \mathcal{A}$. In that case, a useful strategy to avoid learning a Q_{U_θ} separately is to sample through the choice of actions in (3.5). We call the resulting estimator the *intra-option advantage estimator*:

$$\widehat{A}_{U_{\boldsymbol{\theta}}}(S_t, O_t, R_{t+1}, S_{t+1}) \doteq R_{t+1} + \gamma U_{\boldsymbol{\theta}}(S_{t+1}, O_t) - Q_{\boldsymbol{\theta}}(S_t, O_t) \quad . \tag{5.17}$$

While $\widehat{A}_{U_\theta}$ is meant here to be an estimator of the advantage function, it was also found to correspond to the TD error term in the intra-option TD(0) algorithm (5.16) in section 5.3.1. An estimator of this kind was also used by Mnih et al. (2016) in an actor-critic setting so as to maintain only a value function rather than an action-value function in the critic.

Finally, another variant of $\widehat{A}_{U_\theta}$ can be derived by making use of the structure within the utility term. The resulting estimator $\widetilde{A}_{\boldsymbol{\theta}}$ is not only free of primitive actions but also of the $v_{\boldsymbol{\theta}}$ term, which would otherwise have to be computed explicitly or learned separately in addition to $Q_{\boldsymbol{\theta}}$; this at the cost of increased variance compared to (5.17). By recognizing that $U_{\boldsymbol{\theta}}$ is an expectation :

$$\begin{aligned}U_{\boldsymbol{\theta}}(S_{t+1}, O_t) =& (1 - \beta(S_{t+1}, O_t)) Q_{\boldsymbol{\theta}}(S_{t+1}, O) \\ &+ \beta_o(S_{t+1}, O_t) \sum_{o'} \mu_{\boldsymbol{\theta}}\left(o' \,\middle|\, S_{t+1}\right) Q_{\boldsymbol{\theta}}(S_{t+1}, o') \quad ,\end{aligned}$$

we can fully sample through the termination events and option selection process:

$$\widetilde{A}_{\boldsymbol{\theta}}(S_t, O_t, R_{t+1}, S_{t+1}, O_{t+1}) \doteq R_{t+1} + \gamma Q_{\boldsymbol{\theta}}(S_{t+1}, O_{t+1}) - Q_{\boldsymbol{\theta}}(S_t, O_t). \tag{5.18}$$

The distinction between the advantage estimators $\widetilde{A}_{\boldsymbol{\theta}}$ and $\widehat{A}_{U_{\boldsymbol{\theta}}}$ follows from the same reasoning behind augmented TD(0) vs intra-option TD(0) introduced

in section 5.3.1. Hence, we call $\widetilde{A}_{\theta}$ the *augmented advantage estimator*.

5.4 Algorithms and Experiments

This section shows how the option-critic architecture can be instantiated in many different settings.

5.4.1 Four-Rooms Domain

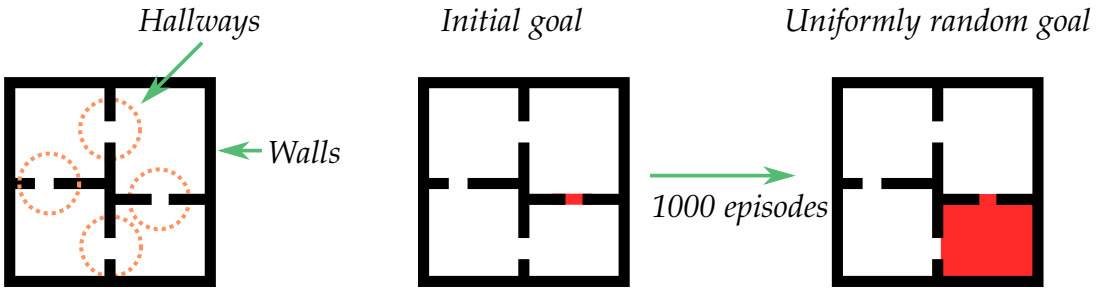


FIGURE 5.3 – Continuing transfer learning experiment in the four-rooms domain. After 1000 episodes, a new goal location is chosen randomly while continuing learning.

The four-rooms domain of Sutton et al. (1999a) has been used frequently to demonstrate new options discovery methods because of its simple structure. While many domains of the same layout can be found in the literature, the experiment presented in this section is based on the description found in (Sutton et al., 1999a, p. 192).

In this instance of the four-room domains, an agent must learn to navigate to a goal initially located in the south *hallway* of the north-east *room* (figure 5.3. Four primitive actions are available in this MDP : move up, down, left or right. Furthermore, primitive actions fail with probability 1/3, in which case the agent moves randomly to any neighboring empty cell reachable under the primitive actions. A primitive action taken in the direction of a wall has no effect and the agent remains in the same state with probability 1. As in Sutton et al. (1999a), the reward structure is 0 on all transitions except for the final transition into the goal state providing a reward of 1. We chose to discount the returns with a discount factor $\gamma = 0.99$.

The goal of this experiment is to assess two important properties of the proposed learning architecture : 1) how fast can we learn to solve this task with options compared to using primitive actions ? 2) can the structure of the learned options provide learning speedups when the task changes ? We put to the test our option-critic architecture for different numbers of options (2, 4, 6 and 8) against agents using SARSA and a policy gradient-based actor-critic architecture using primitive actions only. Each agent would be given 1000 episodes of at most 1000 steps to learn to reach the initial goal from any starting location, after which, a new goal location would be randomly chosen anywhere in the south-east room. We applied this perturbation to the task in a continuing fashion, without resetting the parameters or interrupting the training regime.

Algorithm 5: Option-critic with tabular intra-option Q-learning

$s \leftarrow s_0$
Choose o according to an epsilon-greedy policy of $\mu_{\boldsymbol{\theta}}(s)$
repeat
  Choose a according to $\pi_{\boldsymbol{\theta}}(a \mid s, o)$
  Take action a in s, observe s', r

  $\delta \leftarrow r - Q_{U_{\boldsymbol{\theta}}}(s, o, a)$
  if s' *is non-terminal* **then**
    $\delta \leftarrow \delta + \gamma(1 - \beta_{\boldsymbol{\theta}}(s', o))Q_{\boldsymbol{\theta}}(s', o) + \gamma\beta_{\boldsymbol{\theta}}(s', o) \max_{\bar{o}} Q_{\boldsymbol{\theta}}(s', \bar{o})$
  end
  $Q_{U_{\boldsymbol{\theta}}}(s, o, a) \leftarrow Q_{U_{\boldsymbol{\theta}}}(s, o, a) + \epsilon\delta$

  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} +$
   $\epsilon^{(\boldsymbol{\theta})} \left(\nabla_{\boldsymbol{\theta}} \left(\log \pi_{\boldsymbol{\theta}}(a \mid s)\right) Q_{U_{\boldsymbol{\theta}}}(s, o, a) - \nabla_{\boldsymbol{\theta}} \left(\beta_{\boldsymbol{\theta}}(s', o)\right) \left(Q_{\boldsymbol{\theta}}(s', o) - v_{\boldsymbol{\theta}}(s')\right)\right)$

  if $\beta_{\boldsymbol{\theta}}(s', o)$ *terminates* **then**
    choose new o to the epsilon-greedy policy over options
  end
  $s \leftarrow s'$
until s' *is terminal*

Given the small size of this MDP with its 104 states, we implemented a variant of the option-critic in a simple tabular setting. In this setting, there is no sharing of parameters and every option maintains $|\mathcal{S}| \times |\mathcal{A}|$ weights for its policy and $|\mathcal{S}|$ to represent its termination condition. We choose to learn $Q_{U_{\boldsymbol{\theta}}}$ and $Q_{\boldsymbol{\theta}}$ separately using intra-option Q-learning, which made easier to compute the advantage function as well as to implement policy improvement for the policy over options by an epsilon-greeedy strategy. We also used the log-likelihood

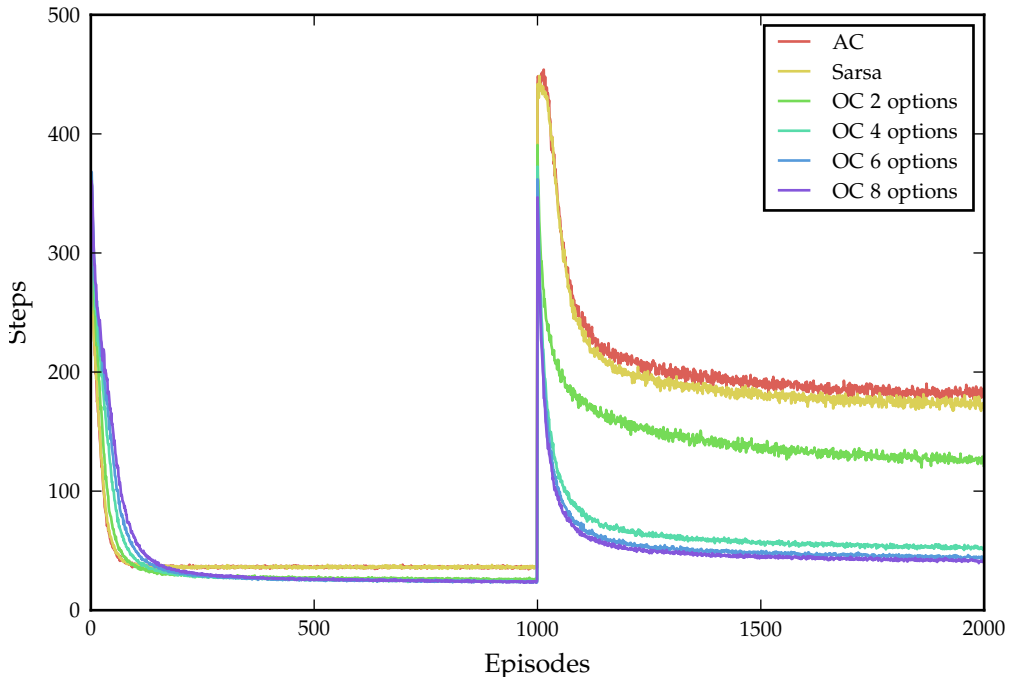


FIGURE 5.4 – Learning in the four-rooms domain: steps per episode (lower is better), averaged over 5000 runs. After 1000 episodes, the goal is moved randomly to a new location. "AC" stands for actor-critic, and "OC" for option-critic.

ratio estimator for estimating gradient for option policies, which were implemented as softmax policies with parameters initialized to zero. We represented the termination conditions using the sigmoid function with no bias term. By initializing the weights of termination conditions to zero, the agent would start with options lasting on average two steps from the beginning of training (because sigmoid(0) = 1/2).

For all the agents learning under the option-critic architecture, we used the following hyperparameters : a baseline for the gradient estimator for option policies, a rate of random options of 1% for the epsilon-greedy policy options, a temperature of 0.01 for the option policies, a learning rate $\epsilon = 0.5$ for the critic and $\epsilon^{(\boldsymbol{\theta})} = 0.25$ for the actor components and no regularization with $\eta = 0$. The agent using only primitive actions were set to comparable parameters. In the actor-critic agent, we set $\epsilon = 0.5$, $\epsilon^{(\mathbf{w})} = 0.25$ and a temperature of 0.01 for the same softmax policy parameterization. Finally, the SARSA agent used an epsilon-greedy policy over primitive actions with 1% randomness and a learning rate of 0.5.

To obtain a more faithful measure of the expected performance of each agent,

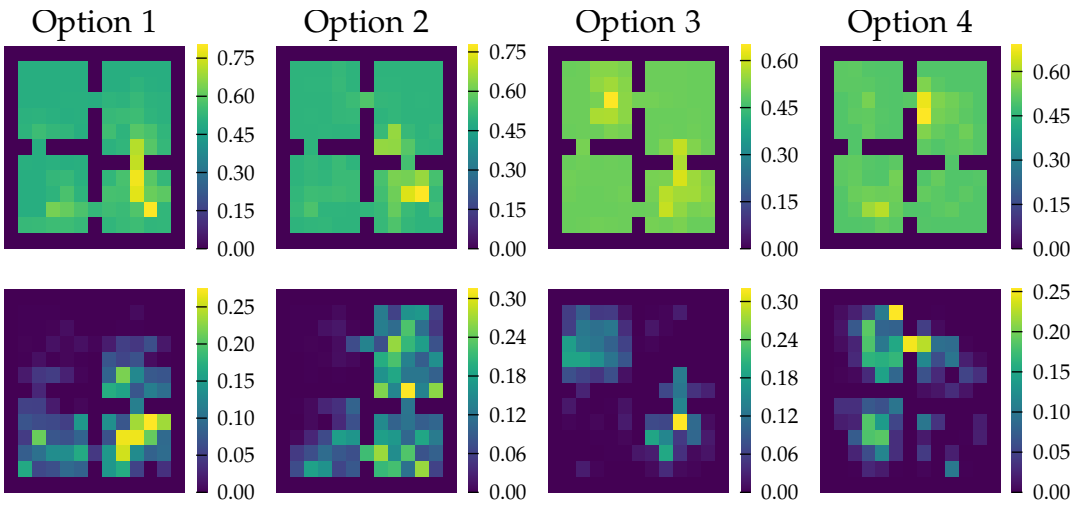


FIGURE 5.5 – Learned options after 2000 episodes. **Top row:** Termination probabilities for every options ie. $\beta_\theta(\cdot, o)$ **Bottom row**: option policy weight corresponding to the preferred action (argmax) at every state

we ran 5000 simulations the same experiment and averaged the learning curves in figure 5.4. We can see that during the first 100 episodes, the learning curves are ordered according to the number of options : learning with 8 options takes more effort than without. This phenomenon is not surprising when learning from scratch, without function approximation and parameters sharing : more options means more parameters to learn. Despite this challenge, the performance gap remains small, with a curve catching up on the next one within a few episodes of difference. A complete reversal of the curves takes place starting from the 1000th episode when the task is moved randomly to a new location. While a small cost had to be incurred in the initial stages of learning, figure 5.4 suggests that building additional knowledge inside more options proves to be useful once the agent is facing the new task.

Despite not using any regularization, figure 5.5 shows that the termination conditions learned after 2000 episodes are not degenerate : the termination probabilities are not saturated and the patterns of activation are specialized locally over the state space. The weight vectors learned for the option policies are also different from each other, preferring different primitive actions in the same state. The bottom panel of figure 5.5 illustrates this characteristic by plotting the weight of maximum value across all actions, at every state, for every option policy. We can note that option 3 seems to value acting in the north-west room more than the other options while option 4 has larger weights around the hallway from the north-west room to the north-east one.

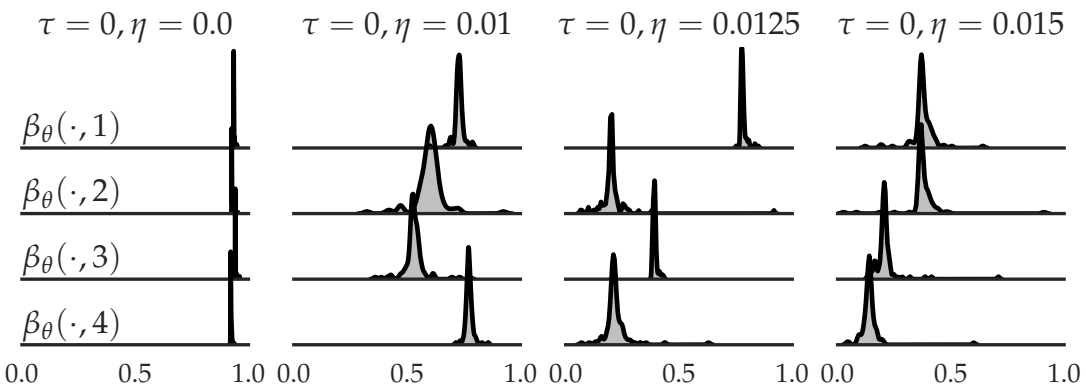


FIGURE 5.6 – Effect of regularization on the termination conditions parameterized with a bias term. Each graph is a kernel density plot over the termination probabilities $\beta_{\boldsymbol{\theta}}(\cdot, o)$ for each option, at all states. The vertical axis spans the options while the horizontal one is over different regularization coefficients.

In this domain, it seems that the choice of initialization to zero weights as well as the lack of bias term provide a sufficient prior favoring well-defined options. Indeed, when using a bias term and no regularization, figure 5.6 shows that the system converges after 2000 steps to a solution where options terminate almost immediately. By increasing the value of the η coefficient (margin), the option-critic architecture learns a set of options that terminate less often ($\beta_{\boldsymbol{\theta}}$ is driven down). For large values of the switching cost regularizer, the system also becomes more prone to using fewer options at the level of μ as they would relinquish control less often.

5.4.2 Deep Q-Network with Option-Critic (DQN-OC)

To showcase the flexibility of the proposed approach and its ability to learn in large environments, we adapted the deep network architecture of Mnih et al. (2015) to learn options in the Arcade Learning Environment (ALE) (Bellemare et al., 2013) under a variant of the option-critic architecture that we call DQN-OC : Deep Q-Network with Option-Critic. As shown in figure 5.7, we designed our network to take as input a concatenation of the last four frames converted to grayscale and scaled to 84×84 pixels. The resulting tensor would then be filtered through a sequence of convolution layers, the first one consisting of 32 filters of size 8×8 with a stride of 4, 64 filters of size 4×4 and stride 2 in the second and finally 64 filters of is size 3×3 with a stride of 1 in the third convolution layer.The resulting filtered output would then be fed through a ReLU activation function to produce a dense output of 512 units.

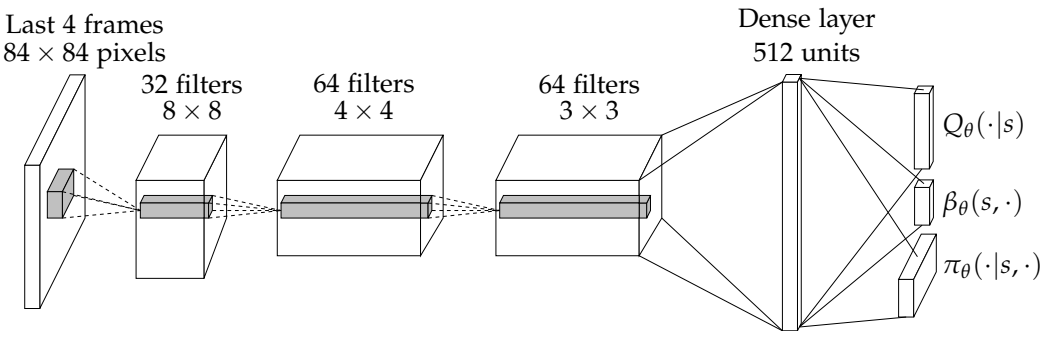


FIGURE 5.7 – Network architecture for DQN-OC. All the layers up to the penultimate one are the same as in the DQN architecture of Mnih et al. (2015).

In the original DQN architecture, the penultimate layers is used to approximate the action-values at the output of the network. Instead of representing options by separate networks, we chose instead to attach two additional *heads* to represent the termination conditions and option policies. Therefore, for each concatenation of four frames fed as input to the network three outputs are simultaneously generated : the option values through a linear mapping $Q_{\boldsymbol{\theta}}(\cdot, s) \in \mathbb{R}^{|\mathcal{O}|}$, the termination probabilities $\beta_{\boldsymbol{\theta}}(s, \cdot) \in \mathbb{R}^{|\mathcal{O}|}$ via a linear-sigmoid function and the action probabilities $\pi_{\boldsymbol{\theta}}(\cdot|s, \cdot) \in \mathbb{R}^{|\mathcal{O}|\times|\mathcal{A}|}$ by a softmax layer.

A benefit of the approach taken in section 5.1 is that it led to the lemmas (5.4) and (5.5) that can be used to learn different parts of a system independently (as opposed to Levy and Shimkin (2012) for example). This allows us to easily combine the control algorithm of Mnih et al. (2015) to learn a policy over options and option values, which can then be used to get gradient estimators for the termination conditions and option policies. To address the instability issues pertaining to nonlinear function approximation, the DQN learning algorithm exploits two main ideas : experience replay (Lin, 1992) and the use of secondary *target* network to form TD update targets. In the context of this work, we used a similar strategy but in combination with intra-option Q-learning. Denoting $Q_{\boldsymbol{\theta}}(S_t, O_t\,;\,\mathbf{w}_t^-)$ for the option-value function under the target network at time t, the intra-option Q-learning loss function $L_{\boldsymbol{\theta}}$ is :

$$L_{\boldsymbol{\theta}}(\mathbf{w}_t) = \mathbb{E}\left[\left(R_{t+1} + \gamma \widehat{U}_{\boldsymbol{\theta}}(S_{t+1}, O_t\,;\,\mathbf{w}_t^-) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t\,;\,\mathbf{w}_t)\right)^2\right] \quad ,$$

where:

$$\widehat{U}_{\boldsymbol{\theta}}(S_{t+1}, O_t\,;\,\mathbf{w}_t^-) \doteq (1-\beta_{\boldsymbol{\theta}}(S_{t+1}, O_t))\widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, O_t\,;\,\mathbf{w}_t^-) + \beta_{\boldsymbol{\theta}}(S_{t+1}, O_t) \max_o \widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, o\,;\,\mathbf{w}_t^-) \ .$$

The expectation here is taken under the distribution of samples:

$$(S_t, O_t, R_{t+1}, S_{t+1}) \ ,$$

drawn uniformly at random from an experience replay buffer. The weight vector $\mathbf{w}_t$ for the option-value function $Q_{\boldsymbol{\theta}}$ is then updated by taking the gradient of the loss $L_{\boldsymbol{\theta}}$. After 10000 parameter updates, the target network is set to the network that had been updated throughout that period.

Having chosen by design to output only $Q_{\boldsymbol{\theta}}$ in our network, we have recourse to the intra-option advantage estimator (5.17) to cope with the lack of explicit $Q_{U_{\boldsymbol{\theta}}}$ estimates in the gradient for option policies. The option policies are then updated at time t with :

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_t + \epsilon_t^{(\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_t} \left(\log \pi_{\boldsymbol{\theta}_t}\left(A_t \mid S_t, O_t\right)\right) \hat{A}_{U_{\boldsymbol{\theta}_t}}(S_t, O_t, R_{t+1}, S_{t+1}\,;\,\mathbf{w}_t)$$

where:

$$\hat{A}_{U_{\boldsymbol{\theta}_t}}(S_t, O_t, R_{t+1}, S_{t+1}\,;\,\mathbf{w}_t) \doteq R_{t+1} + \gamma \widehat{U}_{\boldsymbol{\theta}}(S_{t+1}, O_t\,;\,\mathbf{w}_t) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t\,;\,\mathbf{w}_t) \ .$$

As for the termination conditions, we compute their gradients using the advantage function plus a *margin* term $\eta = 0.001$ and horizon $\tau = 0$ (see section 5.2 for details):

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_t + \epsilon_t^{(\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_t} \left(\beta_{\boldsymbol{\theta}_t}(S_{t+1}, O_t)\right) \left(\widehat{A}_{\boldsymbol{\theta}_t}(S_{t+1}, O_t\,;\,\mathbf{w}_t) + \eta\right) \ .$$

In both cases, the gradients are estimated along the stationary distribution induced by the set of options and the policy over them but not under the distribution over samples from the experience replay buffer (which may not properly reflect the current $\boldsymbol{\theta}_t$).

We used a fixed learning rate of $\epsilon_t^{(\boldsymbol{\theta})} = 1/4000$ for the actor components and updated the critic using RMSProp (Tieleman and Hinton, 2012) as in Mnih et al. (2015). We also adopted the same decay strategy as in Mnih et al. (2015) for the ϵ parameter, starting with $\epsilon = 1$ and ending with $\epsilon = 0.1$ but using $\epsilon = 0.05$

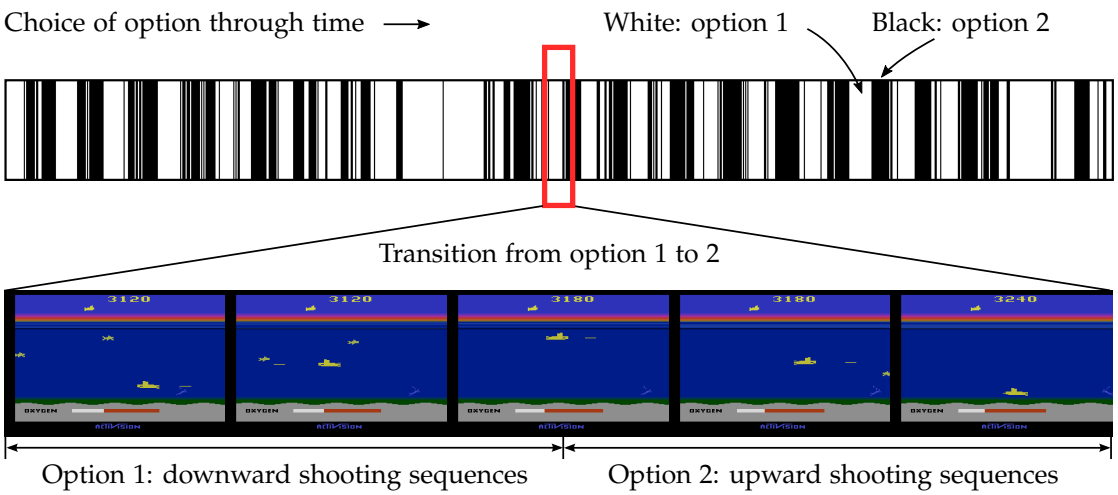


FIGURE 5.8 – Up/down specialization in the solution found by option-critic when learning with 2 options in Seaquest. The top bar shows a trajectory in the game, with "white" representing a segment during which option 1 was active and "black" for option 2.

during the test phase. Finally, to prevent option policies from collapsing to deterministic policies (Williams and Peng, 1991; Mnih et al., 2016) we used an entropy regularizer of 0.01 in the corresponding gradient.

We evaluated option-critic in *Asterisk*, *Ms. Pacman*, *Seaquest* and *Zaxxon*. For comparison, we allowed the system to learn for the same number of episodes as Mnih et al. (2015) and did not perform environment-specific parameter optimization. Despite having more parameters to learn, the learning architecture presented in this section was capable to learn in all games with options, from the ground up, within 200 epochs. In Asterisk, Seaquest and Zaxxon, the final results (table 5.1) surpass the scores reported in Mnih et al. (2015).

To gain an intuitive understanding of the kind of options learned by our approach, we ran a learning experiment with only 2 options in the game of Seaquest. After having learned under the same experimental setup as for the other games, we collected trajectories from the agent interacting with the environment and plotted the choice of option through time. As shown in figure 5.8 that the agent has found a solution that alternates between the two options at a regular interval. Upon closer inspection to the transitions between options, we observed that option 1 specialized in action sequences during a resurfacing maneuver while option 1 would be involved while diving back to the bottom of the seafloor. It is interesting to note that the same kind of options was found by a graph partitioning heuristic in Krishnamurthy et al. (2016).

Algorithm	Amidar	Asterix	Breakout	Hero	Pong	Seaquest	MsPacman	Zaxxon
Mnih et al. 2015 (DQN)	739.5	6012.0	**401.0**	19950.0	18.9	5286.0	2311.0	4977.0
Mnih et al. 2016 (A3C)	283.9	6723.0	551.6	**28765.8**	11.4	2300.2	594.4	2659.0
A2OC No deliberation cost	502.9	**7542.7**	365.7	12253.6	**20.7**	1736.4	1625.2	21.4
A2OC $\lambda = \gamma, \eta = 0.005$	775.6	5326.6	386.2	10752.8	20.6	1702.7	2074.0	**4459.6**
A2OC $\lambda = \gamma, \eta = 0.010$	808.6	6850.6	395.8	26910.6	**20.7**	1688.9	2073.0	3776.0
A2OC $\lambda = \gamma, \eta = 0.015$	**809.5**	5830.3	373.0	24737.4	20.6	1690.8	**2320.4**	2960.0
A2OC $\lambda = \gamma, \eta = 0.020$	741.7	6335.3	384.2	28843.7	4.3	1675.7	2323.3	2405.0
A2OC $\lambda = \gamma, \eta = 0.025$	780.3	4798.1	377.8	24973.7	12.2	1676.0	2115.1	399.1
A2OC $\lambda = \gamma, \eta = 0.030$	764.6	4840.0	376.6	19849.3	20.5	1691.0	2185.6	710.9
A2OC $\lambda = 0., \eta = 0.010$	829.8	6169.5	398.5	25310.2	20.5	1743.9	2085.1	29.4
A2OC $\lambda = 0., \eta = 0.020$	799.2	7798.8	389.8	22691.7	20.4	1652.0	2200.4	32.4
A2OC $\lambda = 0., \eta = 0.030$	808.1	4986.3	381.5	20765.7	17.8	1671.1	2133.5	18.0
DQN-OC No deliberation cost	196.8	207.7	254.2	9547.3	14.8	**5161.6**	2207.2	2908.1
DQN-OC $\lambda = 0., \eta = 0.010$	238.2	7064.3	186.9	8802.2	-12.6	4644.4	2226.4	4081.8

TABLE 5.1 – Final scores for DQN-OC and A2OC averaged over 5 independent runs.

5.4.3 Faster Learning with Advantage Asynchronous Option-Critic (A2OC)

Experience replay tends to be costly in practice when implementing both DQN (Mnih et al., 2015) and its adaptation for learning options (DQN-OC) presented in the previous section. Instead of de-correlating the update target from $Q_{\boldsymbol{\theta}}$ by experience replay, Mnih et al. (2016) found that applying policy gradient updates asynchronously from many parallel threads has a similar effect. Furthermore, this asynchronicity provides a stabilizing effect that renders the target network of DQN superfluous. In our experience with this method, we found the wall-clock time can be cut at least in half and further speed improvements can be made by increasing the number of threads.

In reference to the Asynchronous Advantage Actor-Option-Critic (A3C) (Mnih et al., 2016) architecture, we designed an Asynchronous Advantage Option-Critic (A2OC) variant using the same network architecture (figure 5.7) as for DQN-OC from the previous section. However, rather than updating the option-value estimates using the one-step intra-option Q-learning loss, we used an n-steps target over the augmented state space as previously shown in (5.15). The loss for $\widehat{Q}_{\boldsymbol{\theta}}$ then becomes :

$$L_{\boldsymbol{\theta}}^{(n)}(\mathbf{w}) = \mathbb{E}\left[\left(\sum_{k=0}^{n-1}\gamma^k R_{t+k} + \gamma^n \widehat{Q}_{\boldsymbol{\theta}}(S_{t+n}, O_{t+n}\,;\,\mathbf{w}) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t\,;\,\mathbf{w})\right)^2\right],$$

and where the expectation is no longer taken under the experience replay distribution but rather under the stationary distribution induced by the options

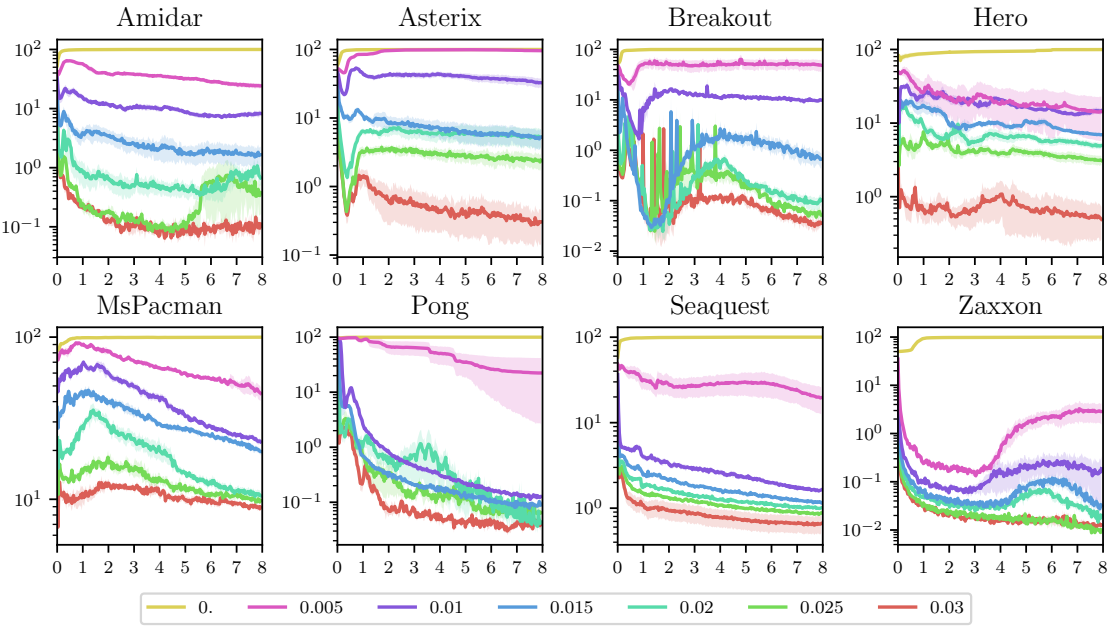


FIGURE 5.9 – Evolution of the empirical termination frequencies for different values of the regularization coefficient during learning (log scale). Without regularization, the options terminate after only one step. Larger values of the regularization coefficient result in more commitment during option execution.

and the policies over them. To avoid potential off-policy complications arising from n-steps return overlapping multiple option decisions, we chose to align n with the number of steps to termination under $\beta_{\boldsymbol{\theta}}$. In the spirit of the *augmented estimators* of section 5.3.2, we also chose to work with the augmented advantage estimator (5.18) when applying the gradient updates to the option policies :

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \epsilon_t^{(\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}} \left(\log \pi_{\boldsymbol{\theta}_t} \left(A_t \mid S_t, O_t \right) \right) \widetilde{A}_{\boldsymbol{\theta}}(S_t, O_t, S_{t+1}, O_{t+1}; \mathbf{w})$$

where:

$$\widetilde{A}_{\boldsymbol{\theta}}(S_t, O_t, R_{t+1}, S_{t+1}, O_{t+1}; \mathbf{w}) = R_{t+1} + \gamma \widehat{Q}_{\boldsymbol{\theta}}(S_{t+1}, O_{t+1}; \mathbf{w}) - \widehat{Q}_{\boldsymbol{\theta}}(S_t, O_t; \mathbf{w})$$

Finally, we applied A2OC not only over various values for the margin ($\tau = 0$) but also to the case where $\tau = \gamma$ and the switching cost is subtracted from the instantaneous reward (section 5.2.1). Hence, whenever a termination event is sampled under the current option, the instantaneous reward is transformed into $\widetilde{R}_{t+1} \doteq R_{t+1} + \eta$. It is over such transformed rewards $\widetilde{R}_{t+1}$ that the above n-steps targets are formed, which then permeates into every component via the option-critic updates.

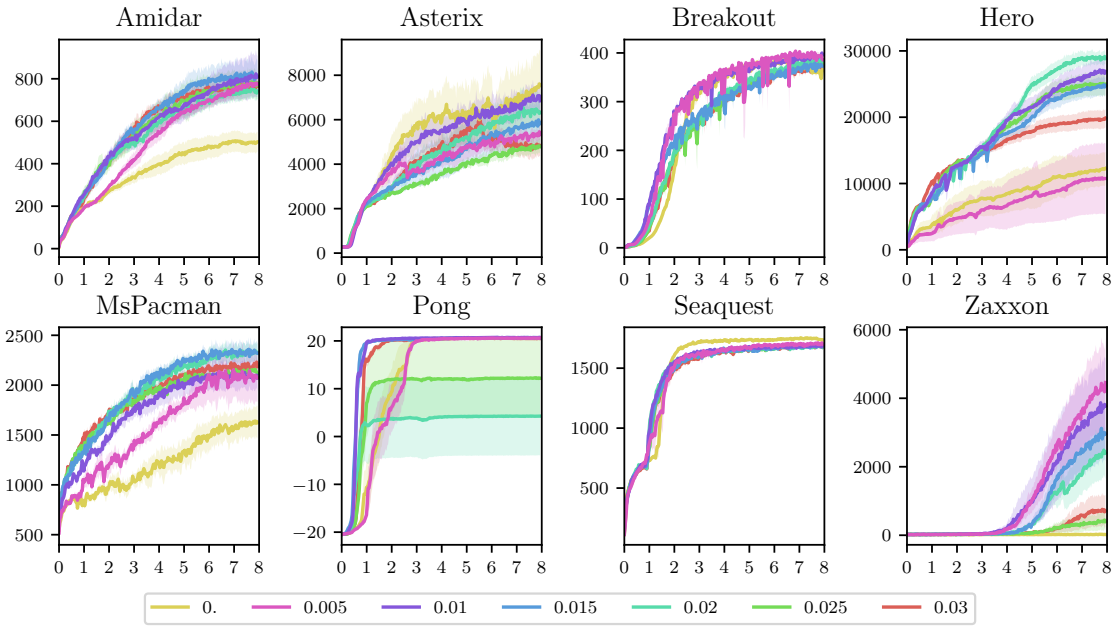
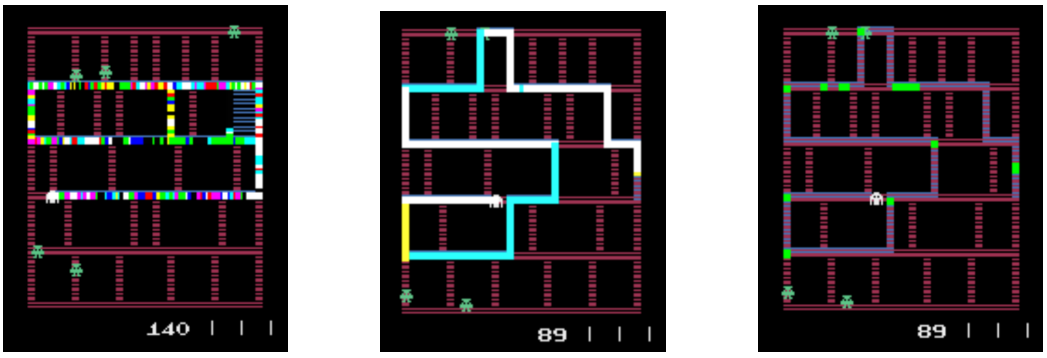


FIGURE 5.10 – Learning curves for A2OC averaged over 5 runs for different values of the regularization coefficient: intermediate values work best but also depend on environmental properties per game.

The effect of the switching cost regularizer at $\tau = \gamma$ can be clearly observed in 5.9 where for $\eta = 0$ (no regularization) the average rate of termination through training is quickly driven up to 100% while larger values of η result in less termination (more commitment). When consulting the corresponding final scores in table 5.1, we see that intermediate values of the regularization coefficient η result in the best performance: neither the one-step options typically found with $\eta = 0$ nor the long and inflexible options for larger values of η perform best. For example, for the games of Amidar and Asterix, the performance is increasing with the regularization coefficient until $\eta = 0.010$ where the performance then starts to degrade. The interplay of the regularization coefficient with the returns makes the choice of η dependent on the reward structure proper to each game. Because its effect is proportional to the ratio with the state values, environments where high rewards are frequently encountered would also call for larger values of the regularization coefficient to lead to a noticeable effect.

The qualitative effect of the regularizer is easy to perceive in figure 5.11 showing a trajectory of the agent in the game Amidar. In this visualization, a segment of color represents a subtrajectory executed under a given option. When



(A) No regularization (B) With regularization (C) Termination events

FIGURE 5.11 – Effect of the switching cost regularizer with $\tau = \gamma$ in the game Amidar. Each segment of color is a subtrajectory under a given option. The last panel (c) shows the location of the termination events along the trajectory shown in (b).

no regularization is applied, the resulting options tend to terminate more frequently and the policy over options tend to use more of the 8 options available. However, when a switching cost penalty is incurred, the A2OC agent learns to persist longer with the same option. The contiguity of the color segment is not due here to the same option being chosen repeatedly when terminating instantly but is rather shows a true – uninterrupted – commitment to the same option policy. Termination was also observed to take place mostly at the intersections from the underlying maze structure that represent key decision points.

5.4.4 Related Optimization-Based Discovery Methods

Comanici and Precup (2010) proposed a gradient-based algorithm for learning termination functions for semi-Markov options. The authors considered a special case where the termination function is parameterized using the logistic function over an accumulation (the semi-Markov aspect) of features.

Levy and Shimkin (2012) extended the work of Comanici and Precup (2010) for learning all components of the options framework (in the Markov case). This work is related to our approach in that both consider the problem of jointly learning parameterized options. However, as shown in 3.3, option-critic is built directly from the intra-option Bellman equations (Sutton et al., 1999a) which led to the structure of the resulting gradients to be exposed directly for the first time. The derivation approach in Levy and Shimkin (2012) is in fact based on

the construction of a policy over both an augmented state and action spaces. While mathematically equivalent (after some simplification) to our construction in section 3.3, Levy and Shimkin (2012) compute the required gradients by representing explicitly their augmented policy. In option-critic, the gradient of each component of the options framework are presented separately (but yet work together to solve the overall task). This allowed us to design new gradient estimators with reduced variance (section 5.3.2 and allowed us to better understand how *long* options can be promoted using switching cost regularizers (section 5.2.

Daniel et al. (2016) showed how parameterized options of a specific kind can be learned via expectation maximization (EM). They derived their approach by taking a graphical model perspective on options execution and learning through which is carried inference. Some of the groundwork for Daniel et al. (2016) was laid in Daniel et al. (2012) for the mixture execution case (and not the call-and-return one usually assumed with options).

Comanici and Precup (2010), Levy and Shimkin (2012) and Daniel et al. (2016) considered learning from the expected discounted return. As we have seen in section 5.2, optimizing directly for this objective can result in degenerate options that terminate at every step. Our work seems to be the first to describe and address this problem through a new regularizer whose interpretation finds its roots in bounded rationality (Simon, 1957).

Also based on the policy gradient theorem, Mankowitz et al. (2016) proposed the ASAP framework for learning parameterized *skills*. The notion of skills defined in this thesis is very much related to options, but differs from the fact that initiation sets and termination conditions are coupled and that the execution model is more akin to a mixture model than call-and-return. In this framework, at every step an agent is executing the skill within the current partition of the state. The boundaries of this partition being *soft*, the agent need not to commit to the same option as in the underlying augmented transition function of the options framework (section 3.3).

Kulkarni et al. (2016) showed the first successful results of temporally extended actions in the challenging game *Montezuma's Revenge*. The learning architecture proposed in this paper is valued-based and relies on the DQN algorithm Mnih et al. (2015). A set of temporally extended actions is specified to the

agent in the form of specific targets to reach in the game (picking the key for example). When the agent gets close to the specified targets, the corresponding option is terminated and the option policy is updated to maximize its *intrinsic reward* (Singh et al., 2004b). In that sense, Kulkarni et al. (2016) is less of an option discovery method than a learning architecture for *learning with* specified options.

The subgoal-based approach of Kulkarni et al. (2016) also relates to Silver and Ciosek (2012) who made use of the intra-option Bellman equations and model composition to discovery options to optimize, by dynamic programming, those targets. In Mann et al. (2015), the existence of a local planner is assumed, which then allows an agent to learn options that navigate between *landmarks*. Andreas et al. (2017) also used a form of partial specification called *policy sketches*, which are also learned in a policy gradient setting.

The decoupled two-tiered architecture of Kulkarni et al. (2016) also resembles the work of Vezhnevets et al. (2017), which in turns is based on (Dayan and Hinton, 1992). The system described by Vezhnevets et al. (2017) also makes use of a gradient-based approach for learning a *Manager* and a *Worker*: the former can be thought as a policy over options, while the latter is amenable to the decision levels of options over primitive actions. The worker component produces primitive actions in response to an intrinsic reward set by the manager.

5.5 Discussion

In this work, we made the assumption that options are available everywhere to simplify the development of our learning architecture. In order to learn initiation sets in the gradient-based framework of option-critic, we would however need a redefinition of initiation sets as functions belonging to a parametric family of functions. It would then be more appropriate to talk of such initiation sets as *parameterized initiation functions*. As required under assumption 4, such parameterized initiation functions would need to be randomized to yield smooth gradients. But what is the execution model associated with such randomized objects ? When should the corresponding *initiation events* be sampled

? Answering this question under the original semantics of the call-and-return execution model is not obvious.

To get a clearer picture of how initiation sets could be incorporated into our framework, we shall first go back to the Bellman equations for options. When μ is a randomized policy, the restriction on the set of available options lead to a sub-probability distribution over options. Hence, a re-normalization on the support $\mathcal{O}(s) \doteq \{o \in \mathcal{O} : s \in \mathcal{I}_o\}$ is needed to be able to sample from μ under the appropriate constraints. We propose at this point to generalize the notion of initiation sets to a new concept of *influence function* $l : \mathcal{S} \times \mathcal{O} \to \mathbb{R}_{\geq 0}$ that enters the Bellman equations in the following manner:

$$\mu_l(o \mid s) \doteq \frac{\mu(o' \mid s')\, l(s', o')}{\sum_x \mu(s', x) l(s', x)}$$
$$Q_{\mathcal{O}}(s, o) = b(s, o) + \sum_{s'} F(s, o, s') \sum_{o' \in \mathcal{O}} \mu_l\left(o' \,\middle|\, s'\right) Q_{\mathcal{O}}(s', o')$$

The 0/1 semantics of initiation sets then still hold under this definition. For example, imagine that for some state and option $s \notin \mathcal{I}_o$ (or equivalently $o \notin \mathcal{O}(s)$). We could specify this restriction as an initiation function where $l(s, o) = 0$ for the given state-option pair. The resulting policy over options would then assign $\mu_l(o \mid s) = 0$ and the agent would never sample o, as required. Function approximation can also be used naturally with an influence function. Despite being deterministic, a parameterized influence function $l_{\boldsymbol{\theta}}$ can be learned under the same gradient-based framework implied by theorem 5.2 for randomized options. To see this, it suffices to note that the parameters of $l_{\boldsymbol{\theta}}$ are also parameters of the composite policy $\mu_{l_{\boldsymbol{\theta}}}$ function. Hence, if $\mu_{l_{\boldsymbol{\theta}}}$ can be evaluated explicitly, taking its gradient is the same as taking the gradient of any other parameterized policy over options.

The *recognizers* framework (Precup et al., 2006) is closely related to the proposed concept of influence function in the normalized form μ_l. Recognizers are defined as functions $c : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that *filter* actions from by a behavior policy in order to induce a related target policy whose importance sampling ratios have lower variance. Putting the problem of off-policy learning aside, influence functions also act as filters on the choice of options by a policy over options μ.

5.5.1 Choice of Objective

In this thesis, we derive the gradient of the expected discounted return with respect to some parameterized options. From this result, we then developed new estimators in section 5.3.2 for the gradients associated with the option policies and termination conditions, which when used for stochastic ascent lead to the option-critic learning architecture. However, as shown in 5.2, optimizing for the expected discounted return might lead to degenerate solutions unless the objective is properly regularized. To this regard, we introduced a family of *switching cost* regularizers in section 5.2.1 that encourages solutions in which options persist for many steps.

Despite promising results based only on the regularized expected discounted return, the question of *what* options should optimize remains largely open. In that sense, the option-critic architecture is more of an answer to the *how* options can be learned to maximize a given reward-like objective. The ability of option-critic to align the construction of options with a given objective will ease future research on what that objective should be. For example, the intrinsic motivation signal defined in Kulkarni et al. (2016) could be optimized readily under the option-critic architecture and in fully an integrated manner. An alternative to the expected discounted return in the option-critic architecture could also be based on the notion of *eigenpurposes* (Machado et al., 2017), which were shown to reflect interesting temporal structure in the ALE domains.

Given that options are not needed for achieving more reward in a single MDP, an important future direction of research is to consider learning options across many tasks (Singh, 1992b; Thrun and Schwartz, 1995) or in a continuum of tasks encountered in a lifelong (Ruvolo and Eaton, 2013) or continual fashion (Ring, 1991). Going back to the perspective of options as a representation (Minsky, 1961; Simon, 1969), good options ought to be those with which learning and planning become faster when facing new tasks. Formulating precisely what *faster* means really is the crux of the matter when it comes to designing a better objective. In this regard, we might gain useful insights on that problem by taking a meta-learning perspective (Hampson and Volper, 1986; Schmidhuber, 1987; Bengio, 1990; Sutton, 1992; Baxter, 1995; Chen et al., 2017; Finn et al., 2017).

Chapter 6

Summary

This thesis showed how options can be learned fully autonomously to maximize the expected sum of discounted rewards. Our approach, option-critic, is based on the blueprints of policy gradient methods which give our system the ability to learn in a continual manner over arbitrary spaces. The system does not require demonstration, expert knowledge or manual decomposition (although these could be leveraged if desired). Furthermore, by virtue of being a gradient ascent method, we know that the options learned by option-critic will be such that they align with the overall objective of simply solving the task at hand. This was demonstrated with experiments (section 5.4) and theory (section 5.1.1).

Our derivation of the option-gradient theorem in section 5.1 helped us gain more insights as to how the optimization process constructs options. For example, we have seen that the gradient of the option policies is such that in order to optimize for the global performance of the system, the corresponding *critic* would have to be defined over triples of state, action and option. This means that a local change to the policy of an option is always made by taking into account its impact on other parts of the system. This allows us, by construction, to guarantee that the learning system will converge to a solution adapted to the task. In previous approaches based on pseudo-reward heuristics, overall convergence would have been difficult to guarantee as they would typically optimize each component in isolation, and assume access to a reset state. Our approach, on the other is fully integrated and can potentially learn

about all other options at all times. In light of recent results obtained by Vezhnevets et al. (2017), it would however be interesting to see if the option-critic gradients can be decoupled while maintaining its convergence properties.

We also found an interesting interpretation of the termination gradient theorem (5.4) as a gradient-based counterpart to the interruption operator from earlier work by Sutton et al. (1999a). Here, if an option is deemed *advantageous*, the termination gradient pushes the parameters of the termination function in the direction which makes the option last longer. Conversely, if the critic values suggest that switching to a different option may yield more return, the termination function corresponding to this option increases so as to terminate more often. In general however, a learning system optimizing only for the discounted return may not see any benefit in committing to the same option for long period of time. Given the opportunity to switch option, it might as well just do it: a phenomenon that we have observed empirically as well in section 5.4.

Based on this insight, we developed a new regularization mechanism that promotes temporal commitment. Inspired by the bounded rationality framework of Simon (1957), we proposed (section 5.2) the idea of a *deliberation cost* that penalizes for frequent switches. The rationale behind this idea is that options ought to be *cheap* to execute compared to choosing primitive actions at every step. Once an option has been chosen, the cost of executing actions *within* is assumed to be negligible compared to re-deciding from scratch. From that perspective, it then makes sense to have long options because it is then easier to amortize the decision cost. Because option-critic can be applied to any objective that is an expected sum of *rewards*, we showed the deliberation cost can be incorporated without any change to the architecture. The effectiveness of this approach was shown empirically and led to more interpretable options.

Our bounded rationality perspective on options led us to view good options as those which make learning and planning *faster*. With our deliberation cost regularizer, we assumed a particular cost structure which may not necessarily reflect the real computation cost involved in the execution of the agent. Future work may consider the possibility of also using actual *measured* computation cost associated with the implementation of an agent running on some given hardware. This approach may prove useful on a system like AlphaGo (Silver et al., 2016) for example, where the cost of querying the next move involves

a lengthy computation during Monte Carlo Tree Search. Another implication of the bounded rationality perspective is that *good* options may also emerge naturally out of an environmental setup that involves an explicit missed opportunity cost. The idea here would depart from the current synchronous execution model, and decouple the agent from its environment so as to introduce opportunity cost: thinking for too long would have real consequences.

This idea that good options play a role in amortizing decision cost was also found in our theoretical analysis of multi-step RL methods and options in section 4.6. We have in fact shown that multi-step methods (including options) have a corresponding interpretation as *matrix splittings* methods (Varga, 1962) which aimed at speeding up linear iterative solvers. Such matrix splitting methods also admit a complementary vocabulary based on the notion of matrix preconditioners. As with options, the design of general and fast preconditioners is a longstanding problem of numerical analysis. Sometimes, good preconditioners can be found when problem-specific knowledge is available. However, the manual design of preconditioners, and of options, quickly become a tedious process for large problems or when only partial knowledge about the domain is available. When solving a single problem with options, it is also clear from the connection with preconditioners that the initial setup cost and subsequent cost per preconditioned iteration should not outweigh the cost associated with the original problem. This new connection between matrix preconditioning and options opens new research avenues for learning and analyzing options in a lifelong learning setting.

Bibliography

Mark B. Ring. Incremental development of complex behaviors through automatic construction of sensory-motor hierarchies. In *ICML*, pages 343–347, 1991.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999a.

Herbert A. Simon. *Models of man: social and rational; mathematical essays on rational human behavior in society setting*. Wiley, 1957.

Richard S. Varga. *Matrix iterative analysis*. Prentice-Hall, Englewood Cliffs, 1962.

Pierre-Luc Bacon and Doina Precup. Constructing temporal abstractions autonomously in reinforcement learning. *AI Magazine*, 39(1):39–50, 2018.

Pierre-Luc Bacon and Doina Precup. A matrix splitting perspective on planning with options. In *NIPS Continual Learning and Deep Networks Workshop*, 2016.

Pierre-Luc Bacon and Doina Precup. Unifying multi-step methods through matrix splitting. In *3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making*, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. Learning options end-to-end. In preparation, 2018.

Pierre-Luc Bacon and Doina Precup. Learning with options: Just deliberate and relax. In *NIPS Bounded Optimality and Rational Metareasoning Workshop*, 2015a.

Pierre-Luc Bacon and Doina Precup. Learning with options: Just deliberate and relax. In *NIPS Bounded Optimality and Rational Metareasoning Workshop*, 2015b.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option : Learning options with a deliberation cost. In *Thirthy-first AAAI Conference On Artificial Intelligence (AAAI)*, 2018.

Jean Harb. Learning options in deep reinforcement learning. Master's thesis, McGill University, 2016.

Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, first edition, 1969.

Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995. ISSN 0001-0782.

Gerry Tesauro, David Gondek, Jonathan Lenchner, James Fan, and John M. Prager. Analysis of watson's strategies for playing jeopardy! *J. Artif. Intell. Res.*, 47:205–251, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529 (7587):484–489, 01 2016.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49 (1):8–30, January 1961.

Richard Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artif. Intell.*, 3(1-3):251–288, 1972.

Benjamin Kuipers. Commonsense knowledge of space: Learning from experience. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'79, pages 499–501, San Francisco, CA, USA, 1979. Morgan Kaufmann Publishers Inc. ISBN 0-934613-47-8.

Richard Earl Korf. *Learning to Solve Problems by Searching for Macro-operators*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1983.

Glenn A. Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3:285–317, 1989.

Gary L. Drescher. *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1991.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*, pages 271–278, 1992.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1094–1099, 1993.

Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *IJCAI*, pages 1121–1127, 1995.

Nils Nilsson. Eye on the prize. *AI Mag.*, 16(2):9–17, September 1995.

Richard S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

Richard S. Sutton. Introduction to reinforcement learning with function approximation. Tutorial Session of the Neural Information Processing Systems Conference, 2015a. URL `http://media.nips.cc/Conferences/2015/tutorialslides/SuttonIntroRL-nips-2015-tutorial.pdf`.

Timothy van Gelder and Robert F. Port. It's about time: An overview of the dynamical approach to cognition. In Robert F. Port and Timothy van Gelder, editors, *Mind As Motion*, pages 1–43. Massachusetts Institute of Technology, Cambridge, MA, USA, 1995. ISBN 0-262-16150-8.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. ISBN 0-8018-5414-8.

D.S. Watkins. *Fundamentals of Matrix Computations*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2004. ISBN 9780471461678.

Jean Piaget. *La construction du réel chez l'enfant.* Neuchâtel ; Paris : Editions Delachaux & Niestlé S.A., 1937.

Richard Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc.*, 60(6):503–516, Jan 1954.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Christopher Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989.

G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical Report 166, Cambridge University Engineering Department, September 1994.

Satinder P. Singh. Reinforcement learning with a hierarchy of abstract models. In *AAAI*, pages 202–207, 1992a.

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *NIPS*, 1995.

Richard S. Sutton. TD models: Modeling the world at a mixture of time scales. In *ICML*, pages 531–539, 1995.

Ronald Parr and Stuart J. Russell. Reinforcement learning with hierarchies of machines. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 1043–1049. MIT Press, 1998.

Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas L. Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *UAI*, pages 220–229, 1998.

Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *ICML*, pages 118–126, 1998.

Ronald A. Howard. Semi-markovian decision processes. In *Proceedings 34th Session International Statistical Institute*, pages 625–652, 1963.

M. L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

Richard S. Sutton. Beyond reward: The problem of knowledge and data. In Stephen H. Muggleton, Alireza Tamaddoni-Nezhad, and Francesca A. Lisi, editors, *Inductive Logic Programming*, pages 2–6, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-31951-8.

Doina Precup and Richard S. Sutton. Multi-time models for temporally abstract planning. In *NIPS*, pages 1050–1056, 1997.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Theoretical results on reinforcement learning with temporally abstract options. In *ECML*, pages 382–393, 1998.

Richard S. Sutton. fourteen declarative principles of experience-oriented intelligence. RLAI Course, 2009. URL `http://incompleteideas.net/RLAIcourse2009/principles2.pdf`.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Intra-option learning about temporally abstract actions. In *ICML*, pages 556–564, 1998.

M. S. Mayzner and R. F. Gabriel. Information "chunking" and short-term retention. *The Journal of Psychology*, 56(1):161–164, 1963.

George Konidaris and Andrew G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *NIPS*, pages 1015–1023, 2009.

Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IEEE IROS*, pages 5239–5246, 2012.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 361–368, 2001.

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *Abstraction, Reformulation and Approximation, 5th International Symposium, SARA 2002, Kananaskis, Alberta, Canada, August 2-4, 2002, Proceedings*, pages 212–223, 2002.

Simsek Özgür and Andrew G. Barto. Skill characterization based on betweenness. In *NIPS 21*, pages 1497–1504, 2009.

Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *ECML*, pages 295–306, 2002.

Simsek Özgür, Alicia P. Wolfe, and Andrew G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *ICML*, pages 816–823, 2005.

Matthew M. Botvinick, Yael Niv, and Andrew C. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262 – 280, 2009.

Arun Tejasvi Chaganty, Prateek Gaur, and Balaraman Ravindran. Learning in a small world. In *AAMAS*, pages 391–397, 2012.

Jake V. Bouvrie and Mauro Maggioni. Efficient solution of markov decision problems with multiscale representations. In *50th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2012*, pages 474–481, 2012.

Pierre-Luc Bacon. On the bottleneck concept for options discovery: Theoretical underpinnings and extension in continuous state spaces. Master's thesis, McGill University, 2013.

Ramnandan Krishnamurthy, Aravind S. Lakshminarayanan, Peeyush Kumar, and Balaraman Ravindran. Hierarchical reinforcement learning using spatio-temporal abstractions and deep neural networks. *CoRR*, abs/1605.05359, 2016.

Marlos C. Machado, Marc G. Bellemare, and Michael H. Bowling. A laplacian framework for option discovery in reinforcement learning. In *ICML*, pages 2295–2304, 2017.

Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artif. Intell. Res.*, 13:227–303, 2000.

Brian Tanner, Vadim Bulitko, Anna Koop, and Cosmin Paduraru. Grounding abstractions in predictive state representations. In *IJCAI*, pages 1077–1082, 2007.

Gian-Carlo Rota. In memoriam of stan ulam - the barrier of meaning. *Physica D: Nonlinear Phenomena*, 22(1):1 – 3, 1986. Proceedings of the Fifth Annual International Conference.

D. Araújo and K. Davids. *What Exactly is Acquired During Skill Acquisition?* Journal of consciousness studies. Imprint Academic, 2011.

Eric V. Denardo. Contraction mappings in the theory underlying dynamic programming. *SIAM Review*, 9(2):165–177, 1967.

D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific optimization and computation series. Athena Scientific, 2012.

Cyrus Derman. *Finite State Markovian Decision Processes*. Academic Press, Inc., Orlando, FL, USA, 1970. ISBN 0122092503.

A. Shwartz. Death and discounting. *IEEE Transactions on Automatic Control*, 46 (4):644–647, 2001.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, in progress, second edition, 2018.

Eric V. Denardo. *Dynamic Programming: Models and Applications*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981. ISBN 0132215071.

A.S. Householder. *The Theory of Matrices in Numerical Analysis*. A Blaisdell book in pure and applied sciences : introduction to higher mathematics. Blaisdell Publishing Company, 1964.

David Matheson Young and Werner Rheinboldt. *Iterative solution of large linear systems*. Academic Press, New York, NY, 1971.

I. Gelfand. Normierte ringe. *Rec. Math. [Mat. Sbornik] N.S.*, 9(51):3–24, 1941.

Erhan Cinlar. *Introduction to Stochastic Processes*. Prentice-Hall, Inc., 1975.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181, 1922.

Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Richard J. Hanson. An extended set of fortran basic linear algebra subprograms. *ACM Trans. Math. Softw.*, 14(1):1–17, 1988. ISSN 0098-3500.

Paul F. Dubois, Konrad Hinsen, and James Hugunin. Numerical python. *Comput. Phys.*, 10(3):262–267, 1996. ISSN 0894-1866.

V. Volkov and J. W. Demmel. Benchmarking gpus to tune dense linear algebra. In *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11, Nov 2008.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. ISSN 00034851.

Albert Benveniste, Pierre Priouret, and Michel Métivier. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag New York, Inc., New York, NY, USA, 1990. ISBN 0-387-52894-6.

Harold Kushner and George Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2003.

J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5): 674–690, May 1997a. ISSN 0018-9286.

A.H. Klopf. *The hedonistic neuron: a theory of memory, learning, and intelligence*. Hemisphere Pub. Corp., 1982.

Matthieu Geist and Bruno Scherrer. Off-policy learning with eligibility traces: a survey. *Journal of Machine Learning Research*, 15(1):289–333, 2014.

K. Chen. *Matrix Preconditioning Techniques and Applications*. Cambridge Monographs on Applied Mathematics. Cambridge University Press, 2005.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5): 674–690, may 1997b.

Ronald Christensen. *Plane Answers to Complex Questions*. Springer New York, 2011.

Justin A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2):233–246, Nov 2002. ISSN 1573-0565.

Michael J. Kearns and Satinder P. Singh. Bias-variance error bounds for temporal difference updates. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, pages 142–147, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-703-X.

Vijaymohan Konda. *Actor-Critic Algorithms*. PhD thesis, Massachusetts Institute of Technology, June 2002.

Martha White and Adam White. A greedy approach to adapting the trace parameter for temporal difference learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 557–565, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-4239-1.

Timothy Arthur Mann, Hugo Penedones, Shie Mannor, and Todd Hester. Adaptive lambda least-squares temporal difference learning. *CoRR*, abs/1612.09465, 2016. URL `http://arxiv.org/abs/1612.09465`.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts, 1960.

David Blackwell. Discounted dynamic programming. *Ann. Math. Statist.*, 36 (1):226–235, 02 1965.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, 1999b.

E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.

Philip Thomas. Bias in natural actor-critic algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 441–448, 2014.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Sham Kakade. Optimizing average reward using discounted rewards. In *Computational Learning Theory, 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 16-19, 2001, Proceedings*, pages 605–615, 2001.

Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *J. Artif. Int. Res.*, 15(1):319–350, November 2001. ISSN 1076-9757.

Satinder P. Singh, Tommi S. Jaakkola, and Michael I. Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, pages 284–292, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-335-2.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 387–395, 2014.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: A new theory for modeling dynamical systems. *Uncertainty in Artificial Intelligence (UAI)*, 2004a.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996. ISBN 1886529108.

Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *10th International Conference on Autonomous Agents and Multiagent Systems*, pages 761–768, 2011.

Adam White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

Richard S. Sutton. True online emphatic td(λ): Quick reference and implementation guide. *CoRR*, abs/1507.07147, 2015b. URL `http://arxiv.org/abs/1507.07147`.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565.

P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of markov reward processes. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, volume 3, pages 2698–2703 vol.3, 1998.

Peter L. Bartlett and Jonathan Baxter. Estimation and approximation bounds for gradient-based reinforcement learning. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, pages 133–141, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-703-X.

Pierre L'Ecuyer. A unified view of the ipa, sf, and lr gradient estimation techniques. *Management Science*, 36(11):1364–1383, nov 1990. doi: 10.1287/mnsc.36.11.1364.

D. R. Cox and Hinkley D. V. *Theoretical Statistics*. Chapman & Hall, 1974. ISBN 0-412-12420-3.

Hajime Kimura and Shigenobu Kobayashi. An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 278–286, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, Sept 1983. ISSN 0018-9472.

Andrew G. Barto, Richard S. Sutton, and Peter S. Brouwer. Associative search network: A reinforcement learning associative memory. *Biological Cybernetics*, 40(3):201–211, May 1981. ISSN 1432-0770.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *NIPS*, pages 1008–1014, 2000.

J. M. Hammersley and D. C. Handscomb. *General Principles of the Monte Carlo Method*, pages 50–75. Springer Netherlands, Dordrecht, 1964. ISBN 978-94-009-5819-7.

Reuven Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1981. ISBN 0471089176.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *J. Mach. Learn. Res.*, 5:1471–1530, December 2004. ISSN 1532-4435.

Leemon C. Baird. Advantage updating. Technical Report WL–TR-93-1146, Wright Laboratory, 1993.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

Martha White. Unifying task specification in reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3742–3750, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Nicholas Roy. *Finding Approximate POMDP solutions Through Belief Compression*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, September 2003.

Steven J. Bradtke and Michael O. Duff. Reinforcement learning methods for continuous-time markov decision problems. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 393–400. MIT Press, 1995.

L. Devroye. *Non-Uniform Random Variate Generation*. Springer New York, 1986. ISBN 9783540963059.

George H. John. When the best move isn't optimal: Q-learning with exploration. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2.*, page 1464, 1994.

Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco A. Wiering. A theoretical and empirical analysis of expected sarsa. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2009, Nashville, TN, USA, March 31 - April 1, 2009*, pages 177–184, 2009. doi: 10.1109/ADPRL.2009.4927542.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed representations. In David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 77–109. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

Richard S. Sutton. Toward a new view of action selection: The subgoal keyboard. 10th Barbados Workshop on Reinforcement Learning, 2016. URL `http://barbados2016.rl-community.org`.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. ISBN 0521642981.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3540–3549, 2017.

Daniel J. Mankowitz, Timothy Arthur Mann, and Shie Mannor. Adaptive skills, adaptive partitions (ASAP). In *Advances in Neural Information Processing Systems 29*, 2016.

Kfir Y. Levy and Nahum Shimkin. *Unified Inter and Intra Options Learning Using Policy Gradient Methods*, pages 153–164. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-29946-9.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

J.A.E.E. van Nunen and J. Wessels. Stopping times and markov programming. Technical Report 76-22, Technische Hogeschool Eindhoven, Eindhoven:, 1976a.

J.A.E.E. van Nunen. *Contracting Markov decision processes*. PhD thesis, Department of Mathematics and Computer Science, 1976. Proefschrift Technische Hogeschool Eindhoven.

J. Wessels. Stopping times and markov programming. In J. Kožešnik, editor, *Transactions of the Seventh Prague Conference on Information Theory, Statistical Decision Functions, Random Processes and of the 1974 European Meeting of Statisticians: held at Prague, from August 18 to 23, 1974*, pages 575–585. Springer Netherlands, Dordrecht, 1977. ISBN 978-94-010-9910-3.

Richard S. Varga. Factorization and normalized iterative methods. In Rudolph E. Langer, editor, *Boundary Problems in Differential Equations*, pages 121–142. University of Wisconsin Press, Madison, 1960.

Herbert B. Keller. On some iterative methods for solving elliptic difference equations. *Quarterly of Applied Mathematics*, 16(3):209–226, 1958. ISSN 0033569X, 15524485.

Evan L. Porteus. Bounds and transformations for discounted finite markov decision chains. *Oper. Res.*, 23(4):761–784, August 1975. ISSN 0030-364X.

Wolfgang Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Springer International Publishing, 2016.

Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent. Convergent tree-backup and retrace with function approximation. In *International Conference on Machine Learning (ICML)*, 2018.

Doina Precup, Richard S. Sutton, and Satinder P. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 759–766, 2000.

Malcolm Ross Kinsella Ryan. *Hierarchical Reinforcement Learning: A Hybrid Approach*. PhD thesis, University of New South Wales, New South Wales, Australia, 2004.

Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowe. Learning with options that terminate off-policy. In *Thirthy-first AAAI Conference On Artificial Intelligence (AAAI)*, 2018.

J. A. E. E. van Nunen and J. Wessels. The generation of successive approximation methods for markov decision processes by using stopping times. Technical Report Memorandum COSOR; Vol. 7622, Eindhoven: Technische Hogeschool Eindhoven., 1976b.

Clement Gehring, Yangchen Pan, and Martha White. Incremental truncated LSTD. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 1505–1511, 2016.

Huizhen Yu, Ashique Rupam Mahmood, and Richard S. Sutton. On generalized bellman equations and temporal-difference learning. In Malek Mouhoub and Philippe Langlais, editors, *Advances in Artificial Intelligence*, pages 3–14, Cham, 2017. Springer International Publishing. ISBN 978-3-319-57351-9.

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8):1–10, 08 2014.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1):159 – 216, 1990. ISSN 0004-3702.

Daniel G. Goldstein and Gerd Gigerenzer. Models of ecological rationality: The recognition heuristic. *Psychological Review*, 109(1):75–90, 2002.

J. van Nunen and J. Wessels. Solving linear systems by methods based on a probabilistic interpretation. *Computing*, 26(3):209–225, Sep 1981. ISSN 1436-5057.

J. van Nunen and S. Stidham. Action-dependent stopping times and markov decision process with unbounded rewards. *Operations-Research-Spektrum*, 3 (3):145–152, Sep 1981.

Thomas E. Morton. Technical note—undiscounted markov renewal programming via modified successive approximations. *Operations Research*, 19(4): 1081–1089, 1971.

Martin L. Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):1127–1137, 1978.

Martin L. Puterman and Shelby L. Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4(1):60–69, 1979.

Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical report, Massachusetts Institute of Technology, 1996.

Dimitri P. Bertsekas. *Lambda-Policy Iteration: A Review and a New Implementation*, chapter 17, pages 379–409. Wiley-Blackwell, 2013. ISBN 9781118453988.

M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. *Neural Information Processing Systems Conference (NIPS)*, 2001.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 06 2013.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1928–1937, 2016.

Ronald J. Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Timothy Arthur Mann, Daniel J. Mankowitz, and Shie Mannor. Time-regularized interrupting options (TRIO). In *ICML*, pages 1350–1358, 2014.

Timothy A. Mann and Shie Mannor. Theoretical analysis of planning with options. In *The 11th European Workshop on Reinforcement Learning (EWRL)*, 2013.

Linn I. Sennott. Constrained discounted markov decision chains. *Probability in the Engineering and Informational Sciences*, 5(4):463–475, 1991.

Kevin Lloyd and Peter Dayan. Interrupting behaviour: Minimizing decision costs via temporal commitment and low-level interrupts. *PLOS Computational Biology*, 14(1):e1005916, jan 2018.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006, October 9-15, 2006, Beijing, China*, pages 2219–2225, 2006.

Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1992.

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

Gheorghe Comanici and Doina Precup. Optimal Policy Switching Algorithms for Reinforcement Learning. In *AAMAS*, pages 709–714, 2010.

C. Daniel, H. van Hoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning, Special Issue*, 104(2):337–357, 2016.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*, pages 273–281, 2012.

Tejas Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Joshua Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems 29*, 2016.

Satinder P. Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, pages 1281–1288, 2004b.

David Silver and Kamil Ciosek. Compositional planning using optimal option models. In *ICML*, 2012.

Timothy Arthur Mann, Shie Mannor, and Doina Precup. Approximate value iteration with temporally extended actions. *J. Artif. Intell. Res.*, 53:375–438, 2015.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 166–175, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Doina Precup, Cosmin Paduraru, Anna Koop, Richard S Sutton, and Satinder P. Singh. Off-policy learning with options and recognizers. In Y. Weiss, P. B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1097–1104. MIT Press, 2006.

Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, May 1992b.

Paul Ruvolo and Eric Eaton. ELLA: an efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 507–515, 2013.

S. E. Hampson and D. J. Volper. Linear function neurons: Structure and training. *Biological Cybernetics*, 53(4):203–217, Feb 1986.

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook. Diploma thesis, Technische Universitat Munchen, Germany, 14 1987.

Yoshua Bengio. Learning a synaptic learning rule. Technical Report 751, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montréal (QC) Canada, 1990.

Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 171–176. AAAI Press, 1992. ISBN 0-262-51063-4.

Jonathan Baxter. *Learning Internal Representations*. PhD thesis, The Flinders University of South Australia, 1995.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 748–756, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.