Faculty of Science and
Bio-Engineering Sciences
Department of Computer Science
Artificial Intelligence

Beyond Single-Step Temporal Difference Learning

Dissertation submitted in fulfilment of the requirements for the degree of Doctor of Science: Computer Science

Anna Harutyunyan

Promotors: Prof. Dr. Ann Nowé (Vrije Universiteit Brussel)
Prof. Dr. Peter Vrancx (Vrije Universiteit Brussel)

2017 Uitgeverij VUBPRESS Brussels University Press
VUBPRESS is an imprint of ASP nv (Academic and Scientific Publishers nv)
Ravensteingalerij 28
B-1000 Brussels
Tel. +32 (0)2 289 26 50
Fax +32 (0)2 289 26 59
E-mail: info@vubpress.be
www.vubpress.be

ISBN TODO
NUR TODO
Legal deposit TODO

Abstract

An intelligent agent performs an action. Exactly one time step later the agent experiences the consequence of its action. It observes the feedback from the environment, and finds itself in a new state. The agent, then, is able to revise its belief about the quality of his action based only on prior beliefs and this momentary feedback. This idea, of updating estimates based on other estimates, roughly explains *temporal difference learning*, one of the most fundamental concepts of reinforcement learning. It is a powerful idea, and its uncanny analogues are observed in the activity of the dopamine-carrying neurons of the human brain. The naive computational form of this idea, however, is understandably limited. In particular: *a single, primitive time-step experience is not sufficiently rich to learn with sophistication.*

In this dissertation, we consider this problem from several partially related and partially complementary directions that all aim to enrich the single step.

Richer updates. Allowing the agent take multiple actions before making the update makes it easier to judge their quality correctly. Unfortunately, doing so is fundamentally difficult when the desired updates are *off-policy*, that is: concerning a course of actions different from the agent's behavior.

We devise novel off-policy multi-step algorithms that rely on the idea of correcting off-policy actions in the value, rather than the conventional probability space, and analyze their convergence,

Richer actions. Consider all of the micro-actions involved in doing something as simple as walking. Yet, we are able to collapse them in a single complex *macro*-action. As the agent gathers experience, it must likewise be able to ascend the levels of acting hierarchy. The *options framework* is a general model for such temporal abstraction in reinforcement learning.

We first make a novel link between the options framework, and multi-step temporal difference planning. Using this link, we devise an algorithm that is able to learn about options that *terminate* off-policy, that is: irrespectively of the behavior options. We then propose and analyze a modification to the options framework, which allows options to represent policies over longer planning horizons.

Richer feedback. Learning is rarely completely insular: when learning a new task, we assimilate multiple sources of feedback. Likewise, in reinforcement learning, the environment feedback alone may be too infrequent to be efficiently exploitable. *Potential-based reward shaping* is a paradigm for augmenting it with additional feedback, notable for its attractive theoretical guarantees. However, it requires the additional feedback to be presented through a specific abstraction, which may be cumbersome to obtain.

We devise a framework that allows one to provide the additional feedback in the natural form directly, while maintaining the desired theoretical guarantees.

Samenvatting

Een intelligente agent voert een actie uit. Precies één tijdstap later observeert de agent de gevolgen van zijn actie. Hij ontvangt een feedbacksignaal van de omgeving en bevindt zich in een nieuwe toestand. De agent kan dan zijn inschatting van de kwaliteit van de genomen actie aanpassen, gebaseerd op deze nieuwe observaties en zijn bestaande schattingen voor het uitvoeren van de actie. Het idee om een schatting aan te passen op basis van andere inschattingen, komt in grote lijnen overeen met temporal difference learning, mogelijk het meest fundamentele concept in modern reinforcement learning. Het is een erg krachtig model dat opvallende gelijkenissen vertoond met de activiteit van dopamine dragende neuronen in het menselijk brein. De naïeve computationele versie ervan is echter eerder beperkt. In het bijzonder: een enkele primitieve tijdstap bevat onvoldoende ervaring om verfijnd te leren. In dit proefschrift beschouwen we drie gerelateerde en deels complementaire richtingen die elk tot doel hebben om deze enkele stap te verrijken.

Uitgebreide updates. In plaats van een enkele stap te gebruiken, is het informatiever om meerdere stappen te beschouwen. Helaas ontstaan fundamentele problemen wanneer dit proces gecombineerd wordt met off-policy leren, dat wil zeggen het leren over acties verschillend van het huidig gedrag van de agent. We ontwikkelen een familie van originele off-policy, meerstaps temporal difference algoritmen en analyseren hun convergentie. Deze algoritmen zijn gebaseerd op het corrigeren van de waarde van off-policy acties, eerder dan de conventionele kansruimte.

Uitgebreide acties. Beschouw al de individuele micro-acties die nodig zijn voor een eenvoudige handeling als stappen. Toch zijn wij mensen in staat om al deze acties te combineren in een enkele complexe actie. Naarmate de agent ervaring opdoet, moet hij op gelijkaardige manier stijgen doorheen de actiehiëarchie. Het options raamwerk is een algemeen model dat een dergelijke abstractie toelaat in het kader van reinforcement learning. We leggen eerst een nieuw verband tussen options en meerstaps temporal difference plannen. We beschouwen dan een vernieuwende methode om discounting toe te passen in de context van options en tonen aan dat deze methode voordelen biedt voor zowel optimalisatie als representatie.

Uitgebreide feedback. Leren gebeurt zelden is een volledige geïsoleerde setting: wanneer we een nieuwe taak leren, verwerken we meerdere bronnen van feedback. Op dezelfde manier kan in reinforcement learning de feedback uit de omgeving te schaars zijn om efficiënt bruikbaar te zijn. Potentiaal gebaseerd reward shaping is een paradigma om het feedbacksignaal te verrijken, dat opvalt door zijn aantrekkelijke theoretische garanties. Deze methode vereist echter dat de extra feedback geleverd wordt via een specifieke abstractie, die vaak onhandig in praktische toepassingen is. We ontwikkelen een nieuw kader dat toelaat om de additionele feedback in een natuurlijke vorm te geven en toch de theoretische garanties van de originele methode te behouden.

Acknowledgments

I am deeply grateful to my advisor Ann Nowé for giving me the opportunity[1] to spend these last 4 years, 8 months and 15 days at the AI Lab, for her good spirits and her confidence in me. Her ability to skillfully run the lab while still caring for its individuals is remarkable, and maybe some day I will learn to multi-task as effectively as she does. I am extremely thankful to my friend and co-advisor Peter Vrancx for his pragmatism, for donating his blackboard to my doodles, and for many a game of Terra Mystica. I am indebted to my jury members for their time invested into this thesis: Bruno Scherrer for his careful proofreading, and for helping me iron out all of the technical kinks, Emma Brunskill for her help in placing the work in a broader context, Bernard Manderick for the many helpful suggestions, and in particular pointing out the Metropolis-Hastings connection of Chapter 3, Ann Dooms for making sure my math was ok, and Wolf De Meuter for chairing, and the practical perspective. Their input has improved this text immeasurably.

My PhD would have been very different if not for a few incredible opportunities, for which I am very grateful. I would like to especially thank Doina Precup for hosting me in 2014, and the later collaboration on the options chapters of the thesis – her intuitive, yet precise approach is something I will always strive to replicate; DeepMind folks for the fantastic environment during my internship in 2015, and Rémi Munos in particular, whom I learnt a disproportionate amount from in the short time, for his patience and clarity of thought. Finally, thanks to the many collaborators and friends made at conferences, and in particular the crew at RL Barbados 2015 for the around-the-clock research excitement, and that jug of rum punch at the beach.

[1] IWT-SBO project MIRAD, grant nr. 120057

I am grateful for the collaboration on the MIRAD project (Joris, Erwin, Friedl, Kevin, Jonas, Amber, Stefan, Maarten and others) for offering a glimpse into the complexity of robotics in the real world, and for always challenging my views with a different perspective on learning.

I could not have wished for a nicer environment to spend the last few years in. (Although perhaps not literally – I cannot say I am not glad to not have to test the heating system for another winter.) Thanks to Tim, Timo, Roxana, Denis, Felipe, Pieter, Steven, Kristof, Kevin, Maarten, Jelena, and many others for the fun reading groups, countless lunches, and the occasional beer. A special thank you to Kirk for being the best listener, not letting me drop my phone, and reminding me to enjoy the little (aka most important) things. I am sorry for always sitting in your spot.

The path that led me to Belgium is extremely nonlinear, and I would like to mention some of its critical nodes. I must thank Cora Borradaile for her infectious enthusiasm about research, and for continuing to be an inspiration for being a positive change in the world; Dan Watson for his good humor and his interest in my future; Dan Bryce whose class on planning may have been one of the reasons I chose to do RL for my PhD; and the Huntsman Foundation for providing me with the opportunity to move to the United States at 17, marking perhaps the first step on my path with a large TD error. I am also grateful for the remnants of the Soviet education system that instilled a passion for mathematics and sciences in me, before I was old enough to learn that this is uncommon.

Despite the futility of the chance of them reading this, I want to thank my climbing buddies (Ben, Rob, Oscar, Sebastian, Hans, Ruan, Ansie, Danouck, Timo, etc) for helping me keep my sanity on the weekends, and many a soft catch. On many occasions I have pondered how pushing for a deadline is similar to pushing through a crux move (in which case writing this thesis has been the world's least scenic big-wall climb).

I am incredibly fortunate to have one of the weirdest families. Thanks to my brother Aram for tolerating my childhood affinity to emulate everything he did (including, later, computer science); to my mother Zhanna for her unbridled enthusiasm for education and discovery that she passed on to us from an early age; and to my father Robert for all that he has done for our family, and for never passing up the opportunity of giving me a good math puzzle.

I must also mention my other family, Ed and Jill Hershberg (as well as Susannah, Karina, Sam, Celeste and Adira – the last two of whom are younger than this thesis!) A special thank you to Ed for probably being the only person to willingly read (and reserve multiple archival copies of) this document.

Finally, the last mention belongs to the person who always gets to see the flip-side, to my dearest Ben. For always inspiring me with his enthusiasm, creativity and humbleness, and for supporting me in all things big and small.

Contents

List of Figures

List of Tables

Notation

We will generally follow the notation from [Sutton and Barto 2017], and use:

lower case	to denote functions (e.g. q, v);
capital case	to denote random variables (e.g. R_{t+1});
calligraphic font	to denote operators (e.g. $\mathcal{T}$, $\mathcal{R}$);
script font	to denote sets (e.g. $\mathcal{S}$, $\mathcal{A}$).

$\mathcal{S}$	the set of states
$\mathcal{A}$	the set of actions
$r(s, a)$	the (expected) reward upon taking the action a in state s
$p(s'\|s, a)$, $p^a_{ss'}$	the probability of transitioning into state s' upon taking action a in state s
γ	the discount factor
S_t	state encountered at time t
A_t	action taken at time t
$R_{t+1} \sim r(S_t, A_t)$	the reward upon taking action A_t at state S_t
δ_t	temporal difference error at time t
G_t	the return generated from time t forward
$G^{(n)}_t$	the n-step return generated from time t forward

π	(target) policy
μ	behavior policy, or policy over options
$p^\pi(s'\|s)$	probability of ending up in s' from s under a policy π
$r^\pi(s)$	reward expected in state s under a policy π
λ	eligibility trace parameter
$\mathcal{P}^\pi$	one step transition operator for a policy π
$\mathcal{T}^\pi$	Bellman operator for a policy π
$\mathcal{T}_\lambda^\pi$	Bellman λ-operator for a policy π
$\mathcal{T}$	Bellman optimality operator
q	Q-function, Q-values, state-action value function
v	V-function, V-values, state value function
q^π, v^π	the Q-and V-values of policy π
q^*, v^*	optimal value functions
$\mathcal{G}(q)$	the set of greedy policies w.r.t. q
α_k	the step-size at phase k
$\mathcal{O}$	the set of options
π^o	policy of option o
β^o	(behavior) termination condition of option o
$\beta^o(s)$	probability of option o terminating in state s
R^o	reward model of option o
P^o	transition model of option o
$\mu(o\|s)$	probability of option o being selected in state s
κ_μ, κ	the flat marginal policy for a policy over options μ
$\mathcal{T}_\mathcal{O}^\mu$	Bellman operator over options for a policy μ
$\mathcal{T}_\mathcal{O}$	Bellman optimality operator over options
ζ^o	target termination condition of option o
h	potential function
f	shaping reward
$\stackrel{\text{def}}{=}$	equal by definition
$\equiv$	notationally equivalent
$\|\cdot\|$	supremum norm
$\mathbf{1}$, e	vector of one-components
$\mathbf{0}$	vector of zero-components
$\mathbb{I}\{a = b\}$	indicator function that returns 1 if $a = b$ and 0 otherwise
$(x_k)_{k\in\mathbb{N}}$, (x_k)	sequence $x_0, x_1, x_2, \ldots$

It's not a human move. I've never seen a human play this move. So beautiful.
— Fan Hui, European Go champion, *on the 37th move of the second game between Lee Sedol and AlphaGo.*

1 | Introduction

Artificial intelligence (AI) in one form or another is an idea that has pervaded the world's intellectual history, *"a dream in urgent need of being realized"* [McCorduck 2004], expressed in humanity's myths, legends, stories, speculation and clockwork automatons.

In the modern day we are surrounded by AI systems: in our phones, computers, and devices. Our wish for these systems is to be able to learn from *us*: how to translate, how to drive, how to make the perfect toast. In March of 2016 for maybe the first time, the world witnessed an AI system that we could learn *from*. It was at a particular match between a human and a machine. The match was in a game of Go, which has always been a special challenge for AI due to its astronomical complexity. Any naive approach is doomed here: there are more possibilities to compute than atoms in the universe. *AlphaGo* [Silver et al. 2017], an AI system that learnt Go by playing itself countless times, was playing Lee Sedol, one of the world's best players. AlphaGo won: four to one, against all expectations. The critical component that set it apart from its predecessors, the mechanism of its self-improvement, was *reinforcement learning*. More victories and remarkable improvements followed, but perhaps the most fascinating outcome has been the revival in the Go community. The play of the AI was qualitatively different and provided truly new insights into the ancient game [Tormanen 2017].

When the most complex board game is solved, the prized challenges evolve with it. While the goal of having reinforcement learning systems provide insights into the important problems of the real world: climate change, healthcare, education, is yet to be achieved, the challenge has never seemed so feasible and inspiring.

1.1 Reinforcement Learning

At the heart of *dynamic programming (DP)* is a recursion. In order to solve a problem, one can often break it down into a small easy-to-solve subproblem and the remainder, for which the procedure can be repeated. Originally intended for solving multi-stage stochastic decision problems, this general idea behind *Bellman's principle of optimality*, has proven to be extremely powerful, and has become fundamental in theoretical computer science, as well as many applied sciences [Dasgupta et al. 2008].

The goal of AI systems is to tackle large, complex problems. As powerful as the formalism of DP is, in the context of AI, exact DP is an analytical, rather than practical framework. This is because it requires accurate knowledge of the underlying dynamics, and suffers from what Bellman called the "curse of dimensionality", that is: the need for the computational requirements to grow exponentially with the dimensions of the state.

Reinforcement learning (RL) can be thought of as the approximate, data-driven counterpart to dynamic programming. It introduces the notion of an actor, an *intelligent agent*. Where DP operates over the entire state space, RL places the agent *in* the state space. Instead of assuming omniscience of the dynamics, the agent *interacts* with them in a closed loop. Everything outside of the agent, including the dynamics, is assumed to be the *environment*. Thus, the agent is able to *act* upon its environment, and *perceive* the consequences of its actions.

The idea of an agent has been formally defined in many ways. At its core, it is nothing but an algorithm that maps input state observations to output actions. By some definitions, in order for an agent to be *intelligent*, it must have an internal representation that evolves over time. It is these algorithms and representations that constitute reinforcement learning research.

> **Definition 1.1: Components of RL**
>
> - An **intelligent agent** is an algorithm that maps input state observations to output actions via an internal, evolving representation.
> - An **environment** is the entity outside of the agent that includes a notion of dynamics, and provides the agent with (possibly incomplete) state observations and evaluative feedback.

The introduction of an agent relates RL to another discipline, of cognitive psychology. The action-feedback loop with the environment can be thought of as a computational formalism for *trial-and-error* learning. Although any learning system can be considered to be trial-and-error – such as a supervised neural network that updates its beliefs about likely outcomes after observing an example of a true outcome – the "error" in RL is principally

different: it contains *evaluative* feedback on an action rather than the underlying truth about whether an action is correct. That is, instead of distinguishing if the action was right or wrong like a supervised *label* would, the feedback in RL is a *reward* that denotes *how* good the action was, with respect to some evaluation criteria. This is much more in line with "pleasure-pain" theories from psychology (e.g. [Thorndike 1911]) and places RL on a peculiar bridge between biological and machine learning.

That this is the case is particularly evident in the history of *temporal difference (TD)* learning, one of the most important ideas of RL. Conceptually, TD learning is related to the Bellman's principle of optimality. But because of the approximate nature of RL, it amounts to improving estimates based on the difference between the estimates at a current and next time instances. Amazingly, such temporal difference *errors* are sufficient to learn the task at hand. Many years after the introduction of the idea computationally, it was shown experimentally that the form of the TD error is remarkably close to the mechanism of activity of dopamine neurons in mammals [Montague et al. 1996], which in turn governs a number of important cognitive processes in the brain.

1.2 Research Question and Contributions

The idea of TD learning is very general: learning from the difference of estimates at different times. Its *single-step* computational form however – prohibitively simple. An agent performs an action, and exactly one time step later observes the environment feedback and the next state, updates its belief about the quality of its course of actions (or a *policy*), and is ready for the next action choice. This naive form is understandably limited. In particular: *a single time step experience is not sufficiently rich to learn with sophistication,* that is: learn about complex problems efficiently and responsibly. This foremost implies being judicious with the environment interactions, since for any AI system to be safely deployed in the real world it is essential for it to interact with its surroundings in an informed careful manner. As such, we ask:

> **Research Question**
>
> How to enrich the basic temporal difference learning step to learn about complex problems in a way that interacts with the environment responsibly, and utilizes each interaction maximally?

This dissertation considers this question from three partially related and partially orthogonal directions:

- Learning off-policy from multiple steps for richer *updates*;

- Temporal abstraction for richer *actions*; and
- Reward shaping for richer *feedback*.

The remainder of this section motivates these directions, each of which is an active area research, and gives a high-level summary of our contributions in each direction.

1.2.1 Learning Off-Policy from Multiple Steps

A physical interaction loop is inherently single-stream: one can only be in a single place at a given instance, take a single action. Yet, there is a multitude of diverse information that can be learned from this single action. In reinforcement learning, an agent acts according to some *behavior*, and this behavior is inherently unique. But in order to have scalable, responsible learning systems, the agent must be able to learn about many possible facts and ways to act that are not limited to the behavior alone. In order to so, the agent must learn *off-policy*. An important example of learning off-policy is the problem of *optimal control*, in which the agent must find the best course of actions while following an exploratory one.

Credit assignment is one of the key challenges of RL. Marvin Minsky in his seminal "Steps Toward Artificial Intelligence" formulated what he called the basic credit-assignment problem for complex reinforcement learning systems: *How do you distribute credit for success among the many decisions that may have been involved in producing it?* [Minsky 1961] This problem becomes even more complex, when we consider off-policy learning. How much should the outcome of the decision one took contribute to one's understanding of the outcomes of other *potential* decisions? The difficulty of answering this question is further exacerbated when considering sequences of decisions.

Contributions

We consider the problem of off-policy learning from multiple steps in Chapter 3 and make the following contributions:

- Off-policy credit assignment is typically done with *importance sampling*, which while being precise, is infeasible for longer decision sequences. We investigate the necessity of its precision, and formulate new algorithms and analysis for off-policy learning without importance sampling, which reveals convergence up to a tradeoff between how many decisions one considers and how off-policy one is.
- Based on the insights from that analysis, we attempt to combine the strengths of the proposed algorithms with several existing ones. The result is a new algorithm, Retrace(λ), that is able to efficiently handle multiple steps, while enjoying general convergence guarantees even in the challenging off-policy control case.

The material in Chapter 3 has been developed in close collaboration with my coauthors Marc G. Bellemare, Tom Stepleton, and especially Rémi Munos, who produced the critical insights behind several of the key results.

1.2.2 Learning with Temporally Abstract Actions

Temporal abstraction enables one to escape the naive measure of a single time-step, and interact with the environment at a variety of timescales, which is essential for learning complex tasks. On the one hand, the wish for temporal abstraction builds on the long history of modeling evolutions of physical systems over time. On the other, the idea that reasoning and learning does or should function on multiple timescales has been prominent in AI from early on (e.g. [Sacerdoti 1974]), and evident in psychology and neuroscience more recently [Newell et al. 2001, Murray et al. 2014].

In RL, temporal abstraction is typically modeled through the *options framework* [Sutton et al. 1999]. It is generally known that longer options result in faster learning. Typically, choosing an option hands over control to it entirely. This introduces a subtle tradeoff: the more one commits to an option, the faster the learning, but the less control one has over its execution. Being able to learn about many possible option durations, while being fully committed to the current one would make the benefits of options available, and remove the reservation of using longer options.

Now let us consider option duration from another angle. One of the strengths of the options framework is its seamless integration with classical TD. This strength comes at a cost, however. Namely, regardless of how sophisticated the options may be, they are anchored to some base notion of a time step. As such, and because the horizon of an agent can only be finite in terms of this time step, if its measure is not sufficient to represent complex tasks, the use of options will make no difference.

Contributions

We consider the problems of option durations and timescales outlined above in Chapters 4 and 5, and make the following contributions:

- In Chapter 4, we show that planning with options under a specific *one-step* model of option execution is exactly equivalent to λ-policy iteration, a well-known multi-step dynamic programming algorithm. This allows one to transfer analogous results with ease.
- Guided by this connection between options and multi-step TD, we formulate and analyze a new algorithm that allows one to learn option termination "off-policy", that is: irrespective of the actual terminating behavior of the options.

- In Chapter 5 we take the reasoning about option durations a step further, and analyze the role of the underlying timescale in options. We formulate a new option discounting scheme that allows representing longer time horizons and introduces a novel bias-variance tradeoff related to option duration.

1.2.3 Learning with Shaping Rewards

If one is to be confined to a single step, another alternative to enable learning with sophistication is to enrich, or *shape* the feedback itself that the step is to receive. Note that where the previous two approaches augmented the agent, this one assumes augmenting the environment, but it is up to the agent, to utilize the modification appropriately.

The term *shaping* in experimental psychology dates at least as far back as [Skinner 1938], and refers to the idea of rewarding all behavior *leading* to the desired behavior, instead of waiting for the subject to exhibit it autonomously. For example, Skinner discovered that, the fastest way to train a rat to push a lever was to reward any movement in the direction of the lever. Analogously, if any behavior of an RL agent leading to the desired behavior is rewarded, learning occurs much faster. A peculiar caveat to this simple intuition is that well-intentioned guidance may have adverse consequences. In the famous example of an agent learning to ride a bicycle to a goal [Randløv and Alstrøm 1998], a shaping reward was given every time the agent moved closer to the goal. But instead of learning to quickly move to the goal and finish the task, the agent quickly learnt to ride around the goal in circles and collect the associated positive reinforcements indefinitely. Similar artifacts were recently discussed in the context of Atari games [OpenAI 2016], and are of great importance when considering AI safety.

Potential-based reward shaping (PBRS) is a reward shaping framework in RL that guarantees that the original goal is kept in sight. It relies on being anchored to a *potential function* which ensures that, as for a potential field in physics, the value of any cycle along this potential function is zero, and there is nothing to exploit. The theoretical appeal of PBRS is balanced with the need of obtaining the potential function, an extra abstraction.

Contributions

We consider the problem of PBRS with unconstrained shaping functions in Chapter 6, and make the following contributions:

- We propose a new algorithm that converts an arbitrary shaping function into the PBRS form, while maintaining the theoretical guarantees.
- We validate this algorithm on a case study of online sparse feedback in the Mario domain, a difficult scenario to handle previously.

1.3 Situation

One may ask, why focus on temporal difference learning in the first place? To answer this question, let us provide some further exposition.[1] TD learning is typically associated with *model-free action-value* methods: ones that learn to *evaluate* the agent's actions from interaction, then infer and improve policies based on these values. In contrast, *model-free policy search* methods learn policies directly, without the intermediate evaluation step. *Model-based* methods use the interactive loop to estimate the dynamics of the environment, then use dynamic programming for obtaining values or simulate trajectories to optimize policies. We will discuss these method families briefly below, and motivate why the impact of good TD algorithms extends to them as well.

1.3.1 Policy Search Methods

Direct policy search methods bypass the evaluation step, and learn to improve policies directly from interaction. These algorithms have distinctly different strengths and weaknesses to those of action-value methods. On the plus side, it can be much simpler to learn what to do (the policy), than exactly how good what is being done is (action-values), and an explicit policy object provides more freedom to specialize in domain-specific policy classes. Furthermore, policy search methods are better suited to deal with high-dimensional action spaces, and are hence ubiquitous in robotic domains [Deisenroth et al. 2013]. However, without the anchoring in values, ensuring optimality is more difficult – indeed like with all gradient descent methods and evolution strategies, convergence is typically guaranteed only to local optima. Furthermore, obtaining direct samples of the policy performance requires long sequential interaction, which may incur high variance. A popular middle ground is the *actor-critic* family of methods [Barto et al. 1983] that maintains values in order to *criticize* the *actor* policy's actions. The values can be used to either entirely replace the samples, or as a *baseline* to mitigate the samples' variance [Williams 1992]. As such, efficient and reliable estimation of action values, typically to be performed with TD, is crucial to the success of many policy search methods.

1.3.2 Model-Based Methods

When the environment is not too vast, it can be much more effective to learn its *model* from interaction, then obtain values and policies from this estimated model. Unfortunately, it can be challenging to scale this process up to complex environments, since accurately estimating the dynamics model can be intractable for a reasonable amount of interactions.

[1] An unfamiliar reader may choose to return to this section after Chapter 2 for better context.

Partial models offer one way of addressing this limitation [Tamar et al. 2012, Jiang et al. 2015a]. This is one of the reasons we consider option models in Chapter 5.

Model-based methods follow the same split into action-value and policy search methods, as model-free methods. Action-value methods typically perform dynamic programming on the estimated models to obtain values [Brafman and Tennenholtz 2002], while policy search methods use the models to generate simulated trajectories for the policy to learn from [Deisenroth and Rasmussen 2011, Levine and Koltun 2013]. Many methods blend model-based and model-free learning by having components of different types [Levine and Koltun 2013]. There are also many methods that use models indirectly, or learn quantities related to them. For example, the popular experience replay mechanism [Lin 1992, Mnih et al. 2015] can be seen as an instance of planning with a non-parametric model [van Seijen and Sutton 2015]. Successor features [Dayan 1993, Barreto et al. 2017] learn the expected future cumulative features of a given policy, and can hence be used to produce the value of that policy w.r.t. any reward function immediately. In all of these cases, it is typically either true that the task of learning the model can be posed as a dynamic programming problem, or one requires learning values alongside for stability.[2]

TD methods hence can be seen as fundamental building blocks that are impactful even in seemingly orthogonal method families.

1.3.3 Criteria

Our focus throughout this thesis is the *guaranteed convergence to the optimal solution*, and the *asymptotic rate* of that convergence.

One other important criteria is the number of interactions (or *samples*) required to find a *near*-optimal solutions. This quantity is a fundamental formalization of learning efficiency in learning theory [Valiant 1984]. In RL, its analysis is made more challenging in the control setting by the coupling of the strategy used to generate samples to the object of optimization itself. In some sense, asymptotic analysis bypasses this difficulty and considers an idealized system. As such, results obtained in the asymptotic setting serve as a good starting point for the more intricate analyses. There is a rich body of literature on sample complexity analysis of RL algorithms (e.g. [Kearns and Singh 1999, Kakade 2003, Dann and Brunskill 2015]) and it would be insightful to analyze the work presented in this thesis from that viewpoint.

One may also consider the *computational* resources required by the learning algorithm. Although it is generally assumed that environment samples are a scarcer resource than computation, this is only true for physical environments, and as computational requirements of the algorithms scale, these considerations will become more pressing.

[2] One notable exception to this is the PILCO agorithm [Deisenroth and Rasmussen 2011].

1.4 Organization of This Document

Key notation used throughout this document is summarized in the front matter. Chapter 2 presents a brief introduction to dynamic programming and reinforcement learning, covering most of the necessary concepts for the remainder of the text. The technical introductions to the more advanced topics of options and shaping are reserved until their corresponding chapters: Sections 4.2.2 and 6.1.1, respectively. Chapters 3 through 6 present the key contributions of this thesis. The results are presented in a logical, rather than chronological order, and for example the earliest results on shaping are presented last. We defer all of the proofs to the appendices to make the narrative and discussions prominent. Chapter 7 concludes the thesis and outlines several directions for future work. Finally, a summary of peer-reviewed and pending publications that underpin each chapter is presented in the back matter.

2 | Dynamic Programming and Reinforcement Learning

This chapter will give a short technical introduction to dynamic programming and reinforcement learning. We will first introduce the basics of dynamic programming, preview convergence results and solution methods. We will then introduce reinforcement learning, the data-driven counterpart to DP, discuss some of its key concepts, and present the basic temporal difference algorithm from first principles of stochastic approximation.

2.1 Markov Decision Processes

The Markov Decision Process (MDP) [Bellman 1957b] framework crystalizes the challenge of sequential decision making. Consider a sequence of *experience* $S_0, A_0, S_1, A_1, S_2, A_2, \ldots$ of states and actions in the environment. The Markov property assumes that the dependencies in this sequence are first-order only:

$$\Pr(S_{t+1}|\underbrace{S_t, A_t, S_{t-1}, A_{t-1}, \ldots, S_0, A_0}_{\mathcal{F}_t}) = \Pr(S_{t+1}|S_t, A_t).$$

That is, each state S_{t+1} is independent of the *history* $\mathcal{F}_{t-1} \stackrel{\text{def}}{=} S_0, A_0, \ldots, S_{t-1}, A_{t-1}$, up to the immediately preceding state-action pair S_t, A_t. The algorithms developed under this simple assumption have transferred to less ideal settings remarkably well.

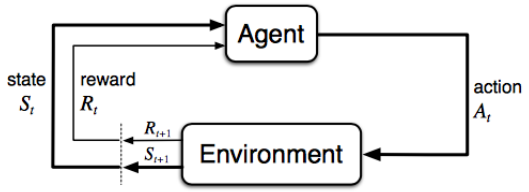

Figure 2.1: The interaction loop between the agent and the environment in reinforcement learning

Definition 2.1: Markov Decision Process

An MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where:

- $\mathcal{S}$ is the (possibly infinite) set of states
- $\mathcal{A}$ is the (possibly infinite) set of actions
- $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function that specifies environment dynamics, with $p(s' \mid s, a)$ denoting the probability of transitioning to state s' upon taking action a in state s
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the (possibly probabilistic) reward function
- $\gamma \in [0, 1]$ is the scalar discount factor that determines the planning horizon of the agent.

The agent interacts with this environment as follows: at each of a sequence of discrete time steps $t = 0, 1, 2, \ldots$, the agent observes the current state of the environment $S_t \in \mathcal{S}$ and selects an action $A_t \in \mathcal{A}$. The clock ticks, the agent receives a numerical reward $R_{t+1} \sim r(S_t, A_t)$ and finds oneself in a new state $S_{t+1} \sim p(\cdot|S_t, A_t)$.

Scope: Environment

In this thesis we will always assume that the set of actions is discrete and finite, and the reward function is bounded $\|r\|_\infty \leq r_{\max}$. We will always be concerned with the *discounted* setting, in which $\gamma < 1$. All of the theoretical results in this thesis are for finite state spaces, but the experiments will often involve function approximation, which we will cover briefly in Section 2.4.6.

2.2 Policies, Returns, and Value Functions

The agent always thinks forward. At a given time t, in an infinite horizon setting, a fundamental quantity of interest is the *return*. Given the experience stream $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, \ldots$, the return of this experience is defined as follows:

$$G_t \stackrel{\text{def}}{=} \sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1}. \tag{2.1}$$

That is: G_t is the future discounted cumulative reward from time t. Equation (2.1) reveals the role of discounting: how does a reward now compare to a reward later? The discounted formulation also allows one to bound G_t easily:

$$|G_t| \leq \frac{1}{1-\gamma} r_{\max}, \tag{2.2}$$

The rule with which an agent selects actions is called a *policy*, and is often denoted by π. Loosely speaking, a policy that collects the most rewards is called *optimal*. In the MDP setting it is known that an optimal policy does not need a memory and may only depend on the current state. Furthermore, it is known that to be optimal over an *infinite horizon* (in the sense of the infinite sum formulation of G_t from (2.1)), a policy can be *stationary*, and not depend on the time that a state was encountered [Puterman 1994]. The agent, hence, may focus its attention to that restricted policy class.

Definition 2.2: Policy

A (memoryless, stationary) policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a probabilistic mapping from states to actions, with $\pi(a \,|\, s)$ denoting the probability of the agent taking action a upon observing state s.

If the actions are chosen with some policy π, the return G_t provides a sample of π's long-term quality. Indeed, the *value* of a policy is defined as the expected return obtained when following it:

$$q^{\pi}(s, a) \stackrel{\text{def}}{=} \mathbb{E}_{\substack{S_{t+1:\infty} \\ A_{t+1:\infty}}} \Big[\underbrace{\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1}}_{G_t} \,|S_t = s, A_t = a; \pi \Big], \tag{2.3}$$

This expectation is implicitly conditioned on the environment dynamics p: each $S_{i+1} \sim p(\cdot|S_i, A_i)$ (while each $A_i \sim \pi(\cdot|S_i)$). Importantly, because both the policy and the

dynamics are stationary, this expectation is independent of time t. Eq, (2.3) describes an *action-value* function. One may also be interested in the overall value of the state, which is defined in a natural way:

$$v^\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_{\substack{S_{t+1:\infty} \\ A_{t:\infty}}} \Big[\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1} | S_t = s, \pi \Big] = \sum_{a \in \mathcal{A}} \pi(a|s) q^\pi(s,a) \tag{2.4}$$

These functions are typically referred to as state-action value functions / Q-values / Q-functions and state value functions / V-values / V-functions, respectively.

Reinforcement learning and dynamic programming are either concerned with *evaluating* a given policy π, or, in a *control* setting, with finding the best such policy. We can now make this notion of optimality precise. Let Π denote the space of all mappings $\mathcal{S} \times \mathcal{A} \to [0,1]$. The *optimal* value functions are defined as follows:

$$v^*(s) \stackrel{\text{def}}{=} \max_{\pi \in \Pi} v^\pi(s) = \max_{a \in \mathcal{A}} q^*(s,a) \stackrel{\text{def}}{=} \max_{a \in \mathcal{A}} \max_{\pi \in \Pi} q^\pi(s,a) \tag{2.5}$$

In the following, we will simply write $\max_\pi$ to imply maximization over the space of policies Π. Finally, the optimal policy π^* can be readily inferred from these value functions due to the fact that there always exists an *optimal deterministic stationary* policy π^*, for which $v^{\pi^*} = v^*$ [Puterman 1994, Bertsekas and Tsitsiklis 1996].[1]

Definition 2.3: Problems

Policy evaluation. How good is a given policy π? What is its value v^π?

Control. What is the *best* policy π^*? What π^* is of maximum value $\max_\pi v^\pi$?

2.3 Dynamic Programming

At the heart of any usage of the term dynamic programming is a recursion: decomposing the solution to the problem into a simple small piece, and the "rest". The Markov property lends the value function to this type of a decomposition naturally:

$$q^\pi(s,a) = \mathbb{E}_{\substack{S_{t+1:\infty} \\ A_{t+1:\infty}}} \left[\sum_{i=t}^{\infty} \gamma^{i-t} R_{i+1} \mid S_t = s, A_t = a; \pi \right]$$

[1] That such π^* exists for all states requires deliberation: why wouldn't there be several policies that are optimal only in some states? The intuition for why this is not the case is that since policies are memoryless, they can always be composed. If π is optimal in s and π' in s', one may simply obtain π^* by following these policies in their respective states. A similar argument can be made for why there is a π^* that is deterministic: if a policy is stochastic over actions, a deterministic one that chooses the action of the higher value will achieve an overall higher reward, and hence there is no reason for the randomization.

$$= \mathbb{E}_{\substack{S_{t+1:\infty} \\ A_{t+1:\infty}}} \left[R_{t+1} + \gamma \sum_{i=t+1}^{\infty} \gamma^{i-t-1} R_{i+1} \,|\, S_t = s, A_t = a; \pi \right]$$

$$= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \,|\, s,a) \underbrace{\mathbb{E}_{\substack{S_{t+2:\infty} \\ A_{t+1:\infty}}} \left[\sum_{i=t+1}^{\infty} \gamma^{i-t-1} R_{i+1} \,|\, S_{t+1} = s'; \pi \right]}_{v^{\pi}(s')} \tag{2.6}$$

$$= r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \,|\, s,a) \sum_{a' \in \mathcal{A}} \pi(a' \,|\, s') q^{\pi}(s', a'). \tag{2.7}$$

This is the *Bellman equation* [Bellman 1957a] for a policy (2.7), but w.r.t. the optimal policy π^*:

$$q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q^*(s', a'). \tag{2.8}$$

This equation is based on Bellman's *principle of optimality* (Def. 2.4), which has become fundamental more broadly in computer science, since its original formulation for decision making. The intuition to why it is especially significant in RL has to do with the fact that while value functions are complex objects that measure future rewards, the first term $r(s,a)$ is the *immediate* one-step reward. It is thus readily available for sampling when learning from simulation, which is the setting we will consider in Section 2.4.

Definition 2.4: Bellman's Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. [Bellman 1957a], Chap. III.3.

The basis of most DP solution methods lies in the fact that iterating an approximate form of (2.8) and (2.7) results in convergence to the desired value. Namely, consider an arbitrary Q-function q and a target policy π. The update that applies the following rule to all state-action pairs s, a:

$$q_{t+1}(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \,|\, s,a) \sum_{a' \in \mathcal{A}} \pi(a'|s') q_t(s', a') \tag{2.9}$$

yields $q_t \to q^{\pi}$, as $t \to \infty$, almost surely. The following section will detail the reasons behind this.

Remark 2.1: Connection between Q- and V-functions

We can use the relationship of Eq. (2.6) to make a more precise connection between Q- and V-functions. In particular, consider an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, and an enhanced MDP $\mathcal{M}'_\pi = (\mathcal{S}^+, \mathcal{A}, p^+, r, \gamma)$ containing all of the states of $\mathcal{M}$, as well as a state for each s, a tuple: $\mathcal{S}^+ = \mathcal{S} \cup \{s, a\}_{s \in \mathcal{S}, a \in \mathcal{A}}$. The transitions from the new states are then made according to the original MDP, while, a state s transitions to the state s, a according to the policy probability: $p^+(s, a|s) = \pi(a|s)$, $p^+(s'|s, a) = p(s'|s, a), \forall s, s' \in \mathcal{S}, a \in \mathcal{A}$. Then, Equations (2.6) and (2.7) can be seen as one- and two-step Bellman equations for $\mathcal{M}'_\pi$, respectively.

2.3.1 Bellman Operators

In this section we will review some of the fundamentals at the basis of convergence of dynamic programming algorithms.

Many operations in DP apply transformations onto the value function object, and it is therefore convenient to think of them as operators. Consider the Q-function q as a mapping $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. An operator $\mathcal{X}$ over Q-functions then takes a function $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and maps it to a function over the same domain and range, $\mathcal{X}q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. First let us consider a one-step transition operator over Q-functions. Given a policy π, define $\mathcal{P}^\pi$ as the following:

$$\mathcal{P}^\pi q(s, a) \stackrel{\text{def}}{=} \sum_{s' \in \mathcal{S}} \sum_{a' \in \mathcal{A}} p(s' \mid s, a) \pi(a' \mid s') q(s', a'). \tag{2.10}$$

Recall the problems from Def. 2.3, and let us expose the case of policy evaluation in detail.

2.3.2 Policy Evaluation

Let us write the *one-step Bellman operator* [Bellman 1957a] for a policy π, that corresponds to applying the update from Eq. (2.9) to all state-action pairs:

$$\mathcal{T}^\pi q \stackrel{\text{def}}{=} r + \gamma \mathcal{P}^\pi q. \tag{2.11}$$

The *fixed point* of an operator is the value of its argument function at which applications of the operator incur no further change.

Definition 2.5: Fixed point

A function f is said to be a fixed point of an operator $\mathcal{X}$, if $\mathcal{X}f = f$.

For example, if we let $\mathcal{D}$ be the differentiation operator, the function e^x is its fixed point: $\mathcal{D}e^x = e^x$. The Bellman equation (2.7) tells us that q^π is the fixed point of the Bellman operator:

$$\mathcal{T}^\pi q^\pi = q^\pi = \sum_{t=0}^{\infty} \gamma^t (\mathcal{P}^\pi)^t r = (I - \gamma \mathcal{P}^\pi)^{-1} r, \tag{2.12}$$

where for any operator $\mathcal{X}$, $\mathcal{X}^t$ denotes t successive applications of $\mathcal{X}$, and the last equality is a standard result for stochastic operators and matrices [Kemény and Snell 1960]. This solution can be computed exactly if the state space is not too large. The key strength of DP is that when that is not the case, *iterative methods, based on repeated applications of $\mathcal{T}^\pi$ converge to q^π*.

So far, we only know that q^π is *a* fixed point of $\mathcal{T}^\pi$, but we do not know whether it is unique, and whether we can find it. The answer to both of these questions is in the affirmative, due to the fact that $\mathcal{T}^\pi$ is a *contracting operator*: it maps two Q-functions closer together with respect to some metric.

Definition 2.6: Contraction

An operator $\mathcal{X}$ over a metric space $(\mathcal{Q}, d)$ is a contraction if $\exists \eta \in [0, 1)$, s.t. $d(\mathcal{X}q, \mathcal{X}z) \leq \eta d(q, z)$, $\forall q, z \in \mathcal{Q}$.

As the metric, we will consider the maximum ℓ_∞-norm:

$$\|q - z\|_\infty \stackrel{\text{def}}{=} \max_{s \in \mathcal{S}, a \in \mathcal{A}} |q(s,a) - z(s,a)|, \quad q, z \in \mathcal{Q}.$$

To verify that $\mathcal{T}^\pi$ is indeed a contraction, first let us verify that the space of Q-functions $\mathcal{Q}$ together with the maximum ℓ_∞-norm is a metric space. Since $\mathcal{Q}$ is a vector space,[2] combining it with any norm gives a normed vector space, which in turn is a metric space. It is then easy to verify that $\mathcal{T}^\pi$ obeys the required inequality w.r.t. the maximum norm:

$$\begin{aligned} \|\mathcal{T}^\pi q_1 - \mathcal{T}^\pi q_2\|_\infty &= \|r + \gamma \mathcal{P}^\pi q_1 - r - \gamma \mathcal{P}^\pi q_2\|_\infty \\ &= \gamma \|\mathcal{P}^\pi q_1 - \mathcal{P}^\pi q_2\|_\infty \end{aligned}$$

[2] This is because Q-functions are defined as arbitrary *candidate* mappings $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, rather than necessarily values q^π of some policy π.

$$
\begin{aligned}
&= \gamma \|\mathcal{P}^\pi (q_1 - q_2)\|_\infty \\
&\overset{(*)}{\leq} \gamma \|q_1 - q_2\|_\infty .
\end{aligned}
$$

where $(*)$ is due to $\|\mathcal{P}^\pi x\|_\infty \leq \|x\|_\infty$ for any vector x, which follows from the definition (2.10). Thus, for discounted problems, where $\gamma < 1$, $\mathcal{T}^\pi$ is a contraction of modulus γ.

Finally, we will need the notion of a *Banach space*: a normed vector space that is *complete*. Intuitively, for a metric space to be complete means that it does not contain any "holes". Slightly more precisely, it is the requirement for a limit of a sequence of elements from the space to be contained in the space. Bounded continuous-valued functions can be shown to be complete, and therefore the space of value functions $\mathcal{Q}$ with the ℓ_∞-norm is a Banach space, if value functions are bounded.

Assumption 2.1: Bounded Values

The value function is bounded: $\|q\|_\infty < +\infty$.

Note that when the reward function is bounded, as we assume in this thesis, this assumption is easily satisfied in the discounted case via Eq. (2.2).

The following theorem is at the basis of many principal convergence results in dynamic programming.

Theorem 2.1: Banach Contraction Mapping Theorem

A contraction map $\mathcal{X}$ on a Banach space $\mathcal{B}$ has a unique fixed point. Furthermore,the sequence $b, \mathcal{X}b, \mathcal{X}^2 b, \ldots$ converges to that unique fixed point, $\forall b \in \mathcal{B}$.

Thus, if Assumption 2.1 holds, q^π is indeed the *unique* fixed point of $\mathcal{T}^\pi$, and repeated applications of $\mathcal{T}^\pi$ are guaranteed to yield it. It is easy to see that the contraction condition implies a worst-case geometric rate of convergence. In our case:

$$
\|(\mathcal{T}^\pi)^n q - q^\pi\| \leq \gamma \|(\mathcal{T}^\pi)^{n-1} q - q^\pi\| \leq \gamma^2 \|(\mathcal{T}^\pi)^{n-2} q - q^\pi\| \leq \ldots \leq \gamma^n \|q - q^\pi\|.
$$

This relation reveals the crucial importance of the discounting rate γ on the convergence rate of DP algorithms: the larger the γ, the slower the convergence.

Note that Theorem 2.1 applies independently of the initial q. A weaker condition can be devised for monotone (but non-contracting) sequences: if their fixed point is unique and they are bounded, they converge to this fixed point. This result is sometimes invoked in the *control* setting.

2.3.3 Control

Recall that in the control setting we are interested in finding the policy of maximum value $q^* = \max_\pi q^\pi$, which corresponds to the Bellman *optimality* operator:

$$\mathcal{T}q \stackrel{\text{def}}{=} r + \gamma \max_\pi \mathcal{P}^\pi q. \tag{2.13}$$

The optimal value function q^* is the fixed point of this operator, as stated by the Bellman optimality equation (2.8):

$$\mathcal{T}q^* = q^*. \tag{2.14}$$

Similarly to $\mathcal{T}^\pi$, $\mathcal{T}$ is a contraction around this fixed point w.r.t. the ℓ_∞-norm:

$$\begin{aligned}
\|\mathcal{T}q_1 - \mathcal{T}q_2\|_\infty &= \|r + \gamma \max_\pi \mathcal{P}^\pi q_1 - r - \gamma \max_\pi \mathcal{P}^\pi q_2\|_\infty \\
&= \gamma \|\max_\pi \mathcal{P}^\pi q_1 - \max_\pi \mathcal{P}^\pi q_2\|_\infty \\
&\leq \gamma \|\max_\pi \mathcal{P}^\pi (q_1 - q_2)\|_\infty \\
&\leq \gamma \|q_1 - q_2\|_\infty.
\end{aligned}$$

Therefore, Theorem 2.1 implies convergence once again, now to q^*.

Finally, note that iterating $\mathcal{T}$ is subtly different from iterating $\mathcal{T}^\pi$, since the underlying policy depends on the approximation q. That is: at each step we consider our best *current guess* for an optimal policy, the policy that would be optimal if the value function were correct. Such a policy is called *greedy*, and simply corresponds to choosing the action of maximum value in each state. We write

$$\mathcal{G}(q) \stackrel{\text{def}}{=} \{\pi \,|\, \pi(a \,|\, s) > 0 \Rightarrow q(s,a) = \max_{a' \in \mathcal{A}} q(s,a')\}$$

to denote the set of greedy policies w.r.t. q. That is: $\mathcal{T}q = \mathcal{T}^\pi q$ for any $\pi \in \mathcal{G}(q)$.

2.3.4 λ-Operators

Clearly, the Bellman equation holds for any number of applications of $\mathcal{T}^\pi$:

$$q^\pi = \mathcal{T}^\pi q^\pi = \mathcal{T}^\pi(\mathcal{T}^\pi q^\pi) = \mathcal{T}^\pi(\mathcal{T}^\pi(\mathcal{T}^\pi q^\pi)) = \ldots = (\mathcal{T}^\pi)^n q^\pi = \ldots, \quad \forall n \in \mathbb{N}.$$

All of the convergence properties carry over, and one may use the repeated n-*step* operator $(\mathcal{T}^\pi)^n$ to converge to q^π. This reduces the number of iterations required for convergence,

but makes each iteration more involved (e.g. [Bertsekas and Tsitsiklis 1996]). In particular, n applications of $\mathcal{T}^\pi$ induce n reward models, followed by the value function:

$$(\mathcal{T}^\pi)^n q = \sum_{t=0}^{n} (\gamma \mathcal{P}^\pi)^t r + (\gamma \mathcal{P}^\pi)^{n+1} q. \tag{2.15}$$

The choice of n has dramatic influence on efficiency and its best choice is problem-specific. A more flexible form of multi-step operators can be obtained by considering a *geometrically weighted sum* of the n-step operators:

$$\mathcal{T}_\lambda^\pi \stackrel{\text{def}}{=} (1-\lambda) \sum_{n=0}^{\infty} \lambda^n ((\mathcal{T}^\pi)^{n+1} q). \tag{2.16}$$

Naturally, q^π remains the fixed point of $\mathcal{T}_\lambda^\pi$. Taking $\lambda = 0$ yields the usual single-step Bellman operator $\mathcal{T}^\pi$, and $\lambda \to 1$ corresponds to an infinite number of applications of $\mathcal{T}^\pi$, which in turn corresponds to the closed form solution from Eq. (2.12).

2.3.5 Iterative Algorithms

Finally, let us formalize the so far implicit algorithms based on iterating $\mathcal{T}^\pi$ and $\mathcal{T}$. In all cases we consider iterations $k = 0, 1, 2, \ldots$, and q_0 initialized arbitrarily.

Following our taxonomy of problems from Def. 2.3, consider first the problem of evaluating a given policy π. If the state space is small, we may simply obtain the value of q^π from Eq. (2.12). Of course, in many problems of interest, this is intractable, and we may have to reach the solution iteratively, that is: by maintaining a sequence of value functions $(q_k)_{k \in \mathbb{N}}$. The Bellman operator $\mathcal{T}^\pi$ is the operator underlying *iterative policy evaluation*:

$$q_{k+1} \leftarrow \mathcal{T}^\pi q_k. \tag{2.17}$$

Now consider the problem of control. Here, we consider a sequence of value functions $(q_k)_{k \in \mathbb{N}}$ *and* their corresponding greedy policies $(\pi_k)_{k \in \mathbb{N}}$. If the state space is small, we may compute the value of each π_k exactly, and perform *policy iteration (PI)*. That is: starting from the arbitrary values q_0, we may at each iteration k *evaluate* this policy in closed form via Eq. (2.12), and try to *improve* on it by devising a new policy:

$$\begin{aligned} \pi_{k+1} &\leftarrow \arg\max_\pi q_{k-1} \\ q_{k+1} &= (I - \gamma \mathcal{P}^{\pi_{k+1}})^{-1} r, \end{aligned} \tag{2.18}$$

If the state space is large, we may wish to bypass the task of exactly evaluating the intermediate policies (π_k), and focus on improving the estimates (q_k) directly, performing

value iteration (VI):

$$q_{k+1} \leftarrow \mathcal{T} q_k. \tag{2.19}$$

Noticing that, for the given greedy policy, VI takes one evaluation step, while PI corresponds to taking infinitely many evaluation steps, it is natural to imagine the middle ground between them. Namely, *modified* policy iteration [Puterman and Shin 1978] takes m evaluation steps, and λ-policy iteration further blends the m-steps via the $\mathcal{T}_\lambda^\pi$ operator from Eq. (2.16) [Bertsekas and Ioffe 1996]. We discuss this latter algorithm in detail in Chapter 4.

The convergence of all of these algorithms is guaranteed by Theorem 2.1. We have so far considered them in the *synchronous* setting: with the updates being applied to all states and actions at once in a single iteration. In online learning, whether prediction or control, the updates ought to be applied to sequentially sampled transitions. Luckily, it is known that convergence holds in such an *asynchronous* setting as well, under reasonable conditions. This theory, originally developed for processing the state space in parallel [Bertsekas 1982], was first related to the online learning setting by [Barto et al. 1995] and has been crucial for the theory of modern RL and the algorithms we are about to discuss.

2.4 Reinforcement Learning

The previous section briefly introduced the fundamentals of dynamic programming, and the ability of iterating $\mathcal{T}$ and $\mathcal{T}^\pi$ to produce accurate Q-functions. The resulting algorithms, value and policy iteration, however, require access to the models r and p. *Approximate* dynamic programming, or *reinforcement learning*, is distinguished by the assumption that these models are not available a priori, but are sampled, that is: the agent interacts with a simulation of the MDP. We will continue considering *action-value*, or *value-based* algorithms, in which the agent's goal is to learn value functions.[3] Depending on the targets of estimation from the obtained samples, value-based solution methods fall into two categories.

Definition 2.7: Value-based RL Algorithms

Model-based approaches use samples to estimate the models r and p, and obtain the desired value functions by applying DP algorithms with these approximate models.

[3] The alternative is policy search methods, discussed in Section 1.3.1.

Model-free approaches use samples to estimate the desired value functions directly, without maintaining approximate models.

We will start from first principles of stochastic approximation, and briefly introduce a naive model-based framework. We will then move to describing *temporal difference learning*, as an instance of a model-free algorithm.

2.4.1 The Blueprint for Stochastic Approximation

Consider a task of estimating an expectation x from samples of the corresponding random variable $(X_k)_{k\in\mathbb{N}}$. By the law of large numbers, this expectation can be estimated iteratively by taking a mixture of the independent samples $(X_k)_{k\in\mathbb{N}}$:

$$\widehat{x}_{k+1} \leftarrow (1-\alpha_k)\widehat{x}_k + \alpha_k X_k, \tag{2.20}$$

where α_k is the step-size.[4] This basic procedure is at the heart of many stochastic approximation algorithms. Indeed, if the sequence of step-sizes $(\alpha_k)_{k\in\mathbb{N}}$ is appropriately decreasing, the sequence $(\widehat{x}_k)_{k\in\mathbb{N}}$ converges to x almost surely, as $k \to \infty$ [Robbins and Monro 1951]. A minimal requirement for the step-sizes to be appropriately decreasing is formalized in the following assumption.

Assumption 2.2: Robbins-Monro

The sequence of step-sizes $(\alpha_k)_{k\in\mathbb{N}}$ satisfies:

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty.$$

The simplest way to perform model-based RL is to use this procedure to estimate p and r, and perform value or policy iteration with the models $\widehat{p}$ and $\widehat{r}$ approximated along the way. There are much more sophisticated model-based methods (e.g. [Brafman and Tennenholtz 2002]), but we limit our discussion to the model-free scope of this thesis.

2.4.2 Learning from Monte Carlo Returns

The term Monte Carlo simply refers to the principle of estimating an expectation via an average of random samples, or *rollouts* (e.g. [Michie and Chambers 1968]). As such, Eq. (2.20) can also be interpreted as an incremental *Monte Carlo* procedure of estimating x.

[4] E.g., if $\alpha_k = \frac{1}{k}$, Eq. (2.20) describes the empirical mean.

In (value-based) RL, the goal is to estimate a value function q^π, or q^*. Let us consider the case of policy evaluation. The difficulty of applying the update (2.20) is in obtaining an unbiased sample $q_k(s,a)$ of q^π for each state-action pair. The simplest way of doing so is to use the complete return $G_k^\pi(s,a)$, where $G_k^\pi(s,a)$ denotes the return G_0 from Eq. (2.1) w.r.t. kth sample of an experience stream $s, a, R_1, S_1, A_1, R_2, \ldots, A_i \sim \pi(\cdot|S_i)$:

$$G_k^\pi(s,a) \stackrel{\text{def}}{=} \sum_{t=0}^{\infty} \gamma^t R_{t+1}.$$

This is exactly analogous to the above description of Monte-Carlo estimation, as $G_k^\pi(s,a)$ is an unbiased sample of $q^\pi(s,a)$, and hence the update for each state and action pair is exactly analogous to the Monte Carlo update of Eq. (2.20):

$$q_{k+1}(s,a) \leftarrow (1-\alpha_k) q_k(s,a) + \alpha_k G_k^\pi(s,a). \tag{2.21}$$

There are some subtleties that arise due to the sequential nature of this process. In particular, if one is to encounter the pair s, a *again* along the trajectory, does one simply ignore it? Or make another update? The former approach is referred to as *first visit* Monte Carlo, while the latter is referred to as *every visit*. Given limited samples, the every visit approach is more savvy, and is often preferred in practice. While its analysis is slightly more involved, convergence results can be obtained for both variants [Singh and Sutton 1996].

Unfortunately, the variance of the *multi-step returns* G_k^π involved in this computation, can be very high. Furthermore, in the infinite horizon setting, one must wait indefinitely for each G_k^π. This is where the Bellman equation proves extremely useful.

2.4.3 Learning from Temporal Differences

Given an estimate q_k of the Q-function, and a short sample experience $s, a, R_{t+1}, S_{t+1}, A_{t+1}$, consider replacing $G_k^\pi(s,a)$ with $\widehat{G_k^\pi}(s,a) \stackrel{\text{def}}{=} R_{t+1} + \gamma q_k(S_{t+1}, A_{t+1})$, or in other words, only sampling the immediate reward, and relying on the current approximation for the remainder:

$$\begin{aligned} q_{k+1}(s,a) &\leftarrow (1-\alpha_k) q_k(s,a) + \alpha_k \widehat{G_k^\pi}(s,a) \\ &= (1-\alpha_k) q_k(s,a) + \alpha_k (R_{t+1} + \gamma q_k(S_{t+1}, A_{t+1})) \\ &= q_k(s,a) + \alpha_k \underbrace{(R_{t+1} + \gamma q_k(S_{t+1}, A_{t+1}) - q_k(s,a))}_{\text{TD error } \delta_t}. \end{aligned} \tag{2.22}$$

This update is at the basis of temporal difference learning [Sutton 1988]. The quantity δ_t is referred to as the *temporal difference (TD) error*, due to the difference of

the estimates $R_{t+1} + q_k(S_{t+1}, A_{t+1})$ of a future time step and the estimate $q_k(s,a) = q_k(S_t, A_t)$ of the current time step. When (q_k) has converged to the value of the policy π that is used to sample A_t, the expected value of the TD error becomes 0. Indeed, $\mathbb{E}_\pi \left[R_{t+1} + \gamma q^\pi(S_{t+1}, A_{t+1}) - q^\pi(s,a) \right] = 0$, due to the Bellman equation.

The convergence conditions for this process are surprisingly unrestrictive: convergence is guaranteed if the step-sizes obey Assumption 2.2, which is indeed the minimal requirement for stochastic approximation convergence in general, and if all states and actions are visited sufficiently often, which can be formalized via the following assumption.[5]

Assumption 2.3: Minimum visit frequency

For any trajectory $S_0, A_0, S_1, A_1, \ldots$, and any state action pair $s \in \mathcal{S}, a \in \mathcal{A}$, there exists a constant D, s.t.:

$$\sum_{t=0}^{\infty} \Pr\{S_t, A_t = s, a\} \geq D > 0.$$

2.4.4 Off-Policy Learning and Exploration

Consider the snippet of experience $s, a, R_{t+1}, S_{t+1}, a'$ used as an input to all of the procedures above. Importantly, the following action a' is not limited to the taken action A_{t+1}. In fact, the choice of it directly determines the limit of convergence of the update. If a' is indeed the next sample A_{t+1}, the problem is that of *policy evaluation*, the algorithm is SARSA [Rummery and Niranjan 1994] (aptly named after the snippet of experience $s, a, R_{t+1}, S_{t+1}, a'$), and the convergence is to q^π, where π is the policy used to sample A_{t+1}. If a' on the other hand is the *greedy* action at S_{t+1} w.r.t. q, the problem is that of *control*, the algorithm is Watkins's Q-Learning [Watkins 1989], and the convergence is to q^*. This brings us directly up to a key dimension in RL: learning on- or off-policy.

Definition 2.8: Off-Policy Learning

Learning is said to be *off-policy* when samples generated by a *behavior* policy μ are used to learn about a distinct *target* policy π. If $\mu = \pi$, the learning is *on-policy*.

Policy evaluation can be either on- or off-policy, while control is almost always off-policy.[6]

[5] This assumption is equivalent to the standard requirement of all states being visited infinitely often, but we will find this form convenient in the analysis of Chapter 3.

[6] An exception is the class of optimal stopping problems [Puterman 1994].

The scenario of online sampling introduces a unique challenge for control in RL: how to ensure that all states and actions are visited sufficiently often? One simple way is to behave w.r.t. a fixed random policy, but this may not lead one to the interesting parts of the state space in a reasonable amount of time (though it will in the limit of infinite time). The other extreme is to follow the agent's best, *greedy* guess of the optimal policy, but then convergence indeed cannot be ensured, since there is the risk of missing an even better policy. This is known as the *exploration-exploitation* tradeoff, and is among the most fundamental problems in RL (e.g. [Thrun 1992]). A simple approach often adopted in practice is using ϵ-*greedy* policies, that is: ones that take the greedy action w.p. $1-\epsilon$ and a random action w.p. ϵ. This simple strategy has been surprisingly effective (e.g. [Mnih et al. 2015]), but many more sophisticated approaches exist that are necessary for the more challenging settings (e.g. [Bellemare et al. 2016]).

2.4.5 λ-Returns

Just like when considering expected operators from the previous section, in the case of learning from simulation one need not update with (or *back up*) a single step, and may instead consider multiple steps from the sampled trajectory $s, a, R_1, S_1, A_1, R_2, \ldots$. Then, the n-*step return* $G^\pi_n(s,a)$ is a sample of $(\mathcal{T}^\pi)^n$, and the λ-return G^π_λ is a sample of $\mathcal{T}^\pi_\lambda$:

$$G^\pi_n(s,a) \stackrel{\text{def}}{=} \sum_{k=1}^{n} \gamma^{k-1} R_{t+k} + \gamma^n q(S_n, A_n),$$

$$G^\pi_\lambda(s,a) \stackrel{\text{def}}{=} (1-\lambda) \sum_{n=0}^{\infty} \lambda^n G^\pi_{n+1}(s,a), \tag{2.23}$$

where $S_0, A_0 = s, a$, as before. That is: $\lambda = 0$ corresponds to the one-step TD update from (2.22), and $\lambda \to 1$ removes the recursion on the approximate Q-function, and restores G^π in the Monte Carlo sense. In this case, of samples, the *computational* expense of applying multi-step operators is replaced by an increase in variance of sampling multi-step returns. Indeed, λ trades off this variance with the bias from bootstrapping with an approximate Q-function [Kearns and Singh 2000], and intermediate values of λ thus usually perform best in practice [Sutton 1996, Singh and Dayan 1998].

From the specification of G^π_λ in (2.23), the update w.r.t. to it seems to require being carried out *offline*: after the reward stream has terminated. In the absence of a finite horizon, this is as infeasible as plain Monte Carlo estimation. Luckily, there is a mechanism to efficiently implement the update *online*, called eligibility traces [Sutton and Barto 2017]. Let e denote the vector of such traces, and consider the following update at time t:

$$e(s,a) \leftarrow \lambda\gamma e(s,a) + \mathbb{I}\{(S_t, A_t) = (s,a)\}, \quad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.24}$$

Like with Monte Carlo updating, there is a choice between every-visit and first-visit updating of the eligibility trace. The equation above is given in the more practical every-visit form, which we will assume throughout. It has recently been argued that the update (2.24) is subtly flawed. [van Seijen and Sutton 2014] showed it does not in fact yield exact equivalence to the offline case, and proposed a *true online* mechanism that does, allowing it to produce more stable estimates.

The parameter λ is commonly highlighted in the algorithm name. Thus, for example, Eq. (2.21) describes SARSA(1), Eq. (2.22) SARSA(0), and an algorithm that updates its estimates with G^{π}_{λ} is referred to as SARSA(λ). In the policy evaluation setting, in particular when estimating V-values, this algorithm is also known as *TD(λ)*.

> **Scope: Algorithms**
>
> The majority of the thesis will be concerned with model-free temporal difference methods. The two exceptions are the first part of Chapter 4 whose focus is exact dynamic programming, and Chapter 5 which considers model-based RL, or approximate dynamic programming, where the models are estimated from samples.

2.4.6 Approximate State Spaces

Everything up to this point has assumed a discrete, finite state space. In most practical settings, the state space is continuous or too large to be represented exactly, and one needs to resort to approximation. The value function is then represented through a set of features ϕ. All of the same principles apply, but the Q-function q_θ becomes parameterized by a parameter vector θ, and the algorithms apply their updates to θ directly.

For example, if the approximation is linear, the Q-function is obtained by a linear combination of θ with ϕ:

$$q_\theta(s, a) = \theta^T \phi_{s,a},$$

and given an experience $s, a, R_{t+1}, S_{t+1}, A_{t+1}$, the update (2.22) becomes:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k(R_{t+1} + \gamma \theta_k^T \phi_{S_{t+1},A_{t+1}} - \theta_k^T \phi_{s,a}).$$

Tile coding is one simple and surprisingly effective example of a linear function approximation scheme [Sutton and Barto 2017].

The approximation can also be performed via a neural network, in which case the Q-function is obtained as the output of the network on an input of a state observation. This setting is particularly powerful, because the features ϕ are then not fixed a priori,

but learnt in the context of the task. This framework of *deep RL* has led to remarkable successes in the last years [Mnih et al. 2015, Silver et al. 2017], and has awakened a new wave of RL research.

2.5 Summary

We have briefly introduced the key ideas from dynamic programming and reinforcement learning. The next chapter will pick up almost exactly where we leave off and consider the problem of learning off-policy from λ-returns.

If it was so, it might be; and if it were so, it would be; but as it isn't, it ain't.
— Lewis Carroll, *Alice's Adventures in Wonderland and Through the Looking-Glass.*

3 | Off-Policy Learning with Corrections

The ability to learn about counterfactuals and ask "what if?" questions is essential, but has been (perhaps unsurprisingly) elusive to perform efficiently. One of the fundamental questions involved is: How does one measure the similarity of the current experience with the experience one wishes to learn about? How close must these experiences be in order for any learning to occur? After all, it seems futile to expect an ability to learn how to swim while walking. The difficulty of answering this question is compounded with the lengths of the experiences.

Reinforcement learning, following suit of stochastic approximation, traditionally answers this question with the help of the *importance sampling ratio* of the probabilities of the involved policies. In this chapter we challenge the necessity of doing this exactly, and advocate using the *value function* as another approximate similarity signal.

We first investigate a *naive* implementation of the idea: what happens when the difference in policies is unaccounted for entirely? The answer to this question substantiates an old algorithm, and yields a new one. Taking a step back, we formulate a unified view of the existing landscape, and describe a novel algorithm that in some sense combines the best of all worlds.

3.1 Introduction

The usual approach to off-policy learning is to disregard, or altogether discard transitions whose target policy probabilities are low. For example, Watkins's Q(λ) [Watkins and Dayan 1992] cuts the trajectory backup as soon as a non-greedy action is encountered. Similarly, in policy evaluation, importance sampling methods [Precup et al. 2000] weight the returns according to the mismatch in the target and behavior probabilities of the corresponding actions. This approach treats transitions conservatively, and hence may unnecessarily terminate backups, or introduce a large amount of variance.

Many off-policy methods, in particular of the Monte Carlo kind, have no other option than to judge off-policy actions in the probability sense. However, temporal difference methods [Sutton 1988] in RL maintain an approximation of the value function along the way, with *eligiblity traces* [Watkins 1989] providing a continuous link between one-step and Monte Carlo approaches. The value function assesses actions in terms of the future expected cumulative reward, and thus provides a way to directly correct immediate *rewards*, rather than transitions. We show in this chapter that:

- such approximate corrections alone can be sufficient for off-policy convergence, subject to a tradeoff condition between the eligibility trace parameter and the distance between the target and behavior policies.
- the corrections, combined with an *approximate*, truncated importance sampling ratio yield guaranteed convergence for arbitrary target and behavior policies.

In particular, we propose an off-policy return operator that augments the return with a correction term, based on the current approximation of the Q-function. We formalize three algorithms stemming from this operator, and analyze their convergence, which we show holds subject to a certain tradeoff between the return "length" λ and the *off-policy-ness* of the behavior.[1]

We take a step back in Section 3.3 and review several related off-policy return-based algorithms. Expressing them in a general common form, we analyze the convergence properties of this general form. We then motivate and derive an improved online algorithm, Retrace(λ), which is both *safe* (that is: enjoys general convergence guarantees for any λ and any amount of off-policy-ness), and *efficient* (that is: able to learn quickly by maximally utilizing the returns). As a corollary to our analysis, the first proof of convergence of Watkins' Q(λ) follows.

[1] The exploration parameter ϵ from the familiar ϵ-greedy exploration is an example of off-policy-ness in this context.

3.2 Naive Off-Policy Corrected Returns

In this section, we consider a *naive* form of our idea, in which the difference in policy probabilities is unaccounted for entirely, and all of the corrections are performed through the value function. We will see that even this scenario can yield convergence, but it is subject to a certain tradeoff.

3.2.1 Operators

Let us describe the Monte Carlo *off-policy corrected return operator* $\mathcal{R}^{\pi,\mu}$ that is at the heart of the algorithms to follow. Given a target policy π, and a return generated by a behavior policy μ, the operator $\mathcal{R}^{\pi,\mu}$ attempts to approximate a return that would have been generated by π, by utilizing a correction built from a current approximation q of q^π. Its application to q at a state-action pair (s,a) is defined as follows:

$$\mathcal{R}^{\pi,\mu}q(s,a) \stackrel{\text{def}}{=} r(s,a) + \mathbb{E}_\mu\big[\sum_{t=1}^{\infty}\gamma^t\big(R_{t+1} + \underbrace{\mathbb{E}_\pi q(S_t,\cdot) - q(S_t,A_t)}_{\text{off-policy correction}}\big)\big], \tag{3.1}$$

where we use the shorthand

$$\mathbb{E}_\pi q(s,\cdot) \equiv \sum_{a\in\mathcal{A}}\pi(a|s)q(s,a),$$

while $\mathbb{E}_\mu[\cdot]$ denotes the expectation $\mathbb{E}_{\substack{S_{1:\infty}\\A_{1:\infty}}}[\cdot]$ of a trajectory drawn w.r.t. a policy μ. That is: $\mathcal{R}^{\pi,\mu}$ gives the usual expected discounted sum of future rewards, but each reward in the trajectory is augmented with an *off-policy correction*, which we define as the difference between the *expected* (with respect to the target policy) Q-value and the Q-value for the taken action. Thus, how much a reward is corrected is determined by both the approximation q, and the target policy probabilities. Notice that if actions are similarly valued, the correction will have little effect, and learning will be roughly on-policy, but if the Q-function has converged to the correct estimates q^π, the correction takes the immediate reward R_{t+1} to the expected reward with respect to π exactly. Indeed, as we will see later, q^π is the fixed point of $\mathcal{R}^{\pi,\mu}$ for any behavior policy μ.

Analogously to Eqs. (2.15) and (2.16), we can consider the n-step[2] and λ versions of $\mathcal{R}^{\pi,\mu}$:

$$\mathcal{R}^{\pi,\mu}_n q(s,a) \stackrel{\text{def}}{=} r(s,a) + \mathbb{E}_\mu\left[\sum_{t=1}^{n}\gamma^t\big(R_{t+1} + \mathbb{E}_\pi q(S_t,\cdot) - q(S_t,A_t)\big) + \gamma^{n+1}\mathbb{E}_\pi q(S_{n+1},\cdot)\right],$$

[2]The bootstrapping term $\mathbb{E}_\pi q(S_{n+1},\cdot)$ is considered as an expectation for symmetry that will be convenient later.

$$\mathcal{R}_\lambda^{\pi,\mu} q \stackrel{\text{def}}{=} (1-\lambda) \sum_{n=0}^{\infty} \lambda^n [\mathcal{R}_n^{\pi,\mu}], \tag{3.2}$$

Note that the λ parameter here takes us from SARSA(0) to the Monte Carlo version of our operator $\mathcal{R}^{\pi,\mu}$, rather than the traditional Monte Carlo form (2.12).

Algorithm 1 Q(λ) with off-policy corrections

Given: Initial Q-function q_0, step-sizes $(\alpha_k)_{k\in\mathbb{N}}$
  for $k = 1, \ldots$ **do**
    Sample a trajectory $S_0, A_0, R_0, \ldots, S_{T_k}$ from μ_k
    $q_{k+1}(s,a) \leftarrow q_k(s,a) \qquad \forall s, a$
    $e(s,a) \leftarrow 0 \qquad \forall s, a$
    for $t = 0, \ldots, T_k - 1$ **do**
      $\delta_t^{\pi_k} \leftarrow R_{t+1} + \gamma \mathbb{E}_{\pi_k} q_{k+1}(S_{t+1}, \cdot) - q_{k+1}(S_t, A_t)$
      for all $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
        $e(s,a) \leftarrow \lambda\gamma e(s,a) + \mathbb{I}\{(S_t, A_t) = (s,a)\}$
        $q_{k+1}(s,a) \leftarrow q_{k+1}(s,a) + \alpha_k \delta_t^{\pi_k} e(s,a)$
      end for
    end for
  end for

On-policy Q$^\pi$(λ): $\mu_k = \pi_k = \pi$.
Off-policy Q$^\pi$(λ): $\mu_k \neq \pi_k = \pi$.
Q*(λ): $\pi_k \in \mathcal{G}(q_k)$.

3.2.2 Algorithms: Q$^\pi$(λ) and Q*(λ)

Recall that we are considering the problems of *off-policy policy evaluation* and *off-policy control* (Def. 2.3). In both problems we are given data generated by a sequence of behavior policies $(\mu_k)_{k\in\mathbb{N}}$. Our algorithm constructs a sequence $(q_k)_{k\in\mathbb{N}}$ of estimates of q^{π_k} from trajectories sampled from μ_k, by applying the $\mathcal{R}_\lambda^{\pi_k,\mu_k}$-operator:

$$q_{k+1} = \mathcal{R}_\lambda^{\pi_k,\mu_k} q_k, \tag{3.3}$$

where π_k is the kth interim target policy. We distinguish between three algorithms and associated operators:

Off-policy Q$^\pi$(λ) for policy evaluation: $\pi_k = \pi$ is the fixed target policy. We write the corresponding operator $\mathcal{R}_\lambda^\pi$.

On-policy Q$^\pi(\lambda)$ for policy evaluation: for the special case of $\mu_k = \mu = \pi$.

Q$^*(\lambda)$ for off-policy control: $(\pi_k)_{k\in\mathbb{N}}$ is a sequence of greedy policies with respect to q_k. We write the corresponding operator $\mathcal{R}^*_\lambda$.

We wish to write the update (3.3) in terms of a simulated trajectory $S_0, A_0, R_1, \ldots, S_{T_k}$ drawn according to μ_k. First, notice that we can rewrite Eq. (3.2) in the following form:[3]

$$\mathcal{R}^{\pi,\mu}_\lambda q(s,a) = q(s,a) + \mathbb{E}_\mu\big[\sum_{t=0}^{\infty}(\lambda\gamma)^t\delta^\pi_t\big], \tag{3.4}$$
$$\delta^\pi_t \stackrel{\text{def}}{=} R_{t+1} + \gamma\mathbb{E}_\pi q(S_{t+1},\cdot) - q(S_t,A_t),$$

where δ^π_t is the *expected* TD-error. The *forward view* of the update is then written:

$$q_{k+1}(s,a) \leftarrow q_k(s,a) + \alpha_k\sum_{t=0}^{T_k}(\gamma\lambda)^t\delta^{\pi_k}_t, \tag{3.5}$$

where T_k is the random variable corresponding to the length of the kth trajectory. While (3.5) resembles many existing TD(λ) algorithms, it subtly differs from all of them, due to $\mathcal{R}^{\pi,\mu}_\lambda$ (rather than $\mathcal{T}^\pi_\lambda$) being at its basis. Section 3.3 discusses the distinctions in detail. The every visit form of Eq. (3.5) is written:

$$q_{k+1}(s,a) \leftarrow q_k(s,a) + \alpha_k\sum_{t=0}^{T_k}\delta^{\pi_k}_t\sum_{i=0}^{t}(\gamma\lambda)^{t-i}\mathbb{I}\{(S_i,A_i)=(s,a)\}, \tag{3.6}$$

and the corresponding *backward view*, or the online form, of all three algorithms is summarized in Algorithm 1. The following theorem states that when μ and π are sufficiently close, the off-policy Q$^\pi(\lambda)$ algorithm converges to its fixed point q^π.

Theorem 3.1

Consider the sequence of Q-functions computed according to Algorithm 1 with fixed policies μ and π. Let $\epsilon = \max_s \|\pi(\cdot|s) - \mu(\cdot|s)\|_1$. If $\lambda\epsilon < \frac{1-\gamma}{\gamma}$, then under the same conditions required for the convergence of TD(λ) (Assumptions 2.2, 2.3, 3.1) we have, almost surely:

$$\lim_{k\to\infty} q_k(s,a) = q^\pi(s,a).$$

We state a similar, albeit weaker result for Q$^*(\lambda)$.

[3] The derivation can be found in Appendix A.9

Theorem 3.2

Consider the sequence of Q-functions computed according to Algorithm 1 with π_k the greedy policy with respect to q_k. If $\lambda < \frac{1-\gamma}{2\gamma}$, then under the same conditions required for the convergence of TD(λ) (Assumptions 2.2, 2.3, 3.1) we have, almost surely:

$$\lim_{k\to\infty} q_k(s,a) = q^*(s,a).$$

The proofs of these theorems rely on showing that $\mathcal{R}_\lambda^\pi$ and $\mathcal{R}_\lambda^*$ are contractions under the stated conditions, and invoking classical stochastic approximation convergence to their fixed point (such as Proposition 4.5 from [Bertsekas and Tsitsiklis 1996]). We will focus on the contraction lemmas, which are the crux of the proofs, then outline the sketch of the online convergence argument.

Discussion

Theorem 3.1 states that for *any* $\lambda \in [0,1]$ there exists some degree of "off-policy-ness" $\epsilon < \frac{1-\gamma}{\lambda\gamma}$ under which q_k converges to q^π. This is the $\lambda-\epsilon$ tradeoff for the off-policy Q$^\pi(\lambda)$ learning algorithm for policy evaluation. In the control case, the result of Theorem 3.2 is weaker as it only holds for values of λ smaller than $\frac{1-\gamma}{2\gamma}$. Notice that this threshold corresponds to the policy evaluation case for $\epsilon = 2$ (arbitrary off-policy-ness). We were not able to prove convergence to q^* for any $\lambda \in [0,1]$ and some $\epsilon > 0$.

The main technical difficulty lies in the fact that in control, the greedy policy with respect to the current q_k may change drastically from one step to the next, while q_k itself changes incrementally (under small learning steps α_k). So the current q_k may not offer a good off-policy correction to evaluate the new greedy policy. In order to circumvent this problem we may want to use slowly changing target policies π_k. For example we could keep π_k fixed for slowly increasing periods of time. This can be seen as a form of optimistic policy iteration [Puterman 1994] where policy improvement steps alternate with approximate policy evaluation steps (and when the policy is fixed, Theorem 3.1 guarantees convergence to the value function of that policy).

The algorithm introduced in Section 3.4.2 overcomes these difficulties and enjoys general convergence guarantees by incorporating an approximate importance sampling ratio in the update.

3.2.3 Convergence Analysis

We begin by verifying that the fixed points of $\mathcal{R}_\lambda^\pi$ and$\mathcal{R}_\lambda^*$, that is: the instances of $\mathcal{R}_\lambda^{\pi,\mu}$ in the policy evaluation and control settings, are q^π and q^*, respectively. We then prove the contractive properties of these operators: $\mathcal{R}_\lambda^\pi$ is always a contraction and will converge to its fixed point, $\mathcal{R}_\lambda^*$ is a contraction for particular choices of λ (given in terms of γ). The contraction coefficients depend on λ, γ, and ϵ: the distance between policies. Finally, we give a proof sketch for online convergence of Algorithm 1.

Before we begin, it will be convenient to rewrite Eq. (3.1) for all state-action pairs:

$$\mathcal{R}^{\pi,\mu}q = r + \sum_{t=1}^{\infty} \gamma^t (\mathcal{P}^\mu)^{t-1}[\mathcal{P}^\mu r + \mathcal{P}^\pi q - \mathcal{P}^\mu q].$$

We can then express $\mathcal{R}_\lambda^\pi$ and $\mathcal{R}_\lambda^*$ from Eq. (3.2) as follows:

$$\mathcal{R}_\lambda^\pi q \stackrel{\text{def}}{=} q + (I - \lambda\gamma\mathcal{P}^\mu)^{-1}[\mathcal{T}^\pi q - q], \tag{3.7}$$

$$\mathcal{R}_\lambda^* q \stackrel{\text{def}}{=} q + (I - \lambda\gamma\mathcal{P}^\mu)^{-1}[\mathcal{T} q - q]. \tag{3.8}$$

It is not surprising that the above along with the Bellman equations (2.12) and (2.14) directly yields that q^π and q^* are the fixed points of $\mathcal{R}_\lambda^\pi$ and $\mathcal{R}_\lambda^*$:

$$\mathcal{R}_\lambda^\pi q^\pi = q^\pi, \qquad \mathcal{R}_\lambda^* q^* = q^*.$$

It then remains to analyze the behavior of $\mathcal{R}_\lambda^{\pi,\mu}$ as it gets iterated.

λ-return for policy evaluation: $\mathbf{Q}^\pi(\lambda)$

We first consider the case with a fixed arbitrary policy π. For simplicity, we take μ to be fixed as well, but the same will hold for any sequence $(\mu_k)_{k\in\mathbb{N}}$, as long as each μ_k is no more off-policy than μ.

Lemma 3.1

Consider the policy evaluation algorithm $q_k = (\mathcal{R}_\lambda^\pi)^k q_0$. Assume the behavior policy μ is ϵ-away from the target policy π, in the sense that $\max_s \|\pi(\cdot|s) - \mu(\cdot|s)\|_1 \leq \epsilon$. Then for $\epsilon < \frac{1-\gamma}{\lambda\gamma}$, the sequence $(q_k)_{k\in\mathbb{N}}$ converges to q^π exponentially fast: $\|q_k - q^\pi\| = O(\eta^k)$, where $\eta = \frac{\gamma}{1-\lambda\gamma}(1 - \lambda + \lambda\epsilon) < 1$.

λ-return for control: $\mathbf{Q}^*(\lambda)$

We next consider the case where the kth target policy π_k is greedy with respect to the value estimate q_k. The following lemma states that is possible to select a small, but nonzero λ and still guarantee convergence.

Lemma 3.2

Consider the off-policy control algorithm $q_k = (\mathcal{R}^*_\lambda)^k q_0$. Then

$$\|\mathcal{R}^*_\lambda q_k - q^*\| \leq \frac{\gamma + \lambda\gamma}{1 - \lambda\gamma}\|q_k - q^*\|,$$

and for $\lambda < \frac{1-\gamma}{2\gamma}$ the sequence $(q_k)_{k\in\mathbb{N}}$ converges to q^* exponentially fast.

Online Convergence

The online convergence of all three algorithm encompassed in Algorithm 1 is proven equivalently, given the conditions of Lemmas 3.2 and 3.1 hold, along with some assumptions required for the online setting.

Assumption 3.1: Finite trajectories

For every sample trajectory τ_k: $\mathbb{E}_{\mu_k} T_k^2 < \infty$, where T_k is the length of τ_k.

Assumption 3.1 requires trajectories to be finite w.p. 1, which is satisfied by e.g. *proper* behavior policies. Equivalently, we may require from the MDP that all trajectories eventually reach a zero-value absorbing state We show that Algorithm 1 converges under Assumption 3.1, together with the standard assumptions 2.2 and 2.3. The proof (in Appendix A.2) closely follows that of Proposition 5.2 from [Bertsekas and Tsitsiklis 1996], and requires rewriting the update in the suitable form, and verifying Assumptions (a) through (d) from their Proposition 4.5.

3.2.4 Experiments

Although we do not have a proof of the $\lambda - \epsilon$ tradeoff (see the discussion in Section 3.2.2) in the control case, we wished to investigate whether such a tradeoff can be observed experimentally. To this end, we applied $\mathrm{Q}^*(\lambda)$ to the Bicycle domain [Randløv and Alstrøm

1998]. Our main interest is in the interplay between the λ parameter in $Q^*(\lambda)$ and the ϵ parameter from an ϵ-greedy exploration policy. We report three findings:

1. Higher values of λ lead to improved learning;
2. Very low values of ϵ exhibit lower performance; and
3. The Q-function diverges when λ is high relative to ϵ.

Together, these findings suggest that there is indeed a $\lambda - \epsilon$ tradeoff in the control case as well, and suggest that with proper care it can be beneficial to do off-policy control with $Q^*(\lambda)$, thus confirming the intuition that such naive $Q(\lambda)$ is "not as naive as one might at first suppose" [Sutton and Barto 1998].

Domain description and Experimental Details

In the Bicycle domain, the agent must simultaneously balance a simulated bicycle and drive it to a goal position. Six real-valued variables describe the state – angle, velocity, etc. – of the bicycle. The reward function is proportional to the angle to the goal, and gives -1 for falling and $+1$ for reaching the goal. The discount factor is 0.99. The Q-function was approximated using multilinear interpolation over a uniform grid of size $10 \times \cdots \times 10$, and the stepsize was tuned to 0.1 from a parameter sweep. Each configuration is reported as an average of five independent trials. Our main performance indicator is the frequency at which the goal is reached by the greedy policy after 500,000 episodes of training.

Results

Learning speed and performance Figure 3.1 (left) depicts the performance of $Q^*(\lambda)$, in terms of the goal-reaching frequency, for three values of ϵ. The agent performs best ($p < 0.05$) for $\epsilon \in [0.003, 0.03]$ and high (w.r.t. ϵ) values of λ.

Divergence For each value of ϵ, we determined the highest *safe* choice of λ which did not result in divergence. As Figure 3.1 (right) illustrates, there is a marked decrease in what a safe value of λ is as ϵ increases. Note the left-hand shaded region corresponding to the *policy evaluation* bound $\frac{1-\gamma}{\gamma\epsilon}$. Supporting our hypothesis on the true bound on λ (Section 3.2.3), it is clear that the maximum safe value of λ depends on ϵ. In particular, notice how $\lambda = 1$ stops diverging exactly where predicted by this bound.

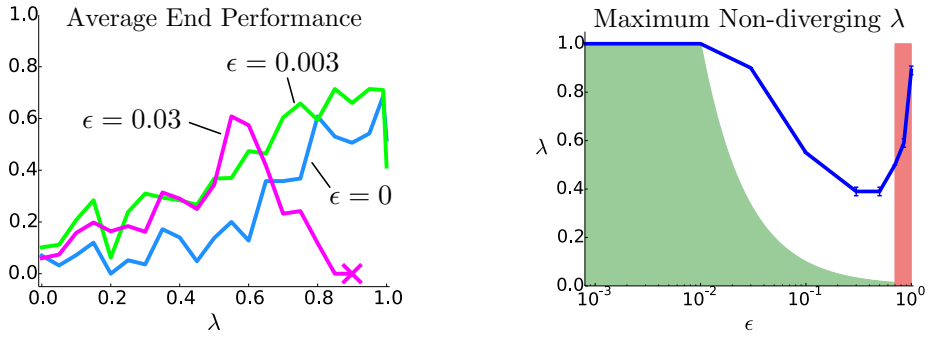

Figure 3.1: **Left.** Q*(λ) on the Bicycle domain. The 'X' marks the lowest value of λ for which $\epsilon = 0.03$ causes divergence. **Right.** The solid land indicates the maximum non-diverging value of λ> The left-hand shaded region corresponds to our hypothesized bound. Parameter settings in the right-hand shaded region do not produce meaningful policies.

3.2.5 Discussion

In control, determining the existence of a non-trivial ϵ-dependent bound for λ remains an open problem. Supported by telling empirical results in the Bicycle domain, we hypothesize that such a bound exists, and closely resembles the $\frac{1-\gamma}{\gamma\epsilon}$ bound from the policy evaluation case.

3.3 Survey of Multi-Step TD Algorithms

In this section, we take a step back and place the presented algorithms in context of the existing work in TD(λ), focusing in particular on action-value methods. Casting them in this common framework will in particular set the stage for our next contribution, the unified multi-step off-policy operator, and Retrace(λ), the improved algorithm.

As usual, let $S_0, A_0, R_1, \ldots$ be a trajectory generated by following a behavior policy μ, i.e. $A_t \sim \mu(\cdot|S_t)$. At time i, SARSA(λ) [Rummery and Niranjan 1994] updates its Q-function as follows:

$$q(S_i, A_i) \leftarrow q(S_i, A_i) + \alpha_i \underbrace{((1-\lambda)\sum_{n=0}^{\infty} \lambda^n G_i^{(n+1)} - q(S_i, A_i))}_{\Delta_i}, \tag{3.9}$$

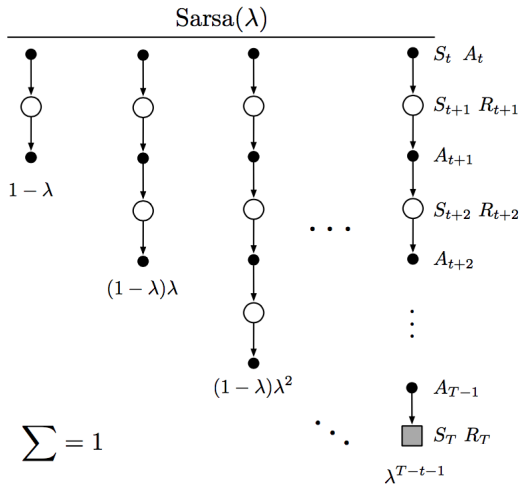

Figure 3.2: Backup diagram for Sarsa(λ) (from [Sutton and Barto 2017]).

$$G_i^{(n)} = \sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} q_k(S_{i+n+1}, A_{i+n+1}), \tag{3.10}$$

where Δ_i denotes the update made at time i, $G_i^{(n)}$ is the n-step return from time i onwards. Δ_i can be rewritten in terms of one-step TD-errors as follows:

$$\Delta_i = \sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t, \tag{3.11}$$
$$\delta_t = R_{t+1} + \gamma q(S_{t+1}, A_{t+1}) - q(S_t, A_t).$$

SARSA(λ) is an on-policy algorithm and converges to the value function q^μ of the behavior policy. Below we will discuss the different algorithms that arise by instantiating $G_i^{(n)}$ or Δ_i from Eq. (3.9) differently. Tables 3.1 and 3.2 provide the full details, while in text we will specify the most revealing components of the updates.

Before we proceed, let us introduce the concept of a *backup* as an instrumental image of thinking about multi-step algorithms. The update in Eq. (3.9) is an example of a backup equation, since we propagate the information from the states and actions $S_1, A_1, S_2, A_2 \ldots$ *back* to S_0, A_0. The backup *diagram* for SARSA(λ) is given in Fig. 3.2.

3.3.1 Policy Evaluation

One can imagine considering *expectations* over action-values at the corresponding states $\mathbb{E}_\pi q(S_t, \cdot) \equiv \sum_a \pi(a|S_t) q(S_t, a)$, in place of the value of the sampled action $q(S_t, A_t)$, i.e.:

$$\delta_t = R_{t+1} + \gamma \mathbb{E}_\pi q(S_{t+1}, \cdot) - \mathbb{E}_\pi q(S_t, \cdot). \tag{3.12}$$

This is the one-step update for *General Q-Learning* [van Hasselt 2011], which is a generalization of *Expected SARSA* [van Seijen et al. 2009] to arbitrary policies. We refer to the direct eligibility trace extensions of these algorithms formed via Equations (3.9)-(3.11) by General Q(λ) and Expected SARSA(λ) (first mentioned by [Sutton et al. 2014]) Unfortunately, in an off-policy setting, General Q(λ) will not converge to the value function q^π of the target policy, as stated by the following proposition.

Proposition 3.1

The stable point of General Q(λ) is $q^{\mu,\pi} = (I - \lambda\gamma(\mathcal{P}^\mu - \mathcal{P}^\pi) - \gamma\mathcal{P}^\pi)^{-1} r$ which is the fixed point of the operator $(1-\lambda)\mathcal{T}^\pi + \lambda\mathcal{T}^\mu$.

Alternatively to replacing both terms with an expectation, one may only replace the value at the *next* state S_{t+1} by $\mathbb{E}_\pi q(S_{t+1}, \cdot)$, obtaining:

$$\delta_t^\pi = R_{t+1} + \gamma \mathbb{E}_\pi q(S_{t+1}, \cdot) - q(S_t, A_t). \tag{3.13}$$

This is exactly our policy evaluation algorithm Q$^\pi(\lambda)$. In particular, when $\pi = \mu$, we get the on-policy Q$^\pi(\lambda)$. The induced *on-policy* correction may serve as a variance reduction term for Expected SARSA(λ) (it may be helpful to refer to the n-step return in Table 3.1 to observe this), but we leave variance analysis of this algorithm for future work. When $\pi \neq \mu$, we recover off-policy Q$^\pi(\lambda)$, which (under the stated conditions) converges to q^π.

Target Policy Probability Methods:

The algorithms above directly descend from basic SARSA(λ), but often learning off-policy requires special treatment. For example, a typical off-policy technique is importance sampling (IS) [Precup et al. 2001]. It is a classical Monte Carlo method that allows one to sample from the available distribution, but obtain (unbiased or consistent) samples of the desired one, by reweighing the samples with their likelihood ratio according to the two distributions. For a behavior policy μ and a target policy π, this yields at time t:

$$\rho_t \stackrel{\text{def}}{=} \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}. \tag{3.14}$$

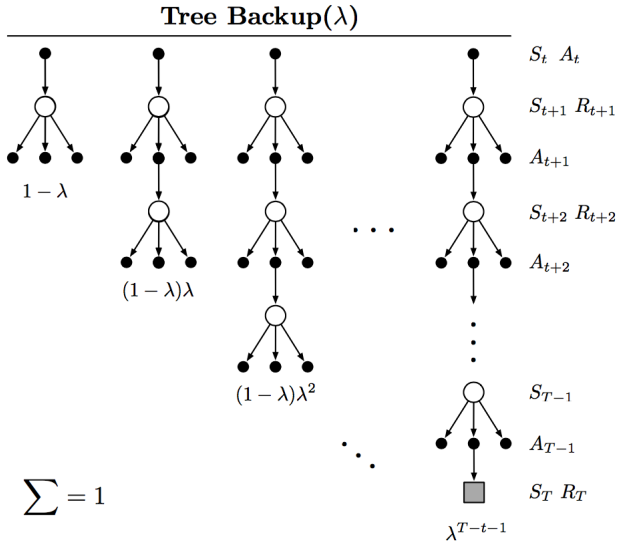

Figure 3.3: Backup diagram for Tree-Backup(λ) (from [Sutton and Barto 2017]).

In order for ρ to be well-defined, it is generally assumed that $\pi(a|s) \implies \mu(a|s)$, or, in order to avoid the coupling of the policies, simply that $\mu(a|s) > 0, \forall s, \in \mathcal{S}, a \in \mathcal{A}$. The updates for the ordinary *per-decision* IS algorithm for policy evaluation are then made as follows:

$$\Delta_i = \sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t \prod_{j=i+1}^{t} \rho_j,$$
$$\delta_t = R_{t+1} + \gamma\rho_{t+1} q(S_{t+1}, A_{t+1}) - q(S_t, A_t).$$

This family of algorithms converges to q^π with probability 1, under any soft, stationary behavior μ [Precup et al. 2000].

However, off-policy Q$^\pi(\lambda)$ is perhaps related closest to the *Tree-Backup (TB) algorithm*, also discussed by [Precup et al. 2000]. Its one-step TD-error is the same as (3.13), the algorithms back up the same tree (Fig. 3.3), and neither requires knowledge of the behavior policy μ. The important difference is in the weighting of the updates. As an off-policy precaution, TB(λ) weighs updates along a trajectory with the cumulative target

probability of that trajectory up until that point:

$$\Delta_i = \sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t^{\pi} \prod_{j=i+1}^{t} \pi(A_j|S_j). \tag{3.15}$$

The weighting simplifies the convergence argument, allowing TB(λ) to converge to q^{π} without further restrictions on the distance between μ and π [Precup et al. 2000]. The drawback of TB(λ) is that in the case of near on-policy-ness (when μ is close to π) the product of the probabilities cuts the traces unnecessarily (especially when the policies are stochastic). What we have shown in this chapter, is that plain TD-learning *can* converge off-policy without policy probability corrections, subject to a tradeoff condition on λ and ϵ. Under that condition, Q$^{\pi}$(λ) applies both on- and off-policy, without modifications. An ideal algorithm should be able to automatically cut the traces (like TB(λ)) in case of extreme off-policy-ness while reverting to Q$^{\pi}$(λ) when being near on-policy.

3.3.2 Control

Perhaps the most popular version of Q(λ) is due to [Watkins and Dayan 1992]. Off-policy, it truncates the return and bootstraps as soon as the behavior policy takes a non-greedy action, as described by the following update:

$$\Delta_i = \sum_{t=i}^{i+k} (\lambda\gamma)^{t-i} \delta_t, \tag{3.16}$$

where $k = \arg\min_{u \geq 1} A_{i+u} \notin \mathcal{G}_{\mathcal{A}}(Q_{i+u})$ is the first time when the behavior policy takes a non-greedy action, and where $\mathcal{G}_{\mathcal{A}}(Q_i)$ denotes the set of greedy actions w.r.t. q at state S_i. Note that this update is a special case of (3.15) for deterministic greedy policies, with $\prod_{j=i+1}^{t} A_j \in \mathcal{G}_{\mathcal{A}}(Q_j)$ replacing the probability product. When the policies μ and π are not too similar, and λ is not too small, the truncation may greatly reduce the benefit of complex backups.

Q(λ) of [Peng and Williams 1996] is meant to remedy this, by being a hybrid between SARSA(λ) and Watkins's Q(λ). Its n-step return $\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} \max q(S_{i+n+1}, \cdot)$ admits the following form of the TD-error:

$$\delta_t = R_{t+1} + \gamma \max q(S_{t+1}, \cdot) - \max q(S_t, \cdot).$$

This is, in fact, the same update rule as the General Q(λ) defined in Eq. (3.12), where π is the greedy policy. Following the same steps as in the proof of Proposition 3.1, the

Table 3.1: Comparison of the update rules of several **policy evaluation** algorithms using the λ-return: SARSA(λ), Expected SARSA(λ), General Q(λ), Per-Decision Importance Sampling (PDIS) (λ), Tree-Backup (TB) (λ), and Q$^\pi$(λ): in both on-policy (i.e. $\pi = \mu$) and off-policy settings ($\pi \neq \mu$). Note the same Q$^\pi$(λ) equation applies in both settings. We write $Q_t = q(S_t, A_t)$, $\mathbb{E}_\pi Q_t = \mathbb{E}_\pi q(S_t, \cdot)$, $\mathbb{E}_\pi^{a \neq b} Q_t = \sum_{a \in \mathcal{A} \setminus b} \pi(a|s) q(S_t, a)$. The **FP** column denotes the stable point of each algorithm (i.e. the fixed point of the expected update), given in red if there no proof of convergence to this fixed point exists in literature.

Algorithm	n**-step return**	**Update rule for the λ-return**	**FP**
TD(λ) (on-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} V_{i+n+1}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t$ $\delta_t = R_{t+1} + \gamma V_{t+1} - V_t$	v^μ
SARSA(λ) (on-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} Q_{i+n+1}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t$ $\delta_t = R_{t+1} + \gamma Q_{t+1} - Q_t$	q^μ
$\mathbb{E}$ SARSA(λ) (on-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} \mathbb{E}_\mu Q_{i+n+1}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t + \mathbb{E}_\mu Q_i - Q_i$ $\delta_t = R_{t+1} + \gamma \mathbb{E}_\mu Q_{t+1} - \mathbb{E}_\mu Q_t$	q^μ
General Q(λ) (off-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} \mathbb{E}_\pi Q_{i+n+1}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t + \mathbb{E}_\pi Q_i - Q_i$ $\delta_t = R_{t+1} + \gamma \mathbb{E}_\pi Q_{t+1} - \mathbb{E}_\pi Q_t$	$q^{\mu,\pi}$
PDIS(λ) (off-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} \prod_{j=i+1}^{t} \rho_j + \gamma^{n+1} Q_{i+n+1} \prod_{j=i+1}^{i+n+1} \rho_j$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t \prod_{j=i+1}^{t} \rho_j$ $\delta_t = R_{t+1} + \gamma \rho_{t+1} Q_{t+1} - Q_t$	q^π
TB(λ) (off-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} \prod_{j=i+1}^{t} \pi_j \left[R_{t+1} + \gamma \mathbb{E}_\pi^{a \neq A_{t+1}} Q_{t+1}\right] + \gamma^{n+1} \prod_{j=i+1}^{i+n+1} \pi_j Q_{i+n+1}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t \prod_{j=i+1}^{t} \pi_j$ $\delta_t = R_{t+1} + \gamma \mathbb{E}_\pi Q_{t+1} - Q_t$	q^π
$\mathbf{Q}^\pi(\lambda)$ (on-policy /off-policy)	$\sum_{t=i}^{i+n} \gamma^{t-i} \left[R_{t+1} + \mathbb{E}_\pi Q_t - Q_t\right] + \gamma^{n+1} \mathbb{E}_\pi Q_{i+n+1}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t$ $\delta_t = R_{t+1} + \gamma \mathbb{E}_\pi Q_{t+1} - Q_t$	q^π

Table 3.2: Comparison of the update rules of several **control** algorithms using the λ-return: Watkins's Q(λ), Peng and Williams's Q(λ), and Q*(λ). We write $Q_t = q(S_t, A_t)$, $Q_t^{\max} = \max q(S_t, \cdot)$, and $\mathcal{G}_{\mathcal{A}}(Q_t)$ denotes the set of greedy actions w.r.t. q at S_t. The **FP** column denotes the stable point of these algorithms (i.e. the fixed point of the expected update), given in red if there no proof of convergence to this fixed point exists in literature.

Algorithm	n**-step return**	**Update rule with** λ**-returns**	**FP**
Q(λ) (Watkins's)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} Q_{i+n+1}^{\max}$ $n < k = \arg\min_{u \geq 1} A_u \notin \mathcal{G}_{\mathcal{A}}(Q_u)$	$\sum_{t=i}^{i+k} (\lambda\gamma)^{t-i} \delta_t$ $\delta_t = R_{t+1} + \gamma Q_{t+1}^{\max} - Q_t$	q^*
Q(λ) (P & W's)	$\sum_{t=i}^{i+n} \gamma^{t-i} R_{t+1} + \gamma^{n+1} Q_{i+n+1}^{\max}$	$\sum_{t=i}^{i+n} (\lambda\gamma)^{t-i} \delta_t + Q_i^{\max} - Q_i$ $\delta_t = R_{t+1} + \gamma Q_{t+1}^{\max} - Q_t^{\max}$	$q^{\mu,*}$
Q*(λ)	$\sum_{t=i}^{i+n} \gamma^{t-i} \left[R_{t+1} + Q_t^{\max} - Q_t \right]$ $+\gamma^{n+1} Q_{i+n+1}^{\max}$	$\sum_{t=i}^{\infty} (\lambda\gamma)^{t-i} \delta_t$ $\delta_t = R_{t+1} + \gamma Q_{t+1}^{\max} - Q_t$	q^*

limit of this algorithm (if it does indeed converge) will be the fixed point of the operator $(1-\lambda)\mathcal{T} + \lambda\mathcal{T}^{\mu}$ which is different from q^* unless the behavior is always greedy.

[Sutton and Barto 1998] mention another, *naive* version of Watkins's Q(λ) that does not cut the trace on non-greedy actions. That is exactly the Q*(λ) algorithm described earlier in this chapter. Notice that despite the similarity to Watkins's Q(λ), the equivalence representation for Q*(λ) is different from the one that would be derived by setting $k = \infty$ in Eq. (3.16), since the n-step return uses the *corrected* immediate reward $R_{t+1} + \gamma \max q(S_t, \cdot) - q(S_t, A_t)$ instead of the immediate reward alone. This correction is invisible in Watkins's Q(λ), since the behavior policy is assumed to be greedy, before the return is cut off.

3.4 Safe and Efficient Off-Policy RL

Of all the algorithms reviewed in the previous section, only Q^π and Tree-Backup don't use importance sampling ratios. Unfortunately, the assumption necessary for convergence of Q^π, that μ and π are close, is restrictive, as well as difficult to uphold in the control case, where the target policy is always greedy with respect to the current Q-function. In that sense the algorithm is not *safe*: it does not handle the case of arbitrary "off-policyness". On the other hand, TB(λ) tolerates arbitrary target/behavior discrepancies by scaling information (here called *traces*) from future temporal differences by the product of target policy probabilities. TB(λ) is not *efficient* in the "near on-policy" case (similar μ and π), though, as traces may be cut prematurely, preventing learning from full returns.

We wish to combine the strengths of these two algorithms. To this end, in the rest of the chapter, we express several of the off-policy, return-based algorithms from the previous section in a common form, and analyze the convergence properties of this general form. We then motivate and derive an improved algorithm, Retrace(λ), which is both *safe* and *efficient*, enjoying convergence guarantees for off-policy policy evaluation and – more importantly – for the control setting. As a corollary to this analysis, the first proof of convergence Watkins' Q(λ) follows.

3.4.1 Unified View

The general form that we consider for comparing several return-based off-policy algorithms is:

$$\mathcal{U}q(s,a) \stackrel{\text{def}}{=} q(s,a) + \mathbb{E}_\mu \left[\sum_{t \geq 0} \gamma^t \Big(\prod_{i=1}^{t} c_i \Big) \big(R_{t+1} + \gamma \mathbb{E}_\pi q(S_{t+1}, \cdot) - q(S_t, A_t) \big) \right], \quad (3.17)$$

for some non-negative coefficients (c_i), where we write $\left(\prod_{i=1}^{t} c_i\right) = 1$ when $t = 0$. By extension of the idea of eligibility traces we informally call the coefficients (c_i) the *traces* of the operator. We review three key algorithms that can be instantiated through this form.

Importance sampling (IS): $c_i = \rho_i = \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}$. Importance sampling is the simplest way to correct for the discrepancy between μ and π when learning from off-policy returns [Precup et al. 2000, Precup et al. 2001, Geist and Scherrer 2014]. The off-policy correction uses the product of the likelihood ratios between π and μ. Notice that the $\mathcal{U}Q$ operator (3.17) defined with this choice of (c_i) yields q^π for any q. For $q = 0$ we recover the basic IS estimate $\sum_{t=0}^{\infty} \gamma^t \left(\prod_{i=1}^{t} c_i\right) R_{t+1}$. Thus, Eq. (3.17) can be seen as a variance

reduction technique (with a baseline q). It is well-known that IS estimates can suffer from large – possibly infinite – variance (mainly due to the variance of the probability ratio product). This in turn has motivated methods that reduce the variance at the cost of adding bias [Mahmood and Sutton 2015, Sutton et al. 2016, Hallak et al. 2016].

Off-policy $Q^\pi(\lambda)$ and $Q^*(\lambda)$: $c_i = \lambda$. In the first part of this chapter, we have introduced an off-policy correction based on a the Q-function (instead of correcting the probability of the sample path like in IS). This approach corresponds to the choice $c_i = \lambda$, and offers the advantage of avoiding the blow-up of the variance of the product of ratios encountered with IS. Interestingly, we have shown that this operator contracts around q^π provided that μ and π are sufficiently close to each other. Unfortunately, $Q^\pi(\lambda)$ requires knowledge of ϵ, and the condition for $Q^*(\lambda)$ is very conservative. Neither $Q^\pi(\lambda)$, nor $Q^*(\lambda)$ are safe as they do not guarantee convergence for arbitrary π and μ.

Tree-backup (TB) (λ): $c_i = \lambda\pi(A_i|S_i)$. The TB($\lambda$) algorithm of [Precup et al. 2000] corrects for the target/behavior discrepancy by multiplying each term of the sum by the product of target policy probabilities. The corresponding operator defines a contraction mapping (not only in expectation but also for any sample trajectory) for any policies π and μ, which makes it a *safe* algorithm. However, it is not efficient, since in the near on-policy case (where μ and π are similar) the algorithm unnecessarily cuts the traces and does not utilize full returns. As shown by our results on $Q^\pi(\lambda)$, we need not discount stochastic on-policy transitions for convergence.

3.4.2 Retrace(λ)

Our contribution is an algorithm – Retrace(λ) – that takes the best of the three algorithms mentioned above. Retrace(λ) uses a *truncated* importance sampling ratio together with the value corrections. It instantiates c_i as:

$$c_i = \lambda \min\left(1, \rho_i\right) = \lambda \min\left(1, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}\right). \tag{3.18}$$

The truncation allows it to avoid the variance explosion of the product of importance sampling ratios. At the same time, unlike the restricted convergence of $Q^\pi(\lambda)$, even a truncated importance ratio is sufficient to guarantee general convergence for any λ and ϵ. Finally, the traces maintained by Retrace(λ) are always larger than those of TB(λ), since

$$\min\left(1, \rho_i\right) = \min\left(1, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}\right) \geq \pi(A_i|S_i).$$

	Definition of c_i	Estimation variance	Guaranteed convergence of $\mathcal{U}$	Use full returns (near on-policy)
Importance sampling	$\frac{\pi(A_i\|S_i)}{\mu(A_i\|S_i)}$	High	for any π, μ	yes
Q(λ)	λ	Low	only for π close to μ	yes
TB(λ)	$\lambda\pi(A_i\|S_i)$	Low	for any π, μ	no
Retrace(λ)	$\lambda \min\left(1, \frac{\pi(A_i\|S_i)}{\mu(A_i\|S_i)}\right)$	Low	for any π, μ	yes

Table 3.3: Properties of several algorithms in terms of the general operator $\mathcal{U}$ from Eq. (3.17).

In particular, when the update is on-policy, $\mu_i = \pi_i$, Retrace(λ) is able to learn from the full returns. Table (3.3) summarizes the instantiations and properties of the relevant algorithms.

In the subsequent sections, we will show the following:

- The operator underlying Retrace(λ) is a γ-contraction around q^π, for *arbitrary* policies μ and π,
- Taking c_i to be no greater than the ratio π/μ is sufficient to guarantee this property,
- Under mild assumptions, the control version of Retrace(λ), where π is replaced by a sequence of increasingly greedy policies, is also a contraction, and
- The online Retrace(λ) algorithm converges a.s. to q^* in the control case, without requiring the GLIE (greedy in the limit of infinite exploration) assumption.
- As a corollary, we prove the convergence of Watkins's Q(λ) to q^* for the first time.

3.4.3 Convergence Theorems

In this section we will in turn consider both off-policy policy evaluation and control settings. The key result is that $\mathcal{U}$ is a contraction mapping in both settings (under a mild additional assumption for the control case).

Policy Evaluation

Consider a fixed target policy π. Let the behavior policy μ be fixed as well, although our results easily extend to sequences of behavior policies $(\mu_k)_{k\in\mathbb{N}}$. Our first result states the γ-contraction of the operator (3.17) defined by any set of non-negative coefficients $c_i = c_i(A_i, \mathcal{F}_i)$ (in order to emphasize that c_i can be a function of the whole history $\mathcal{F}_i$) under the assumption that $c_i \leq \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}$.

Theorem 3.3

The operator $\mathcal{U}$ defined by Eq. (3.17) has a unique fixed point q^π. Furthermore, if for each $A_i \in \mathcal{A}$ and each history $\mathcal{F}_i$ we have $c_i = c_i(A_i, \mathcal{F}_i) \in \left[0, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}\right]$, then for any Q-function q:

$$|\mathcal{U}q(s,a) - q^\pi(s,a)| \leq \eta(s,a)\|q - q^\pi\|,$$

where

$$\eta(s,a) \stackrel{\text{def}}{=} 1 - (1-\gamma)\mathbb{E}_\mu\left[\sum_{t\geq 0}\gamma^t\left(\prod_{i=1}^{t} c_i\right)\right].$$

Thus, $\eta(s,a) \in [0,\gamma]$ is a (s,a)-specific contraction coefficient, which is γ when $c_1 = 0$ (the trace is cut immediately) and can be close to zero when learning from full returns ($c_t \approx 1$ for all t).

Control

In the control setting, the single target policy π is replaced by a sequence of policies which depend on q_k. While most prior work has focused on strictly greedy policies, here we consider the larger class of *increasingly greedy* sequences. We now make this notion precise.

Definition 3.1

We say that a sequence of policies $(\pi_k)_{k\in\mathbb{N}}$ is *increasingly greedy* w.r.t. a sequence $(q_k)_{k\in\mathbb{N}}$ of Q-functions if the following property holds for all k:

$$\mathcal{P}^{\pi_{k+1}} q_{k+1} \geq \mathcal{P}^{\pi_k} q_{k+1}.$$

Intuitively, this means that each π_{k+1} is at least as greedy as the previous policy π_k for q_{k+1}. Many natural sequences of policies are increasingly greedy, including ϵ_k-greedy policies (with non-increasing ϵ_k) and softmax policies (with non-increasing temperature).

We will assume that $c_i = c_i(A_i, \mathcal{F}_i) = c(A_i, S_i)$ is Markovian, in the sense that it depends on S_i, A_i (as well as the policies π and μ) only, but not on the full past history. This allows us to define the (sub)-probability transition operator

$$\mathcal{P}^{c\mu} q(s,a) \stackrel{\text{def}}{=} \sum_{s'}\sum_{a'} p(s'|s,a)\mu(a'|s')c(a',s')q(s',a').$$

Finally, an additional requirement to the convergence in the control case, we assume that q_0 satisfies $\mathcal{T}^{\pi_0} q_0 \geq q_0$, which can be achieved by a pessimistic initialization $q_0 = -r_{\max}/(1-\gamma)$.

Theorem 3.4

Consider an arbitrary sequence of behavior policies $(\mu_k)_{k\in\mathbb{N}}$ (which may depend on (q_k)) and a sequence of target policies $(\pi_k)_{k\in\mathbb{N}}$ that are increasingly greedy w.r.t. the sequence $(q_k)_{k\in\mathbb{N}}$, and consider the update:

$$q_{k+1} = \mathcal{U}_k q_k,$$

where the return operator $\mathcal{U}_k$ is defined by Eq. (3.17) for π_k and μ_k and a Markovian $c_i = c(A_i, S_i) \in [0, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}]$. Assume the target policies π_k are ϵ_k-away from the greedy policies w.r.t. q_k, in the sense that $\mathcal{T}^{\pi_k} q_k \geq \mathcal{T} q_k - \epsilon_k \|q_k\| e$, where e is the vector with 1-components. Further suppose that $\mathcal{T}^{\pi_0} q_0 \geq q_0$. Then for any $k \geq 0$,

$$\|q_{k+1} - q^*\| \leq \gamma \|q_k - q^*\| + \epsilon_k \|q_k\|.$$

In consequence, if $\epsilon_k \to 0$ then $q_k \to q^*$.

Online algorithms

So far we have analyzed the contraction properties of the expected $\mathcal{U}$ operators. We now describe the online Retrace(λ) algorithm which can learn from sample trajectories. The every-visit, backward view of the algorithm is given in Algorithm 2.

In this section, we will only consider the Retrace(λ) algorithm defined with the coefficient $c = \lambda \min(1, \pi/\mu)$. For that c, let us rewrite the operator $\mathcal{P}^{c\mu}$ as $\lambda \mathcal{P}^{\pi \wedge \mu}$, where

$$\mathcal{P}^{\pi \wedge \mu} q(x,a) \stackrel{\text{def}}{=} \sum_y \sum_b \min(\pi(b|y), \mu(b|y)) q(y,b),$$

and write the Retrace operator

$$\mathcal{U} q = q + (I - \lambda\gamma \mathcal{P}^{\pi \wedge \mu})^{-1} (\mathcal{T}^{\pi} q - q).$$

We focus on the control case, noting that a similar (and more general) result can be derived for policy evaluation. We will use the notation π_q to refer to the policy greedy w.r.t. q.

Algorithm 2 Retrace(λ) algorithm.

Given: Initial Q-function q_0, step-sizes $(\alpha_k)_{k\in\mathbb{N}}$
  for $k = 1, \dots$ **do**
    Sample a trajectory $S_0, A_0, R_0, \dots, S_{T_k}$ from μ_k
    $q_{k+1}(s,a) \leftarrow q_k(s,a) \qquad \forall s, a$
    $e(s,a) \leftarrow 0 \qquad \forall s, a$
    for $t = 0, \dots, T_k - 1$ **do**
      $\delta_t^{\pi_k} \leftarrow R_{t+1} + \gamma \mathbb{E}_{\pi_k} q_{k+1}(S_{t+1}, \cdot) - q_{k+1}(S_t, A_t)$
      $c_t \leftarrow \lambda \min\left(1, \frac{\pi_k(S_t,A_t)}{\mu_k(S_t,A_t)}\right)$
      for all $s \in \mathcal{S}, a \in \mathcal{A}$ **do**
        $e(s,a) \leftarrow c_t \gamma e(s,a) + \mathbb{I}\{(S_t, A_t) = (s,a)\}$
        $q_{k+1}(s,a) \leftarrow q_{k+1}(s,a) + \alpha_k \delta_t^{\pi_k} e(s,a)$
      end for
    end for
  end for

Assumption 3.2

The operators $\mathcal{P}^{\pi_k}$ and $\mathcal{P}^{\pi_k \wedge \mu_k}$ asymptotically commute. That is, for any Q-function q:

$$\lim_{k\to\infty} \|(\mathcal{P}^{\pi_k}\mathcal{P}^{\pi_k\wedge\mu_k} - \mathcal{P}^{\pi_k\wedge\mu_k}\mathcal{P}^{\pi_k})q\| = 0.$$

Theorem 3.5

Consider a sequence of sample trajectories, with the kth trajectory $S_0, A_0, R_1, S_1, A_1, R_2, \dots$ generated by following μ_k: $A_t \sim \mu_k(\cdot|S_t)$. For each (s,a) along this trajectory, with ℓ the time of first occurrence of (s,a), update

$$q_{k+1}(s,a) \leftarrow q_k(s,a) + \alpha_k \sum_{t=\ell}^{\infty} \delta_t^{\pi_k} \sum_{j=\ell}^{t} \gamma^{t-j} \left(\prod_{i=j+1}^{t} c_i\right) \mathbb{I}\{(S_j, A_j) = (s,a)\}, \tag{3.19}$$

where $\delta_t^{\pi_k} \stackrel{\text{def}}{=} R_{t+1} + \gamma \mathbb{E}_{\pi_k} q_k(S_{t+1}, \cdot) - q_k(S_t, A_t)$. We consider the Retrace(λ) algorithm where $c_i = \lambda \min\big(1, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}\big)$. Assume that $(\pi_k)_{k\in\mathbb{N}}$ are increasingly greedy w.r.t. $(q_k)_{k\in\mathbb{N}}$ and are each ϵ_k-away from the greedy policies $(\pi_{q_k})_{k\in\mathbb{N}}$, i.e. $\max_s \|\pi_k(\cdot|s) - \pi_{q_k}(\cdot|s)\|_1 \leq \epsilon_k$, with $\epsilon_k \to 0$. Let Assumptions 2.2, 2.3, 3.1 and 3.2 hold. Then $q_k \to q^*$ almost surely.

The proof extends similar convergence proofs of TD(λ) by [Bertsekas and Tsitsiklis 1996] and of optimistic policy iteration by [Tsitsiklis 2003], and is provided in Appendix A.7. Notice that compared to Theorem 3.4 we do not assume that $\mathcal{T}^{\pi_0} q_0 - q_0 \geq 0$ here. However, we make the additional (rather technical) Assumption 3.2. It is satisfied for example when the probability assigned by the behavior policy $\mu_k(\cdot|s)$ to the greedy action $\pi_{q_k}(s)$ is independent of s. Examples include ϵ-greedy policies, or more generally mixtures between the greedy policy π_{q_k} and an arbitrary distribution μ (see Lemma A.4 in the appendix for the proof):

$$\mu_k(a|x) = \epsilon \frac{\mu(a|x)}{1 - \mu(\pi_{q_k}(x)|x)} \mathbb{I}\{a \neq \pi_{q_k}(x)\} + (1-\epsilon)\mathbb{I}\{a = \pi_{q_k}(x)\}. \tag{3.20}$$

Notice that the mixture coefficient ϵ needs not go to 0.

3.4.4 Experiments

To validate our theoretical results, we employ Retrace(λ) in an experience replay [Lin 1993] setting, where sample transitions are stored within a large but bounded *replay memory* and subsequently replayed as if they were new experience. Naturally, older data in the memory is usually drawn from a policy which differs from the current policy, offering an excellent point of comparison for the algorithms presented in Table 3.3.

Our agent adapts the DQN architecture of [Mnih et al. 2015] to replay short sequences from the memory instead of single transitions. The Q-function target for a sample sequence $S_t, A_t, R_{t+1}, \ldots, S_{t+k}$ is

$$\Delta q(S_t, A_t) = \sum_{i=t}^{k-1} \gamma^{i-t} \Big(\prod_{j=t+1}^{i} c_j \Big) \big[R_{i+1} + \gamma \mathbb{E}_\pi q(S_{i+1}, \cdot) - q(S_i, A_i) \big].$$

We compare our algorithms' performance on 60 different Atari 2600 games in the Arcade Learning Environment [Bellemare et al. 2013] using [Bellemare et al. 2013]'s inter-algorithm score distribution. Inter-algorithm scores are normalized so that 0 and 1 respectively correspond to the worst and best score for a particular game, within the set

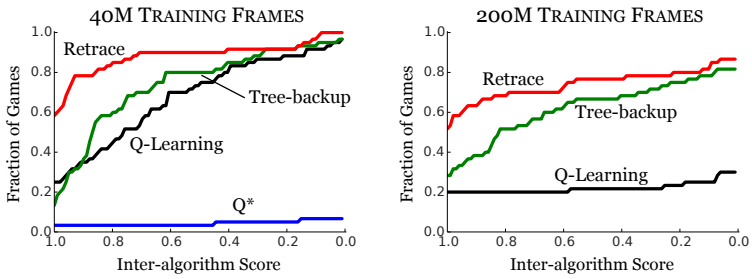

Figure 3.4: Inter-algorithm score distribution for λ-return ($\lambda = 1$) variants and Q-Learning ($\lambda = 0$).

of algorithms under comparison. If $g \in \{1, \ldots, 60\}$ is a game and $z_{g,a}$ the inter-algorithm score on g for algorithm a, then the score distribution function is

$$f(s) \stackrel{\text{def}}{=} \frac{|\{g : z_{g,a} \geq x\}|}{60}.$$

Roughly, a strictly higher curve corresponds to a better algorithm. See Fig. 3.4 for the results.

Across values of λ, $\lambda = 1$ performs best, except for Q* where $\lambda \leq 0.5$ perform better (Fig. 3.5). In general, Q* is highly sensitive to the choice of λ (see Fig. 3.4, left). Both Retrace and TB(λ) achieve dramatically higher performance than Q-Learning early on and maintain their advantage throughout. Compared to TB(λ), Retrace(λ) offers a narrower but still marked advantage. For the full experimental and performance details, we refer the reader to the appendices of the original publication [Munos et al. 2016].

3.4.5 Discussion

This section will present a brief discussion on a number of topics related to our analysis. We will begin by discussing the motivation behind the choice of the shape of trace coefficients for Retrace.

Choice of the trace coefficients c_i

Theorems 3.3 and 3.4 ensure convergence to q^π and q^* for any trace coefficient $c_i \in [0, \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}]$. This in itself is interesting: we may under-estimate, but we may not over-estimate the contribution of an action. However, to make the best choice of c_i, we need to consider the *speed* of convergence, which depends on both (1) the variance of the

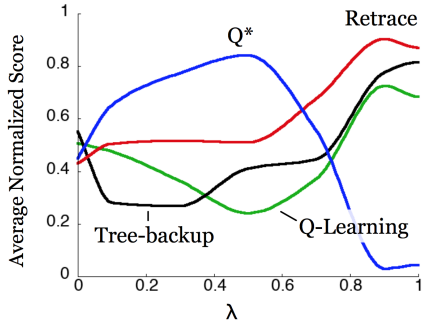

Figure 3.5: Average inter-algorithm scores for each value of λ. The DQN scores are fixed across different λ, but the corresponding inter-algorithm scores vary depending on the worst and best performer within each λ. **Note that average scores are not directly comparable across different values of λ.**

online estimate, which indicates how many online updates are required in a single iteration of $\mathcal{U}$, and (2) the contraction coefficient of $\mathcal{U}$. We summarize the intuitions for these relationships below.

Variance. The variance of the estimate strongly depends on the variance of the product $(c_1 \dots c_t)$, which is not an easy quantity to control in general, as (c_i) are usually not independent. However, assuming independence and stationarity of (c_i), we can deduce that $\mathrm{Var}\left[\sum_t \gamma^t c_1 \dots c_t\right]$ is at least $\sum_t \gamma^{2t} \mathrm{Var}\left[c\right]^t$, which is finite only if $\mathrm{Var}\left[c\right] < 1/\gamma^2$. Thus, an important requirement for a numerically stable algorithm is for $\mathrm{Var}\left[c\right]$ to be as small as possible, and certainly no larger than $1/\gamma^2$. Note that this rules out importance sampling, for which $c \propto \frac{\pi(a|s)}{\mu(a|s)}$, and thus $\mathrm{Var}\left[c|s\right] \propto \sum_a \mu(a|s) \left(\frac{\pi(a|s)}{\mu(a|s)} - 1\right)^2 = \sum_a \frac{\pi(a|s)^2}{\mu(a|s)} - 1$, which may be larger than $1/\gamma^2$ for some π and μ. To ensure this does not happen, we take $c_i \leq 1$.

Contraction speed. The contraction coefficient $\eta \in [0, \gamma]$ of $\mathcal{U}$ depends on how much the traces have been cut, and should be as small as possible (since it takes $\log(\frac{1}{\epsilon}) / \log(\frac{1}{\eta})$ iterations of $\mathcal{U}$ to obtain an ϵ-approximation). μ is smallest when the traces are not cut at all – indeed if $c_i = 1$ for all s, $\mathcal{U}$ is the policy evaluation operator which produces q^π in a single iteration. When the traces are cut, we do not benefit from learning from full returns – in the extreme, $c_1 = 0$ and $\mathcal{U}$ reduces to the single-step Bellman operator with $\eta = \gamma$.

Although the traces (c_i) should be as large as possible, they probably should not be larger than 1, or the update rule would consider the future to be more important than the present.

A reasonable trade-off between low variance (when c_i are small) and high contraction speed (when c_i are large) is given by Retrace(λ), for which we prove the convergence of the online algorithm.

If we relax the assumption that the trace is Markovian (in which case only the result for policy evaluation has been proven so far) we could trade off a low trace at some time for a possibly larger-than-1 trace at another time, as long as their product is less than 1. A possible choice could be:

$$c_t = \lambda \min\Big(\frac{1}{c_1 \dots c_{t-1}}, \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\Big). \tag{3.21}$$

Other points of discussion

No GLIE assumption. The crucial point of Theorem 3.4 is that convergence to q^* occurs for *arbitrary* behavior policies. Thus the online result in Theorem 3.5 does not require the behavior policies to become greedy in the limit of infinite exploration (i.e. GLIE assumption, [Singh et al. 2000]). We believe Theorem 3.5 provides the first convergence result to q^* for a λ-return (with $\lambda > 0$) algorithm that does not require this (difficult-to-uphold) assumption.

Proof of Watkins' Q(λ). As a corollary of Theorem 3.5 when selecting our target policies π_k to be greedy w.r.t. q_k (i.e. $\epsilon_k = 0$), we deduce that Watkins' Q(λ) [Watkins 1989]) converges a.s. to q^* (under the assumption that μ_k commutes asymptotically with the greedy policies, which is satisfied for e.g. μ_k defined by Eq. (3.20)). We believe this is the first such proof.

Increasingly greedy policies. The assumption that the sequence of target policies (π_k) is increasingly greedy w.r.t. the sequence of (q_k) is more general that just considering greedy policies w.r.t. (q_k) (as in Watkins's Q(λ)), and may be more efficient. Indeed, using non-greedy target policies π_k can speed up convergence as the traces will not be cut as frequently. Of course, in order to converge to q^*, we eventually need the target policies (and not, as discussed above, the behavior policies, as mentioned above) to become greedy in the limit (i.e. $\epsilon_k \to 0$ as stated in Theorem 3.4).

Existing extensions. Many of the directions for future work have already been tackled since the time of the publications underlying this chapter. [Gruslys et al. 2018] incorporate Retrace in the actor-critic setting to be able to stably replay experiences off-policy. [Touati et al. 2017] analyze the convergence of the general operator underlying Retrace under linear function approximation, and extend it to a gradient-based form to have this convergence hold. [Mahmood et al. 2017] formulate another general class of stable off-policy algorithms with action-dependent bootstrapping instead of importance sampling, and show that Retrace can be expressed in this way.

Limitations. Although the importance sampling ratio is truncated, Retrace still relies on it being available, and hence for the behavior policy to be stochastic. This can be limiting in general, and is the reason why Chapter 4 relies on the Tree-Backup algorithm instead. The extra Assumption 3.2 for the convergence of the online algorithm, requires that the exploration strategy is state-independent, and hence rules out most strategic exploration strategies. Removing this assumption remains an open problem.

3.5 Related Work

This chapter has focused specifically on *model-free* action-value off-policy learning. The treatment of this problem can be done in other settings as well, and we briefly discuss several related ideas.

3.5.1 Doubly Robust Off-Policy Policy Evaluation

There is an interesting connection of the work described so far and the *doubly robust* framework of off-policy value estimation [Jiang and Li 2016]. This method falls in between model-based and model-free methods. It estimates the models r and p from samples, and uses them to obtain the values $\hat{v}$ and $\hat{q}$ analytically, which are then used to reduce the variance of the model-free importance sampling estimate. The method operates over a fixed horizon H and updates its estimates as follows:

$$V_{H+1-t} = \hat{v}(S_t) + \rho_t(R_{t+1} + \gamma V_{H-t} - \hat{q}(S_t, A_t)), \tag{3.22}$$

with $V_0(s) = 0, \forall s \in \mathcal{S}$, and V_H providing the complete estimate over a horizon H. The doubly robust label refers to this estimate being robust to either the errors in the model (and hence in $\hat{v}$ and $\hat{q}$), or the variance due to the importance sampling estimate. [Thomas and Brunskill 2016] propose a few extensions to this estimator with further theoretical guarantees and reliable empirical performance.

Interestingly, we can make the connection between the estimator of Eq. (3.22) and the off-policy corrected operator of this chapter. In particular, we can rewrite the above as:

$$
\begin{aligned}
V_H &= \hat{v}(S_1) + \rho_1(R_2 + \gamma\hat{v}(S_2) - \hat{q}(S_1, A_1) + \gamma\rho_2(R_3 + \gamma\hat{v}(S_3) - \hat{q}(S_2, A_2) + \gamma\rho_3(\ldots))) \\
&= \hat{v}(S_1) + \sum_{t=1}^{H} \gamma^{t-1} \left(\prod_{j=1}^{t} \rho_j \right) (R_{t+1} + \gamma \underbrace{\hat{v}(S_{t+1})}_{\mathbb{E}_\pi \hat{q}(S_{t+1}, \cdot)} - \hat{q}(S_t, A_t)) \\
&= \hat{v}(S_1) + \sum_{i=1}^{H} \gamma^{t-1} \left(\prod_{j=1}^{t} \rho_j \right) \hat{\delta}_t^\pi,
\end{aligned}
$$

where as before $\rho_i \stackrel{\text{def}}{=} \frac{\pi(A_i|S_i)}{\mu(A_i|S_i)}$. This estimator is very similar to a fixed-horizon, $\lambda = 1$ version of the unified operator of Eq. (3.17) for V-functions in the policy evaluation setting. The main difference is that $\hat{\delta}_t^\pi$ depends on $\hat{q}$, which is a separate model-based estimate, independent of the doubly-robust estimator itself. Unlike Retrace(λ) or Q$^\pi$(λ), the complete importance sampling ratio is used. The authors provide a variance analysis of their estimate, and it would be interesting to adapt it to the Retrace setting. Alternatively, it would be interesting to investigate whether the sufficiency of the truncated importance sampling ratio as demonstrated in the analysis of Retrace yields benefits in the doubly robust setting.

3.5.2 Off-Policy Policy Gradient Methods

Policy gradient algorithms are an important subset of direct policy search methods, in which the gradient of the policy is adjusted in the direction of increasing the return. The problem of off-policy learning is slightly more subtle in this setting. Naive approaches (e.g. REINFORCE [Williams 1992]) consider the generated return directly, and hence have no mechanism of separating behaviors from targets. *Actor-critic* methods mitigate this by replacing the sample of the reward experience with the *critic* value function, which can, potentially, be learnt off-policy with TD learning [Degris et al. 2012]. The absence of stable multi-step off-policy TD algorithms in the past, however, limited both the actor and the critic to single-step sequences. Retrace(λ) makes progress towards removing this limitation and enables efficient off-policy policy gradient algorithms, as seen in a number of recent instances (e.g. [Gruslys et al. 2018, Espeholt et al. 2018]).

It can still be tricky to achieve stable learning when training policies with off-policy samples. To address this, [Gu et al. 2017] introduce an additional degree of freedom that interpolates between on- and off-policy gradient updates, and show that in some sense the best of both worlds could be achieved.

3.5.3 Connection with the Metropolis-Hastings Algorithm

Finally, we make another, intriguing connection to a classical algorithm. Before doing so, let us briefly summarize the Markov Chain Monte Carlo (MCMC) setting. The idea is to estimate some *target* distribution P via sampling a Markov chain whose stationary distribution is P. If P is complex, designing the appropriate Markov chain may be difficult. This is the problem that the Metropolis-Hastings algorithm [Metropolis et al. 1953, Hastings 1970] solves: it is able to achieve the desired target distribution from an *arbitrary* proposal distribution Q that governs the Markov chain. It does so by generating states according to Q, at each transition x, x' querying an "acceptance condition":

$$A(x'|x) = \min\Big(1, \frac{P(x'|x)}{P(x|x')}\frac{Q(x|x')}{Q(x'|x)}\Big), \tag{3.23}$$

and proceeding with x' w.p. $A(x'|x)$, and otherwise remaining in x and repeating the process. The shape of A should look familiar, and reminiscent of the shape of c in Retrace(1) (Eq. (3.18)). It turns out we can take the resemblance further by reinterpreting off-policy learning in the MCMC setting. In particular, recall the augmented MDP interpretation of state-action values given in Remark 2.1: the new MDP $\mathcal{M}'_\pi$ contains all of the original states of the original MDP $\mathcal{M}$, together with all s, a pairs. Transitions of the form $s, a \to s'$ are shared with $\mathcal{M}$ and made according to p, while transitions of the form $s \to s, a$ are made according to π. Then, to say that we learn off-policy is to say that we wish to reconstruct the target Markov chain underlying $\mathcal{M}'_\pi$, while sampling a proposal, or *behavior*, Markov chain underlying $\mathcal{M}'_\mu$. In these Markov chains there are two types of $x \to x'$ transitions to consider.

1) $\mathbf{s \to s, a}$. Recall that $P(x'|x) = P(s, a|s) = \pi(a|s)$ and $Q(x'|x) = Q(s, a|s) = \mu(a|s)$. Noticing that the reverse probability $P(x|x') = P(s|s, a) = Q(s|s, a) = 1$ by construction, we can rewrite Eq. (3.23) as:

$$A(a|s) = \min\left(1, \frac{\pi(a|s)}{\mu(a|s)}\right), \tag{3.24}$$

which is the formula for the Retrace $c_{s,a}$ coefficient exactly.

2) $\mathbf{s, a \to s'}$. The forward probabilities for these transitions are independent of the policy and shared between π and μ: $P(x'|x) = Q(x'|x) = p(s'|s, a)$, but to obtain the acceptance ratio we still need to consider $P(x|x') = P(s, a|s')$ and $Q(s, a|s')$, which do depend on the policy. We know that for an MDP transition matrix p we have: $p(s, a|s')d^\pi(s') =$

$p(s'|s,a)\pi(a|s)d^\pi(s)$, where d^π is the stationary distribution of the policy π *in the original MDP*. Hence:

$$P(s,a|s') = p(s,a|s') = \frac{p(s'|s,a)\pi(a|s)d^\pi(s)}{d^\pi(s')}, \tag{3.25}$$

and we have our acceptance ratio:[4]

$$A(s'|s,a) = \min\left(1, \frac{\pi(a|s)}{\mu(a|s)}\frac{d^\pi(s)}{d^\pi(s')}\frac{d^\mu(s')}{d^\mu(s)}\right). \tag{3.26}$$

There are hence a couple of differences in the settings that both have to do with the difference in the goals of the procedures. Off-policy learning attempts to learn the value function q^π from trajectories generated by μ, while Metropolis-Hastings wishes to estimate the Markov Chain w.r.t. π, while sampling one w.r.t. μ.

- Instead of considering the ratio in Eq. (3.26), Retrace accepts all s' w.p. 1, and only queries the acceptance condition for policies via Eq. (3.24). The introduced bias is mitigated by the fact that off-policy corrections are applied in the value space, inducing the correct fixed point.
- In multi-step off-policy learning one does not resample actions in the case of rejection, but instead the return is terminated in favor of bootstrapping with the value function.

It is interesting to use this interpretation of off-policy learning to further explore connections to MCMC methods.

3.6 Summary

Starting from the *naive* idea that policy probabilities are not always necessary for convergence, we formulated a policy evaluation algorithm that converges subject to a tradeoff between the degree of bootstrapping λ, distance between policies ϵ, and the discount factor γ. Although in control, determining the existence of a non-trivial ϵ-dependent bound for λ remains an open problem, our experiments suggest that such a bound exists, and closely resembles the $\frac{1-\gamma}{\gamma\epsilon}$ bound from the policy evaluation case.

We proceeded to summarize several existing off-policy return-based algorithms along with our new ones, and cast them in a common form. Then, in an attempt to combine their strengths, we propose an improved Retrace(λ) algorithm, that includes an *approximate*

[4]Note that Eq. (3.25) makes it easy to verify that P and Q are reversible, as required by Metropolis-Hastings.

importance sampling correction, which allows it to enjoy general convergence guarantees. Notably, our convergence analysis yields what we believe to be the first convergence proof of Watkins's Q(λ).

In the next chapter we will use much of the machinery developed here in the context of analyzing multi-step, or *temporally abstract* actions.

4 | From Multi-Step Temporal Differences to Options

Temporal abstraction is essential for learning with sophistication, as it enables one to escape the naive measure of a single time-step, and interact with the environment at a variety of timescales. In reinforcement learning, a temporally abstract action (or an *option*) is equal parts knowing what to do and when to stop. In this chapter we study the latter: What are the effects of longer options on the quality of the plan and the ease of finding it? What would happen if we were to stop what we're doing?

We develop insights into these questions by considering the options framework through the looking glass of multi-step temporal differences from the previous chapter. That the two share commonalities should not come as a surprise, but we cast them into a unified framework for the first time. We first consider a simpler option execution model and show that in it, planning with options closely mirrors the λ-policy iteration algorithm, with option termination affecting the rate of convergence to the solution, but not the solution itself. We then show that the role of option termination is more prominent under the general option execution model, and give a novel algorithm that is able to *off-policy* learn about terminations that never happened.

4.1 Introduction

Abstraction is essential for scaling up learning, and there has been a renewed interest in methods that extract, or leverage it [Bacon et al. 2017, Vezhnevets et al. 2016, Kulkarni et al. 2016, Tessler et al. 2016]. The options framework [Sutton et al. 1999] is the de facto standard for modeling temporal abstraction in reinforcement learning.

The temporal aspect of an option is determined by its *termination condition* β, which roughly determines its length. Learning and planning with longer options is known to be more efficient [Mann et al. 2015]. This is partly due to an option having similar properties to the familiar multi-step λ-returns,which are known to yield faster convergence [Bertsekas and Tsitsiklis 1996].[1] The key qualitative difference between the termination β and eligibility trace λ, however, is that in the usual call-and-return model, β directly affects the solution rather than, like λ, just the rate of convergence. If β is not trivial, this couples the quality of the solution with the quality of the options at hand, and can be restrictive especially if the options are not perfect. Indeed, when a set of options is given, we can show that the shorter the options, the more optimal the resulting policy is at the primitive action level. This poses a challenge: on the one hand, we wish for the options to be long to yield fast convergence and meaningful exploration, but on the other, if these options are not ideal, the more we commit to them, the poorer the quality of our solution. *Interrupting* suboptimal options is one way of addressing this [Sutton et al. 1999], but like "cutting" traces in off-policy learning, it may prevent us from following a coherent policy for more than a couple of steps.

To this end, we propose to terminate options *off-policy*, that is: decouple the *behavior* termination condition that the options execute with, from the *target* termination condition that is to be factored into the solution. We describe a new algorithm, Q(β), that achieves this by leveraging connections to several old and new multi-step off-policy temporal difference algorithms.

This chapter is organized as follows. To illustrate the similarity between λ and β, after introducing relevant background, we first treat a simpler *gated* option execution model, in which a new option is chosen at each step. We show that in this case, planning with options resembles λ-policy iteration, and show theoretically and empirically that β obeys the analogous tradeoffs. We then move on to the general *call-and-return* execution model, and show that the role of β here is more prominent, as it affects the solution. We describe and analyze our algorithm Q(β) that is able to learn the solution w.r.t. any termination condition from the available option traces. The behavior β then reverts back to merely

[1] This similarity is particularly relevant since a *full* β*-model* [Sutton 1995] is at the basis of both paradigms.

controlling convergence speed. We validate Q(β) empirically and show that it can learn an optimal solution from suboptimal options quicker than the alternatives.

4.2 Background

In this section we will briefly transfer our operator definitions to state value functions, as required by the first part of this chapter. We will then introduce the option framework formally.

4.2.1 State value function

In the first part of this chapter we will be concerned with the planning setting. In this setting, where the models are available, it isn't necessary to maintain the complete Q-function, and one may instead only maintain the state value function v, obtaining q using Eq. (2.6) and the models r and p. It is then convenient to consider the MDP dynamics jointly with the policy:

$$p^\pi(s,s') \stackrel{\text{def}}{=} \sum_{a\in\mathcal{A}} \pi(s,a)p(s'|s,a), \qquad \forall s,s' \in \mathcal{S}. \tag{4.1}$$

$$r^\pi(s) \stackrel{\text{def}}{=} \sum_{a\in\mathcal{A}} \pi(s,a)r(s,a), \qquad \forall s \in \mathcal{S}. \tag{4.2}$$

The matrix p^π describes the dynamics of the Markov *chain* induced by a policy π on the MDP, and together with r^π, they define a Markov *Reward* Process, or an MDP with the decision aspect removed [Howard 1971]. Analogously to (2.12) we have that:

$$v^\pi = (I - \gamma p^\pi)^{-1} r^\pi.$$

and the Bellman operators can be overloaded to apply to V-functions:

$$\mathcal{T}^\pi v \stackrel{\text{def}}{=} r^\pi + \gamma p^\pi v, \qquad \mathcal{T}^\pi v^\pi = v^\pi,$$
$$\mathcal{T} v \stackrel{\text{def}}{=} \max_\pi (r^\pi + \gamma p^\pi v), \qquad \mathcal{T} v^* = v^*.$$

4.2.2 The Options Framework

Let us now introduce the options framework formally [Sutton et al. 1999]. An option o is a tuple $(\mathcal{I}^o, \beta^o, \pi^o)$, with $\mathcal{I}^o \subseteq \mathcal{S}$ the initiation set, from which an option may start, $\beta^o : \mathcal{S} \to [0,1]$, the probabilistic termination condition, and π^o, the option policy with

which it navigates through the environment. Just like the MDP reward and transition models r and p, options can be seen to induce *semi*-MDP [Puterman 1994] models R and P as follows [Sutton et al. 1999]:

$$P^o_{ss'} \stackrel{\text{def}}{=} \mathbb{E}_{D:s\to s'|o}\left[\gamma^D\right] = \gamma p^{\pi^o}_{ss'}\beta_{s'} + \gamma \sum_{s''} p^{\pi^o}_{ss''}(1-\beta_{s''})P^o_{s''s'}, \tag{4.3}$$

$$R^o_s \stackrel{\text{def}}{=} \mathbb{E}_{D:s|o}\left[\sum_{i=1}^{D} \gamma^{i-1} r^{\pi^o}(S_{t+i})|S_t = s\right] = r^{\pi^o}_s + \gamma \sum_{s'} p^{\pi^o}_{ss'}(1-\beta_{s'})R^o_{s'}. \tag{4.4}$$

where $\mathbb{E}_{D:s|o}\left[\cdot\right]$ and $\mathbb{E}_{D:s\to s'|o}\left[\cdot\right]$ are the expectations of the option duration D from state s and the travel time between state s and s', respectively, w.r.t. option dynamics p^{π^o} and the termination condition β^o. For simplicity, we assume that options can initiate anywhere: $\mathcal{I}^o = \mathcal{S}, \forall o \in \mathcal{O}$. The policy space of interest is now over options: $\mu : \mathcal{S} \times \mathcal{O} \to [0,1]$. It will also be relevant to consider the *marginal* flat policy over primitive actions κ_μ:

$$\kappa_\mu(a|s) \stackrel{\text{def}}{=} \sum_o \mu(o|s)\pi^o(a|s). \tag{4.5}$$

We will omit the μ subscript, where clear from context.

4.3 Planning with Options as λ-Policy Iteration

We begin by considering the *gated* model of option execution, in which a new option choice is made at every primitive time step [Bacon and Precup 2016]. The term gating originates from neuroscience, where it refers to a cognitive model with a unitary working memory [Braver and Cohen 1999]. Such a model is particularly relevant when the planning setting is considered. We show that planning with option models under the gated model resembles λ-policy iteration [Bertsekas and Ioffe 1996]. We give a new proof of convergence with an asymptotic rate explicitly in terms of the termination parameter. The analogous implications thus carry over: given a fixed set of suitable options:[2]

1. The more options terminate (that is: the shorter they are), the closer each iteration of planning with them is to *value iteration* over primitive actions, and thus the faster it is to perform.

2. The less options terminate (that is: the longer they are), the closer the resulting iteration is to *policy iteration* over primitive actions, and the fewer such iterations are required to find the value of the optimal policy.

[2]We will make this notion precise in Section 4.3.3.

This opposite dependence on the termination amount suggests that just like with the λ parameter, an intermediate value for the termination parameter β will usually be best, and "hard" deterministic terminations may not be ideal. In the rest of this section, we will make the correspondence with λ-PI explicit, use it to give a new convergence proof, and validate the theoretical tradeoff on β empirically.

4.3.1 λ-Policy Iteration

Value iteration (VI) is one of the backbones of approximate dynamic programming. It relies on making incremental updates to the value function, in the direction of the value of the current greedy policy, and is guaranteed to converge to the optimum in the limit of infinite time. Unfortunately, as the discount factor gets closer to one, the convergence may occur very slowly [Bertsekas and Tsitsiklis 1996]. *Policy iteration (PI)* represents the other extreme: it computes the value of the greedy policy at each iteration exactly, and guarantees convergence in a finite number of such iterations. Unfortunately, it's rarely practical, as solving the corresponding system of linear equations involving a matrix inverse is hard when the number of states gets large, and may be fatally inaccurate when using approximations. *Modified* PI suggests a middle ground by taking a fixed number of value iteration steps [Puterman and Shin 1978]. Finally, in the spirit of λ-operators, λ-PI elegantly transitions between the extremes by taking a parameterized number of steps towards the value of the greedy policy [Bertsekas and Ioffe 1996].

We will illustrate that doing a single value iteration with option models is equivalent to a variant of λ-policy iteration over primitive actions, where the λ parameter is replaced by a coefficient that depends on option terminations.

4.3.2 The Gated Options Operator

We first take a look at the underlying policy evaluation operators. Since the focus of this section is planning, we will consider the *state* value function v throughout it. For each option, let us define the following (sub-)stochastic matrices:

$$\begin{aligned} p^{\beta\pi^o}(s,s') &\stackrel{\text{def}}{=} p^{\pi^o}(s,s')\beta^o(s'), \\ p^{\beta\kappa}(s,s') &\stackrel{\text{def}}{=} \sum_o \mu(o|s) p^{\beta\pi^o}(s,s') \end{aligned} \tag{4.6}$$

That is: $p^{\beta\pi^o}(s,s')$ denotes the probability of transitioning from s to s' and terminating in s' under an option o, and $p^{\beta\kappa}(s,s')$ denotes the average of these probabilities over some policy over options μ. The matrices $p^{(1-\beta)\pi^o}$ and $p^{(1-\beta)\kappa}$ can be defined analogously.

Under gating, the option models at each step are an average of all options w.r.t. μ, and the reward model writes:

$$R^{\mu}_{s} \stackrel{\text{def}}{=} \sum_{o} \mu(o|s) \left(r^{\pi^o}_{s} + \gamma \sum_{s'} p^{(1-\beta)\pi^o}_{ss'} \sum_{o'} \mu(o'|s') R^{o'}_{s'} \right) = r^{\kappa}_{s} + \gamma \sum_{s'} p^{(1-\beta)\kappa}_{ss'} R^{\mu}_{s'}. \tag{4.7}$$

Analogously, we have the the transition model:

$$P^{\mu}_{ss'} \stackrel{\text{def}}{=} \gamma p^{\beta\kappa}_{ss'} + \gamma \sum_{s''} p^{(1-\beta)\kappa}_{ss''} P^{\mu}_{s''s'}.$$

Finally, the option-level Bellman operator writes in matrix form:

$$\begin{aligned} \mathcal{T}^{\mu}_{\mathcal{O}} v &\stackrel{\text{def}}{=} R^{\mu} + P^{\mu} v = r^{\kappa} + \gamma p^{\beta\kappa} v + \gamma p^{(1-\beta)\kappa}(R^{\mu} + P^{\mu} v) && (4.8) \\ &= (I - \gamma p^{(1-\beta)\kappa})^{-1}(r^{\kappa} + \gamma p^{\beta\kappa} v) \\ &= (I - \gamma p^{(1-\beta)\kappa})^{-1}(r^{\kappa} + \gamma(p^{\kappa} - p^{(1-\beta)\kappa})v). && (4.9) \end{aligned}$$

Policy Evaluation

Now let us review the evaluation operator for λ-policy iteration (λ-PI) [Bertsekas and Ioffe 1996]:

$$\mathcal{T}^{\pi}_{\lambda} v \stackrel{\text{def}}{=} (I - \gamma\lambda p^{\pi})^{-1}(r^{\pi} + \gamma(1-\lambda)p^{\pi} v) = (I - \gamma p^{\lambda\pi})^{-1}(r^{\pi} + \gamma(p^{\pi} - p^{\lambda\pi})v), \tag{4.10}$$

where we write $p^{\lambda\pi} = \lambda p^{\pi}$ by an analogy with Eq. (4.6), whose general form helps highlight the fact that λ need not be a constant and can depend on the state [Yu and Bertsekas 2012]. Inspecting equations (4.9) and (4.10), we can see that $\mathcal{T}^{\pi}_{\lambda}$ is a special case of $\mathcal{T}^{\mu}_{\mathcal{O}}$, for which $p^{(1-\beta)\kappa} = p^{\lambda\pi}$, i.e. the termination function is a constant λ, and the set of options is the set of primitive actions, $\kappa = \mu = \pi$. Thus, just like λ, β has no effect on the solution, and merely affects the *rate* of convergence. The following proposition (Theorem 1 in [Bacon and Precup 2016]) justifies this by showing that the fixed point of $\mathcal{T}^{\mu}_{\mathcal{O}}$ is independent of the termination scheme.

Proposition 4.1

The options operator $\mathcal{T}^{\mu}_{\mathcal{O}}$ as defined in Eq. (4.9) has a unique fixed point, equal to v^{κ}, the value of the marginalized policy κ.

Control

Let us first characterize the control solution $v^*_{\mathcal{O}} = v^{\kappa_{\mu^*}} = \max_\mu v^{\kappa_\mu}$ that planning at the option level ought to converge to. Because options induce a semi-MDP, this solution remains well-defined. If the option set contains primitive actions, $v^*_{\mathcal{O}}$ is simply v^*. Otherwise, there may be an additional loss related to $\|v^*_{\mathcal{O}} - v^*\|$ (see e.g. [Mann et al. 2015]). In the following, we will simply consider convergence to $v^*_{\mathcal{O}}$, the best value expressible by options.

We are now ready to make the correspondence between the control algorithms. It can be made analogously, but with a couple of differences. The λ-PI algorithm performs the following update at iteration k [Bertsekas and Ioffe 1996, Scherrer 2013]:

$$\begin{aligned}\pi_k &\leftarrow \arg\max_\pi (r^\pi + \gamma p^\pi v_k)\\ v_{k+1} &\leftarrow \mathcal{T}_\lambda^{\pi_k} v_k\\ &= (I - \gamma p^{\lambda\pi_k})^{-1}(r^{\pi_k} + \gamma(p^{\pi_k} - p^{\lambda\pi_k})v_k).\end{aligned}$$

Value iteration with option models (that is: w.r.t. policies over options μ) performs the following update at iteration k:

$$\begin{aligned}\mu_k &\leftarrow \arg\max_\mu (R^\mu + P^\mu v_k)\\ v_{k+1} &\leftarrow \mathcal{T}_{\mathcal{O}}^{\mu_k} v_k\\ &= (I - \gamma p^{(1-\beta)\kappa_k})^{-1}(r^{\kappa_k} + \gamma(p^{\kappa_k} - p^{(1-\beta)\kappa_k})v_k).\end{aligned} \tag{4.11}$$

We will refer to this algorithm as β-PI by analogy. There are two key differences between these algorithms which have to do with the qualitative difference of the policies over primitive actions π_k and κ_k. In particular, in β-PI:

1. The policy improvement step is over options, rather than primitive actions: the policy over options μ_k is greedy, but the corresponding marginal policy over primitive actions κ_k is not, in general.
2. The dependence on β is more subtle than in the case of even a state-dependent λ, as it depends on the policy over options μ_k, and may take different values even for μ's that yield the same κ.

We will see that these differences are non-essential to the convergence properties of the operator, given a monotonicity assumption on the marginal policies.

This link explicitly justifies temporally extended actions: a single value iteration step over option models induces a version of λ-PI over primitive actions, which is known to be more efficient for nonzero values of λ [Scherrer 2013], i.e. options that are non-primitive. While it has been known [Mann et al. 2015] that VI with options converges faster, the explicit link with λ-PI is, to our knowledge, novel.

4.3.3 Analysis

We can conduct similar analysis to [Bertsekas and Tsitsiklis 1996] to prove convergence of planning with option models to the optimal policy. We know from options literature [Precup et al. 1998] that the iteration (4.11) converges to $v^*_{\mathcal{O}}$. That line of analysis however, in the semi-MDP fashion, typically treats options as a black box. Here, we are interested in obtaining asymptotic convergence rates explicitly in terms of the termination coefficient β^o, as is done when analyzing λ-operators. To do so, we will require a monotonicity assumption on our option set.

Assumption 4.1: Marginal policies get greedier.

There exists an index k' s.t. for all $k \geq k'$, the sequence of marginal policies (κ_k) is *increasingly greedy*, in the sense that

$$\mathcal{T}^{\kappa_k} v_k \geq \mathcal{T} v_k - \epsilon_k,$$

s.t. $\epsilon_k \to 0$, as $k \to \infty$.

Although this assumption is analytical, rather than practical, we did find it to hold in our experiments. It would be interesting to characterize options that would satisfy it, as they may have favorable convergence properties. We leave this for the future.

In the following we will consider convergence to the best policy expressible by the option set: $v^*_{\mathcal{O}} = v^{\kappa_{\mu^*}} = \max_\mu v^{\kappa_\mu}$, which if the options can express the optimal policy is equal to v^*.

Convergence of Each Iteration

Let us first describe the mapping underlying each iteration (4.11). The following proposition is analogous to Proposition 2.7 from [Bertsekas and Tsitsiklis 1996].

Proposition 4.2

Given an option set $\mathcal{O}$, policy over options μ_k, and an estimate v_k, consider the mapping M_k:

$$M_k v = r^{\kappa_k} + \gamma p^{\beta \kappa_k} v_k + \gamma p^{(1-\beta)\kappa_k} v. \tag{4.12}$$

M_k is a γ-contraction mapping w.r.t. the max norm, whose unique fixed point is the next iterate v_{k+1}. The contraction factor ξ_k is state-dependent and can be written:

$$\xi_k(s) = \gamma(1 - c_k(s)) \leq \gamma, \tag{4.13}$$

where $c_k(s) = \sum_o \mu_k(o|s) \sum_{s'} p^{\pi^o}(s, s')\beta^o_{s'} = \sum_{s'} p^{\beta\kappa_k}_{ss'}$, the expected next state option termination.

Thus, the contraction factor depends on the expected next-state termination c_k, which is nonzero in a state if there is a nonzero probability to transition to a terminating state w.r.t. the options chosen by μ (e.g. the greedy option). In the extreme: $\beta^o_s = 1$ in all s, the options terminate immediately, and M_k converges is one step. This is the case when all actions primitive. In that case, $p^{\beta\kappa_k} = p^{\kappa_k}$, $p^{(1-\beta)\kappa_k}(s, s') = 0, \forall s, s'$, and the iteration (4.12) writes:

$$M_k v = r^{\kappa_k} + \gamma p^{\kappa_k} v_k,$$

a single value iteration. The other extreme is $\beta^o_{s'} = 0$, the options never terminate, and computing v_{k+1} requires full policy iteration step, which returns the value of the next policy κ_k exactly. Then, $p^{\beta\kappa_k}(s, s') = 0, \forall s, s'$, $p^{(1-\beta)\kappa_k} = p^{\kappa_k}$, and the iteration (4.12) writes:

$$M_k v = r^{\kappa_k} + \gamma p^{(1-\beta)\kappa_k} v = r^{\kappa_k} + \gamma p^{\kappa_k} v.$$

Finally, if $\beta^o_{s'} = \beta$ is constant for all states and options, we recover the familiar equations of the λ-operator exactly.

Convergence of β-PI

We are now ready to prove convergence of β-PI to $v^*_{\mathcal{O}}$, the optimal value given the set of options. The proof is analogous to Proposition 2.8 [Bertsekas and Tsitsiklis 1996], but the extension is not trivial since the sequence of policies (κ_k) is not greedy. Instead we use the assumption 4.1 of (κ_k) being *increasingly* greedy. The proof is deferred to the appendix to preserve readability.

Theorem 4.1

Let Assumption 4.1 hold. Then, value iteration with option models (iteration (4.11)) converges to $v^*_{\mathcal{O}}$ almost surely. Furthermore, after some index $k \geq \bar{k}$, the rate of convergence for each state s is linear in $\eta(s)$:

$$|v_{k+1}(s) - v^*_{\mathcal{O}}(s)| \leq \eta(s)\|v_k - v^*_{\mathcal{O}}\|_\infty,$$

$$\eta(s) = 1 - (1-\gamma)C(s) \leq \gamma, \tag{4.14}$$

where

$$C(s) = \mathbb{E}_{\substack{O_t \sim \mu_k(\cdot|S_t) \\ S_{t+1} \sim p^\kappa(\cdot|S_t)}} \left[\sum_{t=0}^{\infty} \gamma^t \left(\prod_{i=1}^{t} (1 - \beta^{O_i}(S_i)) | S_0 = s \right) \right].$$

The contraction coefficient η from Eq. (4.14) determines how many iterations are required, and we want it to be as small as possible. This is achieved when C is as large as possible. In the extreme, if $C = 1$, i.e. options never terminate, the value $v^*_{\mathcal{O}}$ is found in a single iteration. Notice that $C \approx \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \prod_{i=1}^{t} \xi(S_i)\right]$, with ξ from (4.13), i.e. the two convergence rates depend on c in opposite ways, and thus will be optimal for some intermediate value of c. This convergence rate estimate (4.14) applies only for $k \geq \bar{k}$, i.e. when the optimal policy has already been found. It is however qualitatively correct as discussed in [Bertsekas 2015]. Indeed, in the extreme of $(1-\beta) \to 1$, β-PI becomes PI, and converges in the finite number of iterations (contrary to the convergence of VI in the limit of infinite time).

A practical corollary of this discussion is that having intermediate termination values will be most efficient. In particular, having deterministic terminations is similar in effect to "cutting" the trace parameter, which impairs the efficiency of multi-step operators, by reducing the flow of information [Munos et al. 2016], and so may not be the best choice. We will illustrate this point empirically in the next section.

4.3.4 Experiments

We aim to support the claim that in the planning setting, similarly to λ, intermediate termination values, and in particular, soft (non-1) terminations in goal states are more efficient. To better analyze the effects of the termination parameter, we do not consider primitive actions in the option set.

Taxi

We consider the classical Taxi domain [Dietterich 2000] (Fig. 4.1). Here, the goal is to navigate to one of the four target locations that contains the passenger, execute the "pickup" action, navigate to the destination, and "dropoff" the passenger. The agent's state is the location of the taxi, the coordinates of the passenger's origin and destination, and whether the Taxi is full (i.e. if a successful pickup has been executed). There are 6 options: 4 that navigate to each of the target locations, as well as Put and Get macros, which are comprised of navigating to the passenger's origin resp. destination and executing

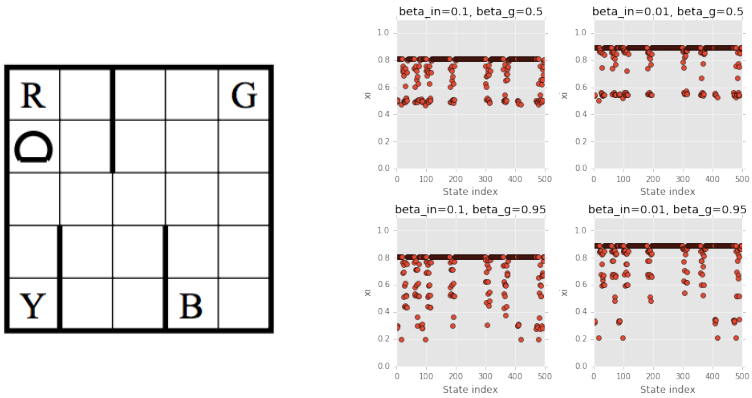

Figure 4.1: **Left.** Taxi domain. The agent navigates between the 4 target locations, executing pickups and drop-offs. **Right.** The contraction factor ξ from (4.13) in episodic Taxi as a function of β_{goal} and β_{in} (average across the 10 VI steps, $\gamma = 0.9$).

a dropoff resp. pickup actions. There is a step-negative reward, a penalty for illegal pickups and dropoffs, and a reward for successfull transfers.[3].

In order to capture the difficulties of long-term planning, we also consider an *infinite* variant of the Taxi task: every time a passenger is successfully dropped off, a new task is instantiated, with the new passenger origin and destination selected at random.

Details

The value function is initialized pessimistically[4] with a mean of $-1000 < -r_{\max}/(1-\gamma)$ and a standard deviation of 100. We report performance for the best-performing discounts for each domain: $\gamma = 0.9$ for Taxi and $\gamma = 0.99$ for Infinite Taxi. We ran 10 and 30 value iteration steps for Taxi and Infinite taxi, respectively, with evaluation occuring every 1 or 2 iterations. Evaluation reports the cumulative rewards of running the greedy policy over options for 500 (primitive) steps. Summary performance refers to an average reward collected over all evaluation steps. All results are an average of 10 independent runs. We evaluate all variants with a fixed set of deterministically terminating options, and only use the termination schemes during planning.

[3]We use the values for these from the Taxi-v2 environment in OpenAI gym

[4]This is to comply with the convergence analysis. In practice, it does not influence the results significantly.

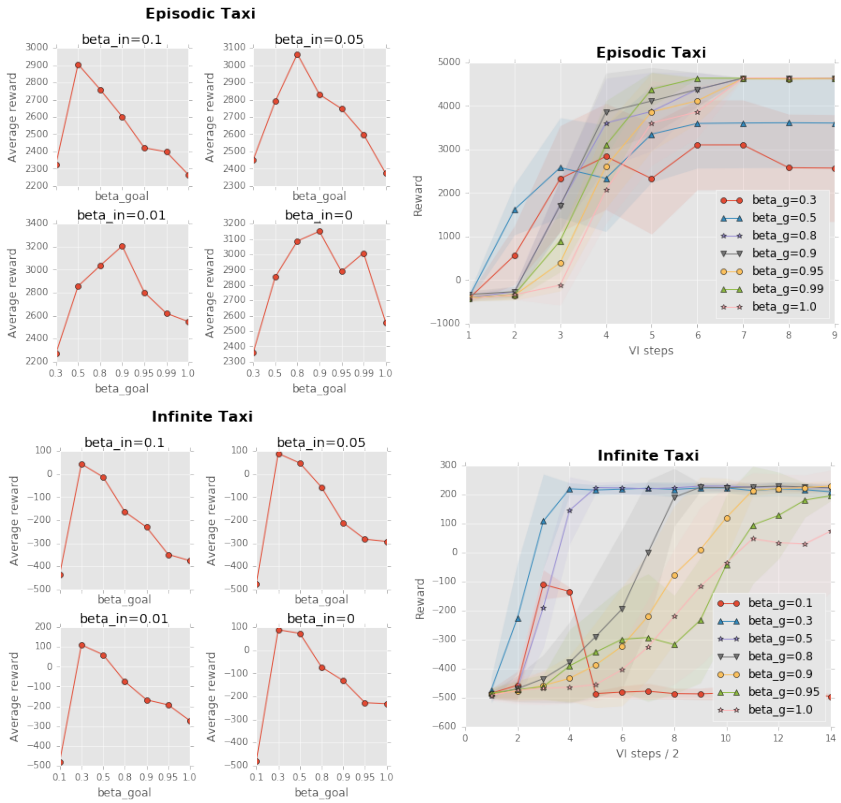

Figure 4.2: **Left.** Summary performance, average of 10 independent runs. The curves are roughly U-shaped, and in particular the best-performing β_{goal} is less than one in all cases. Note the difference in y-axes. **Right.** Learning curves for different values of β_{goal} with the best value of β_{in} for each. The shaded regions show standard deviation.

Since the options are natively deterministic, it will be convenient to distinguish between the values of β in and outside of the terminating regions. Letting G^o denote the set of terminating states of option o, we will write $\beta^o_{goal} \equiv \beta^o_{s_g}$ for $s_g \in G^o$, and $\beta^o_{in} \equiv \beta^o_s$ for $s \notin G^o$. We set the same parameters for all options, and so will drop the ^{o} superscript, and simply refer to the values β_{goal} and β_{in}.

Discussion

The contraction factor ξ from (4.13) depends on both the environment and option dynamics. The intuition for how β_{goal} and β_{in} influence it is as follows: Since the agent spends most of its time in non-goal states, β_{in} has a stronger effect on ξ at each step, while β_{goal} controls the span of ξ (see Figure. 4.1 for an illustration).

The results of our experiments are presented in Figure 4.2. In the tasks we considered, we found it rarely useful to have β_{in} (and hence c) to be large. This makes sense, since our problems have a relatively small number of states, and the effect of efficient value iteration is negligible. β_{goal} however obeyed the tradeoff we predicted readily, with its intermediate values always outperforming the extremes.

4.3.5 Discussion

We have analyzed convergence properties of planning with options in terms of the termination condition parameter. We did so by making the analogy with λ-policy iteration explicit. Our analysis and experiments suggest that intermediate values of β perform best, and in general β should be treated as a parameter.

The learning setting The gating setting is particularly well-suited for planning, since it entails the marginal policy κ, taken simultaneously from all states. When learning with option models, *call-and-return* model of execution is assumed more commonly. In this model the termination function controls the behavior of the agent, and κ is unlikely to be possible to sample.[5] In the rest of the chapter we will consider the call-and-return setting, and observe the corresponding differences and similarities.

[5]Imagine a single never-terminating option: even if there are others in the set that can express the optimal policy, the agent will never have a chance to take them.

4.4 Learning with Options that Terminate Off-Policy

We are now ready to move on to our main contribution. Unlike what we have discussed so far, in the *call-and-return* model of option execution, the solution is no longer indifferent to the termination scheme. The main contribution of this chapter is proposing to decouple the behavior termination condition from the target solution, and giving an algorithm that achieves this.

We will first derive the fixed point of the classical call-and-return option operator. Using its shape for intuition, we will then incrementally build up to our proposed *off-policy* termination option operator, starting from the classical intra-option equations. We will analyze the convergence properties of this operator for both policy evaluation and control, and state the corresponding online algorithm Q(β). Finally, we will validate Q(β) empirically.

4.4.1 The Call-and-Return Operator

In the call-and-return model of option execution, an option is run until completion (according to its termination condition), and only then a new option choice is made [Precup et al. 1998]. This suggests the following state-*option* analogues of the state-action transition operator from Eq. (2.10), and the Bellman operator from Eq. (4.8). For a policy over options μ:

$$\mathcal{P}^{\mu}_{\mathcal{O}} q(s,o) \stackrel{\text{def}}{=} \sum_{s'} P^{o}(s,s') \sum_{o'} \mu(o'|s') q(s',o') \tag{4.15}$$

$$\mathcal{T}^{\mu}_{\mathcal{O}} q(s,o) \stackrel{\text{def}}{=} R^{o}_{s} + \mathcal{P}^{\mu}_{\mathcal{O}} q(s,o). \tag{4.16}$$

We wish to derive the fixed point of this operator, but at the primitive action resolution. Let ν be an arbitrary policy over options, and $c : \mathcal{S} \times \mathcal{O} \to [0,1]$ a coefficient function. Consider the following transition operator and its corresponding Bellman operator:

$$\mathcal{P}^{c\nu} q(s,o) \stackrel{\text{def}}{=} \sum_{s'} p^{\pi^{o}}_{ss'} c(s',o) \sum_{o'} \nu(o'|s') q(s',o'), \tag{4.17}$$

$$\mathcal{T}^{c\nu} q \stackrel{\text{def}}{=} c r^{\pi} + \gamma \mathcal{P}^{c\nu} q,$$

where r^{π} is the $|\mathcal{S}| \times |\mathcal{O}|$-vector of $r^{\pi^{o}}$ for all options. Note that $\mathcal{T}^{c\nu}$, like $\mathcal{T}^{\pi}$, is a one-step operator, whereas $\mathcal{T}^{\mu}_{\mathcal{O}}$ is an option-level operator. In particular, $\mathcal{P}^{c\nu}$ defines the following operators corresponding to option *continuation* and *termination*, respectively:

$$\mathcal{P}^{(I-\beta)\iota}(s,o) \stackrel{\text{def}}{=} \sum_{s'} p^{\pi^{o}}_{ss'} (1 - \beta^{o}(s')) q(s',o),$$

$$\mathcal{P}^{\beta\mu}(s,o) \stackrel{\text{def}}{=} \sum_{s'} p^{\pi^o}_{ss'} \beta^o(s') \sum_{o'} \mu(o'|s') q(s',o').$$

That is: ι (*"iota"*) is the policy over options that maintains the current (argument) option. Using these operators, we can express $\mathcal{P}^{\mu}_{\mathcal{O}}$ from Eq. (4.15) and the reward model from Eq. (4.4) concisely for all state-option tuples:

$$\mathcal{P}^{\mu}_{\mathcal{O}} q = (I - \gamma \mathcal{P}^{(I-\beta)\iota})^{-1} \gamma \mathcal{P}^{\beta\mu} q, \qquad R = (I - \gamma \mathcal{P}^{(I-\beta)\iota})^{-1} r^{\pi},$$

and rewrite the option-level Bellman operator from Eq. (4.16):

$$\mathcal{T}^{\mu}_{\mathcal{O}} q = (I - \gamma \mathcal{P}^{(I-\beta)\iota})^{-1} (r^{\pi} + \gamma \mathcal{P}^{\beta\mu} q). \tag{4.18}$$

The following proposition derives the fixed point of $\mathcal{T}^{\mu}_{\mathcal{O}}$ in terms of the one-step operators $\mathcal{T}^{(I-\beta)\iota}$ and $\mathcal{T}^{\beta\mu}$.

Proposition 4.3

The fixed point of $\mathcal{T}^{\mu}_{\mathcal{O}}$ is the same as the fixed point of the operator $\mathcal{T}^{(I-\beta)\iota} + \mathcal{T}^{\beta\mu}$, and writes:

$$q^{\mu,\iota}_{\beta} = (I - \gamma(\mathcal{P}^{\beta\mu} - \mathcal{P}^{\beta\iota}) - \gamma \mathcal{P}^{I\iota})^{-1} r^{\pi}. \tag{4.19}$$

Thus, the termination scheme directly affects the convergence limit: in the extreme, if $\beta = \mathbf{0}$, options never terminate, and we have the fixed point of $\mathcal{T}^{I\iota}$: $q^{\mu,\iota}_{\mathbf{0}}(s,o) = v^{\pi^o}(s)$, the value of the option o. In the other extreme, $\beta = \mathbf{1}$, the options terminate at every step and we have the fixed point of $\mathcal{T}^{I\mu}$, which can be shown to correspond to the value of the marginal policy κ from Eq. (4.5) (Prop 4.1):

$$q^{\mu,\iota}_{\mathbf{1}}(s,o) = v^{\kappa}(s), \forall o \in \mathcal{O}. \tag{4.20}$$

Comparison with the Gated Model

Before we proceed, let us briefly highlight the differences with the setting in the previous section. Unlike the gated model, in which β transitioned between value and policy iteration, in call-and-return we get a hybrid of value iteration and policy *evaluation*. First, consider the λ-PI evaluation operator w.r.t. action-values q:

$$\mathcal{T}^{\pi}_{\lambda} q = (I - \gamma \mathcal{P}^{\lambda\pi})^{-1} (r + \gamma(\mathcal{P}^{\pi} - \mathcal{P}^{\lambda\pi}) q), \tag{4.21}$$

where again we write $\mathcal{P}^{\lambda\pi} = \lambda \mathcal{P}^{\pi}$ by an analogy with (4.17). The equations (4.18) and (4.21) are still similar, but the key difference is the discrepancy between the policies ι and μ. Since option continuation is non-myopic:

$$\mathcal{P}^{\beta\mu} = \mathcal{P}^{\mu} - \mathcal{P}^{(1-\beta)\mu} \neq \mathcal{P}^{\mu} - \mathcal{P}^{(1-\beta)\iota},$$

we achieve the fixed point from Prop. 4.3, instead of that of Prop. 4.1. This discrepancy introduces *off-policy-ness* on the primitive action level, which we will discuss in detail further. Similarly, in control, the operator

$$\mathcal{M}_k q = r + \gamma \mathcal{P}^{\beta\mu_k} q_k + \gamma \mathcal{P}^{(1-\beta)\iota} q,$$

replaces the mapping $M_k v = r^{\kappa_k} + \gamma p^{\beta\kappa_k} v_k + \gamma p^{(1-\beta)\kappa_k} v$. The new operator $\mathcal{M}_k$ transition between value iteration $r + \gamma \mathcal{P}^{\beta\mu_k} q_k$ and policy *evaluation* $r + \gamma \mathcal{P}^{(1-\beta)\iota} q$, which just evaluates the current option policy π^o. At the primitive action level, this is crucial, since longer options no longer imply more progress towards the solution, if π^o is not aligned with it.

4.4.2 Off-Policy Option Termination

We would like to decouple the *behavior* termination condition ζ (*"zeta"*) that governs for how long the options are followed from the *target* termination condition β that factors into the solution. Apart from the theoretical appeal of the freedom that this allows, a key motivation is the fact that on the one hand just like with multi-step returns, the less options terminate the faster the convergence, but on the other the more options terminate, the better the control solution (as we show formally in the next section). One possibility to address this is to pick an intermediate behavior termination condition that trades these off. If we are able to decouple the two, however, then we in fact achieve the best of both worlds, which is exactly what we propose to do in this chapter.

The critical insight in our approach is the *off-policy-ness* at the primitive action level that is introduced by the discrepancy between policies μ (that picks a new option) and ι (that maintains the current option) in Eq. (4.19). The *degree* of this off-policy-ness is modulated exactly by the termination condition β. In a nutshell, we propose to leverage multi-step off-policy learning and correct *some* of the off-policy-ness, with the extent to which we choose to do so determining the target termination condition ζ. In the extreme, we can learn the marginal policy κ directly. In the other extreme, we can learn the value of the current option. The two extremes are traded off via ζ. This algorithm is analogous to the unifying algorithm Q(σ) in which σ modulates the degree of off-policy-ness [De Asis et al. 2018, Sutton and Barto 2017].

This section will incrementally build intuition and formulate the proposed *off-policy termination* operator, while the next will analyze its convergence.

Unifying Operator and Algorithm

Before we begin, we remind the reader of the general form of the operator (4.22) underlying several off-policy return-based algorithms from the previous chapter:

$$\mathcal{U}q(s,a) \stackrel{\text{def}}{=} q(s,a) + \mathbb{E}_\mu\left[\sum_{t\geq 0}\gamma^t\Big(\prod_{i=1}^{t}c_i\Big)\delta_t\right], \tag{4.22}$$

Recently, [De Asis et al. 2018, Sutton and Barto 2017] formulated an algorithm that can be expressed in a similar form, and unifies existing algorithms even more generally. Instead of the binary taxonomy of on-and off-policy algorithms, the parameter σ smoothly transitions between the two, for the following forms of δ_t and c_i:

$$\delta_t = R_{t+1} + \gamma\left(\sigma_i q(S_{t+1},A_{t+1}) + (1-\sigma_i)\mathbb{E}_\pi q(S_{t+1},\cdot)\right) - q(S_t,A_t), \tag{4.23}$$
$$c_i = (1-\sigma_i)\pi_b(S_i,A_i) + \sigma_i, \tag{4.24}$$

where as usual $R_{t+1} \sim r(S_t,A_t)$. In particular, $\sigma = 1$ corresponds to the on-policy SARSA(0) algorithm, while $\sigma = 0$ to Tree-Backup(0). In the following we will begin from the standard intra-option equations, and using the general form of Eq. (4.22), arrive at an analogous update to Eq. (4.23)-(4.24).

From one step intra-option learning to General Q(λ)

To begin, let us first re-derive the target from Proposition 4.3 starting from the familiar intra-option equations [Sutton and Precup 1998]. Letting Δ denote the update on the estimated Q-function, we have at time t and the current option o:

$$\Delta q(S_t,o) \sim R_{t+1} + \gamma\tilde{q}(S_{t+1},o) - q(S_t,o), \tag{4.25}$$
$$\tilde{q}(s,o) = (1-\beta^o(s))q(s,o) + \beta^o(s)\mathbb{E}_\mu q(s,\cdot),$$

where as before we write $\mathbb{E}_\mu q(s,\cdot) \stackrel{\text{def}}{=} \sum_o \mu(o|s)q(s,o)$. Notice that this is exactly a sample of the one-step update corresponding to $\mathcal{T}^{(I-\beta)\iota} + \mathcal{T}^{\beta\mu}$. In fact, if we roll it out over multiple steps we obtain Eq. (4.18) exactly:

$$q(s,o) = \mathbb{E}_{\pi^o}\Big[\sum_{t=0}^{\infty}\gamma^t\Big(\prod_{i=1}^{t}(1-\beta^o(S_i))\Big)[R_{t+1} + \gamma\beta^o(S_{t+1})\mathbb{E}_\mu q(S_{t+1},\cdot)]\Big], \tag{4.26}$$

from which some simple algebra yields:

$$\Delta q(s,o) = r^{\pi_o}(s) + \gamma\mathbb{E}_\mu q(S_1,\cdot) - q(s,o) + \mathbb{E}_{\pi^o}\Big[\sum_{t=1}^{\infty}\gamma^t\Big(\prod_{i=1}^{t}(1-\beta^o(S_i))\Big)\delta_t^{\mu\mu}\Big], \tag{4.27}$$

$$\delta_t^{\mu\mu} = R_{t+1} + \gamma \mathbb{E}_\mu q(S_{t+1}, \cdot) - \mathbb{E}_\mu q(S_t, \cdot).$$

This is the same update as Peng's Q(λ) for greedy policies [Peng and Williams 1996] or General Q(λ) (i.e. off-policy Expected SARSA(λ) [van Seijen et al. 2009]) for arbitrary μ, with $1 - \beta^o(S_i)$ being the state-option analogue of λ from those algorithms. The fixed point of both of those algorithms is in fact derived in Proposition 3.1 from the previous chapter and is indeed analogous to that given in Proposition 4.3 in this chapter.

General Q(λ) to Tree-Backup(λ)

Starting from the multi-step intra-option update, we wish to cast it in the form of the general off-policy operator from Eq. (4.22). We do this by replacing the second expectation in the TD-error with the point-estimate $q(S_t, o)$, which introduces *off-policy corrections* from the previous chapter. This gives us Eq. (4.27), but with δ_t^μ instead of $\delta_t^{\mu\mu}$:

$$\delta_t^\mu = R_{t+1} + \gamma \mathbb{E}_\mu q(S_{t+1}, \cdot) - q(S_t, o).$$

If we further augment the "trace" $1 - \beta^o(S_i)$ with the policy probability coefficient $\mu(o|S_i)$, we obtain option-level Tree-Backup(λ) [Precup et al. 2000], whose target policy is μ, and behavior policy is ι:

$$\Delta q(s, o) = \mathbb{E}_{\pi^o}\Big[\sum_{t=0}^{\infty} \gamma^t \Big(\prod_{i=1}^{t} c_i^o\Big) \delta_t^\mu | S_0 = s\Big], \tag{4.28}$$
$$c_i^o = \mu(o|S_i)(1 - \beta^o(S_i)).$$

From the convergence guarantees of Tree-Backup, we know that this update converges to the fixed point of $\mathcal{T}^{I\mu}$, which in turn corresponds to q^κ (Eq. (4.20)).

The off-policy termination operator

We are now ready to present the operator underlying the Q(β) algorithm, which is the key contribution of this chapter. Eq. (4.28) can be considered a special case where we correct *all* of the off-policy-ness, thus implicitly assuming $\zeta = \mathbf{1}$. To get the general case, we need to split the target in each TD-error into two terms, that are weighted by $\zeta^o(S_t)$ or $(1 - \zeta^o(S_t))$:

$$\mathcal{R}_{\zeta,\beta}^\mu q(s, o) = q(s, o) + \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t} c_i^o\Big) \delta_t^{\zeta,\mu}\Big], \tag{4.29}$$
$$\delta_t^{\zeta,\mu} = R_{t+1} + \gamma \tilde{q}(S_{t+1}, o) - q(S_t, o),$$

Algorithm 3 Q(β) algorithm

Given: Option set $\mathcal{O}$, target termination function ζ, initial Q-function q_0, step-sizes $(\alpha_k)_{k\in\mathbb{N}}$

for $k = 0, 1, \ldots$ **do**
  Sample an option o from μ_k
  Sample the return $S_0, R_1, S_1, R_2 \ldots, S_{D_k}$ from π^o, D_k is determined by sampling $1 - \beta^o$.
  for $t = 0, 1, \ldots D_k - 1$ **do**
    $q_{k+1}(S_t, o) \leftarrow q_k(S_t, o) + \alpha_k \Delta_t$
    $\Delta_t = \sum_{i=t}^{D_k-1} \gamma^{i-t} \Big(\prod_{j=t+1}^{i} c_j^o \Big) \delta_t^{\zeta,\mu_k}$
    $\delta_t^{\zeta,\mu_k} = R_{t+1} + \gamma \tilde{q}_{\mu_k}(S_{t+1}, o) - q(S_t, o)$
    $\tilde{q}_{\mu_k}(s, o) = [1 - \zeta^o(s)] q(s, o) + \zeta^o(s) \mathbb{E}_{\mu_k} q(s, \cdot)$
    $c_j^o = ([1 - \zeta^o(S_j)] + \zeta^o(S_j)\mu(o|S_j))\,.$
  end for
end for

$$\tilde{q}(s, o) = [1 - \zeta^o(s)] q(s, o) + \zeta^o(s) \mathbb{E}_\mu q(s, \cdot),$$
$$c_i^o = ([1 - \zeta^o(S_i)] + \zeta^o(S_i)\mu(o|S_i))\,(1 - \beta^o(S_i)).$$

We will drop the β superscript, and simply write $\mathcal{R}_\zeta^\mu$ when β is clear from context, or not relevant. Note that the first factor in c_i^o is explicit, while $1 - \beta^o(S_i)$ is sampled from the current option during learning. Algorithm 3 presents the forward view of this algorithm for the general case of an evolving policy μ_k.

This algorithm is a very similar to the recently formalized Q(σ) [Sutton and Barto 2017, De Asis et al. 2018], in which σ controls the degree of off-policy-ness. There, $\sigma = 1$ corresponds to SARSA, and $\sigma = 0$ to Tree-Backup. The parameter ζ is the state-option generalization of $1 - \sigma$, with $\zeta = \mathbf{0}$ learning the value of the current option o (i.e. the policy ι), and $\zeta = \mathbf{1}$ the value of the marginal policy κ (i.e. the policy μ). The behavior termination β on the other hand has a role analogous to that of the eligibility trace parameter λ.

Relationship with intra-option learning

The off-policy-ness discussed so far is subtly different than that in the more familiar off-policy intra-option setting. The intra-option learning algorithm suggests applying the update (4.25) (in its one-step form) to all options o "consistent with" the experience stream $S_1, A_1, \ldots$ [Sutton and Precup 1998]. For stochastic policies this amounts to

applying importance sampling to the update. That is, given a behavior option b, the trace coefficient $1 - \beta^o(S_i)$ from Eq. (4.25) now becomes

$$c_i^{ob} = \frac{\pi^o(A_i|S_i)}{\pi^b(A_i|S_i)}(1 - \beta^o(S_i)). \tag{4.30}$$

Writing β^o_{t+1} for $\beta^o(S_{t+1})$, the value for option o writes:

$$q(s,o) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi^b}\Big[\Big(\prod_{i=1}^{t} c_i^{ob}\Big)[R_{t+1} + \gamma\beta^o_{t+1}\mathbb{E}_\mu q(S_{t+1},\cdot)]\Big]$$

Now, notice that there are two sources of off-policy-ness in these formulas. One is π^b vs. π^o, the contrast between option policies, and the other is in the target: ι vs. μ itself. Indeed if we write the above in a form from Eq. (4.27) we get a different correction:

$$\Delta q(s,o) = r^{\pi_o}(s) + \gamma\mathbb{E}_\mu q(S_{t+1},\cdot) - q(s,o) + \mathbb{E}_{\pi^b}\Big[\sum_{t=1}^{\infty}\gamma^t\Big(\prod_{i=1}^{t-1} c_i^{ob}\Big)(1-\beta^o(S_t))\delta_t^{ob}\Big],$$

$$\delta_t^{ob} = \frac{\pi^o(A_t|S_t)}{\pi^b(A_t|S_t)}[R_{t+1} + \gamma\mathbb{E}_\mu q(S_{t+1},\cdot)] - \mathbb{E}_\mu q(S_t,\cdot).$$

Since the corrections for the two sources of off-policy-ness are orthogonal, it could be possible to combine them. We leave this for the future.

4.4.3 Analysis

In this section we will analyze the convergence behavior of the off-policy termination operator in both policy evaluation and control, and show that learning about shorter target options off-policy is generally asymptotically more efficient than on-policy. We will then consider the relationship of the solution quality in control with option duration, and show that shorter options generally yield better solutions.

We will prove that the evaluation operator $\mathcal{R}^\mu_\zeta$ is contractive around the appropriate fixed point, and that its contraction factor is less than that of the respective *on*-policy operator, if the target options terminate more than the behavior ones.

Theorem 4.2: Policy evaluation

The operator $\mathcal{R}^\mu_\zeta$ defined in Eq. (4.29) has a unique fixed point $q^{\mu,\iota}_\zeta$, as defined in Eq. (4.19). Furthermore, if for each state $S_i \in \mathcal{S}$ and option $o \in \mathcal{O}$ we have

$c_i^o \leq (1 - \zeta^o(S_i)) + \zeta^o(S_i)\mu(o|S_i)$, then for any Q-function q:

$$|\mathcal{R}_\zeta^\mu q(s,o) - q_\zeta^{\mu,\iota}(s,o)| \leq \eta(s,o)\|q(s,o) - q_\zeta^{\mu,\iota}(s,o)\|,$$

where $\eta(s,o) \stackrel{\text{def}}{=} 1 - (1-\gamma)\mathbb{E}_{\pi^o}\Big[\sum_{t=0}^{\infty}\gamma^t\Big(\prod_{i=1}^{t} c_i^o\Big)\Big] \leq \gamma$.

Proof. The proof is analogous to that of Theorem 1 from [Munos et al. 2016] and is given in appendix. □

The contraction coefficient η controls the convergence speed of this operator: the smaller η (and the larger $\prod_{i=1}^{t} c_i^o$) the fewer iterations are needed to converge, but the larger the computational expense when planning, or the variance when learning [Bertsekas and Ioffe 1996, Munos et al. 2016]. Since $c_i^o \leq 1$, and options terminate eventually, the variance is less significant here, and generally, larger c_i^o will yield faster convergence. In our case, since a behavior option is assumed (i.e. the $(1 - \beta_i^o)$ factor in Eq. (4.29) is fixed), the additional ζ-term in c_i^o can only reduce the existing trace. However, since we are interested in learning about a different target, we ought to compare traces with the setting when that target is learnt *on*-policy. The following corollary derives the condition when Q(β) maintains larger traces than its on-policy counterpart.

Corollary 4.1

The convergence of the iteration corresponding to the operator $\mathcal{R}_{\zeta,\beta}^\mu$ defined in Eq. (4.29) is faster than that of its on-policy counterpart $\mathcal{R}_{\zeta,\zeta}^\mu$, if

$$\zeta^o(s) \geq 1 - \frac{\mu(o|s)(1-\beta^o(s))}{\mu(o|s)(1-\beta^o(s)) + \beta^o(s)}.$$

In particular $\beta^o(s) = 0$, any $\zeta^o(s) > 0$ satisfies this, irrespective of μ. If μ is deterministic, this holds for all $\zeta^o(s) > \beta^o(s)$ for the chosen o. In general, the intuition here is that it's easier to learn from longer option traces about shorter options than vice versa.

Let us now formulate the control analogue of Theorem 4.2. Note that Assumption 4.1 is no longer required. As before we consider convergence to the best policy *expressible by options*: $q_{\mathcal{O}}^* = \max_\mu q_\zeta^{\mu,\iota}$.

Theorem 4.3: Control

Consider a sequence of policies over options $(\mu_k)_{k\in\mathbb{N}}$ that are greedy w.r.t. the sequence of estimates $(q_k)_{k\in\mathbb{N}}$, and consider the update:

$$q_{k+1} = \mathcal{R}_\zeta^{\mu_k} q_k,$$

where the operator $\mathcal{R}_\zeta^{\mu_k}$ is defined by Eq. (4.29) for the k-th policy μ_k. Let $\mathcal{T}_\zeta^{\mu,\iota} q \stackrel{\text{def}}{=} \mathcal{T}^{(I-\zeta)\iota} q + \mathcal{T}^{\zeta\mu} q$. Suppose that $\mathcal{T}_\zeta^{\mu_0,\iota} q_0 \geq q_0$. Then for any $k \geq 0$,

$$\|q_{k+1} - q_{\mathcal{O}}^*\| \leq \gamma \|q_k - q_{\mathcal{O}}^*\|.$$

It follows that $q_k \to q_{\mathcal{O}}^*$ as $t \to \infty$.

Proof. The proof is a simpler version of that of Theorem 2 from [Munos et al. 2016], and we omit it here. □

We do not give a proof of online convergence at this time, but verify it empirically in our experiments.

Option duration and solution quality

Our convergence results show that learning about shorter options off-policy is more efficient than on-policy. Here, we motivate why one would want to learn about shorter options in the first place. In particular, we will show that given a set of options, the more they terminate, the better the resulting control solution at the primitive action resolution. Intuitively, this is because upon termination, the learner picks the current best option, whereas during option execution, the target value includes the potentially suboptimal current option (Proposition 4.3). The following theorem formalizes this intuition (proof in appendix), and may be of independent interest.

Theorem 4.4: The more options terminate, the better the solution.

Given a set of options $\mathcal{O}$, and a greedy policy over options μ. Let $\zeta \geq \beta$ be two termination conditions for the options in $\mathcal{O}$. Then: $q_\zeta^{\mu,\iota} \geq q_\beta^{\mu,\iota}$.

Note that this result refers to the target solution. During learning, the more decisions there is to make, the more potential there is for error. As such, in reasonably complex tasks we expect the performance to obey a tradeoff on ζ.

4.4.4 Experiments

Finally, let us evaluate our algorithm empirically. We aim to illustrate the following claims:

- The learning speed improves as β gets smaller.
- The control performance improves, as ζ gets larger.
- Q(β) converges with off-policy terminations

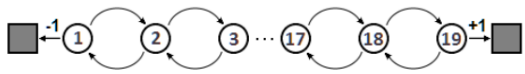

Figure 4.3: The 19-state random walk task. The agent starts in the middle. Transitions are deterministic, and the task terminates in each end.

For simplicity, we assume that options terminate deterministically in a set of goal states.[6] We hence reduce ζ and β to single parameters that determine the likelihood of terminating before reaching the goal. The "plain" variant refers to the on-policy intra-option update from Eq. (4.26).

Policy evaluation

First, we show that Q(β) learns the correct values on the 19-state random walk task (Figure 4.3). There are two options, one leads all the way to the left, the other to the right. The policy over options is uniform. The task is to estimate the value function w.r.t. target terminations ζ. The results are given in Figure 4.5. Q(β) is able to learn the correct values (up to an irreducible exploration-related error). As expected, Q(β) gets more efficient as behavior options get longer (β gets smaller). The opposite is true for the plain algorithm, since without interrupting sufficiently, it does not have a chance to update the intermediate states with anything other than the option policy. There is a small inflection point in the performance of the plain algorithm at the *on-* policy value of β.

Control

To demonstrate the benefit of decoupled off- and on- policy terminations, we compare our algorithm with the plain on-policy variant (labelled: *onpolicy-plain*) that uses $\beta = \zeta$

[6] Note the difference with the setting in the first part of the chapter, where we considered probabilistic goal terminations. In the learning call-and-return setting, where terminations are sampled and a new option is not chosen until the current one has terminated, probabilistic goal terminations simply cause the agent to unproductively jitter around the goal region.

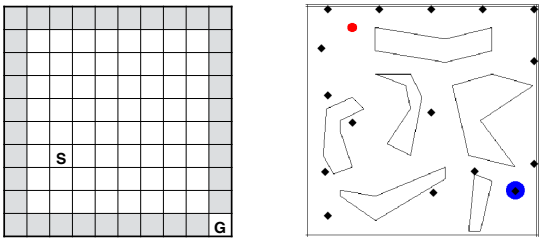

Figure 4.4: **Left:** The modified cliffwalk task. Shaded regions are cliffs. **Right:** Pinball domain configuration used. The red ball must be moved to the blue hole. Each black diamond indicates an option landmark.

during both learning and evaluation. In order to demonstrate that the learning target plays a role, we also compare it with the plain algorithm that uses ζ at evaluation only (labelled: *offpolicy-plain*). The behavior $\beta = 0$, unless specified otherwise.

Modified Cliffwalk We illustrate the benefits of off-policy termination on a modified Cliffwalk example. The agent starts in a position inside a $n \times n$ grid with the goal of getting to a corner where a positive reward is given. The step reward is zero, but there are small cliffs along the border that aren't fatal, but induce a penalty. We have four options, one for each cardinal direction, that take the agent up until the corresponding border (and cliff). Thus, while these options are able to learn to reach the goal in an optimal number of *steps*, they are unable to learn the optimal policy which only moves inside the grid, so as to not encounter cliffs. To ensure adequate exploration we consider ϵ_{opt}-soft option policies, as well as a usual ϵ-greedy policy over options during learning (but not during evaluation). The results are plotted in Fig. 4.6. Q(β) outperforms the alternatives in all cases, and its performance improves with larger ζ. Note that is the only variant able to surpass the value of the suboptimal policy.

Pinball We finally evaluate our algorithm on a variation of the Pinball domain [Konidaris and Barto 2009]. Here, a small ball must be maneuvered through a set of obstacles into a hole. Observations consist of 4 continuous variables describing the ball's x, y positions and velocities. There are 5 primitive actions: the first 4 actions apply a small force to either the x or y velocity, the final action leaves all velocities unchanged. There is a step penalty and a final reward. We define a set of *landmark* options [Mann et al. 2015] that move the ball near a target goal location on the board. The agent can initiate and terminate each option from within some initiation and termination distances from the respective landmark. To

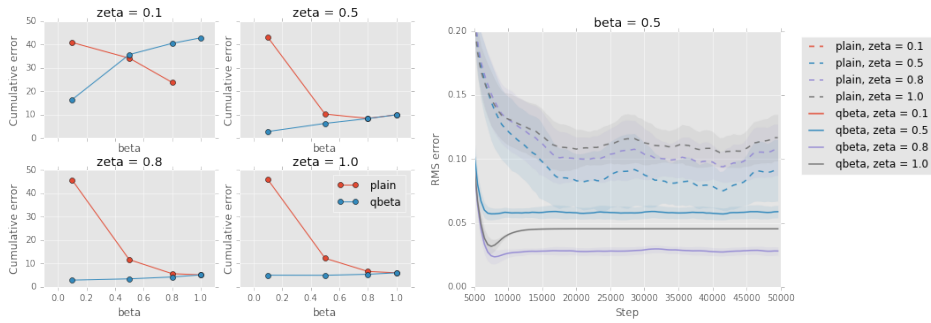

Figure 4.5: Prediction error on the 19-state chain task. Each variant is an average of 10 seeds. **Left:** Sum error for each β-ζ combination. Q(β) always gets more efficient as β decreases (the options get longer). **Right:** Example learning curves. The lines corresponding to $\zeta = 0.1$ are outside the axes' bounds. The shaded region covers standard deviation.

illustrate the benefits of terminating suboptimal options, landmarks were placed in such a way that the paths from start to the goal via the landmarks are suboptimal. The results are plotted in Fig. 4.7. As expected, the performance of Q(β) improves with longer options. The target solution on the other hand is best for intermediate ζ-s. In a comparison, Q(β) outperforms the on-policy variant that learns with $\beta = \zeta$. However, in this domain, the off-policy variant (that learns with $\beta = 0$, but evaluates with ζ) performs comparably to Q(β). This may in part be due to the use of function approximation, whose generalization allows the plain target to update meaningfully within the option trajectory, and in part due to the noisy nature of Pinball, in which there are many optimal policies of similar values. Since, as we have seen, Q(β) is the only variant to learn accurate values, we expect it to stand out more in settings where the reward scheme is more intricate.

4.4.5 Details

The setting is as follows: an option o is picked according to μ, and a trajectory $s, R_1, S_2, R_2, \ldots, R_{D-1}, S_D$ is generated according to π^o and β^o. Then for each state S_i in the trajectory $q(S_i, o)$ is updated according to the considered algorithm.

19-chain ζ and β are evaluated in the range of $\{0.1, 0.5, 0.8, 1\}$, and the step-sizes set via a linear search over $\alpha \in \{0.1, 0.2, 0.3, 0.4\}$. The termination conditions ζ and β are

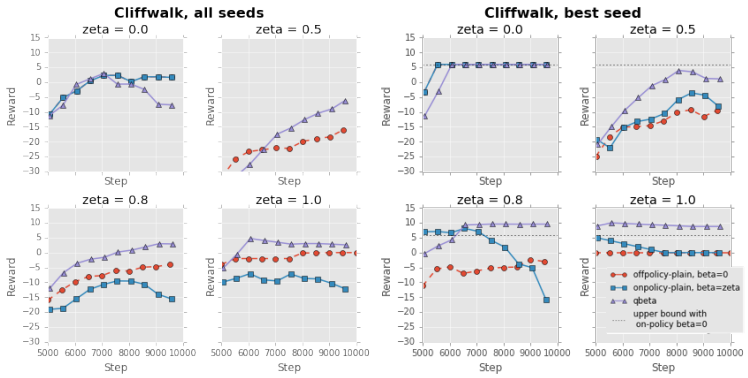

Figure 4.6: Control performance on Cliffwalk. Each variant is evaluated on 5 seeds for 10 runs each. *Left:* Average performance per value of ζ on all seeds. *Right:* Learning curves for the best seeds per variant. Notice how Q(β) is the only variant that escapes the plateau of the suboptimal policy.

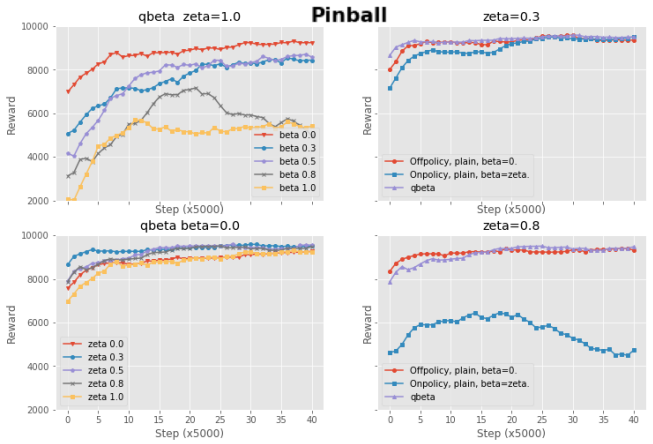

Figure 4.7: Control performance on Pinball. Each variant is evaluated on 20 independent runs. *Left:* Influence of ζ and β on Q(β): performance improves as β gets larger; intermediate target ζ-s are best. *Right:* Comparison within the variants.

evaluated in the range of $\{0.1, 0.5, 0.8, 1\}$, with the first value being positive to ensure adequate state visitation. The discount factor $\gamma = 0.99$.

Modified Cliffwalk The reward scheme is: $r_{goal} = 10$ and $r_{cliff} = -2$, and the grid size $n = 10$. We set $\epsilon = 0.1$, and $\epsilon_{opt} = 0.3$, and determine the step-size for each variant from a linear search over $\alpha \in \{0.1, 0.2, 0.3, 0.4\}$, behavior $\beta = 0$, and target ζ is evaluated on the range of $\{0, 0.5, 0.8, 1\}$. The discount factor $\gamma = 0.99$.

Pinball The reward is -1 on every step, except the final step which receives a reward of 10000. We use initiation distance of 0.3 and termination distance of 0.03. The state option value function was approximated using tile coding with 16, 10 by 10 tilings. All algorithms used a learning rate $\alpha = 0.01$, discount $\gamma = 0.99$ and an exploration rate $\epsilon = 0.05$ and $\epsilon_{opt} = 0.01$ during learning. The target ζ is evaluated on the range of $\{0, 0.3, 0.5, 0.8, 1\}$.

4.4.6 Discussion

We propose decoupling behavior and target termination conditions, like it is done with policies in off-policy learning. We formulate an algorithm for learning target terminations off-policy, analyze its expected convergence, and validate it empirically, confirming the theoretical intuition that learning shorter options from longer options is beneficial both computationally and qualitatively. More generally, we cast learning with options into a common framework with well-studied multi-step off-policy temporal difference learning, which allows us to carry over existing results with ease.

Learning longer options from shorter options. We have assumed here that the options are given, but may not express the optimal policy well. This scenario applies when the options describe simple rules of thumb, or are transferred from a different task. If the options are not given, but learnt end-to-end, our wish typically is to distill meaningful behavior in them. However, instead, the result often ends up reducing to degenerate options [Bacon et al. 2017, Mann et al. 2014]. Being able to impose longer durations on the target off-policy may mitigate this. Though it should be noted that our convergence results suggest that learning may not be as efficient then.

Action-level importance sampling. The option policy term in the trace is ambivalent to the action choice. Thus if $\mu(o|S_i)$ is small, c_i^o will be small, even if the taken action is consistent with the option policy π^o. It would be interesting to replace this term with the importance sampling ratio at the primitive action level, like $\frac{\kappa(A_i|S_i)}{\pi^o(A_i|S_i)}$, which corresponds to

another multi-step off-policy algorithm, Retrace(λ) [Munos et al. 2016]. Another direction towards this goal is to incorporate the intra-option correction from Eq. (4.30).

Limitations. Proving online convergence of Q(β) in the control setting remains an open problem. The technical issue is that a certain matrix in the proof of Theorem 4.3 is not a contraction unless an additional assumption holds, such as the asymptotic commutativity assumption 3.2 from Chapter 3 which unfortunately does not apply in this case. Supported by reliable empirical behavior, we hypothesize that convergence holds under reasonable conditions and plan to investigate it further in the future.

4.5 Related Work

Much of the related work has already been discussed throughout. We mention a few more relevant works below. In general, the analysis in our work is related to that in λ-operator literature [Bertsekas and Ioffe 1996, Munos et al. 2016, Scherrer 2013], while the intuitions to options literature [Mann et al. 2015, Bacon and Precup 2015, Bacon and Precup 2016].

[Bacon and Precup 2016] analyze the policy evaluation setting, derive Proposition 4.1 and show that the options operator (4.18) induces a matrix splitting. In an earlier work, [Bacon and Precup 2015] argue that the main computational expense when planning, is the *deliberation* of choosing an option, and point out that more terminations incur a higher rate of deliberation, which is computationally expensive. Off-policy terminations allow to lower the cost of deliberation during learning (which is when the bulk of computation happens).

[Mann et al. 2015] give concentrability coefficients for convergence of approximate value iteration with options. Their Section 3.2 conveys intuitions similar to the ones found here, but does so from the semi-MDP view. While they also consider option durations, they do not express them in terms of the termination function. It would be interesting to reconcile these two lines of analysis.

[Yu and Bertsekas 2012] consider general λ-operators with state-dependent λ, with Section 5.1.1 specifically discussing λ-PI. In the case of options, $1 - \beta^o$ takes the role of a state-option-dependent λ. [White 2017] proposed to consider $1 - \beta^o$ as part of the transition-based *discount* instead.

[Mann et al. 2014] propose an algorithm for multi-step option interruption that stems from the same motivation of mitigating poor-quality options. In order to avoid the resulting options being too short, they introduce a time-regularization term. Our approach bypasses the need to do so by interrupting off-policy and the ability to explicitly specify target terminations.

4.6 Summary

In this chapter, we have drawn parallels between planning and learning with options and multi-step temporal differences, and leveraged them to gain insights about the qualitative and quantitative influence of the option termination condition β.

We showed that planning with options in the gated model corresponds to λ-policy iteration, and gave new convergence results in terms of the termination condition parameter. Our analysis and experiments suggest that intermediate values of β perform best, and in general β should be treated as a parameter.

In call-and-return, where the termination condition is coupled with the solution directly, there is an additional tradeoff imposed by the relationship of β with the quality of the solution. Using the intuition that the underlying algorithms are inherently *off-policy* at the primitive action level, we suggest a new algorithm that escapes that tradeoff by learning the solution w.r.t. *any* termination condition, irrespective of the one used for behavior. We demonstrate the benefits of doing so empirically.

5 | Discounting Options

The previous chapter provided insights into what the effects of longer multi-step actions, or *options*, really are. In this chapter, we take that question a level deeper, and step outside the standard options framework. Sidestepping the usual assumption that options live on a shared clock, we ask: what happens when each option decides its own timescale? Is option duration sometimes irrelevant, so long as its goal is achieved? This chapter analyzes the answers to these questions.

5.1 Introduction

The discount factor γ in reinforcement learning is traditionally treated as something in between a mathematical convenience and a meaningful time horizon parameter. Indeed, the instrumental variable in learning: the discounted return

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

is roughly capable of representing a *horizon* of $\frac{1}{1-\gamma}$ steps, while the convergence speed is strongly related to that same quantity. This makes it challenging to learn about very distant goals in a feasible amount of time. While there have been important generalizations of the naive constant γ to state-, state-action, or transition-based matrices [Yu 2015, White 2017], little has been done in the way of concrete instantiations. In this chapter, we propose a mechanism for extending the agent's horizon by tying discounting in with temporal abstraction, in particular the options framework [Sutton et al. 1999].

Options induce reward and transition models that act as higher-order analogues of those from the primitive MDP. The discount factor is an integral part of these models – indeed the previous chapter drew the clear parallels between options and multi-step temporal difference methods. As such, an option that takes a hundred steps incurs an associated factor of γ^{100}, and hence, regardless of the sophistication of our options, if they are long, we remain at the mercy of the step-discount capturing the necessary horizon. On the one hand, this ensures the consistency of options with one-step methods, on the other: the fact that the agent's horizon is ambivalent to the use of options is highly unsatisfying, as temporal abstraction offers no additional temporal representation power.

In this chapter, we propose to address this by generalizing the options framework to allow for *time dilation*. In particular (1) we allow the step-discount in the option transition model to be independent of that in the environment, and (2) so as to mitigate the resulting bias, we introduce an option-level discounting matrix Γ that augments the transition model irrespectively of option duration. This simple generalization allows for options to extend the agent's horizon, while preserving desirable convergence properties.

We analyze the properties of planning with such time-dilated options and devise novel bias-variance bounds that apply to the classical framework, as a special case. In particular, we show that *larger* step-discounts in the transition model actually *reduce* the variance of the estimated solution, which is contrary to the familiar intuitions about e.g. multi-step returns. This is the case in particular due to the variance incurred by the random duration of an option. Notably, we verify the shape of the bounds empirically on a classical problem.

Time dilation provides a vehicle for extending the agent's horizon proportionally to the options it has available, and gives options the power to "abstract away" time. One simple illustration of where this is crucial is a setting of multiple versions of the same task of varying sizes, whose solution over options is the same, but where option length reflects the size of the task. Time dilation allows for the same policy over options to be learnt irrespectively of the size of the task, while any "flat" discount will fail to capture this policy for some size of the task. We discuss this in more detail in Section 5.3.1, and illustrate it with an empirical example in Section 5.6.2.

5.2 Notation and Setting

Throughout, we parameterize the models and operators with by the associated discount:

$$P^o_\gamma(s'|s) \stackrel{\text{def}}{=} \mathbb{E}_{D:s\to s'|o}\left[\gamma^D\right] \tag{5.1}$$

$$R^o_\gamma(s) \stackrel{\text{def}}{=} \mathbb{E}_{D:s|o}\left[\sum_{i=1}^{D}\gamma^{i-1}r^{\pi^o}(S_{t+i})|S_t=s\right] \tag{5.2}$$

$$\mathcal{P}^{\mu}_{\mathfrak{O}_\gamma} q(s,o) \stackrel{\text{def}}{=} \sum_{s'} P^o_\gamma(s,s') \sum_{o'} \mu(o'|s') q(s',o') \tag{5.3}$$

$$\mathcal{T}^{\mu}_{\mathfrak{O}_\gamma} q(s,o) \stackrel{\text{def}}{=} R^o_\gamma(s) + \mathcal{P}^{\mu}_{\mathfrak{O}_\gamma} q(s,o). \tag{5.4}$$

where, as before, $\mathbb{E}_{D:s|o}[\cdot]$ and $\mathbb{E}_{D:s\to s'|o}[\cdot]$ are the expectations of the option duration D from state s and the travel time between state s and s', respectively, w.r.t. option dynamics p^{π^o} and the termination condition β^o.

We distinguish the MDP discount factor by γ_{env}. We will occasionally consider settings where the agent seeks to optimize policies over a very long, nearly undiscounted horizons of $\gamma_{eval} >> \gamma_{env}$. We say that a discount γ is able to *represent* a policy w.r.t. $\gamma' > \gamma$ if $\pi^*_\gamma = \pi^*_{\gamma'}$.

5.3 Discounting Options

Now let us motivate and formalize introducing time dilation into the options framework. We first discuss in further detail the effect that the discount has on the policy learnt by an agent, and how it persists despite the use of options. We then describe time dilation and discuss how it alleviates the issue.

5.3.1 Horizon Length and Discounting

Let us leave the options framework for a moment, and consider the following example. The agent needs to choose between a closer, worse goal with a reward of z and a farther, better goal with a reward of $Z > z$.[1] Now, consider the role of the discount factor γ_{env} in this decision. In order for the agent to pick the higher reward, it would need

$$\gamma^D_{env} Z > \gamma^d_{env} z,$$

where D and d are the distances from Z and z to the agent's location. Thus, there is a minimum value of

$$\gamma_{env} > \left(\frac{z}{Z}\right)^{\frac{1}{D-d}} \tag{5.5}$$

required to represent the optimal policy $\pi^*_{\gamma_{eval}}$. See Figure 5.1 for an illustration.

Now, consider the same task, but abstracted. Instead of primitive actions, we have d and D *options* separating the agent from the respective goals, and so on a trajectory

[1] Assume the step reward to be zero, and the environment to be deterministic.

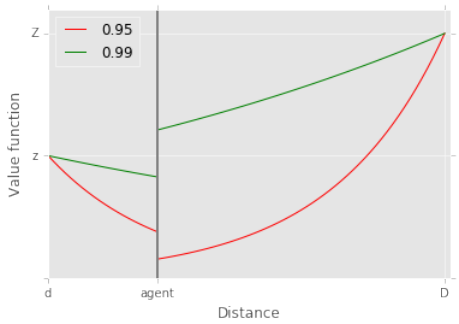

Figure 5.1: The agent needs to choose between a closer, worse goal with a reward of z and a farther, better goal with a reward of $Z > z$. The lines represent the values of the left and right actions split at the agent's location and w.r.t. two different discounts. The outcome of the agent's choice in the current location thus depends on the discount: the red discounting scheme of $\gamma_{env} = 0.95$ is too short-sighted to prefer the correct goal Z. Note that for any discount $\gamma_{env} < 1$, the distances d and D can be proportionally increased (to $d + K$ and $D + K$ for some $K < \infty$) for γ_{env} to be insufficient to capture the correct ordering of the goals.

$S_0, S_1, \ldots$, instead of γ_{env}, we have $P^o_{\gamma_{env}}(S_{i+1}|S_i)$ discounting at each step i. The same condition (5.5) (but in matrix form) then is required of P^o:

$$\|P^o_{\gamma_{env}}\|_\infty > \left(\frac{z}{Z}\right)^{\frac{1}{D-d}}. \tag{5.6}$$

We would like for this condition to hold irrespectively of the length of the options, so long as the plan over them retains the same shape. However, despite the number of *decisions* (over options) remaining the same, an increased number of steps causes $\|P^o_{\gamma_{env}}\|_\infty$ to get smaller, and eventually stop satisfying Eq. (5.6).[2] Hence, the policy over options remains tied critically to the step-discount.

5.3.2 Options with Time Dilation

We suggest to allow the option transition model to dilate time by varying its associated step-discount from γ_{env} to γ_o. In order to mitigate the bias that this introduces, as well as

[2]To see this note that $\gamma^L_{env} \leq \|P^o_{\gamma_{env}}\|_\infty \leq \gamma^\ell_{env}$, where ℓ and L denote minimum and maximum option durations. As such, for any value $\gamma_{env} < 1$ there is an option duration L for which Eq. 5.6 is not satisfied, in particular when $\gamma^L_{env} < \left(\frac{z}{Z}\right)^{\frac{1}{D-d}}$.

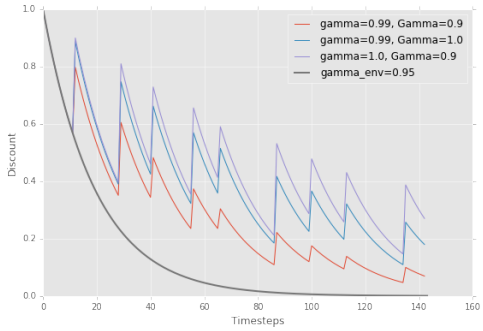

Figure 5.2: The shape of the coefficients induced by different constant values of Γ and γ for random option durations drawn from a Poisson distribution with $\lambda = 10$. The spikes represent a new option choice, and are induced by the fact that there is now discrepancy between the discounting in the transition and reward models. The reward model remains unchanged and discounted with γ_{env}.

ensure favorable convergence guarantees, we additionally impose an *option-level* discount Γ^o. This replaces Eq. (5.1) with the following:

$$P^o_{\Gamma\gamma}(s'|s) \stackrel{\text{def}}{=} \Gamma^o_{ss'} \mathbb{E}_{D:s\to s'|o}\left[\gamma_o^D\right]. \tag{5.7}$$

We will use the $\Gamma\gamma$ suffix to denote models of this form. Since our focus is extending the agent's horizon, we will always consider $\gamma_o \geq \gamma_{env}$. Clearly $P^o_{\Gamma\gamma} = P^o_{\gamma_{env}}$ if $\gamma_o = \gamma_{env}$ and $\Gamma = I$. Figure 5.2 plots a simple instance of the coefficients induced by this manner of discounting. The two-timescale structure is incurred due to the discrepancy between the reward model (which is still discounted with γ_{env}) and the transition model. Such structure is particularly relevant when good options are known, and a policy over options captures the desired policy well.

We will show that γ_o imposes a bias-variance tradeoff on the complexity of estimating the transition model and the solution it incurs. In the extreme, if $\gamma_o = 1$, all of the variance of a given transition s, s' associated with the random variable D that determines the number of steps from s to s' is removed. When compared to the target w.r.t. the native γ_{env}, this comes at the cost of introducing bias which is in terms of the distance of γ_{env} and γ_o. The additional option-level discount Γ^o can help reduce this bias. In fact, we show that there is a value of Γ^o for which the bias is zero, which occurs when Γ^o captures P^o in some sense.

The new option model allows one to effectively redefine the primitive resolution of the agent, simply by considering $\gamma_o = 1$. This in turn provides options with the power to represent policies over horizons that would otherwise be too large to capture with a fixed

step-discount. Indeed, consider the example from above but with $P^o_{\Gamma\gamma}$ with $\gamma_o = 1$, in place of $P^o_{\gamma_{env}}$. Then, $\|P^o_{\Gamma\gamma}\|_\infty = \|\Gamma^o\|_\infty$ becomes independent of the step-discount, and able to represent the same policy over options regardless of their length. This is one of the key motivations of our approach.

Note that the reward model of the agent remains unchanged, and hence the new options are internally inconsistent. In the next section we derive an equivalence with a step-discounted setting with consistent options, that is: ones with transition and reward models discounted on the same scale.

For the sake of simplicity, throughout the rest of the chapter we take $\gamma_o = \gamma$ to be the same for all options, but all of the results can be transferred to the general case of option-dependent γ_o.

Terminal state option-level discount

Although we introduce the framework for the general case of a full Γ^o matrix, it is more practical both in theory and practice to consider a diagonal Γ^o that specifies a common option-level discount upon arrival to a state s', regardless of where the trajectory started. Let Γ^o be a $|\mathcal{S}| \times |\mathcal{S}|$ diagonal matrix whose entries are in $[0, 1]$, s.t. $\Gamma^o(s')$ (or $\Gamma^o_{s'}$) denotes the discount upon arrival at state s'. This form of Γ^o lets us rewrite $P^o_{\Gamma\gamma}$ in an intuitive way:

$$P^o_{\Gamma\gamma}(s'|s) = \gamma\Gamma^o_{s'}\beta^o_{s'}\left(p^{\pi^o}_{ss'} + \gamma\sum_{s''} p^{\pi^o}_{ss''}(1-\beta^o_{s''})\left(p^{\pi^o}_{s''s'} + \gamma\sum_{s'''} p^{\pi^o}_{s''s'''}(1-\beta^o_{s'''})\,(\ldots)\right)\right).$$

This equation makes it evident that Γ^o thus controls the *inter-option* discounting of an option o, while γ is the *intra-option* discount that accounts for the variability in the trajectory.

We return to the full matrix view briefly in Section 5.5.2.

5.4 Convergence Analysis

In this section we will derive an equivalence from the model we propose to a step-discounted setting with consistent options, and use it to prove expected convergence under mild conditions. Our analysis both here and in the next section is for the policy evaluation setting of a fixed policy μ, but our experiments test the control setting, and illustrate the insights found in the theory.

Let us write the new option operators. Note that we continue using $R^o_{\gamma_{env}}$, since time dilation does not affect the reward model. We have:

$$\mathcal{P}^\mu_{\mathcal{O}_{\Gamma\gamma}} q(s,o) = \sum_{s'} P^o_{\Gamma\gamma}(s,s') \sum_{o'} \mu(o'|s') q(s',o'), \tag{5.8}$$

$$\mathcal{T}^{\mu}_{\mathfrak{O}_{\Gamma_\gamma}} q(s,o) \stackrel{\text{def}}{=} R^o_{\gamma_{env}}(s) + \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma_\gamma}} q(s,o). \tag{5.9}$$

We can show that $\mathcal{T}^{\mu}_{\mathfrak{O}_{\Gamma_\gamma}}$ is a contraction so long as a discounted terminating state is reachable. The assumption that options have finite duration is standardly imposed, here we only require that there is a chance for an option to terminate in a state whose discount is less than one.

Assumption 5.1

For each $o \in \mathfrak{O}$ and for each $s \in \mathcal{I}^o$, the initiation set of o, $\exists s'$ that is reachable by π^o, s.t. $\beta^o_{s'} > 0$ and $\Gamma^o_{s'} < 1$, or $\gamma < 1$.

The following theorem proves that $\mathcal{T}^{\mu}_{\mathfrak{O}_{\Gamma_\gamma}}$ is a contraction, and derives the equivalent problem with a modified reward model, termination scheme, and a generalized step-discount.

Theorem 5.1

The operator $\mathcal{T}^{\mu}_{\mathfrak{O}_{\Gamma_\gamma}}$ from Eq. (5.9) is a contraction for $\gamma < 1$ or if a reachable terminating state with $\Gamma_{s'} < 1$ exists (Assumption 5.1). The fixed point of $\mathcal{T}^{\mu}_{\mathfrak{O}_{\Gamma_\gamma}}$ is equivalent to that of a κ-discounted options operator $\mathcal{T}^{\mu}_{\mathfrak{O}_\kappa}$ from Eq. (5.4) for

$$\kappa(s,o,s') = \gamma(\Gamma^o_{s'}\beta^o_{s'} + 1 - \beta^o_{s'}) = \gamma(1 - \beta^o_{s'}(1 - \Gamma^o_{s'})) \leq \gamma,$$

w.r.t. a scaled reward function:

$$z^{\pi^o} = (I - \gamma p^{(1-\beta)\pi^o})(I - \gamma_{env} p^{(1-\beta)\pi^o})^{-1} r^{\pi^o},$$

and termination schemes

$$\zeta^o_{s'} = \frac{\Gamma^o_{s'}\beta^o_{s'}}{\Gamma^o_{s'}\beta^o_{s'} + 1 - \beta^o_{s'}},$$

where as before we write $p^{(1-\beta)\pi^o}_{ss'} \stackrel{\text{def}}{=} (1 - \beta^o_{s'}) p^{\pi^o}_{ss'}$.

This theorem implies that the step discount is controlled by Γ^o, γ, but also β^o. This is appropriate, since β^o controls the inner timescale of an option. For example, if $\gamma = 1$ and $\Gamma^o_{s'} = 0$ for some s', the discount at s' is $1 - \beta^o_{s'}$ exactly.

On the other hand, the value of Γ directly impacts the new implicitly induced termination scheme. For example, if $\Gamma^o_{s'} = 0$ and no bootstrapping occurs, then ζ accounts for it by not permitting any termination. In general, it can be observed that any $\Gamma^o_{s'} < 1$ implies $\zeta^o_{s'} < \beta^o_{s'}$, and hence a solution w.r.t. less terminating, longer options.

5.5 The Bias-Variance Tradeoff in the Option Transition Model

We will now analyze the computational effects of allowing for time dilation in the option transition model. We will show that considering a larger discount in the transition model generally reduces variance, but at the cost of introducing bias. The inter-option discount Γ then helps control this bias, with a particular shape of Γ removing it altogether.

We will show that varying γ in $\mathcal{P}^{\mu}_{\mathfrak{O}_\gamma}$ from Eq. (5.3), and more generally replacing $\mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}}$ with $\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}}$ from Eq. (5.8) induces a novel bias-variance tradeoff on the approximate loss when $\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}}$ is estimated from samples.

Let

$$q^{\mu}_{\gamma_{env}} = (I - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}})^{-1} R_{\gamma_{env}}, \quad q^{\mu}_{\Gamma\gamma} = (I - \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}})^{-1} R_{\gamma_{env}}, \quad q^{\mu}_{\widehat{\Gamma\gamma}} = (I - \widehat{\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}}})^{-1} R_{\gamma_{env}},$$

where $\widehat{\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}}}$ is the approximate transition model estimated from samples. The approximate loss has the following form:

$$\begin{aligned} \mathcal{E} &= \|q^{\mu}_{\widehat{\Gamma\gamma}} - q^{\mu}_{\gamma_{env}}\| = \|q^{\mu}_{\widehat{\Gamma\gamma}} - q^{\mu}_{\Gamma\gamma} + q^{\mu}_{\Gamma\gamma} - q^{\mu}_{\gamma_{env}}\| \\ &\leq \underbrace{\|q^{\mu}_{\widehat{\Gamma\gamma}} - q^{\mu}_{\Gamma\gamma}\|}_{\mathcal{E}_{estim}} + \underbrace{\|q^{\mu}_{\Gamma\gamma} - q^{\mu}_{\gamma_{env}}\|}_{\mathcal{E}_{targ}}, \end{aligned} \tag{5.10}$$

The first term $\mathcal{E}_{estim}$ here is the estimation error that contains the variance, while the second term $\mathcal{E}_{targ}$ is the bias in the targets. We will analyze them separately below.

5.5.1 Variance

It is widely known that larger discounts, and larger eligibility traces typically incur more variance [Jiang et al. 2015b, Petrik and Scherrer 2009, Kearns and Singh 2000]. In the case of options, somewhat counterintuitively, it is also the case that larger transition discounts γ incur *less* estimation variance, when sufficiently large. This becomes evident when considering $\gamma = 1$, for which the variance in γ^D due to the random length of the trajectory is entirely removed. The reason this seems at odds to our knowledge of variance properties of e.g. λ-returns is because the variance incurred by random option duration is not present there. We have the following result (proof in appendix):

Lemma 5.1

Let $d_{\min}$ and $d_{\max}$ be the minimum and maximum option durations across the option set $\mathcal{O}$, and let $\Gamma_{\max}$ be the maximum inter-option discount $\Gamma_{\max} = \max_{o \in \mathcal{O}} \|\Gamma^o\|_1$. Let each $P^o_{\Gamma\gamma}$ be estimated from n i.i.d. samples, and let $R^o_{\gamma_{env}}$ be given. Then, for any policy μ, with probability $1 - \delta$:

$$\mathcal{E}_{estim} = \|q^{\mu}_{\widehat{\Gamma\gamma}} - q^{\mu}_{\Gamma\gamma}\| \leq \frac{r_{\max}}{1 - \gamma_{env}} B_{\Gamma\gamma} \sqrt{\frac{1}{2n} \log \frac{2|\mathcal{S}||\mathcal{O}|}{\delta}},$$
$$B_{\Gamma\gamma} = \frac{\Gamma_{\max}(\gamma^{d_{\min}} - \gamma^{d_{\max}})}{(1 - \Gamma_{\max}\gamma^{d_{\min}})^2}. \tag{5.11}$$

The factor $B_{\Gamma\gamma}$ from Eq. (5.11) thus monotonically increases with $\Gamma_{\max}$, and monotonically *decreases* with γ, when γ is large. See Fig. 5.3 for example shapes. We will observe this behavior empirically on a control task.

5.5.2 Bias

Now, let us turn to the error $\mathcal{E}_{targ}$ incurred by the discrepancy in the targets.

Lemma 5.2

Let $d_{\min}$ and $d_{\max}$ be the minimum and maximum option durations across the option set $\mathcal{O}$, and let $\Gamma_{\max}$ be the maximum inter-option discount $\Gamma_{\max} = \max_{o \in \mathcal{O}} \|\Gamma^o\|_1$. Let μ be a policy over options and consider the difference in the value of μ w.r.t. the option models $\{(R^o_{\gamma_{env}}, P^o_{\gamma_{env}})\}_{o \in \mathcal{O}}$ and $\{(R^o_{\gamma_{env}}, P^o_{\Gamma\gamma})\}_{o \in \mathcal{O}}$. We have:

$$\mathcal{E}_{targ} = \|q^{\mu}_{\Gamma\gamma} - q^{\mu}_{\gamma_{env}}\|_{\infty} \leq \frac{r_{\max}\left((\gamma - \gamma_{env})(\gamma^{d_{\min}} + 1) + \gamma(1 - \Gamma_{\max})\right)}{(1 - \gamma_{env})^2(1 - \Gamma_{\max}\gamma^{d_{\min}})}.$$

Consider the second factor in the numerator of the bound. It is in turn composed of two terms, one that reflects the difference between γ and γ_{env}, and another additive term that has to do with the inter-option discount Γ. If $\Gamma_{\max} = 1$, and there is no inter-option discounting, this term vanishes, and the error reduces to that incurred by the difference in the discounts. Otherwise, there is some bias introduced by $\Gamma_{\max} \neq 1$, and some bias introduced by $\gamma \neq \gamma_{env}$. Even though the worst-case bound is additive, these biases can sometimes be "in opposite directions", and reduce the overall error when compared to either one in isolation. In fact, there is a value of Γ that reduces bias all the way to zero, even if $\gamma \neq \gamma_{env}$. The following proposition derives a sufficient condition for this.

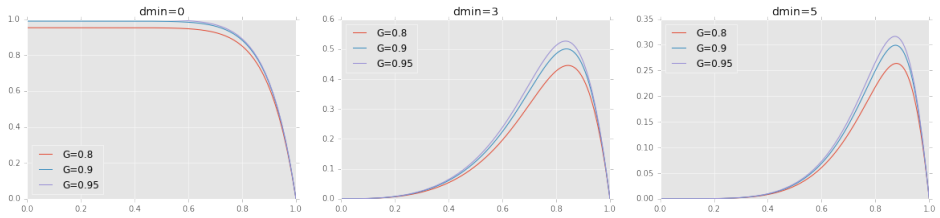

Figure 5.3: $B_{\Gamma\gamma}$ from Eq. (5.11) for $d_{\max} = 10$ and different values of $d_{\min}$. We see that there is a decrease in variance near $\gamma = 1$. Note that the lower values of γ corresponding to the other low-variance region may not be sufficient to represent complex policies.

Proposition 5.1

If $\Gamma^o = (P^o_\gamma)^{-1} P^o_{\gamma_{env}}$, there is no bias in the value function, regardless of γ.

Note that in order for the form of Γ^o from this proposition to hold, Γ^o must be a full (rather than diagonal) matrix, whose value is closely related to that of the option transition model[3] While it is unlikely to be able to achieve this, even an approximate Γ^o can help balance the bias. We leave a precise characterization of the general case of this for the future.

Finally, from Eq. (5.10) and Lemmas 5.1 and 5.2, we have our result:

Theorem 5.2

Let $d_{\min}$ and $d_{\max}$ be the minimum and maximum option durations across the option set $\mathcal{O}$, and let $\Gamma_{\max}$ be the maximum inter-option discount $\Gamma_{\max} = \max_{o \in \mathcal{O}} \|\Gamma^o\|_1$. Let μ be a policy over options, and let each $P^o_{\Gamma\gamma}$ be estimated from n i.i.d. samples. Then, with probability $1 - \delta$, the error in the estimate $q^\mu_{\widehat{\Gamma\gamma}}$ is bounded by:

$$\mathcal{E} = \|q^\mu_{\widehat{\Gamma\gamma}} - q^\mu_{\gamma_{env}}\| \leq \frac{r_{\max}}{1-\gamma_{env}} \times \left(\underbrace{\frac{\Gamma_{\max}(\gamma^{d_{\min}} - \gamma^{d_{\max}})}{(1-\Gamma_{\max}\gamma^{d_{\min}})^2} \sqrt{\frac{1}{2n} \log \frac{2|\mathcal{S}||\mathcal{O}|}{\delta}}}_{variance} + \underbrace{\frac{(\gamma - \gamma_{env})(\gamma^{d_{\min}} + 1) + \gamma(1 - \Gamma_{\max})}{(1-\gamma_{env})(1-\Gamma_{\max}\gamma^{d_{\min}})}}_{bias} \right).$$

[3] e.g. if $\gamma = 1$, Γ^o must be $P^o_{\gamma_{env}}$ exactly.

Note that while this result is for the setting of policy evaluation, it is representative of the control setting, as evidenced by our experimental results.

5.6 Experiments

We investigate the two key ideas of this chapter empirically, namely we first demonstrate the bias-variance tradeoff obtained in Theorem 5.2, and we then illustrate the ability of time dilation to extend the agent's horizon and preserve far-sighted policies, irrespectively of the size of the environment. Our approximate planning setting is similar to that described in [Jiang et al. 2015b]. Similarly to that work, and since the reward model is unaffected by our proposed framework, we do not estimate the reward model in our experiments, but use its true value.

5.6.1 Bias-Variance

We investigate whether the analytical bias-variance tradeoff can be observed in practice in the control setting on the classical Four Rooms domain [Sutton et al. 1999], see Fig. 5.4, left. Here, the agent aims to navigate to a goal location via options that navigate from inside of each room to its hallways. To evaluate the effects of varied option duration, we add ϵ-noise to the typically deterministic option policies. That is: an option takes an action recommend by its original π^o w.p. $1-\epsilon$, and a random action w.p. ϵ. To obtain a clear picture, we consider a very noisy case of $\epsilon = 0.5$.

For each option o, and for each state $s \in \mathcal{I}^o$, we sample n trajectories to obtain an estimate $\widehat{P^o_{\Gamma\gamma}}$ of $P^o_{\Gamma\gamma}$. We then perform policy iteration w.r.t. the approximate models $\widehat{P^o_{\Gamma\gamma}}$ and the true reward models $R^o_{\gamma_{env}}$ to obtain the approximate optimal policy $\pi^*_{\widehat{\Gamma\gamma}}$. We then report the certainty equivalence (CE) loss[4] $-\frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} v^*_{\widehat{\Gamma\gamma}}(s)$ for the value of this $\pi^*_{\widehat{\Gamma\gamma}}$. See Fig. 5.5 for the results and considered parameter ranges.

Notice how the loss curves mimic the bound on the variance term from Lemma 5.1 closely for reasonably high γ, while the bias term dominates the performance of the low γ-s.

5.6.2 Horizon Invariance

Recall the scenario described in Sec. 5.3.1. We simulate an experiment that mimics this scenario and observe that the claims hold in practice numerically. In particular, we consider

[4] CE control is the term used in stochastic control theory for the setting of acting according to a policy obtained via planning with an inaccurate model (e.g. [Jiang et al. 2015b]).

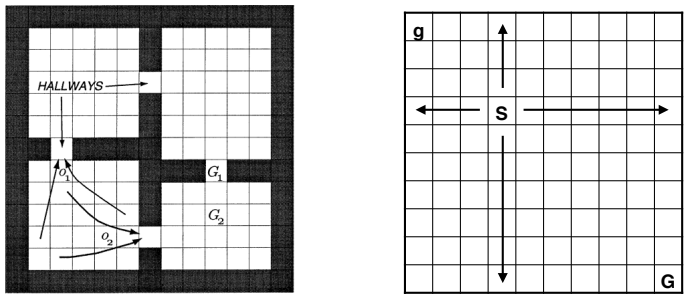

Figure 5.4: The domains used in our experiments. **Left** Four Rooms. The agent starts in the top left room, and aims to navigate to G_1 via options that navigate to hallways. The option policies are ϵ-soft and extremely noisy with $\epsilon = 0.5$. **Right** Growing Gridworld. The agent's task is to get from the start state S to the goal G. There is another distractor goal g with a smaller reward.

a simple Growing Gridworld task (Fig. 5.4, right). There are two terminal states: g with a smaller reward (of 1) and G with a larger reward (of 2). The preference of the agent between them is entirely determined by its discount factor γ. As in the previous experiment we estimate $\widehat{P^o_{\Gamma\gamma}}$ from n samples, and obtain $\pi^*_{\widehat{\Gamma\gamma}}$ by policy iteration. We take the value of n to be 2 here. For the estimation to be less trivial, we consider ϵ-soft option policies, as described above, with $\epsilon = 0.05$. We then consider both the value of the optimal policy $\pi^*_{\widehat{\Gamma\gamma}}$ w.r.t. approximate model, and the value of the optimal policy $\pi^*_{\Gamma\gamma}$ w.r.t. the true model $P^o_{\Gamma\gamma}$, both evaluated with a very high $\gamma_{eval} = 1 - 10^{-8}$ to capture our desired evaluation metric. We denote these values by $v^{\widehat{\pi^*}}_{\gamma_{eval}}$ and $v^{\pi^*}_{\gamma_{eval}}$, respectively.

We compare two variants: one with $\gamma < 1, \Gamma = 1$ (corresponding to the classical option model), and the other with $\gamma = 1, \Gamma < 1$ (exploiting time dilation). The reward model is computed with the same value of $\gamma_{env} < 1$ for both cases. Figure 5.6 reports the certainty equivalence gain $\frac{1}{|\mathcal{S}|}\sum_s v(s)$ for both the exact and approximate optimal values of these variants.

We see that the same pattern is induced in both the exact and approximate case, and the values of the optimal policies diminish, as the size of the grid gets larger. As should be expected, the effect is more pronounced in the approximate case. Time dilation on the other hand allows the options to maintain the same performance regardless of the size of the grid.

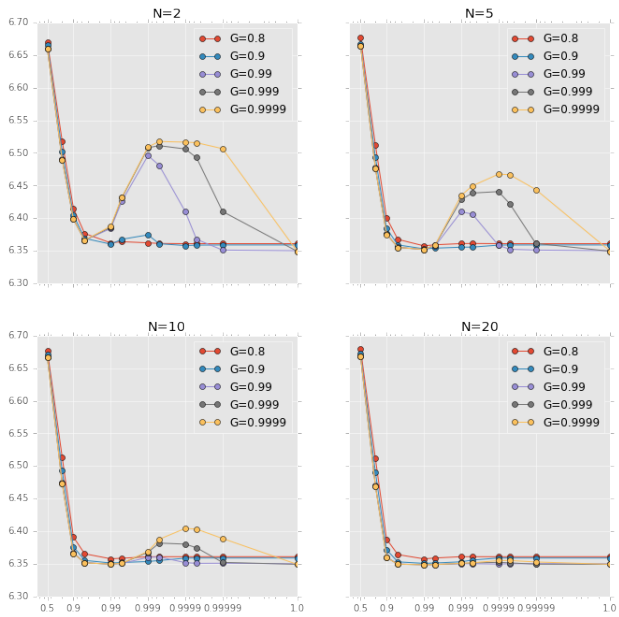

Figure 5.5: The certainty equivalence loss $-\frac{1}{|\mathcal{S}|}\sum_s v^*_{\widehat{\Gamma}_\gamma}(s)$ as a function of γ and for different values of Γ (lower is better). The reward model is known, the transition model is estimated from N samples, and $v^*_{\widehat{\Gamma}_\gamma}$ is obtained from solving it. Average of 100 independent runs. Notice the similarity with Fig. 5.3, which diminishes as N increases, since the effects of the variance then diminish. The large error in the small γ-s on the other hand is due to a large bias. Note the log scale, where we have biased the value at 1.0 to be finite.

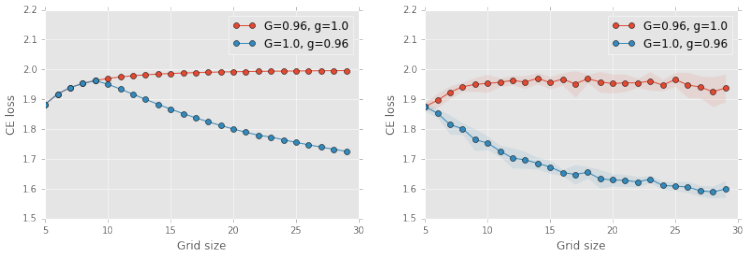

Figure 5.6: The certainty equivalence *gain* $\frac{1}{|\mathcal{S}|}\sum_s v(s)$ as a function of the grid size (higher is better). The value function v is the value w.r.t. a high γ_{eval} of the optimal policies w.r.t. **Left** the exact model $P^o_{\Gamma\gamma}$ **Right** the approximate model $\widehat{P^o_{\Gamma\gamma}}$. The shaded area denotes standard deviation. We compare two variants: one with $\gamma < 1, \Gamma = 1$ (corresponding to the classical option model), and the other with $\gamma = 1, \Gamma < 1$ (exploiting time dilation). The reward model is computed with the same value of $\gamma_{env} < 1$ for both cases. We see that in both cases the performance of the variant $\gamma < 1$ deteriorates with the size of the grid, while the variant with $\gamma = 1, \Gamma < 1$ is indifferent to the size of the grid. Note that this pattern is irrespective of the chosen value of γ and would occur for some grid size for any γ.

5.7 Related Work

The analysis in our work is closely related to that in [Jiang et al. 2015b], and the earlier results along the same lines of [Petrik and Scherrer 2009]. In both works, the authors consider the tradeoff on the estimation variance and the target bias in the quality of the approximate planning solution incurred by using a lower discount factor. [Jiang et al. 2015b] show that it is beneficial to use a lower discount, when the number of samples of the transition model is small. These implications carry over to Γ^o in the context of options, while γ controls a more subtle tradeoff that has to do with random option duration. [Petrik and Scherrer 2009] focus on the bias aspect, and show that in some problems the bias due to using a lower discount can be better than predicted by the worst case. In particular, the authors show that is true for problems whose rewards are sparse. It is interesting to identify a similar structure for the case of options.

General transition-based discounting is introduced in [White 2017]. There, the author proposes to use the discount as a formalism for reinforcement learning *tasks*, and argue that each option then represents a task, since the termination condition of each option together with the step discount incurs a transition discount. We propose to alter the option discounting explicitly, and as a result incur option-*transition* discounts.

5.8 Discussion

Control case. While the experiments we have conducted are in the control setting, and support the implications of the analysis, the analysis applies to a fixed policy μ. Most of the results can be extended to the control case with an extra relatively standard step, but in order to extend Lemma 5.1 to apply to all policies, we need to consider the relationship of the number of optimal policies under a given model $|M_{\Gamma\gamma}|$ to Γ and γ. [Jiang et al. 2015b] give an interesting interpretation of γ as a policy complexity control parameter, and show that the number of optimal policies grows monotonically with γ. This is less straightforward in the case of options due to there being two parameters Γ and γ, instead of one. We plan to investigate this further in the future.

Reward model. We have assumed throughout that the reward model R is unaffected by the new discounting and is w.r.t. γ_{env}. Such an assumption is natural – the option may not have control over the way that the environment provides it with rewards, and it is convenient to maintain the reward models on the same scale. For complete generality, it is possible to consider an option-specific γ_r^o, which would introduce another bias-variance tradeoff, this time in the option return. This could be useful when the reward model is estimated from noisy samples, and a lower γ_r^o is sufficient to capture it.

Limitations. We expect there to be interaction between the effects of time dilation and the sign of the rewards, hence a renormalization to a positive reward range may be required for predictable behavior. The method introducess more parameters, when compared to the naive scalar γ case, and just like with γ, the effect of the parameters on performance can be very strong. Finding a way to adjust them from data (in the style of the termination condition of option-critic algorithhms [Bacon et al. 2017] is an interesting future direction.

Future work. We plan to evaluate the benefits of horizon invariance with option transfer from Section 5.3.1 in a larger problem. If there is generalization in the state representation, it may be possible to successively transfer option policies learnt on smaller instances of a task to larger instances of the same task, as a form of *scaffolding*. Furthermore, it is interesting to use time dilation as an alternative to deliberation cost [Harb et al. 2018] for incentivizing longer option duration.

5.9 Summary

We propose a generalization to the options framework that provides an option with autonomy over its timescale, by introducing time dilation into the option transition model. We have analyzed the bias-variance tradeoff incurred by doing so, and verified the analytical shapes empirically. We have shown that the proposed framework allows one to continue to represent policies over options for which the step discount is inadequate.

Representing long horizons is unquestionably key to learning complex problems. Unfortunately, the incurred computational complexity of doing so via the usual step discount is often prohibitive. The time dilation mechanism we propose in this work may sidestep this issue by tying discounting in with the options framework, and allowing options to be truly temporally abstract.

"The answer to the Great Question... of Life, the Universe and Everything... is... Forty-two," said Deep Thought, with infinite majesty and calm.

— Douglas Adams, *The Hitchhiker's Guide to the Galaxy.*

6 | Potential-Based Shaping with Arbitrary Rewards

In this final chapter, we turn to another approach to enriching the basic learning step. Namely, we consider a case, where the environment reward may be very sparse, but other sources of auxiliary information are available. This is common in realistic settings, since learning is rarely insular, and there is a multitude of additional signals one could attempt to leverage. The key challenge in doing so is ensuring that it is not harmful. This is a tough requirement to meet, because it requires a careful inspection of the additional feedback.[1] The framework of potential-based reward shaping (PBRS) guarantees that its application preserves optimality, and is hence ideal for incorporating auxiliary information, whose quality cannot be assured.

The drawback of PBRS is that it requires rendering the additional information in a specific form, which may be restrictive. In this chapter we will describe an algorithm to use PBRS to encode information of general form and obtain the required specific form through learning. We will then validate our approach on a case study of online feedback in the game of Mario, which is a scenario that was difficult to handle soundly in the past.

[1] See e.g. a recent blog post by OpenAI [OpenAI 2016] for a modern take on this issue.

6.1 Introduction

The term *shaping* in experimental psychology (dating at least as far back as [Skinner 1938]) refers to the idea of rewarding all behavior *leading* to the desired behavior, instead of waiting for the subject to exhibit it autonomously (which, for complex tasks, may take prohibitively long). For example, Skinner discovered that, in order to train a rat to push a lever, any movement in the direction of the lever had to be rewarded. The typical *tabula rasa* RL paradigm guarantees the agent to learn the desired behavior eventually. However, as with Skinner's rat, the RL agent may take a very long time to stumble upon the target lever, if the only reinforcement (or *reward*) it receives is after that fact. Shaping can hence be used to speed up the learning process by providing additional rewards. Shaping in RL has been linked to reward functions from very early on; [Mataric 1994] interpreted shaping as designing a more complex reward function, [Dorigo and Colombetti 1997] used shaping on a real robot to translate expert instructions into reward for the agent, as it executed a task, and [Randløv and Alstrøm 1998] proposed learning a hierarchy of RL signals in an attempt to separate the extra reinforcement function from the base task. It is in the same paper that they uncover the issue of modifying the reward signals in an unconstrained way: when teaching an agent to ride a bicycle, and encouraging progress towards the goal, the agent would get "distracted", and instead learn to ride in a loop and collect the positive reward forever. This issue of *positive reward cycles* is addressed by [Ng et al. 1999], where they devise their *potential-based* reward shaping (PBRS) framework, which constrains the shaping reward to have the form of a difference of a potential function of the transitioning states. In fact, they prove a stronger claim that such a form is *necessary*[2] for leaving the original task unchanged. This elegant and implementable framework led to an explosion of reward shaping research and proved to be extremely effective [Asmuth et al. 2008], [Devlin et al. 2011], [Brys et al. 2014], [Snel and Whiteson 2014]. [Wiewiora et al. 2003] extended PBRS to state-action *advice* potentials, and [Devlin and Kudenko 2012] recently generalized PBRS to handle *dynamic* potentials, allowing potential functions to change online whilst the agent is learning.

6.1.1 Potential-Based Reward Shaping

The most general form of reward shaping in RL can be described as modifying the reward function of the underlying MDP:

$$r' = r + f,$$

[2]Given no knowledge of the MDP dynamics.

where f is the *shaping* reward function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, with $f(s, a, s')$ giving the additional reward on the transition (s, a, s'), and where here and throughout this chapter we assume r to be defined on transitions. *Potential-based reward shaping* [Ng et al. 1999] maintains a potential function $h : \mathcal{S} \to \mathbb{R}$, and constrains the shaping reward function f to the following form:

$$f(s, a, s') = \gamma h(s') - h(s), \qquad \forall a \in \mathcal{A}, \tag{6.1}$$

where γ is the discounting factor of the MDP. [Ng et al. 1999] show that this form is both necessary and sufficient for policy invariance. [Wiewiora et al. 2003] extend PBRS to *advice* potential functions defined over the state-action space. The authors consider two types of advice: look-ahead and look-back, providing the theoretical framework for the former. For any snippet s, a, s', a':

$$f(s, a, s', a') = \gamma h(s', a') - h(s, a). \tag{6.2}$$

Finally, [Devlin and Kudenko 2012] generalize the form in Eq. (6.1) to dynamic potentials, by including a time parameter, and show that all theoretical properties of PBRS hold. For any snippet of experience s, t, s', t', where t and t' are the times of occurrence of s and s' we have:

$$f(s, t, s', t') = \gamma h(s', t') - h(s, t).$$

6.2 Expressing Arbitrary Reward Functions as Potential-Based Advice

Additive reward functions from early reward shaping research, while dangerous to policy preservation, were able to convey *behavioral* knowledge (e.g. expert instructions) directly. Potential functions require an additional abstraction, and restrict the form of the additional *effective* reward, but provide crucial theoretical guarantees. We seek to bridge this gap between the available behavioral knowledge and the potential-based shaping rewards.

In this work, we provide a novel way to specify the shaping rewards, *directly* through an arbitrary reward function, while implicitly maintaining the grounding in potentials, necessary for policy invariance. For this, we first extend Wiewiora's advice framework to *dynamic advice potentials*. We then propose to in parallel learn a *secondary* value function w.r.t. a variant of our arbitrary reward function, and use its successive estimates as our dynamic advice potentials. We show that the effective shaping rewards then reflect the input reward function in expectation. Empirically, we first demonstrate our method to avoid the issue of positive reward cycles on a grid-world task, when given the same

behavior knowledge that trapped the bicyclist from [Randløv and Alstrøm 1998]. We then show an application, where our *dynamic (PB) value-function advice* outperforms other reward-shaping methods that encode the same knowledge, as well as a shaping w.r.t. a different popular heuristic.

6.2.1 From Reward Functions to Dynamic Potentials

There are two (inter-related) problems in PBRS: *efficacy* and *specification*. The former has to do with designing *the best* potential functions, i.e. those that offer the quickest and smoothest guidance. The latter refers to capturing the *available* domain knowledge into a potential form, in the easiest and most effective way. This work primarily deals with that latter question.

Locking knowledge in the form of potentials is a convenient theoretical paradigm, but may be restrictive, when considering all types of domain knowledge, in particular *behavioral* knowledge, which is likely to be specified in terms of *actions*. Say, for example, an expert wishes to encourage an action a in a state s. If following the advice framework, she sets $h(s,a) = 1$, with h zero-valued elsewhere, the shaping reward associated with the transition (s,a,s') and some function a' will be $f(s,a,s',a') = h(s',a') - h(s,a) = 0 - 1 = -1$, so long as the pair (s',a') is different from (s,a).[3] The favorable behavior (s,a) will then factually be *dis*couraged. She could avoid this by further specifying h for state-actions reachable from s via a, but that would require knowledge of the MDP. What she would thus like to do is to be able to specify the desired *effective* shaping reward f directly, but without sacrificing optimality provided by the potential-based framework.

This work formulates a framework to do just that. Given an arbitrary reward function r^*, we wish to achieve $f \approx r^*$, while maintaining policy invariance. This question is equivalent to seeking a potential function h, based on r^*, s.t. $f^h \approx r^*$, where (and in the future) we take f^h to mean a potential-based shaping reward w.r.t. h.

The core idea of our approach is to learn h in parallel as a secondary (state-action) value function on the negation of the expert-provided r^*, and use the consecutively updated values of h_t as a dynamic potential function, thus making the translation into potentials *implicit*. Specifically, for an experience s, a, S_{t+1}, A_{t+1}:

$$h_{t+1}(s,a) = h_t(s,a) + \beta_t \delta_t^h,$$

$$\delta_t^h = R_{t+1}^h + \gamma h_t(S_{t+1}, A_{t+1}) - h_t(s,a), \tag{6.3}$$

$$R_{t+1}^h = -r^*(s,a), \tag{6.4}$$

[3] Assume the example is undiscounted for clarity.

where β_t is the learning rate at time t, and the action A_{t+1} is chosen on-policy w.r.t. some policy π. The shaping reward is then of the form:

$$F_{t+1} = \gamma h_{t+1}(S_{t+1}, A_{t+1}) - h_t(S_t, A_t). \tag{6.5}$$

The intuition of the correspondence between r^* and f lies in the relation between the Bellman equation (for h):

$$h^\pi(s,a) = -r^*(s,a) + \gamma \mathbb{E}_\pi \left[h^\pi(s',a') \right]$$

and shaping rewards from an advice potential function:

$$f(s,a) = \gamma h(s',a') - h(s,a) = r^*(s,a).$$

This intuition will be made more precise later.

6.2.2 Analysis

This section is organized at follows. First, we extend the potential-based advice framework to *dynamic* potential-based advice, and ensure that the desired guarantees hold. (Our dynamic (potential-based) *value-function* advice is then an instance of dynamic potential-based advice.) We then turn to the question of correspondence between r^* and f, showing that f captures r^* in expectation. Finally, we ensure that these expectations are meaningful, by arguing convergence.

Dynamic Potential-Based Advice

Analogously to [Devlin and Kudenko 2012], we augment Wiewiora's look-ahead advice function (Eq. (6.2)) with a time parameter to obtain our dynamic potential-based advice:

$$f(s,a,t,s',a',t') \stackrel{\text{def}}{=} \gamma h(s',a',t') - h(s,a,t) = h_{t'}(s',a') - h_t(s,a) \tag{6.6}$$

where t/t' is the time of the agent visiting state s/s' and taking action a/a'. The following theorem shows the relationship of the values of a policy in the original and shaped MDPs.

Theorem 6.1

Let q^π be the value of a policy π in $M = (r, \mathcal{S}, \mathcal{A}, p, \gamma)$, and q_h^π that in $M' = (r + f, \mathcal{S}, \mathcal{A}, p, \gamma)$, with f defined as in Eq. (6.6). We have that:

$$q_h^\pi = q^\pi - h_0.$$

Thus, once the optimal policy w.r.t. $r + f$ is learnt, to uncover the optimal policy w.r.t. r, one may use the *biased greedy* action-selection [Wiewiora et al. 2003] w.r.t. the *initial* values of the dynamic advice function.

$$\pi(s) = \arg\max_a \left(q(s,a) + h_0(s,a)\right).$$

Notice that when the advice function is initialized to 0, the biased greedy action-selection above reduces to the basic greedy policy, allowing one to use dynamic advice equally seamlessly to simple state potentials.

Shaping in Expectation

The previous section proved the soundness of the general framework of dynamic advice. We will now show that the specific algorithm we propose indeed captures the auxiliary r^* in expectation.

Let h be the state-action value function that learns on $r^h = -r^*$, while following some fixed policy π. The shaping reward at timestep t w.r.t. h as a dynamic advice function is given by:

$$\begin{aligned}
F_{t+1} &\stackrel{\text{def}}{=} f(S_t, A_t, t, S_{t+1}, A_{t+1}, t+1) \\
&= \gamma h_{t+1}(S_{t+1}, A_{t+1}) - h_t(S_t, A_t) \\
&= \gamma h_t(S_{t+1}, A_{t+1}) - h_t(S_t, A_t) + \gamma h_{t+1}(S_{t+1}, A_{t+1}) - \gamma h_t(S_{t+1}, A_{t+1}) \\
&\stackrel{(6.3)}{=} \delta_t^h - R_{t+1}^h + \gamma \Delta h(S_{t+1}, A_{t+1}) \\
&= r^*(S_t, A_t) + \delta_t^h + \gamma \Delta h(S_{t+1}, A_{t+1}). \qquad (6.7)
\end{aligned}$$

where we denote the change in the estimate h from time t to time $t+1$ by Δh. Now assume the process has converged to the TD fixed point h^π. $h_{t+1} = h_t = h^\pi$. For any t and t' we have:

$$\begin{aligned}
f(s, a, t, s', a', t') &= \gamma h^\pi(s', a') - h^\pi(s, a) \\
&= -r^h(s, a) + r^h(s, a) + \gamma \mathbb{E}_\pi \left[h^\pi(s', a')\right] - h^\pi(s, a)
\end{aligned}$$

$$
\begin{aligned}
&+ \gamma h^\pi(s', a') - \gamma \mathbb{E}_\pi \left[h^\pi(s', a') \right] \\
&= r^*(s, a) + \gamma \left(h^\pi(s', a') - \mathbb{E}_\pi \left[h^\pi(s', a') \right] \right). \qquad (6.8)
\end{aligned}
$$

Thus, we obtain that the shaping reward f w.r.t. the converged values h^π, reflects the *expected* auxiliary reward $r^*(s, a)$ plus a term that measures how different the sampled next state-action value is from the expected next state-action value. This term will at each transition further encourage transitions that are "better than expected" (and vice versa), similarly, e.g., to "better-than-average" (and vice versa) rewards in R-learning [Schwartz 1993].

Now let f^* be the *expected* shaping reward on a state-action pair (s, a), Taking the expectation w.r.t. the transition matrix p, and the policy π with which a' is chosen, we have:

$$
\begin{aligned}
f^*(s, a) &= \mathbb{E}_\pi \left[f(s, a, s', a') \right] \\
&= r^*(s, a) + \gamma \mathbb{E}_\pi \left[h^\pi(s', a') - \mathbb{E}_\pi \left[h^\pi(s', a') \right] \right] = r^*(s, a). \qquad (6.9)
\end{aligned}
$$

Thus, Eq. (6.7) gives the shaping reward while h's are not yet converged, (6.8) gives the component of the shaping reward on a transition after h^π are correct, and (6.9) establishes the equivalence of f^* and r^* in expectation.

Convergence of h

If the policy π is fixed, and the q^π estimates are correct, the expectations in the previous section are well-defined, and h converges to the corresponding TD fixed point. However, h is learnt at the same time as q. This process can be shown to converge by formulating the framework on two timescales [Borkar 1997], and using the ODE method of [Borkar and Meyn 2000]. We thus require[4] the step size schedules $(\alpha_k)_{k \in \mathbb{N}}$ and $(\beta_k)_{k \in \mathbb{N}}$ satisfy the following:

$$
\lim_{k \to \infty} \frac{\alpha_k}{\beta_k} = 0 \qquad (6.10)
$$

q and h correspond to the slower and faster timescales, respectively. Given that step-size schedule difference, we can rewrite the iterations (for q and h) as one iteration, with a combined parameter vector, and show that the assumptions (A1)-(A2) from [Borkar and Meyn 2000] are satisfied, which allows to apply their Theorem 2.2. This analysis is analogous to that of convergence of TD with Gradient Correction (Theorem 2 in [Sutton et al. 2009]), and is left out for clarity of exposition.

[4] In addition to the standard stochastic approximation assumptions, that is Assumption 2.2 for both α_k and β_k, and Assumptions 2.3, 3.1.

Note that this convergence is needed to assure that h indeed captures the expert reward function r^*. The form of general dynamic advice from Theorem 6.1 itself does not pose any requirements on the convergence properties of h to guarantee policy invariance.

6.2.3 Experiments

We first demonstrate our method correctly solving a grid-world task, as a simplified instance of the bicycle problem. We then assess the practical utility of our framework on a larger cart-pole benchmark, and show that our dynamic (PB) VF advice approach outperforms other methods that use the same domain knowledge, as well as a popular static shaping w.r.t. a different heuristic.

Grid-World

We formulate a minimal working example of the bicycle problem [Randløv and Alstrøm 1998], illustrating the issue of positive reward cycles. Given a 20×20 grid, the goal is located at the bottom right corner. The agent must reach it from its initial position at the top left corner, upon which event it will receive a positive reward. The reward on the rest of the transitions is 0. The actions correspond to the 4 cardinal directions, and the state is the agent's position coordinates (x, y) in the grid. The episode terminates when the goal was found, or when 10000 steps have elapsed.

Given approximate knowledge of the problem, a natural heuristic to encourage is transitions that move the agent to the right, or down, as they are to advance the agent closer to the goal. A reward function r^* encoding this heuristic can be defined as

$$r^*(s, \text{right}) = r^*(s, \text{down}) = c, \quad c \in \mathbb{R}^+, \quad \forall s \in \mathcal{S}.$$

When provided naïvely (i.e. with $f = r^*$), the agent is at a risk of getting "distracted": getting stuck in a positive reward cycle, and never reaching the goal. We apply our framework, and learn the corresponding h w.r.t. $r^h = -r^*$,setting f accordingly (Eq. (6.5)). We compare that setting with the base learner and with the non-potential-based naïve learner.[5]

Learning was done via Sarsa with ϵ-greedy action selection, $\epsilon = 0.1$. The learning parameters were tuned to the following values: $\gamma = 0.99$, $c = 1$, $\alpha_{t+1} = \tau\alpha_t$ decaying exponentiallywith $\alpha_0 = 0.05$, $\tau = 0.999$ and $\beta_t = 0.1$.

[5] To illustrate our point more clearly, we omit the *static* PB variant with (state-only) potentials $h(x, y) = x + y$. It depends on a different type of knowledge (about the state), while this experiment compares two ways to utilize the behavioral reward function r^*. The static variant does not require learning, and hence performs better in the beginning.

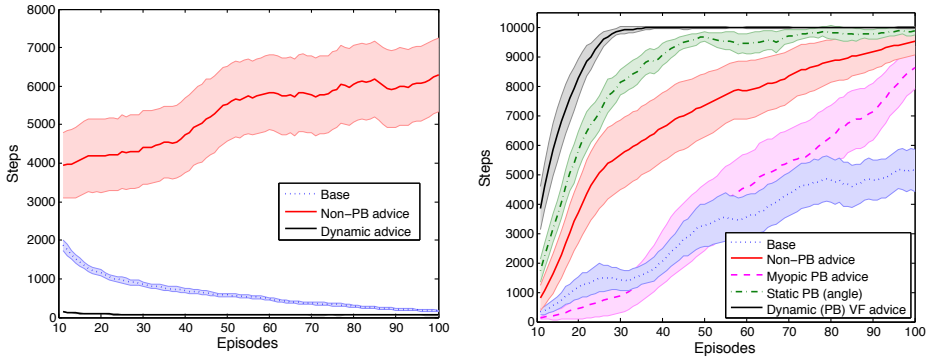

Figure 6.1: Mean learning curves. Shaded areas correspond to the 95% confidence intervals. The plot is smoothed by taking a running average of 10 episodes. (a) The same reward function added directly to the base reward function (non-PB advice) diverges from the optimal policy, whereas our automatic translation to dynamic-PB advice accelerates learning significantly. (b) Our dynamic (PB) VF advice learns to balance the pole the soonest, and has the lowest variance.

We performed 50 independent runs, 100 episodes each (Fig. 6.1). Observe that the performance of the (non-PB) agent learning with $f = r^*$ actually got *worse* with time, as it discovered a positive reward cycle, and got more and more disinterested in finding the goal. Our agent, armed with the same knowledge, used it properly (in a true potential-based manner) and the learning was accelerated significantly, compared to the base agent.

Cart-Pole

We now evaluate our approach on a more difficult cart-pole benchmark [Michie and Chambers 1968]. The task is to balance a pole on top a moving cart for as long as possible. The (continuous) state contains the angle ξ and angular velocity $\dot{\xi}$ of the pole, and the position x and velocity $\dot{x}$ of the cart. There are two actions: a small positive and a small negative force applied to the cart. A pole falls if $|\xi| > \frac{\pi}{4}$, which terminates the episode. The track is bounded within $[-4, 4]$, but the sides are "soft"; the cart does not crash upon hitting them.[6] The reward function penalizes a pole drop, and is 0 elsewhere. An episode terminates successfully, if the pole was balanced for 10000 steps.

An intuitive behavior to encourage is moving the cart to the right (or left) when the pole is leaning rightward (or leftward). Let $i : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$ be the indicator function

[6] This is to focus our attention to the balancing problem.

	Best γ values		Base $\gamma = 0.8$	
Variant	Final	Overall	Final	Overall
Base	5114.7±188.7	3121.8±173.6	5114.7±165.4	3121.8±481.3
Non-PB advice	9511.0±37.2	6820.6±265.3	6357.1±89.1	3405.2±245.2
Myopic PB shaping	8618.4±107.3	3962.5±287.2	80.1±0.3	65.8±0.9
Static PB	9860.0±56.1	8292.3±261.6	3744.6±136.2	2117.5±102.0
Dynamic (PB) VF advice	**9982.4±18.4**	**9180.5±209.8**	**8662.2±60.9**	**5228.0±274.0**

Table 6.1: Cart-pole results. Performance is indicated with standard error. The final performance refers to the last 10% of the run. Dynamic (PB) VF advice has the highest mean, and lowest variance both in tuned and fixed γ scenarios, and is the most robust, whereas myopic shaping proved to be especially sensitive to the choice of γ.

denoting such orientation of state s and action a. A reward function to encompass the rule can then be defined as:

$$r^*(s,a) = i(s,a) \times c, \quad c \in \mathbb{R}^+.$$

We compare the performance of our agent to the base learner and two other reward shaping schemes that reflect the same knowledge about the desired behavior, and one that uses different knowledge (about the angle of the pole).[7] The variants are described more specifically below:

Base The base learner, $f_1 = 0$.

Non-PB advice Advice is received simply by adding r^* to the main reward function, $f_2 = r^*$. This method will lose some optimal policies.

Myopic PB advice Potentials are initialized and maintained with r^*, i.e. $f_3 = f^h$ with $h = r^*$. This is closest to Wiewiora's look-ahead advice framework.

Static PB shaping with angle The agent is penalized proportionally to the angle with which it deviates from equilibrium. $f_4 = f^h$ with $h \sim -|\xi|^2$.

Dynamic (PB) VF advice We learn h as a value function w.r.t. $r^h = -r^*$, and set $f_5 = f^h$ accordingly (Eq. (6.5)).

We used tile coding [Sutton and Barto 2017] with 10 tilings of 10×10 to represent the continuous state. Learning was done via Sarsa(λ) with eligibility traces and ϵ-greedy

[7] Note that unlike our behavioral encouragement, the angle shaping requires *precise* information about the state, which is more demanding in a realistic setup, where the advice comes from an external observer.

action selection, $\epsilon = 0.1$. The learning parameters were tuned to the following: $\lambda = 0.9$, $c = 0.1$, $\alpha_{t+1} = \tau\alpha_t$ decaying exponentially with $\alpha_0 = 0.05$, $\tau = 0.999$, and $\beta_t = 0.2$. We found γ to affect the results differently across variants, with the following best values: $\gamma_1 = 0.8$, $\gamma_2 = \gamma_3 = \gamma_4 = 0.99$, $\gamma_5 = 0.4$. M_i is then the MDP $(\mathcal{S}, \mathcal{A}, \gamma_i, p, r + f_i)$. Fig. 6.1 gives the comparison across M_i (i.e. the best γ values for each variant), whereas Table 6.1 also contains the comparison w.r.t. the base value $\gamma = \gamma_1$.

We performed 50 independent runs of 100 episodes each (Table 6.1). Our method outperforms the alternatives in both fixed and tuned γ scenarios, converging to the optimal policy reliably after 30 episodes in the latter (Fig. 6.1). Paired t-tests on the sums of steps of all episodes per run for each pair of variants confirm all variants as significantly different with $p < 0.05$. Notice that the non-potential-based variant for this problem does not perform as poorly as on the grid-world task. The reason for this is that getting stuck in a positive reward cycle can be *good* in cart-pole, as the goal is to continue the episode for as long as possible. However, consider the policy that achieves keeping the pole at an equilibrium (at $\xi = 0$). While clearly optimal in the original task, this policy will not be optimal in M_2, as it will yield 0 additional rewards.

6.2.4 Discussion

Choice of r^*. The given framework is general enough to capture any form of the reward function r^*. Recall, however, that $f^* = r^*$ holds *after* h values have converged. Thus, the simpler the provided reward function r^*, the sooner will the effective shaping reward capture it. In this work, we have considered reward functions r^* of the form $r^*(\mathcal{B}) = c, c > 0$, where $\mathcal{B}$ is the set of encouraged behavior transitions. This follows the convention of shaping in psychology, where punishment is *implicit* as absence of positive encouragement. Due to the expectation terms in f, we expect such form (of all-positive, or all-negative r^*) to be more robust. Another assumption is that all encouraged behaviors are encouraged equally; one may easily extend this to varying preferences $c_1 < \ldots < c_k$, and consider a choice between expressing them within a single reward function, or learning a separate value function for each signal c_i.

Role of discounting. Discounting factors γ in RL determine how heavily the future rewards are discounted, i.e. the *reward horizon*. Smaller γ's (i.e. heavier discounting) yield quicker convergence, but may be insufficient to convey long-term goals. In our framework, the value of γ plays two separate roles in the learning process, as it is shared between h and q. Firstly, it determines how quickly h values converge. Since we are only interested in the *difference* of consecutive h-values, smaller γ's provide a more stable estimate, without losses. On the other hand, if the value is too small, q will lose sight of the

long-term rewards, which is detrimental to performance, if the rewards are for the base task alone. We, however, are considering the *shaped* rewards. Since shaped rewards provide informative immediate feedback, it becomes less important to look far ahead into the future. This notion is formalized by [Ng 2003], who proves (in Theorem 3) that a "good" potential function shortens the reward horizon of the original problem. Thus γ, in a sense, balances the stability of learning h with the length of the shaped reward horizon of q.

Limitations. As with all shaping methods, one must be careful about the magnitude of the reward for the base problem, and that of the shaping reward [Harutyunyan et al. 2015a]. In the case of our algorithm this concerns both the magnitude of the reward provided to the secondary value function, and the relative step-size schedule.

In the remainder of this chapter we will apply the framework we formulated in the first part to the challenging scenario of online advice.

6.3 Case Study: Advising Mario with Dynamic PBRS

Advice is an integral part of learning, both for humans and machines. While priceless in some situations, it is heuristic in nature, and may be extremely suboptimal. In RL, where *learning* implies optimizing a given reward function that specifies the task, care must be taken so as to maintain focus on solving that task, and use advice only as guidance. The alternative is to learn to optimize the *advice* itself, which may solve the problem if the advice comes from a perfect oracle, but is likelier to result in suboptimal behaviors, and even prevent solving the problem altogether [Randløv and Alstrøm 1998]. We wish to ensure that regardless of the quality of advice, the agent does not suffer negative consequences for heeding it.

For this, we place ourselves into the PBRS framework [Ng et al. 1999], which gives the necessary form of modifying the reward function of an MDP without altering its (near-)optimal policies. We now make explicit the implicit assumption above of advice being expressed as a reward function r^A. While there is evidence that reward shaping offers more advantages than pure exploration guidance [Harutyunyan et al. 2014, Laud and DeJong 2003], previous attempts of integrating human feedback into the reward scheme have not shown much promise [Knox and Stone 2012, Griffith et al. 2013]. This is unsurprising, as these attempts are either not potential-based, or do not capture the advice properly, since previously there has been no clear way to translate the advice function r^A into h.The framework we described in the beginning of this chapter allows one to express *any* arbitrary reward function in the potential-based form of Eq. (6.1). An important

implication is that we may then leverage r^A that is being provided sporadically *online* (e.g. by a human), while maintaining all guarantees of PBRS.

6.3.1 Setting and Method

We assume a *reward-centric* feedback strategy [Loftin et al. 2014], i.e. all feedback is positive, and punishment is implicit in the absence of feedback. The advice function is then an indicator i defined over the state-action space. We render i as a numerical reward function r^A in a natural way:

$$r^A(s,a) = c \times i(s,a)$$

where c is a scaling constant. Then, following our framework, we learn h^A to express r^A, with the following update rules at each step:

$$\begin{aligned} h^A_{t+1}(S_t,A_t) &\leftarrow h^A_t(S_t,A_t) + \beta_t(-R^A_{t+1} + \gamma h^A_t(S_{t+1},A_{t+1}) - h^A_t(S_t,A_t)) \\ f^A_{t+1} &\leftarrow \gamma h^A_{t+1}(S_{t+1},A_{t+1}) - h^A_t(S_t,A_t) \\ q(S_t,A_t) &\leftarrow q(S_t,A_t) + \alpha_t \left(R_{t+1} + f^A_{t+1} + \gamma q(S_{t+1},A_{t+1}) - q(S_t,A_t)\right) \end{aligned}$$

Note that we do not attempt to solve the advice *delay* problem. In our framework, the advice is implicitly propagated down the trajectory via the eligibility trace, with the remaining effect of delay being treated as noise.

6.3.2 Mario Domain

The Mario benchmark problem [Karakovskiy and Togelius 2012] is based on Infinite Mario Bros, a public reimplementation of Super Mario Bros®. There are 12 discrete actions, corresponding to the buttons (with valid combinations) on a NES controller. Environment rewards correspond to the points collected in the game: the agent is rewarded for killing an enemy, collecting a coin, etc, and punished for getting hurt by a creature or dying. The state space includes information about Mario's state (can jump, can shoot, etc), as well as the coordinates of the closest enemy within a given range. The states are represented tabularly, following the architecture of [Brys et al. 2015]. The state-action values are initialized to 0, resulting in near-random starting behavior.

6.3.3 Experiments

The participants were asked to advise Mario for the first 5 episodes by watching the agent play (at full speed of 25 decisions per second), and pressing a key, whenever in their opinion

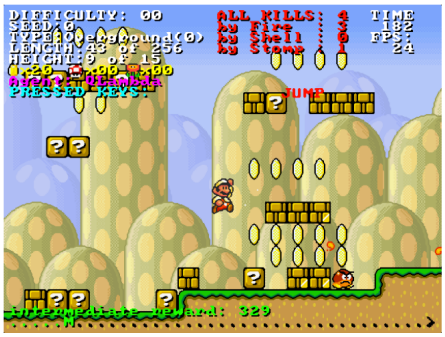

Figure 6.2: A screenshot from Mario executing the level used in the demonstration

it had performed a good action.[8] Recall that feedback is exclusively positive. After the advice had stopped, Mario continued learning on its own for another 95 episodes. We have considered two classes of advice:

Expert advice The advice is provided by a domain expert with detailed knowledge of the state space

Non-expert advice Advisors are unaware of the state space (and, occasionally, of Mario)

Tab. 6.2 gives the comparison between the performance of Mario learning without any advice, with expert advice and with non-expert advice. Each variant is an average of 21 independent runs. The average advising rate was recorded to be 0.015, amounting to ≈ 45 advising steps per trial. Note that there is no significant difference between expert and non-expert advice, suggesting robustness to advice quality.

These results show that even with incredibly sparse advice rates, a large state space, noise incurred by the complexities of the domain and the delay in advice, our method is able to significantly improve the learning performance of Mario.

6.4 Related Work

The correspondence between value and potential functions has been known since the conceivement of the latter. [Ng et al. 1999] point out that the optimal potential function is the true value function itself (as in that case the problem reduces to learning the trivial zero value function). With this insight, there have been attempts to simultaneously learn

[8] See https://vimeo.com/121085629 for an example video.

Variant	Advice phase	Cumulative
Baseline	-376±51	470±83
Non-expert	**401±54**	**677±60**
Expert	**402±62**	**774±47**

Table 6.2: Points collected by Mario in the three considered scenarios (indicated with standard error of the mean). The best ($p < 0.05$) performance is given in bold.

the base value function at coarser and finer granularities (of function approximation), and use the (quicker-to-converge) former as a potential function for the latter [Grzes and Kudenko 2008]. Our approach is different in that our value functions learn on *different* rewards with the same state representation, and it tackles the question of *specification* rather than *efficacy*.

On the other hand, there has been a lot of research in human-provided advice [Thomaz and Breazeal 2006], [Knox et al. 2012]. This line of research (*interactive shaping*) typically uses the human advice component heuristically as a (sometimes annealed) additive component in the reward function, which does not follow the potential-based framework, and thus does not in general preserve policies. [Knox and Stone 2012] do consider PBRS as one of their methods, but (a) stay strictly myopic (similarly to the third variant in the cart-pole experiment), and (b) limit themselves to state potentials. Our approach is different in that it incorporates the external advice through a *value function*, and stays entirely sound in the PBRS framework.

6.5 Summary

In this chapter, we proposed a new reward shaping framework which allows to specify the *effective* shaping reward directly. Given an arbitrary reward function, we learned a secondary value function, w.r.t. a variant of that reward function, concurrently to the main task, and used the consecutive estimates of that value function as dynamic advice potentials. We showed that the shaping reward resulting from this process captures the input reward function in expectation. We presented empirical evidence that the method behaves in a true potential-based manner, and that such encoding of the behavioral domain knowledge speeds up learning significantly more, compared to its alternatives. The framework induces little added complexity: the maintenance of the auxiliary value function is linear in time and space [Modayil et al. 2012], and, when initialized to 0, the optimal base value function is unaffected.

With the inherent noise and lack of qualitative guarantees in human advice, it is imperative to be cautious when integrating it in the RL process. As PBRS specifies the *necessary* form of modifying an MDP without altering optimality, it seems a natural choice for modeling human advice. We demonstrated the performance of a framework that is able to do this for the first time, showing promise for reward shaping methods in this avenue.

7 | Conclusions

In an effort to enrich the basic temporal difference step to enable learning with sophistication, this dissertation touched upon three major research areas: learning off-policy from multiple steps, learning with options, and learning with shaping rewards. There are three motivating threads that can be traced through our work in these research areas: *efficiency*, *safety*, and *optimality*.

Learning from multiple steps is more efficient. On the other hand, it is crucial for safety to be able to learn about courses of actions, without explicitly trying them, to learn off-policy. Further yet, if one is to learn off-policy from multiple steps, we wish for such learning to utilize the steps as efficiently as possible, and for us to be sure that the learning will optimally converge to the desired outcomes. Chapter 3 proposes algorithms to achieve this.

Learning with longer, temporally abstract options is more efficient for the same reasons, as it is with multiple steps. But it is also safer, as with long sensible options there is less potential for noise and jitter during learning. On the other hand, the shorter the options, the more chances one has to pick among them optimally. Chapter 4 describes an algorithm that removes this conflict between efficiency and safety on one hand, and optimality on the other.

When optimality cannot be captured with primitive actions with a finite horizon, Chapter 4 gives an option discounting scheme that allows for options to do so. This discounting introduces a general tradeoff when using options: between bias and variance, optimality and efficiency.

One of the motivations of reward shaping is its use as a mechanism to guide exploration, which is crucial for safety. Potential-based reward shaping ensures optimality is preserved. Chapter 6 addresses the need to efficiently utilize arbitrary information in the required form.

7.1 Future Directions

The work in this dissertation has inspired many tangential ideas. Below we briefly outline a few of the ones we find particularly fruitful.

7.1.1 Learning Longer Options

End-to-end option discovery often ends up converging to degenerate, one-step options [Bacon et al. 2017, Mann et al. 2014]. Longer options help in the beginning, but lose significance at the end. This is unsatisfying, since one typically wishes to distill meaningful behavior in the options, perhaps in order to transfer them to a different task.

One possible reason for this could be the continued anchoring to the primitive time step. With it, each option is indeed no different from a sequence of primitive actions, and there is no benefit in using options within the existing framework, as soon as the equivalent sequence of primitive actions has been found. Adding a *deliberation* cost [Bacon and Precup 2015, Harb et al. 2018], which incurs a penalty at decision points is one possibility of addressing this. The ideas in Chapter 5 lend themselves to another. We can imagine optimizing each option internally on its own timescale, while externally treating the options irrespectively of their duration. This may make primitive actions less appealing to fall back on, since each option will now be itself in some sense no different from a primitive action, albeit more effective.

Another possible solution is to use the ideas from Chapter 4 and off-policy impose longer durations on the options that are being learnt. This is similar to learning *time-regularized* options from [Mann et al. 2014], but instead of the explicit regularization and option interruption, here the desired effect would be imposed latently off-policy.

7.1.2 Distributing Time with Options

Consider the following example. An agent is starting to learn a task that takes a thousand steps to solve. If the agent is to have hope to solve it, its discount factor γ must be able to capture the thousand steps, that is it must be at least 0.999. Now, imagine that at the end of learning, the agent has discovered that the task is composed of two options, each taking five hundred steps. Each option must be able to represent its own horizon, which is

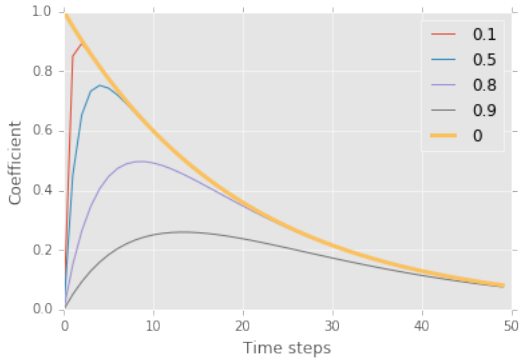

Figure 7.1: Second-order discounting coefficients for $\gamma_1 = 0.95$.

now shorter, but the outer task has reduced to the mere two steps! A plan over two steps can be represented with a γ of as low as 0.5. Of course, one is free to continue using the higher discount, but it is unnecessary, and, as is known, may yield slower learning.

Motivated by this example, one can imagine using the ideas in Chapter 5 to *distribute* the time horizon of the agent, as its actions get more sophisticated, with the slowest discounting found at the bottom of the hierarchy.

7.1.3 Second-Order Discounting

Consider two discounts γ_1 and γ_2, and consider the *difference* in values of a policy π w.r.t. these discounts:

$$v^{\pi}_{\gamma_1,\gamma_2}(s) = v^{\pi}_{\gamma_1}(s) - v^{\pi}_{\gamma_2}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}(\gamma_1^t - \gamma_2^t)R_{t+1}\right].$$

Figure 7.1 plots the coefficients of this function for a fixed γ_1 and different values of γ_2.

The two discounting rates together incur a specific case of *second-order* exponential decay: the rewards at nearby time steps weigh *less* that those temporally removed. Such a setting may be useful in noisy problems where closer rewards whose discount coefficients are larger drown out the more important long-horizon information. It is also appropriate in real-world scenarios in which the environment incurs a physical *time-delay* in displaying the consequences of an agent's action. In fact, to handle such time delay a similar shape has been considered for *eligibility traces* in a predictive neuroscience model [Kettner et al. 1997]. In RL, eligibility traces do not impact the solution, and to capture the value function w.r.t. the delayed rewards, the mechanism must be realized through discounting.

7.1.4 Shaping in Time

The classical meaning of the term shaping is to build up to solving a complex task via a sequence of simpler related tasks. This idea, sometimes referred to as *scaffolding* can be realized in PBRS via considering a sequence of potential functions that increase in sophistication.

In line with the successive approximation work of [Chow and Tsitsiklis 1991], there have been efforts to scaffold the potential function from fast-to-learn coarse features to slow-but-accurate fine features [Grzes and Kudenko 2009]. It is possible to envision something similar, but in *time*, rather than in space, that is: scaffolding potential functions from more to less discounting.[1] The more discounting (i.e. the smaller the γ), the easier it is to learn a task, but the more short-sighted the resulting value function.

More precisely, given an MDP with some discount factor γ_{env}, one can simultaneously estimate value functions (v_{γ_i}) w.r.t. $\gamma_1 < \gamma_2 < \ldots < \gamma_n$, and use these value functions as potentials in the increasing order of γ. The benefit of the indirect way of using potentials is that due to the optimality guarantees of PBRS, the schedule of transitioning between potential functions is less important.

7.1.5 Formal Recommendations of Potential Functions

There has been very little work on formal properties of what constitutes good potential functions, apart from the seminal result of [Ng et al. 1999], in which they show that the ideal potential function is the true value function itself. This is crucial, but hardly helpful in practice.

The question is particularly interesting when considering *dynamic* PBRS, since there are two dimensions at play there: (1) the distance of the potential function and the target value function, and (2) the evolution of the estimates of the two quantities. In particular, it is possible that the potential function evolves "in the same direction" of the target, and reduces the worst case errors along the way, affecting the contraction coefficient itself.

More generally, it is interesting to devise general guidelines for potential functions that guarantee a speed up in learning, under some assumptions on the MDP.

[1] Note that coarser features reduce the granularity of the agent's horizon, but do not cause it to be more myopic, as a smaller discount does.

Publications by the Author

Most of the chapters in this thesis are based on peer-reviewed publications. We briefly summarize the relationships here. Chapter 3 is based on [Harutyunyan et al. 2016] and the subsequent [Munos et al. 2016]. Chapter 4 is in part based on [Harutyunyan et al. 2018] and in part on previously unpublished results concerning the gated model. The content in Chapter 5 is novel, and is currently in submission. Chapter 6 is based on [Harutyunyan et al. 2015c] and the related demonstration [Harutyunyan et al. 2015b].

We entirely omit [Harutyunyan et al. 2014] and [Harutyunyan et al. 2015a] which describe the Horde of shapings architecture. In these works, the key idea is to instead of relying on a single potential function, as is common, to learn an *ensemble* of potentials, and recombine them online. The benefit of this is that it is difficult to construct a single good potential a priori, not only because the underlying heuristic may be flawed, but even because the numerical scale of the potential impacts the results significantly [Harutyunyan et al. 2015a]. We show that learning a multitude of variants online, and recombining them even with a simple voting scheme, is able to match or exceed the performance of the best pre-selected variant. We refer the reader to [Brys et al. 2017] for a detailed treatment of these and related ideas.

A significant part of the PhD was spent working on the MIRAD lower limb exoskeleton, for which event prediction and detection were performed. In particular, we deployed neural networks and Hidden Markov Models to perform predictive gait segmentation and prediction of certain events (such as *seat-off*), which were necessary for smooth control. We choose to omit this content here, as the learning methods were all supervised. [Tanghe et al. 2016], [Lambrecht et al. 2017], as well as a manuscript in preparation cover some of that work.

Key Publications for this Dissertation

Proceedings of Conferences with International Referees

1. **Anna Harutyunyan**, Peter Vrancx, Pierre-Luc Bacon, Doina Precup and Ann Nowé. *Learning with Options that Terminate Off-Policy.* In Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI). To appear. 2018.

2. **Anna Harutyunyan**, Marc G. Bellemare, Tom Stepleton, and Rémi Munos. *Q(λ) with off-policy corrections.* In Proceedings of Algorithmic Learning Theory (ALT), pages 305–320. Lecture Notes in Computer Science, vol 9925. 2016.

3. Rémi Munos, Tom Stepleton, **Anna Harutyunyan**, and Marc G. Bellemare. *Safe and efficient off-policy reinforcement learning.* In Proceedings of Neural Information Processing Systems (NIPS), pages 1054–1062, 2016.

4. **Anna Harutyunyan**, Sam Devlin, Peter Vrancx, and Ann Nowé. *Expressing Arbitrary Reward Functions as Potential-Based Advice.* In Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI), pages 2652–2658, 2015.

Manuscripts

1. **Anna Harutyunyan**, Peter Vrancx, Ann Nowé and Doina Precup. *Discounting Options.* In submission.

Complete List of Publications

Proceedings of International Conferences

1. **Anna Harutyunyan**, Peter Vrancx, Pierre-Luc Bacon, Doina Precup and Ann Nowé. *Learning with Options that Terminate Off-Policy.* In Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI). To appear. 2018.

2. Rémi Munos, Tom Stepleton, **Anna Harutyunyan**, and Marc G. Bellemare. *Safe and efficient off-policy reinforcement learning.* In Proceedings of Neural Information Processing Systems (NIPS), pages 1054–1062, 2016.

3. **Anna Harutyunyan**, Marc G. Bellemare, Tom Stepleton, and Rémi Munos. *Q(λ) with off-policy corrections.* In Proceedings of Algorithmic Learning Theory (ALT), pages 305-320, 2016. Lecture Notes in Computer Science, vol 9925. 2016.

4. Tim Brys, **Anna Harutyunyan**, Halit Bener Suay, Sonia Chernova, Matthew E. Taylor, and Ann Nowé. *Reinforcement Learning from Demonstration through Shaping.* In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pages 3352-3358, 2015.

5. **Anna Harutyunyan**, Sam Devlin, Peter Vrancx, and Ann Nowé. *Expressing Arbitrary Reward Functions as Potential-Based Advice.* In Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI), pages 2652–2658, 2015.

6. Tim Brys, **Anna Harutyunyan**, Matthew E. Taylor, and Ann Nowé. *Policy Transfer using Reward Shaping.* In Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pages 181–188, 2015.

7. Tim Brys, **Anna Harutyunyan**, Peter Vrancx, Matthew E. Taylor, Daniel Kudenko, and Ann Nowé. *Multi-Objectivization of Reinforcement Learning Problems by Reward Shaping.* In Proceedings of the International Joint Conference on Neural Networks (IJCNN), pages 2315–2322, 2014.

Extended Abstracts

1. **Anna Harutyunyan**, Tim Brys, Peter Vrancx, and Ann Nowé. *Multi-Scale Reward Shaping via an Off-Policy Ensemble.* In Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pages 1641–1642, 2015.

2. **Anna Harutyunyan**, Tim Brys, Peter Vrancx, and Ann Nowé. *Off-Policy Shaping Ensembles in Reinforcement Learning.* In Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI), pages 1021–1022, 2014.

3. **Anna Harutyunyan**, Tim Brys, Peter Vrancx, and Ann Nowé. *Shaping Mario with Human Advice (Demonstration).* In Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pages 1913–1914, 2015.

Refereed Workshops

1. **Anna Harutyunyan**, Tim Brys, Peter Vrancx, and Ann Nowé. *Off-Policy Reward Shaping with Ensembles.* In Autonomous Learning Agents (ALA) Workshop at AAMAS, 2015. **Best Paper Award.**

2. **Anna Harutyunyan**, Peter Vrancx, Pierre-Luc Bacon, Doina Precup and Ann Nowé. *Learning with Options that Terminate Off-Policy.* In Hierarchical Reinforcement Learning Workshop at NIPS, 2017. **Best Paper Award.**

International Journals

1. Stefan Lambrecht, **Anna Harutyunyan**, Kevin Tanghe, Maarten Afschrift, Joris De Schutter, Ilse Jonkers. *Real-Time Gait Event Detection Based on Kinematic Data Coupled to a Biomechanical Model.* Sensors 17, no. 4: 671. 2017.

2. Tim Brys, **Anna Harutyunyan**, Peter Vrancx, Matthew E. Taylor and Ann Nowé. *Multi-objectivization and Ensembles of Shapings in Reinforcement Learning.* Neurocomputing 263: 48–59. 2017.

3. Kevin Tanghe, **Anna Harutyunyan**, Erwin Aertbelien, Friedl De Groote, Joris De Schutter, Peter Vrancx, and Ann Nowé. *Predicting Seat-Off and Detecting Start-of-Assistance Events for Assisting Sit-to-Stand with an Exoskeleton.* Robotics and Automation Letters, IEEE, 1(2):792–799, 2016.

In Preparation

1. **Anna Harutyunyan**, Stefan Lambrecht, Kevin Tanghe, Erwin Aertbelien, Friedl De Groote, Joris De Schutter, and Ann Nowé. *Predictive Gait Segmentation with Artificial Neural Networks.*

A | Proofs from Chapter 3

A.1 Proofs of Lemma 3.1 and Lemma 3.2

Proof of Lemma 3.1. First notice that

$$\begin{aligned}\|(\mathcal{P}^\pi - \mathcal{P}^\mu)q\| &= \max_{s,a} \Big| \sum_{s'} p(s'|s,a) \sum_{a'} \left((\pi(a'|s') - \mu(a'|s'))\, q(s',a')\right) \Big| \\ &\leq \max_{s,a} \sum_{s'} p(s'|s,a) \sum_{a'} |\pi(a'|s') - \mu(a'|s')| \|q\| \leq \epsilon \|q\|. \end{aligned} \tag{A.1}$$

Let $\mathcal{B} = (I - \lambda\gamma\mathcal{P}^\mu)^{-1}$ be the resolvent. From (3.7) we have

$$\begin{aligned}\mathcal{R}_\lambda^\pi q - q^\pi &= \mathcal{B}\left(\mathcal{T}^\pi q - q + (I - \lambda\gamma\mathcal{P}^\mu)(q - q^\pi)\right) \\ &= \mathcal{B}\left(r + \gamma\mathcal{P}^\pi q - q^\pi - \lambda\gamma\mathcal{P}^\mu(q - q^\pi)\right) \\ &= \mathcal{B}\left(\gamma\mathcal{P}^\pi(q - q^\pi) - \lambda\gamma\mathcal{P}^\mu(q - q^\pi)\right) \\ &= \gamma\mathcal{B}\big[(1-\lambda)\mathcal{P}^\pi + \lambda(\mathcal{P}^\pi - \mathcal{P}^\mu)\big](q - q^\pi).\end{aligned}$$

Taking the sup-norm, since μ is ϵ-away from π as in Eq. (A.1):

$$\|\mathcal{R}_\lambda^\pi q - q^\pi\| \leq \eta \|q - q^\pi\|$$

for $\eta = \frac{\gamma}{1-\lambda\gamma}(1 - \lambda + \lambda\epsilon) < 1$. Thus $\|q_k - q^\pi\| = O(\eta^k)$. □

Proof of Lemma 3.2. Fix μ. Using (3.8), we write

$$\mathcal{R}_\lambda^* q - q^* = (I - \lambda\gamma\mathcal{P}^\mu)^{-1}\left[\mathcal{T}q - q + (I - \lambda\gamma\mathcal{P}^\mu)(q - q^*)\right]$$

$$= (I - \lambda\gamma\mathcal{P}^{\mu})^{-1} \left[\mathcal{T}q - q^* - \lambda\gamma\mathcal{P}^{\mu}(q - q^*)\right].$$

Taking the sup-norm, since $\|\mathcal{T}q - q^*\| \leq \gamma\|q - q^*\|$, we deduce the result:

$$\|\mathcal{R}^*_{\lambda}q - q^*\| \leq \frac{\gamma + \lambda\gamma}{1 - \lambda\gamma} \|q - q^*\|.$$

□

A.2 Proof of Theorems 3.1 and 3.2

We will appeal to Proposition 5.2 from [Bertsekas and Tsitsiklis 1996] to prove the online convergence of the algorithms. This will require rewriting the update in the suitable form, and verifying Assumptions (a) through (d) from their Proposition 4.5. The sketch below applies to both policy evaluation and control, for the values of λ prescribed by Lemmas 3.1 and 3.2.

Proof (Sketch). Let $z_{k,t}(s,a) \stackrel{\text{def}}{=} \sum_{i=0}^{t}(\gamma\lambda)^{t-i}\mathbb{I}\{(S_i, A_i) = (s,a)\}$ denote the accumulating trace. It follows from Assumptions 2.3 and 3.1 that the total update at phase k is bounded, which allows us to write the online version of (3.6) as

$$\begin{aligned} q^o_{k+1}(s,a) &\leftarrow (1 - D_k\alpha_k)q^o_k(s,a) + D_k\alpha_k\big(\mathcal{R}^{\pi_k,\mu_k}_{\lambda}q^o_k(s,a) + w_k + u_k\big) \\ w_k &\stackrel{\text{def}}{=} (D_k)^{-1}\Big[\sum_{t\geq 0} z_{k,t}\delta^{\pi_k}_t - \mathbb{E}_{\mu_k}\big[\sum_{t\geq 0} z_{k,t}\delta^{\pi_k}_t\big]\Big], \\ u_k &\stackrel{\text{def}}{=} (D_k\alpha_k)^{-1}\big(q^o_{k+1}(s,a) - q_{k+1}(s,a)\big), \end{aligned}$$

where $D_k(s,a) \stackrel{\text{def}}{=} \sum_{t\geq 0}\Pr\{S_t, A_t = s,a\}$, and we drop the (s,a) argument for α_k, D_k, w_k, u_k, and $z_{k,t}$. Combining Assumptions 2.3 and 3.1, we have $0 < D \leq D_k(s,a) < \infty$, which, combined in turn with Assumption 2.2, assures that the new stepsize sequence $\tilde{\alpha}_k(s,a) = (D_k\alpha_k)(s,a)$ satisfies Assumption (a) of Prop. 4.5. Assumptions (b) and (d) require the variance of the noise term $w_k(s,a)$ to be bounded, and the residual $u_k(s,a)$ to converge to zero, both of which can be shown identically to the corresponding results from [Bertsekas and Tsitsiklis 1996], if Assumption 3.1 and Assumption (a) are satisfied. Finally, Assumption (c) is satisfied by Lemmas 3.1 and 3.2 for the policy evaluation and control cases, respectively. Therefore, the sequence $(q^o_k)_{k\in\mathbb{N}}$ converges to q^{π} or q^* in the respective settings, w.p. 1. □

A.3 Proof of Proposition 3.1

Proof. Writing the algorithm in operator form, we get

$$\begin{aligned}\mathcal{R}q &= (1-\lambda)\sum_{n\geq 0}\lambda^n\Big[\sum_{t=0}^{n}\gamma^t(\mathcal{P}^\mu)^t r + \gamma^{n+1}(\mathcal{P}^\mu)^n\mathcal{P}^\pi q\Big] \\ &= \sum_{t=0}^{\infty}(\lambda\gamma)^t(\mathcal{P}^\mu)^t\Big[r+(1-\lambda)\gamma\mathcal{P}^\pi q\Big] = (I-\lambda\gamma\mathcal{P}^\mu)^{-1}\Big[r+(1-\lambda)\gamma\mathcal{P}^\pi q\Big].\end{aligned}$$

Thus, the fixed point $q^{\mu,\pi}$ of $\mathcal{R}$ satisfies the following:

$$q^{\mu,\pi} = (I-\lambda\gamma\mathcal{P}^\mu)^{-1}\left[r+(1-\lambda)\gamma\mathcal{P}^\pi q^{\mu,\pi}\right] = (1-\lambda)\mathcal{T}^\pi q^{\mu,\pi} + \lambda\mathcal{T}^\mu q^{\mu,\pi}.$$

Solving for $q^{\mu,\pi}$ yields the result. □

A.4 Proof of Theorem 3.3

The following lemma will be useful in proving Theorem 3.3.

Lemma A.1

The difference between $\mathcal{U}q$ and its fixed point q^π is

$$\mathcal{U}q(s,a) - q^\pi(s,a) = \mathbb{E}_\mu\Big[\sum_{t\geq 1}\gamma^t\Big(\prod_{i=1}^{t-1}c_i\Big)\Big(\big[\mathbb{E}_\pi[(q-q^\pi)(S_t,\cdot)] - c_t(q-q^\pi)(S_t,A_t)\big]\Big)\Big].$$

Proof. The fact that q^π is the fixed point of the operator $\mathcal{U}$ is obvious from (3.17) since $\mathbb{E}_{S_{t+1}\sim p(\cdot|S_t,A_t)}\big[R_{t+1}+\gamma\mathbb{E}_\pi q^\pi(S_{t+1},\cdot) - q^\pi(S_t,A_t)\big] = (\mathcal{T}^\pi q^\pi - q^\pi)(S_t,A_t) = 0$, since q^π is the fixed point of $\mathcal{T}^\pi$. Now, let $\Delta q \stackrel{\text{def}}{=} q - q^\pi$. We begin by rewriting Eq. (3.17):

$$\mathcal{U}q(s,a) = \sum_{t\geq 0}\gamma^t\mathbb{E}_\mu\left[\Big(\prod_{i=1}^{t}c_i\Big)\Big(R_{t+1}+\gamma\left[\mathbb{E}_\pi q(S_{t+1},\cdot) - c_{t+1}q(S_{t+1},A_{t+1})\Big)\right]\right].$$

$$q^\pi(s,a) = \mathcal{U}q^\pi(s,a) = \sum_{t\geq 0}\gamma^t\mathbb{E}_\mu\left[\Big(\prod_{i=1}^{t}c_i\Big)\Big(R_{t+1}+\gamma\left[\mathbb{E}_\pi q^\pi(S_{t+1},\cdot) - c_{t+1}q^\pi(S_{t+1},A_{t+1})\Big)\right]\right],$$

from which we deduce that

$$\begin{aligned}\mathcal{U}q(s,a) - q^\pi(s,a) &= \sum_{t\geq 0}\gamma^t \mathbb{E}_\mu\Big[\Big(\prod_{i=1}^{t} c_i\Big)\Big(\gamma\big[\mathbb{E}_\pi \Delta q(S_{t+1},\cdot) - c_{t+1}\Delta q(S_{t+1},A_{t+1})\big]\Big)\Big] \\ &= \sum_{t\geq 1}\gamma^t \mathbb{E}_\mu\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\Big(\big[\mathbb{E}_\pi \Delta q(S_t,\cdot) - c_t\Delta q(S_t,A_t)\big]\Big)\Big].\end{aligned}$$

□

Proof (Theorem 3.3). Since q^π is the fixed point of $\mathcal{U}$, from Lemma A.1, and defining $\Delta q \overset{\text{def}}{=} q - q^\pi$, we have

$$\begin{aligned}\mathcal{U}q(s,a) - q^\pi(s,a) &= \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\Big(\big[\mathbb{E}_\pi \Delta q(S_t,\cdot) - c_t\Delta q(S_t,A_t)\big]\Big)\Big] \\ &= \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t-1}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\Big(\big[\mathbb{E}_\pi \Delta q(S_t,\cdot) - \mathbb{E}_{A_t}[c_t(A_t,\mathcal{F}_t)\Delta q(S_t,A_t)|\mathcal{F}_t]\big]\Big)\Big] \\ &= \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t-1}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\sum_b \big(\pi(b|S_t) - \mu(b|S_t)c_t(b,\mathcal{F}_t)\big)\Delta q(S_t,b)\Big].\end{aligned}$$

Now since $\pi(b|S_t) - \mu(b|S_t)c_t(b,\mathcal{F}_t) \geq 0$, we have that $\mathcal{U}q(s,a) - q^\pi(s,a) = \sum_{y,b} w_{y,b}\Delta q(y,b)$, i.e. a linear combination of $\Delta q(y,b)$ weighted by non-negative coefficients:

$$w_{y,b} \overset{\text{def}}{=} \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t-1}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\big(\pi(b|S_t) - \mu(b|S_t)c_t(b,\mathcal{F}_t)\big)\mathbb{I}\{S_t = y\}\Big].$$

The sum of those coefficients is:

$$\begin{aligned}\sum_{y,b} w_{y,b} &= \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t-1}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\sum_b \big(\pi(b|S_t) - \mu(b|S_t)c_t(b,\mathcal{F}_t)\big)\Big] \\ &= \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t-1}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)\mathbb{E}_{A_t}[1 - c_t(A_t,\mathcal{F}_t)|\mathcal{F}_t]\Big] = \sum_{t\geq 1}\gamma^t \mathbb{E}_{S_{1:t},A_{1:t}}\Big[\Big(\prod_{i=1}^{t-1} c_i\Big)(1 - c_t)\Big] \\ &= \mathbb{E}_\mu\Big[\sum_{t\geq 1}\gamma^t\Big(\prod_{i=1}^{t-1} c_i\Big) - \sum_{t\geq 1}\gamma^t\Big(\prod_{i=1}^{t} c_i\Big)\Big] = \gamma C - (C-1),\end{aligned}$$

where $C = \mathbb{E}_\mu\big[\sum_{t\geq 0}\gamma^t\big(\prod_{i=1}^{t} c_i\big)\big]$. Since $C \geq 1$, we have that $\sum_{y,b} w_{y,b} \leq \gamma$. Thus $\mathcal{U}q(s,a) - q^\pi(s,a)$ is a sub-convex combination of $\Delta q(y,b)$ weighted by non-negative

coefficients $w_{y,b}$ which sum to (at most) γ, thus $\mathcal{U}$ is a γ-contraction mapping around q^π. □

A.5 Increasingly Greedy Policies

Recall the definition of an increasingly greedy sequence of policies.

Definition A.1

We say that a sequence of policies (π_k) is increasingly greedy w.r.t. a sequence of functions (q_k) if the following property holds for all k:

$$\mathcal{P}^{\pi_{k+1}} q_{k+1} \geq \mathcal{P}^{\pi_k} q_{k+1}.$$

It is obvious to see that this property holds if all policies π_k are greedy w.r.t. q_k. Indeed in such case, $\mathcal{T}^{\pi_{k+1}} q_{k+1} = \mathcal{T} q_{k+1} \geq \mathcal{T}^{\pi} q_{k+1}$ for any π. We now prove that this property holds for ϵ_k-greedy policies (with non-increasing (ϵ_k)) in the following lemma. Of course not all policies satisfy this property (a counter-example being $\pi_k(a|s) = \arg\min_{a'} q_k(s, a')$).

Lemma A.2

Let (ϵ_k) be a non-increasing sequence. Then the sequence of policies (π_k) which are ϵ_k-greedy w.r.t. the sequence of functions (Q_k) is increasingly greedy w.r.t. that sequence.

Proof. From the definition of an ϵ-greedy policy we have:

$$\begin{aligned}
\mathcal{P}^{\pi_{k+1}} q_{k+1}(s,a) &= \sum_y p(y|s,a)\big[(1-\epsilon_{k+1}) \max_b q_{k+1}(y,b) + \epsilon_{k+1} \frac{1}{|\mathcal{A}|} \sum_b q_{k+1}(y,b)\big] \\
&\geq \sum_y p(y|s,a)\big[(1-\epsilon_{k}) \max_b q_{k+1}(y,b) + \epsilon_{k} \frac{1}{|\mathcal{A}|} \sum_b q_{k+1}(y,b)\big] \\
&\geq \sum_y p(y|s,a)\big[(1-\epsilon_{k}) q_{k+1}(y, \arg\max_b q_k(y,b)) + \epsilon \frac{1}{|\mathcal{A}|} \sum_b q_{k+1}(y,b)\big] \\
&= \mathcal{P}^{\pi_k} q_{k+1},
\end{aligned}$$

where we used the fact that $\epsilon_{k+1} \leq \epsilon_k$. □

A.6 Proof of Theorem 3.4

As mentioned in the main text, since c_i is Markovian, we can define the (sub)-probability transition operator

$$(\mathcal{P}^{c\mu}q)(s,a) \stackrel{\text{def}}{=} \sum_{s'}\sum_{a'} p(s'|s,a)\mu(a'|s')c(a',s')q(s',a').$$

The Retrace(λ) operator then writes

$$\mathcal{U}_k q = q + \sum_{t\geq 0}\gamma^t(\mathcal{P}^{c\mu_k})^t(\mathcal{T}^{\pi_k}q - q) = q + (I-\gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q - q).$$

Proof. We now lower- and upper-bound the term $q_{k+1} - q^*$.

Upper bound on $q_{k+1} - q^*$. Since $Q_{k+1} = \mathcal{U}_k q_k$, we have

$$\begin{aligned}
Q_{k+1} - q^* &= q_k - q^* + (I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\mathcal{T}^{\pi_k}q_k - q_k\big] \\
&= (I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\mathcal{T}^{\pi_k}q_k - q_k + (I-\gamma\mathcal{P}^{c\mu_k})(q_k - q^*)\big] \\
&= (I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\mathcal{T}^{\pi_k}q_k - q^* - \gamma\mathcal{P}^{c\mu_k}(q_k - q^*)\big] \\
&= (I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\mathcal{T}^{\pi_k}q_k - \mathcal{T}q^* - \gamma\mathcal{P}^{c\mu_k}(q_k - q^*)\big] \\
&\leq (I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\gamma\mathcal{P}^{\pi_k}(q_k - q^*) - \gamma\mathcal{P}^{c\mu_k}(q_k - q^*)\big] \\
&= \gamma(I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k}\big](q_k - q^*), \\
&= \mathcal{A}_k(q_k - q^*), \qquad \text{(A.2)}
\end{aligned}$$

where $\mathcal{A}_k = \gamma(I-\gamma\mathcal{P}^{c\mu_k})^{-1}\big[\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k}\big]$.

Now let us prove that $\mathcal{A}_k$ has non-negative elements, whose sum over each row is at most γ. Let e be the vector with 1-components. By rewriting $\mathcal{A}_k$ as $\gamma\sum_{t\geq 0}\gamma^t(\mathcal{P}^{c\mu_k})^t(\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k})$ and noticing that

$$(\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k})e(s,a) = \sum_{s'}\sum_{a'} p(s'|s,a)[\pi_k(a'|s') - c(a',s')\mu_k(a'|s')] \geq 0, \qquad \text{(A.3)}$$

it is clear that all elements of $\mathcal{A}_k$ are non-negative. We have

$$\begin{aligned}
\mathcal{A}_k e &= \gamma\sum_{t\geq 0}\gamma^t(\mathcal{P}^{c\mu_k})^t\big[\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k}\big]e \\
&= \gamma\sum_{t\geq 0}\gamma^t(\mathcal{P}^{c\mu_k})^t e - \sum_{t\geq 0}\gamma^{t+1}(\mathcal{P}^{c\mu_k})^{t+1}e
\end{aligned}$$

$$\begin{aligned} &= e - (1-\gamma)\sum_{t\geq 0}\gamma^t(\mathcal{P}^{c\mu_k})^t e \\ &\leq \gamma e, \end{aligned} \tag{A.4}$$

(since $\sum_{t\geq 0}\gamma^t(\mathcal{P}^{c\mu_k})^t e \geq e$). Thus $\mathcal{A}_k$ has non-negative elements, whose sum over each row, is at most γ. We deduce from (A.2) that $q_{k+1}-q^*$ is upper-bounded by a sub-convex combination of components of $q_k - q^*$; the sum of their coefficients is at most γ. Thus

$$q_{k+1} - q^* \leq \gamma\|q_k - q^*\|e. \tag{A.5}$$

Lower bound on $q_{k+1} - q^*$. We have

$$\begin{aligned} q_{k+1} &= q_k + (I - \gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k) \\ &= q_k + \sum_{i\geq 0}\gamma^i(\mathcal{P}^{c\mu_k})^i(\mathcal{T}^{\pi_k}q_k - q_k) \\ &= \mathcal{T}^{\pi_k}q_k + \sum_{i\geq 1}\gamma^i(\mathcal{P}^{c\mu_k})^i(\mathcal{T}^{\pi_k}q_k - q_k) \\ &= \mathcal{T}^{\pi_k}q_k + \gamma\mathcal{P}^{c\mu_k}(I - \gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k). \end{aligned} \tag{A.6}$$

Now, from the definition of ϵ_k we have $\mathcal{T}^{\pi_k}q_k \geq \mathcal{T}q_k - \epsilon_k\|q_k\| \geq \mathcal{T}^{\pi^*}q_k - \epsilon_k\|q_k\|$, thus

$$\begin{aligned} q_{k+1} - q^* &= q_{k+1} - \mathcal{T}^{\pi_k}q_k + \mathcal{T}^{\pi_k}q_k - \mathcal{T}^{\pi^*}q_k + \mathcal{T}^{\pi^*}q_k - \mathcal{T}^{\pi^*}q^* \\ &\geq q_{k+1} - \mathcal{T}^{\pi_k}q_k + \gamma P^{\pi^*}(q_k - q^*) - \epsilon_k\|q_k\|e \end{aligned}$$

Using (A.6) we derive the lower bound:

$$q_{k+1} - q^* \geq \gamma\mathcal{P}^{c\mu_k}(I - \gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k) + \gamma\mathcal{P}^{\pi^*}(q_k - q^*) - \epsilon_k\|q_k\|. \tag{A.7}$$

Lower bound on $\mathcal{T}^{\pi_k}q_k - q_k$. By hypothesis, (π_k) is increasingly greedy w.r.t. (q_k), thus

$$\begin{aligned} \mathcal{T}^{\pi_{k+1}}q_{k+1} - q_{k+1} &\geq \mathcal{T}^{\pi_k}q_{k+1} - q_{k+1} \\ &= \mathcal{T}^{\pi_k}\mathcal{U}q_k - \mathcal{U}q_k \\ &= r + (\gamma\mathcal{P}^{\pi_k} - I)\mathcal{U}q_k \\ &= r + (\gamma\mathcal{P}^{\pi_k} - I)\big[q_k + (I - \gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k)\big] \\ &= \mathcal{T}^{\pi_k}q_k - q_k + (\gamma\mathcal{P}^{\pi_k} - I)(I - \gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k) \\ &= \gamma\big[\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k}\big](I - \gamma\mathcal{P}^{c\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k) \end{aligned}$$

$$= \quad \mathcal{B}_k(\mathcal{T}^{\pi_k} q_k - q_k), \tag{A.8}$$

where $\mathcal{B}_k = \gamma[\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k}](I - \gamma\mathcal{P}^{c\mu_k})^{-1}$. Since $\mathcal{P}^{\pi_k} - \mathcal{P}^{c\mu_k}$ has non-negative elements (as proven in (A.3)) as well as $(I - \gamma\mathcal{P}^{c\mu_k})^{-1}$, then $\mathcal{B}_k$ has non-negative elements as well. Thus

$$\mathcal{T}^{\pi_k} q_k - q_k \geq \mathcal{B}_{k-1}\mathcal{B}_{k-2} \dots \mathcal{B}_0(\mathcal{T}^{\pi_0} q_0 - q_0) \geq 0,$$

since we assumed $\mathcal{T}^{\pi_0} q_0 - q_0 \geq 0$. Thus (A.7) implies that

$$q_{k+1} - q^* \quad \geq \quad \gamma\mathcal{P}^{\pi^*}(q_k - q^*) - \epsilon_k \|q_k\|.$$

and combining the above with (A.5) we deduce

$$\|q_{k+1} - q^*\| \leq \gamma\|q_k - q^*\| + \epsilon_k\|q_k\|.$$

Now assume that $\epsilon_k \to 0$. We first deduce that q_k is bounded. Indeed as soon as $\epsilon_k < (1-\gamma)/2$, we have

$$\|q_{k+1}\| \leq \|q^*\| + \gamma\|q_k - q^*\| + \frac{1-\gamma}{2}\|q_k\| \leq (1+\gamma)\|q^*\| + \frac{1+\gamma}{2}\|q_k\|.$$

Thus $\lim \|q_k\| \leq \frac{1+\gamma}{1-(1+\gamma)/2}\|q^*\|$. Since q_k is bounded, we deduce that $\lim q_k = q^*$. □

A.7 Proof of Theorem 3.5

We first prove the convergence of a general online algorithm.

Theorem A.1

Consider the algorithm

$$q_{k+1}(s,a) = (1-\alpha_k(s,a))q_k(s,a) + \alpha_k(s,a)(\mathcal{U}_k q_k(s,a) + \omega_k(s,a) + \upsilon_k(s,a)), \tag{A.9}$$

and assume that (1) ω_k is a centered, $\mathcal{F}_k$-measurable noise term of bounded variance, and (2) υ_k is bounded from above by $\theta_k(\|q_k\| + 1)$, where (θ_k) is a random sequence that converges to 0 a.s. Then, under the same assumptions as in Theorem 3.5, we have that $q_k \to q^*$ almost surely.

Proof. We write $\mathcal{U}$ for $\mathcal{U}_k$. Let us prove the result in three steps.

Upper bound on $\mathcal{U}q_k - q^*$. The first part of the proof is similar to the proof of (A.5), so we have

$$\mathcal{U}q_k - q^* \leq \gamma\|q_k - q^*\|e. \tag{A.10}$$

Lower bound on $\mathcal{U}q_k - q^*$. Again, similarly to (A.7) we have

$$\begin{aligned}\mathcal{U}q_k - q^* &\geq \gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k}(I-\gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k)\\ &\quad +\gamma\mathcal{P}^{\pi^*}(q_k - q^*) - \epsilon_k\|q_k\|. \end{aligned} \tag{A.11}$$

Lower-bound on $\mathcal{T}^{\pi_k}q_k - q_k$. Since the sequence of policies (π_k) is increasingly greedy w.r.t. (q_k), we have

$$\begin{aligned}\mathcal{T}^{\pi_{k+1}}q_{k+1} - q_{k+1} &\geq \mathcal{T}^{\pi_k}q_{k+1} - q_{k+1}\\ &= (1-\alpha_k)\mathcal{T}^{\pi_k}q_k + \alpha_k\mathcal{T}^{\pi_k}(\mathcal{U}q_k + \omega_k + \upsilon_k) - q_{k+1}\\ &= (1-\alpha_k)(\mathcal{T}^{\pi_k}q_k - q_k) + \alpha_k\big[\mathcal{T}^{\pi_k}\mathcal{U}q_k - \mathcal{U}q_k + \omega'_k + \upsilon'_k\big], \end{aligned} \tag{A.12}$$

where $\omega'_k \stackrel{\text{def}}{=} (\gamma\mathcal{P}^{\pi_k} - I)\omega_k$ and $\upsilon'_k \stackrel{\text{def}}{=} (\gamma\mathcal{P}^{\pi_k} - I)\upsilon_k$. It is easy to see that both ω'_k and υ'_k continue to satisfy the assumptions on ω_k, and υ_k. Now, from the definition of the $\mathcal{U}$ operator, we have

$$\begin{aligned}\mathcal{T}^{\pi_k}\mathcal{U}q_k - \mathcal{U}q_k &= r + (\gamma\mathcal{P}^{\pi_k} - I)\mathcal{U}q_k\\ &= r + (\gamma\mathcal{P}^{\pi_k} - I)\big[q_k + (I-\gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k)\big]\\ &= \mathcal{T}^{\pi_k}q_k - q_k + (\gamma\mathcal{P}^{\pi_k} - I)(I-\gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k)\\ &= \gamma(\mathcal{P}^{\pi_k} - \lambda\mathcal{P}^{\pi_k\wedge\mu_k})(I-\gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k})^{-1}(\mathcal{T}^{\pi_k}q_k - q_k). \end{aligned}$$

Using this equality into (A.12) and writing $\xi_k \stackrel{\text{def}}{=} \mathcal{T}^{\pi_k}q_k - q_k$, we have

$$\xi_{k+1} \geq (1-\alpha_k)\xi_k + \alpha_k\big[\mathcal{B}_k\xi_k + \omega'_k + \upsilon'_k\big], \tag{A.13}$$

where $\mathcal{B}_k \stackrel{\text{def}}{=} \gamma(\mathcal{P}^{\pi_k} - \lambda\mathcal{P}^{\pi_k\wedge\mu_k})(I-\gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k})^{-1}$. The matrix underlying $\mathcal{B}_k$ is non-negative but may not be a contraction mapping (the sum of its components per row may be larger than 1). Thus we cannot directly apply Proposition 4.5 of [Bertsekas and Tsitsiklis 1996]. However, as we have seen in the proof of Theorem 3.4, $\mathcal{A}_k = \gamma(I - \gamma\lambda\mathcal{P}^{\pi_k\wedge\mu_k})^{-1}(\mathcal{P}^{\pi_k} - \lambda\mathcal{P}^{\pi_k\wedge\mu_k})$ is a γ-contraction mapping. So now we relate $\mathcal{B}_k$ to $\mathcal{A}_k$ using our assumption that $\mathcal{P}^{\pi_k}$ and $\mathcal{P}^{\pi_k\wedge\mu_k}$ commute asymptotically, i.e. $\|(\mathcal{P}^{\pi_k}\mathcal{P}^{\pi_k\wedge\mu_k} - \mathcal{P}^{\pi_k\wedge\mu_k}\mathcal{P}^{\pi_k})q\| = \eta_k$ with $\eta_k \to 0$. For any (sub)-transition matrices U and V, we have

$$\begin{aligned}U(I-\lambda\gamma V)^{-1} &= \sum_{t\geq 0}(\lambda\gamma)^t UV^t\\ &= \sum_{t\geq 0}(\lambda\gamma)^t\Big[\sum_{s=0}^{t-1}V^s(UV-VU)V^{t-s-1} + V^tU\Big] \end{aligned}$$

$$= (I-\lambda\gamma V)^{-1}U + \sum_{t\geq 0}(\lambda\gamma)^t \sum_{s=0}^{t-1} V^s(UV-VU)V^{t-s-1}.$$

Replacing U by $\mathcal{P}^{\pi_k}$ and V by $\mathcal{P}^{\pi_k\wedge\mu_k}$, we deduce

$$\|(\mathcal{B}_k - \mathcal{A}_k)e\| \leq \gamma \sum_{t\geq 0} t(\lambda\gamma)^t \eta_k = \gamma \frac{1}{(1-\lambda\gamma)^2}\eta_k.$$

Thus, from (A.13),

$$\xi_{k+1} \geq (1-\alpha_k)\xi_k + \alpha_k\big[\mathcal{A}_k\xi_k + \omega'_k + \upsilon''_k\big], \tag{A.14}$$

where $\upsilon''_k \stackrel{\text{def}}{=} \upsilon'_k + \gamma\sum_{t\geq 0} t(\lambda\gamma)^t\eta_k\|\xi_k\|$ continues to satisfy the assumptions on υ_k (since $\eta_k \to 0$).

Now, let us define another sequence ξ'_k as follows: $\xi'_0 = \xi_0$ and

$$\xi'_{k+1} = (1-\alpha_k)\xi'_k + \alpha_k(\mathcal{A}_k\xi'_k + \omega'_k + \upsilon''_k).$$

We can now apply Proposition 4.5 of [Bertsekas and Tsitsiklis 1996] to the sequence (ξ'_k). The entries of $\mathcal{A}_k$ are non-negative, and the sum of their coefficients per row is bounded by γ, see (A.4), thus $\mathcal{A}_k$ are γ-contraction mappings and have the same fixed point which is 0. The noise ω'_k is centered and $\mathcal{F}_k$-measurable and satisfies the bounded variance assumption, and υ''_k is bounded above by $(1+\gamma)\theta'_k(\|q_k\|+1)$ for some $\theta'_k \to 0$. Thus $\lim_k \xi'_k = 0$ almost surely.

Now, it is straightforward to see that $\xi_k \geq \xi'_k$ for all $k \geq 0$. Indeed by induction, let us assume that $\xi_k \geq \xi'_k$. Then

$$\begin{aligned}
\xi_{k+1} &\geq (1-\alpha_k)\xi_k + \alpha_k(\mathcal{A}_k\xi_k + \omega'_k + \upsilon''_k) \\
&\geq (1-\alpha_k)\xi'_k + \alpha_k(\mathcal{A}_k\xi'_k + \omega'_k + \upsilon''_k) \\
&= \xi'_{k+1},
\end{aligned}$$

since all elements of $\mathcal{A}_k$ are non-negative. Thus we deduce that

$$\liminf_{k\to\infty} \xi_k \geq \lim_{k\to\infty} \xi'_k = 0 \tag{A.15}$$

Conclusion. Using (A.15) in (A.11) we deduce the lower bound:

$$\liminf_{k\to\infty} \mathcal{U}q_k - q^* \geq \liminf_{k\to\infty} \gamma\mathcal{P}^{\pi^*}(q_k - q^*), \tag{A.16}$$

almost surely. Now combining with the upper bound (A.10) we deduce that

$$\|\mathcal{U}q_k - q^*\| \leq \gamma\|q_k - q^*\| + O(\epsilon_k\|q_k\|) + O(\xi_k).$$

The last two terms can be incorporated to the $v_k(s,a)$ and $\omega_k(s,a)$ terms, respectively; we thus again apply Proposition 4.5 of [Bertsekas and Tsitsiklis 1996] to the sequence (q_k) defined by (A.9) and deduce that $q_k \to q^*$ almost surely. □

It remains to rewrite the update (3.19) in the form of (A.9), in order to apply Theorem A.1.

Let $z^k_{s,t}$ denote the accumulating trace:

$$z^k_{i,t} \stackrel{\text{def}}{=} \sum_{j=i}^{t} \gamma^{t-j}\Big(\prod_{\ell=j+1}^{t} c_\ell\Big)\mathbb{I}\{(S_j,A_j) = (S_i,A_i)\}.$$

Let us write $q^o_{k+1}(S_i,A_i)$ to emphasize the online setting. Then (3.19) can be written as

$$q^o_{k+1}(S_i,A_i) \leftarrow q^o_k(S_i,A_i) + \alpha_k(S_i,A_i)\sum_{t=i}^{\infty}\delta^{\pi_k}_t z^k_{i,t}, \tag{A.17}$$
$$\delta^{\pi_k}_t = R_{t+1} + \gamma\mathbb{E}_{\pi_k} q^o_k(S_{t+1},\cdot) - q^o_k(S_t,A_t),$$

Using our assumptions on finite trajectories, and $c_i \leq 1$, we can show that:

$$\mathbb{E}\Big[\sum_{t\geq s} z^k_{s,t}|\mathcal{F}_k\Big] < \mathbb{E}\left[T^2_k|\mathcal{F}_k\right] < \infty \tag{A.18}$$

where T_k denotes trajectory length. Now, let $D_k \equiv D_k(S_i,A_i) \stackrel{\text{def}}{=} \sum_{t=i}^{\infty}\Pr\{(S_t,A_t) = (S_i,A_i)\}$. Then, using (A.18), we can show that the total update is bounded, and rewrite

$$\mathbb{E}_{\mu_k}\Big[\sum_{t\geq s}\delta^{\pi_k}_t z^k_{s,t}\Big] = D_k(S_i,A_i)\big(\mathcal{U}_k q_k(S_i,A_i) - q(S_i,A_i)\big).$$

Finally, using the above, and writing $\alpha_k = \alpha_k(S_i,A_i)$, (A.17) can be rewritten in the desired form:

$$q^o_{k+1}(S_i,A_i) \leftarrow (1-\tilde{\alpha}_k)q^o_k(S_i,A_i) + \tilde{\alpha}_k\big(\mathcal{U}_k q^o_k(S_i,A_i) + \omega_k(S_i,A_i) + v_k(S_i,A_i)\big), \tag{A.19}$$

$$\omega_k(S_i,A_i) \stackrel{\text{def}}{=} (D_k)^{-1}\left(\sum_{t\geq s}\delta^{\pi_k}_t z^k_{s,t} - \mathbb{E}_{\mu_k}\left[\sum_{t\geq s}\delta^{\pi_k}_t z^k_{s,t}\right]\right),$$

$$\upsilon_k(S_i, A_i) \stackrel{\text{def}}{=} (\tilde{\alpha}_k)^{-1}\big(q^o_{k+1}(S_i, A_i) - q_{k+1}(S_i, A_i)\big),$$
$$\tilde{\alpha}_k \stackrel{\text{def}}{=} \alpha_k D_k.$$

It can be shown that the variance of the noise term ω_k is bounded, using (A.18) and the fact that the reward function is bounded. It follows from Assumptions 2.2, 2.3 and 3.1 that the modified stepsize sequence $(\tilde{\alpha}_k)$ satisfies the conditions of Assumption 2.2. The second noise term $\upsilon_k(S_i, A_i)$ measures the difference between online iterates and the corresponding offline values, and can be shown to satisfy the required assumption analogously to the argument in the proof of Prop. 5.2 in [Bertsekas and Tsitsiklis 1996]. The proof relies on the eligibility coefficients (A.18) and rewards being bounded, the trajectories being finite, and the conditions on the stepsizes being satisfied.

We can thus apply Theorem A.1 to the update (A.19), and conclude that the iterates $q^o_k \to q^*$ as $k \to \infty$, w.p. 1.

A.8 Asymptotic Commutativity of $\mathcal{P}^{\pi_k}$ and $\mathcal{P}^{\pi_k \wedge \mu_k}$

Lemma A.3

Let (π_k) and (μ_k) two sequences of policies. If there exists α such that for all s, a,

$$\min(\pi_k(a|s), \mu_k(a|s)) = \alpha\pi_k(a|s) + o(1), \tag{A.20}$$

then the transition operators $\mathcal{P}^{\pi_k}$ and $\mathcal{P}^{\pi_k \wedge \mu_k}$ asymptotically commute: $\|(\mathcal{P}^{\pi_k}\mathcal{P}^{\pi_k \wedge \mu_k} - \mathcal{P}^{\pi_k \wedge \mu_k}\mathcal{P}^{\pi_k})q\| = o(1)$.

Proof. For any q, we have

$$\begin{aligned}
(\mathcal{P}^{\pi_k}\mathcal{P}^{\pi_k \wedge \mu_k})q(s,a) &= \sum_y p(y|s,a) \sum_b \pi_k(b|y) \sum_z p(z|y,b) \sum_c (\pi_k \wedge \mu_k)(c|z) q(z,c) \\
&= \alpha \sum_y p(y|s,a) \sum_b \pi_k(b|y) \sum_z p(z|y,b) \sum_c \pi_k(c|z) q(z,c) + \|q\| o(1) \\
&= \sum_y p(y|s,a) \sum_b (\pi_k \wedge \mu_k)(b|y) \sum_z p(z|y,b) \sum_c \pi_k(c|z) q(z,c) + \|q\| o(1) \\
&= (\mathcal{P}^{\pi_k \wedge \mu_k}\mathcal{P}^{\pi_k})q(s,a) + \|q\| o(1).
\end{aligned}$$

□

Lemma A.4

Let (π_{q_k}) a sequence of (deterministic) greedy policies w.r.t. a sequence (q_k). Let (π_k) a sequence of policies that are ϵ_k away from (π_{q_k}), in the sense that, for all s,

$$\|\pi_k(\cdot|s) - \pi_{q_k}(s)\|_1 = 1 - \pi_k(\pi_{q_k}(s)|s) + \sum_{a \neq \pi_{q_k}(s)} \pi_k(a|s) \leq \epsilon_k.$$

Let (μ_k) a sequence of policies defined by:

$$\mu_k(a|s) = \frac{\alpha\mu(a|s)}{1 - \mu(\pi_{q_k}(s)|s)} \mathbb{I}\{a \neq \pi_{q_k}(s)\} + (1-\alpha)\mathbb{I}\{a = \pi_{q_k}(s)\}, \tag{A.21}$$

for some arbitrary policy μ and $\alpha \in [0,1]$. Assume $\epsilon_k \to 0$. Then the transition operators $\mathcal{P}^{\pi_k}$ and $\mathcal{P}^{\pi_k \wedge \mu_k}$ asymptotically commute.

Proof. The intuition is that asymptotically π_k gets very close to the deterministic policy π_{q_k}. In that case, the minimum distribution $(\pi_k \wedge \mu_k)(\cdot|s)$ puts a mass close to $1-\alpha$ on the greedy action $\pi_{q_k}(s)$, and no mass on other actions, thus $(\pi_k \wedge \mu_k)$ gets very close to $(1-\alpha)\pi_k$, and Lemma A.3 applies (with multiplicative constant $1-\alpha$).

Indeed, from our assumption that π_k is ϵ-away from π_{q_k} we have:

$$\pi_k(\pi_{q_k}(s)|s) \geq 1 - \epsilon_k, \text{ and } \pi_k(a \neq \pi_{q_k}(s)|s) \leq \epsilon_k.$$

We deduce that

$$\begin{aligned}(\pi_k \wedge \mu_k)(\pi_{q_k}(s)|s) &= \min(\pi_k(\pi_{q_k}(s)|s), 1-\alpha) \\ &= 1 - \alpha + O(\epsilon_k) \\ &= (1-\alpha)\pi_k(\pi_{q_k}(s)|s) + O(\epsilon_k),\end{aligned}$$

and

$$\begin{aligned}(\pi_k \wedge \mu_k)(a \neq \pi_{q_k}(s)|s) &= O(\epsilon_k) \\ &= (1-\alpha)\pi_k(a|s) + O(\epsilon_k).\end{aligned}$$

Thus Lemma A.3 applies (with a multiplicative constant $1-\alpha$) and $\mathcal{P}^{\pi_k}$ and $\mathcal{P}^{\pi_k \wedge \mu_k}$ asymptotically commute. □

A.9 Derivation of Eq. (3.4)

From the definition of $\mathcal{U}_\lambda^{\pi,\mu}$ in Eq. (3.2), we have:

$$
\begin{aligned}
\mathcal{U}_\lambda^{\pi,\mu} q(s,a) &= (1-\lambda)\mathcal{U}_0^{\pi,\mu} + (1-\lambda)\lambda\mathcal{U}_1^{\pi,\mu} + (1-\lambda)\lambda^2\mathcal{U}_2^{\pi,\mu} + \dots \\
&= (1-\lambda)\sum_{t=0}^{\infty}\lambda^t r(s,a) + (1-\lambda)\mathbb{E}_\mu\Big[\gamma\mathbb{E}_\pi q(S_1,\cdot) \\
&\quad + \lambda\Big(\gamma(R_2 + \mathbb{E}_\pi q(S_1,\cdot) - q(S_1,A_1)) + \gamma^2\mathbb{E}_\pi q(S_2,\cdot)\Big) \\
&\quad + \lambda^2\Big(\gamma(R_2 + \mathbb{E}_\pi q(S_1,\cdot) - q(S_1,A_1)) \\
&\quad + \gamma^2(R_3 + \mathbb{E}_\pi q(S_2,\cdot) - q(S_2,A_2)) + \gamma^3\mathbb{E}_\pi q(S_3,\cdot)\Big) + \dots\Big] \\
&\overset{(1)}{=} r(s,a) + \mathbb{E}_\mu\Big[(1-\lambda)\gamma\mathbb{E}_\pi q(S_1,\cdot) \\
&\quad + \gamma\lambda(R_2 + \mathbb{E}_\pi q(S_1,\cdot) - q(S_1,A_1)) + (1-\lambda)\gamma^2\lambda\mathbb{E}_\pi q(S_2,\cdot) \\
&\quad + \gamma^2\lambda^2(R_3 + \mathbb{E}_\pi q(S_2,\cdot) - q(S_2,A_2)) + (1-\lambda)\gamma^3\lambda^2\mathbb{E}_\pi q(S_3,\cdot) + \dots\Big] \\
&\overset{(2)}{=} r(s,a) + \mathbb{E}_\mu\Big[\gamma\mathbb{E}_\pi q(S_1,\cdot) \\
&\quad + \gamma\lambda(R_2 - q(S_1,A_1)) + \gamma^2\lambda\mathbb{E}_\pi q(S_2,\cdot) \\
&\quad + \gamma^2\lambda^2(R_3 - q(S_2,A_2)) + \gamma^3\lambda^2\mathbb{E}_\pi q(S_3,\cdot) + \dots\Big] \\
&\overset{(3)}{=} q(s,a) + \mathbb{E}_\mu\Big[R_1 + \gamma\mathbb{E}_\pi q(S_1,\cdot) - q(S_0,A_0) \\
&\quad + \gamma\lambda(R_2 + \gamma\mathbb{E}_\pi q(S_2,\cdot) - q(S_1,A_1)) + \\
&\quad + \gamma^2\lambda^2(R_3 + \mathbb{E}_\pi q(S_3,\cdot) - q(S_2,A_2)) + \dots\Big] \\
&= q(s,a) + \mathbb{E}_\mu\Big[\sum_{t=0}^{\infty}(\gamma\lambda)^t(R_{t+1} + \gamma\mathbb{E}_\pi q(S_{t+1},\cdot) - q(S_t,A_t))\Big].
\end{aligned}
$$

where (1) is due to the fact that $(1-\lambda)\sum_{t=0}^{\infty}\lambda^t = 1$, (2) is obtained by expanding the $(1-\lambda)$ factors, and cancelling terms, and (3) is due to the implicit conditioning on $S_0, A_0 = s, a$.

B | Proofs from Chapter 4

B.1 Proof of Proposition 4.2

Proof. Verifying that v_{k+1} is the fixed point of M_k is immediate given (4.12) and the update for v_{k+1} from (4.11). Let us now derive contraction factor of M_k. We have that:

$$M_k v - M_k v' = \gamma p^{(1-\beta)\kappa_k}(v - v').$$

Since $0 \leq \beta_s^o \leq 1, \forall s, o$, the entries in $p^{(1-\beta)\kappa_k}$ are nonnegative. Consider the sum of $p^{(1-\beta)\kappa_k}$ over each row:

$$\begin{aligned}
\xi_k(s) &= \gamma \sum_{s'} p^{(1-\beta)\kappa_k}(s, s') \\
&= \gamma \sum_{s'} \sum_{o} \mu_k(o|s) p^{\pi^o}(s, s')(1 - \beta_{s'}^o) \\
&= \gamma (1 - \sum_{o} \mu_k(o|s) \sum_{s'} p^{\pi^o}(s, s') \beta_{s'}^o) \\
&= \gamma (1 - c_k(s)) \leq \gamma.
\end{aligned}$$

This concludes the proof. □

B.2 Proof of Theorem 4.1

Proof. We will conduct a proof by induction. Assume that $v_0 \leq \mathcal{T}^{\kappa_k} v_0$ (which can be attained by pessimistic initialization, $v_s = -r_{\max}/(1-\gamma)$), and write v^* for $v^*_{\mathcal{O}}$. We will show that for all k:

$$v_k \leq \mathcal{T}^{\kappa_k} v_k \leq v_{k+1} \leq \mathcal{T}^{\kappa_{k+1}} v_{k+1} \leq v^* \tag{B.1}$$

To this end, fix k, and assume $v_k \leq \mathcal{T}^{\kappa_k} v_k$. We will show (B.1) in three steps.

Step 1: $v_k \leq \mathcal{T}^{\kappa_k} v_k \leq v_{k+1}$.

We have that $v_k \leq \mathcal{T}^{\kappa_k} v_k = M_k v_k$, due to the inductive assumption.

Then, by monotonicity of $\mathcal{T}^{\kappa_k}$, which implies monotonicity of M_k: $v_k \leq \mathcal{T}^{\kappa_k} v_k \leq M_k^m v_k \leq M_k^{m+1} v_k$ for all positive integers m, and so by taking $m \to \infty$, we get

$$v_k \leq \mathcal{T}^{\kappa_k} v_k \leq \lim_{m\to\infty} (M_k)^m v_k = v_{k+1}. \tag{B.2}$$

Step 2: $v_{k+1} \leq \mathcal{T}^{\kappa_{k+1}} v_{k+1}$.

From the definition of M_k we have

$$\begin{aligned} M_k v_{k+1} &= r^{\kappa_k} + \gamma p^{\beta\kappa_k} v_k + \gamma(p^{\kappa_k} - p^{\beta\kappa_k}) v_k \\ &= \mathcal{T}^{\kappa_k} v_{k+1} - \gamma p^{\beta\kappa_k}(v_{k+1} - v_k) \end{aligned}$$

Then, due to the monotonicity of $p^{\beta\kappa_k}$ and the fact that $v_{k+1} - v_k \geq 0$ as shown in the previous step, the above equation implies that

$$v_{k+1} = M_k v_{k+1} \leq \mathcal{T}^{\kappa_k} v_{k+1}, \tag{B.3}$$

where the first equality follows from the fact that v_{k+1} is the fixed point of M_k. On the other hand we have by A2, the assumption that the policies are increasingly greedy, that

$$\mathcal{T}^{\kappa_k} v_{k+1} \leq \mathcal{T}^{\kappa_{k+1}} v_{k+1} \tag{B.4}$$

The desired inequality then follows from combining (B.4) and (B.3).

Step 3: $\mathcal{T}^{\kappa_{k+1}} v_{k+1} \leq v^*$

From the monotonicity of $\mathcal{T}^{\kappa_{k+1}}$ and the relation shown in Step 2, we have that $\mathcal{T}^{\kappa_{k+1}} v_{k+1} \leq (\mathcal{T}^{\kappa_{k+1}})^m v_{k+1}$ for all positive integers m. Thus taking the limit as $m \to \infty$:

$$\mathcal{T}^{\kappa_{k+1}} v_{k+1} \leq \lim_{m\to\infty} (\mathcal{T}^{\kappa_{k+1}})^m v_{k+1} = v^{\kappa_{k+1}} \leq v^*$$

where the last inequality comes from the definition of v^* as $\sup_\pi v^\pi$.

Putting the three steps together we see by induction that that the desired relation (B.1) holds for all k. In particular:

$$v_k \leq v_{k+1} \leq v^*,$$

which implies that the sequence (v_k) is monotonic and bounded, and thus converges to some limit $\bar{v} \leq v^*$. From the definition of M_k in (4.12) we have that

$$v_{k+1} = M_k v_{k+1} = \mathcal{T}^{\kappa_k} v_k + \gamma p^{\beta\kappa_k}(v_{k+1} - v_k).$$

By taking the limit $k \to \infty$ and using the fact that $v_{k+1} - v_k \to 0$ and by Assumption 4.1: $\lim_{k\to\infty} \mathcal{T}^{\kappa_k} v_k = \mathcal{T} \lim_{k\to\infty} v_k = \mathcal{T}\bar{v}$, we obtain $\bar{v} = \mathcal{T}\bar{v}$, thus $\bar{v} = v^*$, since v^* is the unique fixed point of $\mathcal{T}$. Finally, the fixed points of $\mathcal{T}^{\kappa^*}$ and $\mathcal{T}_{\mathcal{O}}^{\mu^*}$ are the same. This concludes the proof of convergence, and we turn to deriving its rate.

Convergence rate First let us rewrite Eq. (4.18) as:

$$\mathcal{T}_{\mathcal{O}}^{\mu} v = v + (I - \gamma p^{(1-\beta)\kappa})^{-1}(\mathcal{T}^{\kappa} v - v).$$

Since $v_k \to v^*$, it follows that for all k larger than some index $\bar{k}$, κ_k is optimal, and $\mathcal{T}^{\kappa_k} v^* = v^*$. Thus, using the above, we have for all $k \geq \bar{k}$:

$$\begin{aligned}
v_{k+1} - v^* &= v_k - v^* + \underbrace{(I - \gamma p^{(1-\beta)\kappa_k})}_{B_k}^{-1}(\mathcal{T}^{\kappa_k} v_k - v_k) \\
&= B_k^{-1}(\mathcal{T}^{\kappa_k} v_k - \mathcal{T} v^* - \gamma p^{(1-\beta)\kappa_k}(v_k - v^*)) \\
&= \gamma B_k^{-1}(p^{\kappa_k}(v_k - v^*) - \gamma(p^{\kappa_k} - p^{\beta\kappa_k})(v_k - v^*)) \\
&= \gamma(I - \gamma p^{(1-\beta)\kappa_k})^{-1} p^{\beta\kappa_k}(v_k - v^*) \\
&= A_k(v_k - v^*).
\end{aligned}$$

where

$$\begin{aligned}
A_k &= \gamma(I - \gamma(p^{\kappa_k} - p^{\beta\kappa_k})^{-1} p^{\beta\kappa_k} \\
&= \gamma \sum_{t\geq 0} \gamma^t (p^{\kappa_k} - p^{\beta\kappa_k})^t p^{\beta\kappa_k}.
\end{aligned}$$

Since $p^{\beta\kappa_k} \leq p^{\kappa_k}$ elementwise, A_k is composed of nonnegative elements. We now look at its sum over each row. Let e be the vector of all 1s:

$$A_k e = \gamma \sum_{t\geq 0} (\gamma^t (p^{(1-\beta)\kappa_k})^t (p^{\kappa_k} - p^{(1-\beta)\kappa_k}) e$$

$$
\begin{aligned}
&= \gamma \underbrace{\sum_{t\geq 0} \gamma^t (p^{(1-\beta)\kappa_k})^t\, p^{\kappa_k}}_{C} e - \sum_{t\geq 1} (p^{(1-\beta)\kappa_k})^t e \\
&= \gamma C e - (Ce - e) \\
&= e - (1-\gamma) Ce \\
&\leq \gamma e,
\end{aligned}
$$

since $Ce \geq e$. Thus

$$\|v_{k+1} - v^*\|_\infty \leq \gamma \|v_k - v^*\|_\infty.$$

The coefficient C is state-dependent and we have that:

$$|v_{k+1}(s) - v^*(s)| \leq \eta(s) \|v_k - v^*\|_\infty,$$

with

$$\eta(s) = 1 - (1-\gamma) \mathbb{E}_{\substack{o_t \sim \mu(\cdot|s_t) \\ s_{t+1} \sim p^{\pi^{o_t}}(\cdot|s_t)}} \left[\sum_{t\geq 0} \gamma^t \left(\prod_{i=1}^{t} (1 - \beta^{o_t}(s_i)) \Big| s_0 = s \right) \right],$$

where we write $\prod_{i=1}^{0} x = 1$ for simplicity. □

B.3 Proof of Proposition 4.3

Proof. From Eq. (4.18), and writing q for $q_\beta^{\mu,\iota}$ for less clutter:

$$
\begin{aligned}
q &= (I - \gamma \mathcal{P}^{(I-\beta)\iota})^{-1} (r^\pi + \gamma \mathcal{P}^{\beta\mu} q), \\
q - \gamma \mathcal{P}^{(I-\beta)\iota} q &= r^\pi + \gamma \mathcal{P}^{\beta\mu} q, \\
q &= r^\pi + \gamma \mathcal{P}^{\beta\mu} q + \gamma \mathcal{P}^{(I-\beta)\iota} q \\
&= r^\pi + \gamma \mathcal{P}^{\beta\mu} q + \gamma \mathcal{P}^{\iota} q - \gamma \mathcal{P}^{\beta\iota} q \\
&= r^\pi + \gamma (\mathcal{P}^{\beta\mu} - \mathcal{P}^{\beta\iota}) q + \gamma \mathcal{P}^{\iota} q.
\end{aligned}
$$

Solving for q we have:

$$
\begin{aligned}
q &= (I - \gamma(\mathcal{P}^{\beta\mu} - \mathcal{P}^{\beta\iota}) - \gamma \mathcal{P}^{\iota})^{-1} r^\pi \\
&= (I - \gamma \mathcal{P}^{\beta\mu} - \gamma \mathcal{P}^{(I-\beta)\iota})^{-1} r^\pi
\end{aligned}
$$

□

B.4 Proof of Theorem 4.2

Proof. Write $\mathcal{U}$ for $\mathcal{U}^{\mu}_{\zeta}$. Recall that:

$$c^o_i = ((1-\zeta^o(S_i)) + \zeta^o(S_i)\mu(o|S_i))(1-\beta^o(S_i)),$$
$$\tilde{q}(s,o) = [1-\zeta^o(s)]q(s,o) + \zeta^o(s)\mathbb{E}_\mu q(s,\cdot).$$

The fact that the fixed point is the desired one is clear from Eq. (4.29) and Proposition 4.3:

$$\begin{aligned}\mathcal{U}q^{\mu,\iota}_\zeta(s,o) &= q^{\mu,\iota}_\zeta(s,o) + \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t}c^o_i\Big)[R_{t+1} + \gamma((1-\zeta^o_i)q^{\mu,\iota}_\zeta(S_t,o) \\ &\quad + \zeta^o_i(S_t)\mathbb{E}_\mu q^{\mu,\iota}_\zeta(S_t,\cdot) - q^{\mu,\iota}_\zeta(S_t,o)]\Big] \\ &= q^{\mu,\iota}_\zeta(s,o) + \sum_{t=0}^{\infty}\gamma^t\mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t}c^o_i\Big)(\mathcal{T}^{(I-\beta)\mu}q^{\mu,\iota}_\zeta + \mathcal{T}^{\beta\mu}q^{\mu,\iota} - q^{\mu,\iota}_\zeta)(S_t,o))] \\ &= q^{\mu,\iota}_\zeta(s,o)\end{aligned}$$

Now, let us derive the contraction result. Eq. (4.29) can be written:

$$\mathcal{U}q(s,o) = \sum_{t=0}^{\infty}\mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t}c^o_i\Big)T^{oc}_t\Big]$$
$$T^{co}_t = R_{t+1} + \gamma[\tilde{q}(S_{t+1},o) - c^o_{t+1}q(S_{t+1},o)]$$

Since $\mathcal{U}q^{\mu,\iota}_\zeta = q^{\mu,\iota}_\zeta$, and after shifting the sum index forward, we have:

$$\mathcal{U}q(s,o) - q^{\mu,\iota}_\zeta(s,o) = \sum_{t=1}^{\infty}\mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t-1}c^o_i\Big)\Delta T^{co}_t\Big],$$
$$\begin{aligned}\Delta T^{co}_t &= (1-\zeta^o_t)\Delta q(S_t,o) + \zeta^o_t\mathbb{E}_\mu\Delta q(S_t,\cdot) - c^o_t\Delta q(S_t,o) \\ &= ((1-\zeta^o_t) - c^o_t)\Delta q(S_t,o) + \zeta^o_t\sum_b\mu(b|S_t)\Delta q(S_t,b) \\ &= \sum_b\Big(\mathbb{I}_{b=o}((1-\zeta^b_t) - c^b_t) + \zeta^o_t\mu(b|S_t)\Big)\Delta q(S_t,b).\end{aligned}$$

Since $c^o_t \le (1-\zeta^o_t) + \zeta^o_t\mu(o|S_t)$ and $\zeta^o_t\mu(b|S_t) \ge 0, \forall b \ne o$, we have a linear combination of $\Delta q(S_t,b)$ weighted by non-negative coefficients $w_{y,b}$ defined as:

$$w_{y,b} = \sum_{t=1}^{\infty}\mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t-1}c^o_i\Big)(\mathbb{I}_{b=o}((1-\zeta^b_t) - c^b_t) + \zeta^o_t\mu(b|S_t))\mathbb{I}_{S_t=y}\Big]$$

whose sum is:

$$\begin{aligned}
\sum_b w_{y,b} &= \sum_{t=1}^{\infty} \mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t-1} c_i^o\Big) \sum_b \mathbb{I}_{b=o}((1-\zeta_t^b) - c_t^b) + \zeta_t^o \mu(b|S_t)\Big] \\
&= \sum_{t=1}^{\infty} \mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t-1} c_i^o\Big)(((1-\zeta_t^o) - c_t^o) + \sum_b \mu(b|S_t)\zeta_t^o\Big] \\
&= \sum_{t=1}^{\infty} \mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t-1} c_i^o\Big)(((1-\zeta_t^o) - c_t^o) + \zeta_t^o)\Big] \\
&= \sum_{t=1}^{\infty} \mathbb{E}_{\pi^o}\Big[\Big(\prod_{i=1}^{t-1} c_i^o\Big)(1 - c_t^o)\Big] \\
&= \mathbb{E}_{\pi^o}\left[\sum_{t=1}^{\infty} \gamma^t \Big(\prod_{i=1}^{t-1} c_i^o\Big) - \sum_{t=1}^{\infty} \gamma^t \Big(\prod_{i=1}^{t} c_i^o\Big)\right] \\
&= \gamma C - (C-1) \leq \gamma
\end{aligned}$$

where $C = \mathbb{E}_{\pi^o}\left[\sum_{t=0}^{\infty} \gamma^t \Big(\prod_{i=1}^{t} c_i^o\Big)\right]$, and the last inequality is due to $C \geq 1$. It follows that $\mathcal{U}$ is a γ-contraction around $q_\zeta^{\mu,\iota}$. □

B.5 Proof of Corollary 4.1

Proof. We would like for the off-policy trace $c_i^o = ((1-\zeta^o(S_i)) + \zeta^o(S_i)\mu(o|S_i))(1-\beta^o(S_i))$ to be larger than the equivalent on-policy trace $(1-\zeta^o(S_i))$.

$$\begin{aligned}
(1-\zeta_t^o) &\leq ((1-\zeta_t^o) + \zeta_t^o \mu(o|S_t))(1-\beta_t^o) \\
(1-\zeta_t^o)\beta_t^o &\leq \zeta_t^o \mu(o|S_t)(1-\beta_t^o) \\
\beta_t^o &\leq \zeta_t^o(\mu(o|S_t)(1-\beta_t^o) + \beta_t^o) \\
\zeta_t^o &\geq \frac{\beta_t^o}{\mu(o|S_t)(1-\beta_t^o) + \beta_t^o} \\
&= 1 - \frac{\mu(o|S_t)(1-\beta_t^o)}{\mu(o|S_t)(1-\beta_t^o) + \beta_t^o}
\end{aligned}$$

So if ζ obeys this bound, it is more beneficial to learn it off- rather than on- policy, from the point of view of convergence speed. □

B.6 Proof of Theorem 4.4

Proof. From Proposition 4.3 we have:

$$\begin{aligned} q_{\zeta}^{\mu,\iota} - q_{\beta}^{\mu,\iota} &= \sum_{t=1}^{\infty} \gamma^t ((\mathcal{P}^{\zeta\mu} + \mathcal{P}^{(I-\zeta)\iota})^t - (\mathcal{P}^{\beta\mu} + \mathcal{P}^{(I-\beta)\iota})^t) r^{\pi} \\ &= (\mathcal{P}^{\zeta\mu} + \mathcal{P}^{(I-\zeta)\iota} - \mathcal{P}^{\beta\mu} - \mathcal{P}^{(I-\beta)\iota}) \sum_{t=1}^{\infty} \gamma^t \mathcal{A}_t r^{\pi}, \end{aligned}$$

where $\mathcal{A}_t = (\mathcal{P}^{\zeta\mu} + \mathcal{P}^{(I-\zeta)\iota} - \mathcal{P}^{\beta\mu} - \mathcal{P}^{(I-\beta)\iota})^{-1}((\mathcal{P}^{\zeta\mu} + \mathcal{P}^{(I-\zeta)\iota})^t - (\mathcal{P}^{\beta\mu} - \mathcal{P}^{(I-\beta)\iota})^t)$. Let $f = \sum_{t=1}^{\infty} \gamma^t \mathcal{A}_t r^{\pi}$ be a Q-function. Then simple manipulations yield:

$$\begin{aligned} q_{\zeta}^{\mu,\iota} - q_{\beta}^{\mu,\iota} &= (\mathcal{P}^{(\zeta-\beta)\mu} + \mathcal{P}^{(I-\zeta-I+\beta))\iota}) f \\ &= (\mathcal{P}^{(\zeta-\beta)\mu} - \mathcal{P}^{(\zeta-\beta)\iota}) f \\ &= (\mathcal{P}^{I\mu} - \mathcal{P}^{I\iota})(\zeta - \beta) f \\ &\geq (\mathcal{P}^{I\mu} - \mathcal{P}^{I\iota}) f \geq \mathbf{0}, \end{aligned}$$

since the operators $\mathcal{P}^{I\mu}$ and $\mathcal{P}^{I\iota}$ are monotone, $\zeta \geq \beta$ and $\mathcal{P}^{I\mu} f = \max_{\nu} \mathcal{P}^{1\nu} f \geq \mathcal{P}^{I\iota} f$ for any Q-function f. □

C | Proofs from Chapter 5

C.1 Proof of Theorem 5.1

Proof. While we assume that the reward model is unchanged, in order to preserve the appealing form of the option equations, it will be convenient to renormalize the *one-step* reward model to be in terms of γ. We hence wish for $R^o_{\gamma_{env}}$ and R^o_γ to be the same:

$$R^o_{\gamma_{env}} = (I - \gamma_{env} p^{(1-\beta)\pi^o})^{-1} r^{\pi^o}_{\gamma_{env}} = (I - \gamma p^{(1-\beta)\pi^o})^{-1} r^{\pi^o}_\gamma = R^o_\gamma,$$

solving which for $r^{\pi^o}_\gamma$ yields the result.

$$r^{\pi^o}_\gamma = (I - \gamma p^{(1-\beta)\pi^o})(I - \gamma_{env} p^{(1-\beta)\pi^o})^{-1} r^{\pi^o}_{\gamma_{env}}.$$

The Bellman operator can then be rewritten:

$$\mathcal{T}^\mu_{\mathcal{O}_{\Gamma_\gamma}} q \stackrel{\text{def}}{=} R + \mathcal{P}^\mu_{\mathcal{O}_{\Gamma_\gamma}} q = (I - \gamma \mathcal{P}^{(I-\beta)\iota})^{-1} (r^\pi_\gamma + \gamma \mathcal{P}^{\Gamma\beta\mu} q).$$

Let us now derive the fixed point of $\mathcal{T}^\mu_{\mathcal{O}_{\Gamma_\gamma}}$:

$$\begin{aligned} q &= (I - \gamma \mathcal{P}^{(1-\beta)\pi})^{-1} (r^\pi_\gamma + \gamma \mathcal{P}^{\Gamma\beta\mu} q) \\ q - \gamma \mathcal{P}^{(1-\beta)\pi} q &= r^\pi_\gamma + \gamma \mathcal{P}^{\Gamma\beta\mu} q \\ q &= r^\pi_\gamma + \gamma \left(\mathcal{P}^{\Gamma\beta\mu} q + \mathcal{P}^{(1-\beta)\pi} q \right) \\ &= \left(I - \gamma (\mathcal{P}^{\Gamma\beta\mu} + \mathcal{P}^{(1-\beta)\pi}) \right)^{-1} r^\pi_\gamma \end{aligned}$$

$$= \sum_{t=0}^{\infty} \gamma^t \left(\mathcal{P}^{\Gamma\beta\mu} + \mathcal{P}^{(1-\beta)\pi}\right)^t r_\gamma^\pi.$$

So $\mathcal{B} = \mathcal{P}^{\Gamma\beta\mu} + \mathcal{P}^{(1-\beta)\pi}$ is the corresponding one-step operator. Let us verify that it induces the new step-discount and termination scheme. Let $\gamma^o(s,s') = \Gamma^o(s')\beta^o(s') + 1 - \beta^o(s')$. We have:

$$\begin{aligned}
\mathcal{B}q(s,o) &= (\mathcal{P}^{\Gamma\beta\mu} + \mathcal{P}^{(1-\beta)\iota})q(s,o) \\
&= \sum_{s'} p^{\pi^o}(s'|s) \left(\sum_{o'} \mu(o'|s')\Gamma^o(s')\beta^o(s')q(s',o') + (1-\beta^o(s'))q(s',o) \right) \\
&= \sum_{s'} p^{\pi^o}(s'|s)\gamma^o(s,s') \left(\sum_{o'} \mu(o'|s') \frac{\Gamma^o(s')\beta^o(s')}{\gamma^o(s,s')} q(s',o') + \frac{1-\beta^o(s')}{\gamma^o(s,s')} q(s',o) \right) \\
&= \sum_{s'} p^{\pi^o}(s'|s)\gamma^o(s') \left(\sum_{o'} \mu(o'|s')\zeta^o(s')q(s',o') + (1-\zeta^o(s'))q(s',o) \right)
\end{aligned}$$

Thus, we have a new termination scheme $\zeta^o(s) = \frac{\Gamma^o(s)\beta^o(s)}{\gamma^o(s)}$, and combining γ^o with the step-discount γ, we get our new step discount

$$\gamma(s,o,s') = \gamma\gamma^o(s,s') = \gamma\left(\Gamma^o(s')\beta^o(s') + 1 - \beta^o(s')\right) = \gamma\left(1 - \beta^o(s')(1-\Gamma^o(s'))\right).$$

This implies that we have a state-option-dependent contraction factor:

$$\begin{aligned}
\eta(s,o) &= \mathbb{E}_{S_1,S_2,\ldots\sim p^{\pi^o}} \left[\prod_{i=1}^{\infty} \gamma(S_i, o, S_{i+1}) \right] \\
&= \mathbb{E}_{S_1,S_2,\ldots\sim p^{\pi^o}} \left[\prod_{i=1}^{\infty} \gamma\left(1 - \beta^o(S_i)(1-\Gamma^o(S_i))\right) \right] \leq \gamma.
\end{aligned}$$

The operator $\mathcal{T}^{\mu}_{\mathcal{O}_{\Gamma_\gamma}}$ is a contraction if $\eta(s,o) < 1$. This is trivially true if $\gamma < 1$. If $\gamma = 1$, in order for $\eta(s,o) < 1$, we need $1 - \beta^o(S_i)(1-\Gamma^o(S_i)) < 1$ for some S_i along the trajectory, which is the same as $\beta^o(S_i)(1-\Gamma^o(S_i)) > 0$ and holds if Assumption 5.1 holds: if such an S_i is reachable by π^o, is terminating in the sense that $\beta^o(S_i) > 0$, and $\Gamma^o(S_i) < 1$. □

C.2 Proof of Lemma 5.1

Throughout we will refer to $\Gamma_{\max} \stackrel{\text{def}}{=} \max_{o\in\mathcal{O}, s\in\mathcal{S}} \Gamma_s$, and, for each option o, its minimum and maximum durations $d^o_{\min}$ and $d^o_{\max}$ for an option o: that is $d^o_{\min}$ denote the minimum

duration of o between any s and s' in $\mathcal{I}^o$. We will also use $d_{\min} \stackrel{\text{def}}{=} \min_{o\in\mathcal{O}} d^o_{\min}$, $d_{\max} \stackrel{\text{def}}{=} \max_{o\in\mathcal{O}} d^o_{\max}$ as the global minimum and maximum durations across options.

Before we proceed, let us show two helper bounds.

Lemma C.1

For each option $o \in \mathcal{O}$:

$$|P^o_{\Gamma\gamma}(\cdot|s)|_1 \leq \Gamma^o_{\max}\gamma^{d^o_{\min}}.$$

Proof. For each s and s', the transition model $P^o_{\Gamma\gamma}(s'|s) \leq \Gamma^o_{s'}\gamma^{d_{\min}}$, the minimum duration. Taking a max over the states s' yields the result. □

Lemma C.2

Let γ_{env} be the environment discount factor used by the reward model. Then, the value function is bounded:

$$\|q^\mu_{\Gamma\gamma}\|_\infty \leq \frac{r_{\max}}{1-\gamma_{env}}\frac{1}{1-\Gamma_{\max}\gamma^{d_{\min}}}.$$

Proof. From the definition of $q^\mu_{\Gamma\gamma}$, Lemma C.1 and the definition of the reward model R:

$$\begin{aligned}\|q^\mu_{\Gamma\gamma}\|_\infty &= (I - \mathcal{P}^\mu_{\mathcal{O}_{\Gamma\gamma}})^{-1}R \\ &\leq \frac{1}{1-\max_{s\in\mathcal{S},o\in\mathcal{O}}|P^o_{\Gamma\gamma}(\cdot|s)|_1}\|R\|_\infty \\ &\leq \frac{1}{1-\Gamma_{\max}\gamma^{d_{\min}}}\frac{r_{\max}}{1-\gamma},\end{aligned}$$

since $\|\mathcal{P}^\mu_{\mathcal{O}_{\Gamma\gamma}}q\|_\infty \leq \max_{s\in\mathcal{S},o\in\mathcal{O}}|P^o_{\Gamma\gamma}(\cdot|s)|_1\|q\|_\infty$. □

It will now be convenient to change our notation slightly. Let M denote the model $M = (\mathcal{S}, \mathcal{O}, \mathcal{P}_\mathcal{O}, R)$, and $\widehat{M} = (\mathcal{S}, \mathcal{O}, \widehat{\mathcal{P}_\mathcal{O}}, R)$, the approximate model with an approximate transition model. Then q^μ_M and $q^\mu_{\widehat{M}}$ are the respective value functions of a policy μ.[1]

We will show the Lemma in two steps.

[1] E.g., if $M = (\mathcal{S}, \mathcal{O}, \mathcal{P}_{\mathcal{O}_{\Gamma\gamma}}, R)$, $q^\mu_M = q^\mu_{\Gamma\gamma}$ and $q^\mu_{\widehat{M}} = q^\mu_{\widehat{\Gamma\gamma}}$.

1) Bounding $\mathcal{E}_{estim}$ in terms of the one-step error Similarly to Lemma 4 from [Jiang et al. 2015b], we can relate the error in the value functions due to the approximate model to the maximum one-step error:

Lemma C.3

For any $M = (\mathcal{S}, \mathcal{O}, \mathcal{P}_\mathcal{O}, R)$, and $\forall \mu : \mathcal{S} \to \mathcal{O}$, we have:

$$\left\| q^\mu_{\widehat{M}} - q^\mu_M \right\|_\infty \leq \frac{1}{1 - \Gamma_{\max}\gamma^{d_{\min}}} \max_{s\in\mathcal{S}, o\in\mathcal{O}} \left| R^o(s) + \widehat{\mathcal{P}^\mu_\mathcal{O}} q^\mu_M(s,o) - q^\mu_M(s,o) \right|.$$

Proof. Consider the evolution

$$q_m(s,o) = R^o(s) + \widehat{\mathcal{P}^\mu_\mathcal{O}} q_{m-1}(s,o). \tag{C.1}$$

We can bound the difference between successive estimates:

$$\begin{aligned} \|q_m - q_{m-1}\|_\infty &= \|\widehat{\mathcal{P}^\mu_\mathcal{O}}(q_{m-1} - q_{m-2})\|_\infty \\ &\leq \max_{s\in\mathcal{S}, o\in\mathcal{O}} |\widehat{P^o}(\cdot|s)|_1 \|q_{m-1} - q_{m-2}\|_\infty \\ &= \Gamma_{\max}\gamma^{d_{\min}} \|q_{m-1} - q_{m-2}\|_\infty, \end{aligned}$$

due to Lemma C.1 (which of course still applies to the approximate model.) Thus

$$\|q_m - q_0\|_\infty \leq \sum_{k=0}^{m-1} \|q_{k+1} - q_k\|_\infty \leq \|q_1 - q_0\|_\infty \sum_{k=1}^{m-1} (\Gamma_{\max}\gamma^{d_{\min}})^{k-1}.$$

Since as $m \to \infty$, $q_m = q^\mu_{\widehat{M}}$, we have that $\|q^\mu_{\widehat{M}} - q_0\|_\infty \leq \frac{1}{1-\Gamma_{\max}\gamma^{d_{\min}}} \|q_1 - q_0\|_\infty$. Finally, since q_0 can be initialized to q^μ_M, and from Eq. C.1 for $m = 1$, we have our result:

$$\left\| q^\mu_{\widehat{M}} - q^\mu_M \right\|_\infty \leq \frac{1}{1 - \Gamma_{\max}\gamma^{d_{\min}}} \max_{s\in\mathcal{S}, o\in\mathcal{O}} \left| R^o(s) + \widehat{\mathcal{P}^\mu_\mathcal{O}} q^\mu_M(s,o) - q^\mu_M(s,o) \right|.$$

□

Bounding the one-step error with the Hoeffding's bound Now let us bound the one-step error in terms of the number of samples. The following Lemma is similar to Lemma 2 from [Jiang et al. 2015b].

Lemma C.4

Let $\widehat{P^o_{\Gamma\gamma}}$ denote the modified transition model of an option o averaged over n samples, and $\widehat{\mathcal{P}^\mu_{\mathcal{O}_{\Gamma\gamma}}}$ the corresponding operator w.r.t. some policy over options μ. We have, with probability $1-\delta$:

$$\|R+\widehat{\mathcal{P}^\mu_{\mathcal{O}_{\Gamma\gamma}}} q^\mu_{\Gamma\gamma} - q^\mu_{\Gamma\gamma}\| \leq \frac{r_{\max}}{1-\gamma}\frac{1}{1-\Gamma_{\max}\gamma^{d_{\min}}}\Gamma_{\max}(\gamma^{d_{\min}}-\gamma^{d_{\max}})\sqrt{\frac{1}{2n}\log\frac{2|\mathcal{S}||\mathcal{O}|}{\delta}}, \tag{C.2}$$

Proof. The transition model estimate $\widehat{P^o_{\Gamma\gamma}}(s'|s)$ is an average of samples of the form $\Gamma^o_{s'}\gamma^D$, where D is the random variable corresponding to option duration. Let $X = R^o(s) + \gamma^D\Gamma^o_{s'}v^\mu_{\Gamma\gamma}(s')$, where $v^\mu_{\Gamma\gamma}(s) = \sum_o \mu(o|s)q^\mu_{\Gamma\gamma}(s,o)$. We have that:

$$R^o(s) + \gamma^{d_{\max}}\Gamma^o_{s'}v^\mu_{\Gamma\gamma}(s') \leq X \leq R^o(s) + \gamma^{d_{\min}}\Gamma_{s'}v^\mu_{\Gamma\gamma}(s').$$

Thus the range $a = X_{\min} - X_{\max}$ of X is:

$$a = \Gamma^o_{s'}(\gamma^{d_{\min}} - \gamma^{d_{\max}})v^\mu_{\Gamma\gamma}(s').$$

Then, since $\widehat{P^o_{\Gamma\gamma}}(s'|s)$ is sampled i.i.d., and $q^\mu_{\Gamma\gamma}(s,o)$ is the average of $R^o(s)+\sum_{s'}\widehat{P^o_{\Gamma\gamma}}(s'|s)v^\mu_{\Gamma\gamma}(s')$, we have by the Hoeffding's bound:

$$\Pr(|X - \mathbb{E}[X]| \geq t) \leq 2\exp\left(-\frac{2nt^2}{a^2}\right) = 2\exp\left(-\frac{2nt^2}{(v^\mu_{\Gamma\gamma}(s')\Gamma^o_{s'}(\gamma^{d_{\min}}-\gamma^{d_{\max}}))^2}\right)$$

We can get the uniform bound by setting the right hand side of the above to $\frac{\delta}{|S||\mathcal{O}|}$, and solving for t:

$$\begin{aligned}
\frac{\delta}{|S||\mathcal{O}|} &= 2\exp\left(-\frac{2nt^2}{(\|v^\mu_{\Gamma\gamma}\|_\infty\Gamma_{\max}(\gamma^{d_{\min}}-\gamma^{d_{\max}}))^2}\right)\\
\log\frac{2|S||\mathcal{O}|}{\delta} &= \frac{2nt^2}{(\|v^\mu_{\Gamma\gamma}\|_\infty\Gamma_{\max}(\gamma^{d_{\min}}-\gamma^{d_{\max}}))^2}\\
t &= \|v^\mu_{\Gamma\gamma}\|_\infty\Gamma_{\max}(\gamma^{d_{\min}}-\gamma^{d_{\max}})\sqrt{\frac{1}{2n}\log\frac{2|S||\mathcal{O}|}{\delta}}
\end{aligned}$$

Substituting the bound on $\|v^\mu_{\Gamma\gamma}\|_\infty = \|q^\mu_{\Gamma\gamma}\|_\infty$ from Lemma C.2 yields the result. □

Lemma 5.1 then follows from combining Lemmas C.3 and C.4

C.3 Proof of Lemma 5.2

Proof. We have

$$\begin{aligned} q^{\mu}_{\Gamma\gamma} - q^{\mu}_{\gamma_{env}} &= \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\Gamma\gamma} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}} \\ &= \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\Gamma\gamma} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} + \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}} \\ &= \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} (q^{\mu}_{\Gamma\gamma} - q^{\mu}_{\gamma_{env}}) + (\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}}) q^{\mu}_{\gamma_{env}} \\ &= (I - \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}})^{-1} \underbrace{(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}})}_{\mathcal{A}} q^{\mu}_{\gamma_{env}} \end{aligned} \tag{C.3}$$

Let us now bound this expression. We will start with the inner term first. Noticing that $\mathcal{P}^{\Gamma\beta^{\mu}} = \mathcal{P}^{\beta\mu}\Gamma$, and from the definitions of the operators, we can expand:

$$\begin{aligned} \|\mathcal{A} q^{\mu}_{\gamma_{env}}\| &= \|(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}}) q^{\mu}_{\gamma_{env}}\| \\ &= \left|\left| \gamma \mathcal{P}^{\beta\mu}\Gamma q^{\mu}_{\gamma_{env}} + \gamma \mathcal{P}^{(1-\beta)\iota} \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \left(\gamma_{env} \mathcal{P}^{\beta\mu} q^{\mu}_{\gamma_{env}} + \gamma_{env} \mathcal{P}^{(1-\beta)\iota} \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}} \right) \right|\right| \\ &\leq \left|\left| \mathcal{P}^{\beta\mu}(\gamma\Gamma - \gamma_{env}) q^{\mu}_{\gamma_{env}} + \mathcal{P}^{(1-\beta)\iota} \left(\gamma \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \gamma_{env} \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}} \right) \right|\right| \\ &= \left|\left| \mathcal{P}^{\beta\mu}(\gamma\Gamma - \gamma_{env}) q^{\mu}_{\gamma_{env}} + \mathcal{P}^{(1-\beta)\iota} \Big(\gamma \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \gamma_{env} \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} \right. \right. \\ &\qquad \left. \left. + \gamma_{env} \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \gamma_{env} \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}} \right) \right|\right| \\ &= \left|\left| \mathcal{P}^{\beta\mu}(\gamma\Gamma - \gamma_{env}) q^{\mu}_{\gamma_{env}} + \mathcal{P}^{(1-\beta)\iota} \Big((\gamma - \gamma_{env}) \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} \right. \right. \\ &\qquad \left. \left. + \gamma_{env} (\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}}) \Big) \right|\right| \\ &\leq |\Gamma_{\max}\gamma - \gamma_{env}| \|q^{\mu}_{\gamma_{env}}\| + (\gamma - \gamma_{env}) \|\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}}\| + \gamma_{env} \|\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}}\| \\ &\leq |\Gamma_{\max}\gamma - \gamma_{env}| \|q^{\mu}_{\gamma_{env}}\| + (\gamma - \gamma_{env}) \gamma^{d_{\min}} \|q^{\mu}_{\gamma_{env}}\| + \gamma_{env} \|(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}}) q^{\mu}_{\gamma_{env}}\|, \end{aligned}$$

where the last inequality is due to Lemma C.1. Let us simply the first coefficient:

$$|\Gamma_{\max}\gamma - \gamma_{env})| = ||\Gamma_{\max}\gamma - \gamma + \gamma - \gamma_{env})| \leq \gamma(1 - \Gamma_{\max}) + \gamma - \gamma_{env}.$$

Solving for $\|(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}}) q^{\mu}_{\gamma_{env}}\|$, and by Lemma C.2, we get:

$$\begin{aligned} \|(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}}) q^{\mu}_{\gamma_{env}}\| &\leq \frac{1}{1 - \gamma_{env}} \left(\gamma(1 - \Gamma_{\max}) + \gamma - \gamma_{env} + (\gamma - \gamma_{env}) \gamma^{d_{\min}} \right) \|q^{\mu}_{\gamma_{env}}\| \\ &\leq \frac{r_{\max}}{(1 - \gamma_{env})^2} \left((\gamma - \gamma_{env})(\gamma^{d_{\min}} + 1) + \gamma(1 - \Gamma_{\max}) \right). \end{aligned}$$

Finally, from Eq. (C.3) and using the bound on $\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}}$ from Lemma C.1:

$$\mathcal{E}_{targ} \leq \frac{r_{\max}}{(1-\gamma_{env})^2} \frac{(\gamma-\gamma_{env})(\gamma^{d_{\min}}+1)+\gamma(1-\Gamma_{\max})}{1-\Gamma_{\max}\gamma^{d_{\min}}}.$$

□

C.4 Proof of Proposition 5.1

Proof. From Lemma 5.2:

$$\mathcal{E}_{estim} = (I - \mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}})^{-1}(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}})q^{\mu}_{\gamma_{env}}$$

If the inner term is zero, the bias will be zero as well:

$$\begin{aligned}
(\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} - \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}})q^{\mu}_{\gamma_{env}}(s,o) &= 0 \\
\mathcal{P}^{\mu}_{\mathfrak{O}_{\Gamma\gamma}} q^{\mu}_{\gamma_{env}}(s,o) &= \mathcal{P}^{\mu}_{\mathfrak{O}_{\gamma_{env}}} q^{\mu}_{\gamma_{env}}(s,o) \\
\sum_{s'} P^{o}_{\gamma}(s'|s)\Gamma^{o}_{s'} \sum_{o'} \mu(o'|s') q^{\mu}_{\gamma_{env}}(s',o') &= \sum_{s'} P^{o}_{\gamma_{env}}(s'|s) \sum_{o'} \mu(o'|s') q^{\mu}_{\gamma_{env}}(s',o').
\end{aligned}$$

Let $v^{\mu}_{\gamma_{env}} \stackrel{\text{def}}{=} q^{\mu}_{\gamma_{env}}$. For each option o:

$$\begin{aligned}
P^{o}_{\gamma}\Gamma^{o}v^{\mu}_{\gamma_{env}} &= P^{o}_{\gamma_{env}}v^{\mu}_{\gamma_{env}} \\
P^{o}_{\gamma}\Gamma^{o} &= P^{o}_{\gamma_{env}}v^{\mu}_{\gamma_{env}}(v^{\mu}_{\gamma_{env}})^{-1} \\
\Gamma^{o} &= (P^{o}_{\gamma})^{-1}P^{o}_{\gamma_{env}}.
\end{aligned}$$

□

D | Proofs from Chapter 6

D.1 Proof of Theorem 6.1

Proof. We examine the change in the optimal Q-function of the original MDP, resulting from adding f to the base reward function r. Write $F_{t+1} = f(S_t, A_t, t, S_{t+1}, A_{t+1}, t+1)$. We have:

$$
\begin{aligned}
q_h^\pi(s,a) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t(R_{t+1}+F_{t+1})|S_0=s, A_0=a\right] \\
&\overset{(6.5)}{=} \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t(R_{t+1}+\gamma h_{t+1}(S_{t+1},A_{t+1})-h_t(S_t,A_t))\right] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\right] + \mathbb{E}_\pi\left[\sum_{t=1}^{\infty}\gamma^t h_t(S_t,A_t)\right] - \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t h_t(S_t,A_t)\right] \\
&= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t R_{t+1}\right] - h_0(s,a) = q^\pi(s,a) - h_0(s,a).
\end{aligned}
$$

□

Bibliography

[Asmuth et al. 2008] John Asmuth, Michael L Littman, and Robert Zinkov (2008). *Potential-based Shaping in Model-based Reinforcement Learning*. In *Proceedings of 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 604–609: AAAI Press.

[Bacon et al. 2017] Pierre-Luc Bacon, Jean Harb, and Doina Precup (2017). *The option-critic architecture*. In *Proceedings of 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 1726–1734: AAAI Press.

[Bacon and Precup 2015] Pierre-Luc Bacon and Doina Precup (2015). *Learning with options: Just deliberate and relax*. In *NIPS 2015 Workshop on Bounded Optimality and Rational Metareasoning*.

[Bacon and Precup 2016] Pierre-Luc Bacon and Doina Precup (2016). *A Matrix Splitting Perspective on Planning with Options*. arXiv preprint arXiv:1612.00916.

[Barreto et al. 2017] Andre Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver (2017). *Successor Features for Transfer in Reinforcement Learning*. In *Advances in Neural Information Processing Systems 30*, pp. 4055–4065: Curran Associates.

[Barto et al. 1995] Andrew G Barto, Steven J Bradtke, and Satinder P Singh (1995). *Learning to act using real-time dynamic programming*. Artificial intelligence, 72(1-2), 81–138.

[Barto et al. 1983] Andrew G Barto, Richard S Sutton, and Charles W Anderson (1983). *Neuronlike adaptive elements that can solve difficult learning control problems*. IEEE transactions on systems, man, and cybernetics, SMC-13(5), 834–846.

[Bellemare et al. 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling (2013). *The Arcade Learning Environment: An evaluation platform for general agents*. Journal of Artificial Intelligence Research, 47, 253–279.

[Bellemare et al. 2016] Marc G Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos (2016). *Unifying count-based exploration and intrinsic motivation*. In *Advances in Neural Information Processing Systems 29*, pp. 1471–1479: Curran Associates.

[Bellman 1957a] Richard Bellman (1957a). *Dynamic programming*: Princeton University Press.

[Bellman 1957b] Richard Bellman (1957b). *A Markovian decision process*. Journal of Mathematical Mechanics, 6, 679–684.

[Bertsekas 1982] Dimitri P Bertsekas (1982). *Distributed dynamic programming*. IEEE transactions on Automatic Control, 27(3), 610–616.

[Bertsekas 2015] Dimitri P Bertsekas (2015). *Lambda-policy iteration: A review and a new implementation*. arXiv preprint arXiv:1507.01029.

[Bertsekas and Ioffe 1996] Dimitri P Bertsekas and Sergey Ioffe (1996). *Temporal differences-based policy iteration and applications in neuro-dynamic programming*. Lab. for Info and Decision Systems Report LIDS-P-2349, MIT.

[Bertsekas and Tsitsiklis 1996] Dimitri P. Bertsekas and John N. Tsitsiklis (1996). *Neuro-Dynamic Programming*: Athena Scientific.

[Borkar 1997] Vivek S. Borkar (1997). *Stochastic approximation with two time scales*. Systems and Control Letters, 29(5), 291–294.

[Borkar and Meyn 2000] Vivek S Borkar and Sean P Meyn (2000). *The ODE method for convergence of stochastic approximation and reinforcement learning*. SIAM Journal on Control and Optimization, 38(2), 447–469.

[Brafman and Tennenholtz 2002] Ronen I Brafman and Moshe Tennenholtz (2002). *R-max-a general polynomial time algorithm for near-optimal reinforcement learning*. Journal of Machine Learning Research, 3(Oct), 213–231.

[Braver and Cohen 1999] Todd S Braver and Jonathan D Cohen (1999). *Dopamine, cognitive control, and schizophrenia: the gating model*. Progress in brain research, 121, 327–349.

[Brys et al. 2015] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé (2015). *Reinforcement Learning from Demonstration through Shaping.*. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, pp. 3352–3358: AAAI Press.

[Brys et al. 2017] Tim Brys, Anna Harutyunyan, Peter Vrancx, Ann Nowé, and Matthew E Taylor (2017). *Multi-objectivization and ensembles of shapings in reinforcement learning*. Neurocomputing, 263(Supplement C), 48–59. Multiobjective Reinforcement Learning: Theory and Applications.

[Brys et al. 2014] Tim Brys, Ann Nowé, Daniel Kudenko, and Matthew E Taylor (2014). *Combining Multiple Correlated Reward and Shaping Signals by Measuring Confidence*. In *Proceedings of 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, pp. 1687–1693: AAAI Press.

[Chow and Tsitsiklis 1991] C-S Chow and John N Tsitsiklis (1991). *An optimal one-way multigrid algorithm for discrete-time stochastic control*. IEEE transactions on automatic control, 36(8), 898–914.

[Dann and Brunskill 2015] Christoph Dann and Emma Brunskill (2015). *Sample complexity of episodic fixed-horizon reinforcement learning*. In *Advances in Neural Information Processing Systems 28*, pp. 2818–2826: Curran Associates.

[Dasgupta et al. 2008] Sanjoy Dasgupta, Christos H Papadimitriou, and Umesh Vazirani (2008). *Algorithms*. New York, NY, USA: McGraw-Hill, 1 ed.

[Dayan 1993] Peter Dayan (1993). *Improving generalization for temporal difference learning: The successor representation*. Neural Computation, 5(4), 613–624.

[De Asis et al. 2018] Kristopher De Asis, J Fernando Hernandez-Garcia, G Zacharias Holland, and Richard S Sutton (2018). *Multi-step Reinforcement Learning: A Unifying Algorithm*. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, p. To Appear: AAAI Press.

[Degris et al. 2012] Thomas Degris, Martha White, and Richard S Sutton (2012). *Off-Policy Actor-Critic*. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pp. 457–464: Omnipress.

[Deisenroth and Rasmussen 2011] Marc Deisenroth and Carl E Rasmussen (2011). *PILCO: A model-based and data-efficient approach to policy search*. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 465–472: Omnipress.

[Deisenroth et al. 2013] Marc P Deisenroth, Gerhard Neumann, and Jan Peters (2013). *A survey on policy search for robotics*. Foundations and Trends® in Robotics, 2(1–2), 1–142.

[Devlin et al. 2011] Sam Devlin, Marek Grzes, and Daniel Kudenko (2011). *An Empirical Study of Potential-Based Reward Shaping and Advice in Complex, Multi-Agent Systems*. Advances in Complex Systems, 14(2), 251–278.

[Devlin and Kudenko 2012] Sam Devlin and Daniel Kudenko (2012). *Dynamic potential-based reward shaping*. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-12)*, pp. 433–440: International Foundation for Autonomous Agents and Multiagent Systems.

[Dietterich 2000] Thomas G Dietterich (2000). *Hierarchical reinforcement learning with the MAXQ value function decomposition*. Journal of Artificial Intelligence Research, 13, 227–303.

[Dorigo and Colombetti 1997] Marco Dorigo and Marco Colombetti (1997). *Robot Shaping: Experiment In Behavior Engineering*: MIT Press, 1 ed.

[Espeholt et al. 2018] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam. Doron, Vlad Firoiu, Tim Harley, Ian Dunning, Shane Legg, and Koray Kavukcuoglu (2018). *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*. arXiv preprint arXiv:1802.01561.

[Geist and Scherrer 2014] Matthieu Geist and Bruno Scherrer (2014). *Off-policy learning with eligibility traces: A survey*. The Journal of Machine Learning Research, 15(1), 289–333.

[Griffith et al. 2013] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles Isbell, and Andrea L Thomaz (2013). *Policy Shaping: Integrating Human Feedback with Reinforcement Learning*. In *Advances in Neural Information Processing Systems 26*, pp. 2625–2633: Curran Associates.

[Gruslys et al. 2018] Audrunas Gruslys, Will Dabney, Mohammad G Azar, Bilal Piot, Marc G Bellemare, and Rémi Munos (2018). *The Reactor: A Fast and Sample-Efficient*

Actor-Critic Agent for Reinforcement Learning. In *Proceedings of the 6th International Conference on Learning Representations (ICLR-18)*.

[Grzes and Kudenko 2008] Marek Grzes and Daniel Kudenko (2008). *Multigrid Reinforcement Learning with Reward Shaping*. In *Artificial Neural Networks - ICANN 2008*, vol. 5163 of *Lecture Notes in Computer Science*, pp. 357–366: Springer Berlin Heidelberg.

[Grzes and Kudenko 2009] Marek Grzes and Daniel Kudenko (2009). *Reinforcement learning with reward shaping and mixed resolution function approximation*. International Journal of Agent Technologies and Systems (IJATS), 1(2), 36–54.

[Gu et al. 2017] Shixiang Gu, Tim Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine (2017). *Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning*. In *Advances in Neural Information Processing Systems 30*, pp. 3849–3858.

[Hallak et al. 2016] Assaf Hallak, Aviv Tamar, Rémi Munos, and Shie Mannor (2016). *Generalized Emphatic Temporal Difference Learning: Bias-Variance Analysis*. In *Proceedings of 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pp. 1631–1637: AAAI Press.

[Harb et al. 2018] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup (2018). *When Waiting is not an Option: Learning Options with a Deliberation Cost*. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, p. To Appear: AAAI Press.

[Harutyunyan et al. 2015a] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé (2015a). *Multi-Scale Reward Shaping via an Off-Policy Ensemble*. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-15)*, pp. 1641–1642: International Foundation for Autonomous Agents and Multiagent Systems.

[Harutyunyan et al. 2015b] A. Harutyunyan, T. Brys, P. Vrancx, and A. Nowé (2015b). *Shaping Mario with Human Advice (Demonstration)*. In *Proceedings of AAMAS*, pp. 1913–1914: International Foundation for Autonomous Agents and Multiagent Systems.

[Harutyunyan et al. 2018] A. Harutyunyan, Vrancx P., Bacon P.-L., Precup D., and Nowé A. (2018). *Learning with Options that Terminate Off-Policy*. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, p. To Appear: AAAI Press.

[Harutyunyan et al. 2016] Anna Harutyunyan, Marc G. Bellemare, Tom Stepleton, and Rémi Munos (2016). *Q(λ) with Off-Policy Corrections*. In *Proceedings of the 27th*

International Conference on Algorithmic Learning Theory (ALT-2016), vol. 9925 of *Lecture Notes in Artificial Intelligence*, pp. 305–320: Springer International Publishing.

[Harutyunyan et al. 2014] Anna Harutyunyan, Tim Brys, Peter Vrancx, and Ann Nowé (2014). *Off-Policy Shaping Ensembles in Reinforcement Learning*. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI-14)*, pp. 1021–1022.

[Harutyunyan et al. 2015c] Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé (2015c). *Expressing Arbitrary Reward Functions as Potential-Based Advice*. In *Proceedings of 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, pp. 2652–2658: AAAI Press.

[van Hasselt 2011] van Hado Hasselt (2011). *Insights in reinforcement learning: formal analysis and empirical evaluation of temporal-difference learning algorithms*. PhD thesis, Utrecht University.

[Hastings 1970] W Keith Hastings (1970). *Monte Carlo sampling methods using Markov chains and their applications*. Biometrika, 57(1), 97–109.

[Howard 1971] Ronald A Howard (1971). *Dynamic Probabilistic Systems, Vol. II: Semi-Markov and Decision Processes*. New York: Wiley.

[Jiang et al. 2015a] Nan Jiang, Alex Kulesza, and Satinder Singh (2015a). *Abstraction selection in model-based reinforcement learning*. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 179–188: PMLR.

[Jiang et al. 2015b] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis (2015b). *The dependence of effective planning horizon on model accuracy*. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-15)*, pp. 1181–1189: International Foundation for Autonomous Agents and Multiagent Systems.

[Jiang and Li 2016] Nan Jiang and Lihong Li (2016). *Doubly Robust Off-policy Value Evaluation for Reinforcement Learning*. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, pp. 652–661: PMLR.

[Kakade 2003] Sham M Kakade (2003). *On the sample complexity of reinforcement learning*. PhD thesis, University College London.

[Karakovskiy and Togelius 2012] Sergey Karakovskiy and Julian Togelius (2012). *The Mario AI Benchmark and Competitions*. Computational Intelligence and AI in Games, IEEE Transactions on, 4(1), 55–67.

[Kearns and Singh 1999] Michael J Kearns and Satinder P Singh (1999). *Finite-sample convergence rates for Q-learning and indirect algorithms*. In *Advances in neural information processing systems 12*, pp. 996–1002: MIT Press.

[Kearns and Singh 2000] Michael J. Kearns and Satinder P. Singh (2000). *Bias-Variance Error Bounds for Temporal Difference Updates*. In *Proceedings of the 13th Annual Conference on Computational Learning Theory (COLT-00)*, pp. 142–147: Morgan Kaufmann.

[Kemény and Snell 1960] John G Kemény and J Laurie Snell (1960). *Finite markov chains*. University series in undergraduate mathematics: Van Nostrand.

[Kettner et al. 1997] RE Kettner, S Mahamud, H-C Leung, N Sitkoff, JC Houk, BW Peterson, and AG Barto (1997). *Prediction of complex two-dimensional trajectories by a cerebellar model of smooth pursuit eye movement*. Journal of neurophysiology, 77(4), 2115–2130.

[Knox et al. 2012] W Bradley Knox, Brian D Glass, Bradley C Love, W Todd Maddox, and Peter Stone (2012). *How Humans Teach Agents: A New Experimental Perspective*. International Journal of Social Robotics, 4, 409–421.

[Knox and Stone 2012] W Bradley Knox and Peter Stone (2012). *Reinforcement learning from simultaneous human and MDP reward*. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-12)*, pp. 475–482. International Foundation for Autonomous Agents and Multiagent Systems.

[Konidaris and Barto 2009] George D Konidaris and Andrew G Barto (2009). *Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining*. In *Advances in Neural Information Processing Systems 22*, pp. 1015–1023: Curran Associates.

[Kulkarni et al. 2016] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum (2016). *Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation*. In *Advances in Neural Information Processing Systems 29*, pp. 3675–3683: Curran Associates.

[Lambrecht et al. 2017] Stefan Lambrecht, Anna Harutyunyan, Kevin Tanghe, Maarten Afschrift, Joris De Schutter, and Ilse Jonkers (2017). *Real-Time Gait Event Detection Based on Kinematic Data Coupled to a Biomechanical Model*. Sensors, 17(4), 671.

[Laud and DeJong 2003] Adam Laud and Gerald DeJong (2003). *The influence of reward on the speed of reinforcement learning: An analysis of shaping*. In *In Proceedings of the 20th International Conference on Machine Learning (ICML-03)*: Morgan Kaufmann.

[Levine and Koltun 2013] Sergey Levine and Vladlen Koltun (2013). *Guided policy search*. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1–9: Omnipress.

[Lin 1992] Long-Ji Lin (1992). *Self-improving reactive agents based on reinforcement learning, planning and teaching*. Machine learning, 8(3-4), 293–321.

[Lin 1993] Long-Ji Lin (1993). *Scaling up reinforcement learning for robot control*. In *Proceedings of the 10th International Conference on Machine Learning (ICML-93)*, pp. 182–189: Morgan Kaufmann.

[Loftin et al. 2014] Robert Loftin, James MacGlashan, and Matthew Taylor (2014). *A strategy-aware technique for learning behaviors from discrete human feedback*. In *Proceedings of 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, pp. 937–943. International Foundation for Autonomous Agents and Multiagent Systems.

[Mahmood and Sutton 2015] A Rupam Mahmood and Richard S Sutton (2015). *Off-policy learning based on weighted importance sampling with linear computational complexity*. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI-15)*, pp. 552–561: ???

[Mahmood et al. 2017] A Rupam Mahmood, Huizhen Yu, and Richard S Sutton (2017). *Multi-step Off-policy Learning Without Importance Sampling Ratios*. arXiv preprint arXiv:1702.03006.

[Mann et al. 2014] Timothy Mann, Daniel Mankowitz, and Shie Mannor (2014). *Time-regularized interrupting options (TRIO)*. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1350–1358: PMLR.

[Mann et al. 2015] Timothy Mann, Shie Mannor, and Doina Precup (2015). *Approximate Value Iteration with Temporally Extended Actions*. Journal of Artificial Intelligence Research, 53, 375–438.

[Mataric 1994] Maja J Mataric (1994). *Reward Functions for Accelerated Learning*. In *In Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, pp. 181–189: Morgan Kaufmann.

[McCorduck 2004] Pamela McCorduck (2004). *Machines who think*: Natick, Mass: A.K. Peters.

[Metropolis et al. 1953] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller (1953). *Equation of state calculations by fast computing machines*. The journal of chemical physics, 21(6), 1087–1092.

[Michie and Chambers 1968] Donald Michie and Roger A Chambers (1968). *BOXES: An experiment in adaptive control*. Machine Intelligence, 2(2), 137–152.

[Minsky 1961] Marvin Minsky (1961). *Steps toward artificial intelligence*. Proceedings of the IRE, 49(1), 8–30.

[Mnih et al. 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). *Human-level control through deep reinforcement learning*. Nature, 518(7540), 529–533.

[Modayil et al. 2012] Joseph Modayil, Adam White, Patrick M Pilarski, and Richard S Sutton (2012). *Acquiring a broad range of empirical knowledge in real time by temporal-difference learning*. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 1903–1910. IEEE.

[Montague et al. 1996] P Read Montague, Peter Dayan, and Terrence J Sejnowski (1996). *A framework for mesencephalic dopamine systems based on predictive Hebbian learning*. Journal of neuroscience, 16(5), 1936–1947.

[Munos et al. 2016] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare (2016). *Safe and efficient off-policy reinforcement learning*. In *Advances in Neural Information Processing Systems 29*, pp. 1046–1054: Curran Associates.

[Murray et al. 2014] John D Murray, Alberto Bernacchia, David J Freedman, Ranulfo Romo, Jonathan D Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang (2014). *A hierarchy of intrinsic timescales across primate cortex*. Nature neuroscience, 17(12), 1661–1663.

[Newell et al. 2001] Karl M Newell, Yeou-Teh Liu, and Gottfried Mayer-Kress (2001). *Time scales in motor learning and development.*. Psychological review, 108(1), 57.

[Ng 2003] Andrew Y Ng (2003). *Shaping and policy search in reinforcement learning*. PhD thesis, University of California, Berkeley.

[Ng et al. 1999] Andrew Y Ng, Daishi Harada, and Stuart J Russell (1999). *Policy invariance under reward transformations: Theory and application to reward shaping*. In *In Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pp. 278–287: Morgan Kaufmann.

[OpenAI 2016] OpenAI (2016). *Faulty Reward Functions in the Wild*. https://blog.openai.com/faulty-reward-functions/.

[Peng and Williams 1996] Jing Peng and Ronald J Williams (1996). *Incremental multi-step Q-learning*. Machine Learning, 22(1-3), 283–290.

[Petrik and Scherrer 2009] Marek Petrik and Bruno Scherrer (2009). *Biasing Approximate Dynamic Programming with a Lower Discount Factor*. In *Advances in Neural Information Processing Systems 21*, pp. 1265–1272: Curran Associates.

[Precup et al. 2001] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta (2001). *Off-policy temporal-difference learning with function approximation*. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pp. 417–424: Morgan Kaufmann.

[Precup et al. 1998] Doina Precup, Richard S Sutton, and Satinder Singh (1998). *Theoretical results on reinforcement learning with temporally abstract options*. In *Proceedings of the 13th European conference on machine learning (ECAI-98)*, pp. 382–393. Springer.

[Precup et al. 2000] Doina Precup, Richard S Sutton, and Satinder Singh (2000). *Eligibility Traces for Off-Policy Policy Evaluation*. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, pp. 759–766: Morgan Kaufmann.

[Puterman 1994] Martin L Puterman (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, 1st ed.

[Puterman and Shin 1978] Martin L Puterman and Moon Chirl Shin (1978). *Modified policy iteration algorithms for discounted Markov decision problems*. Management Science, 24(11), 1127–1137.

[Randløv and Alstrøm 1998] Jette Randløv and Preben Alstrøm (1998). *Learning to Drive a Bicycle Using Reinforcement Learning and Shaping*. In *Proceedings of the 15th International Conference on Machine Learning(ICML-98)*, pp. 463–471: Morgan Kauffman.

[Robbins and Monro 1951] Herbert Robbins and Sutton Monro (1951). *A stochastic approximation method*. The annals of mathematical statistics, 22(3), 400–407.

[Rummery and Niranjan 1994] Gavin A Rummery and Mahesan Niranjan (1994). *Online Q-learning using connectionist systems*. Technical Report, Cambridge University Engineering Department.

[Sacerdoti 1974] Earl D Sacerdoti (1974). *Planning in a hierarchy of abstraction spaces*. Artificial intelligence, 5(2), 115–135.

[Scherrer 2013] Bruno Scherrer (2013). *Performance bounds for λ-policy iteration and application to the game of Tetris*. Journal of Machine Learning Research, 14(Apr), 1181–1227.

[Schwartz 1993] Anton Schwartz (1993). *A Reinforcement Learning Method for Maximizing Undiscounted Rewards*. In *Proceedings of the 10th International Conference on Machine Learning (ICML-93)*, pp. 298–305: Morgan Kaufmann.

[van Seijen and Sutton 2014] van Harm Seijen and Richard S Sutton (2014). *True online TD(λ)*. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 692–700: PMLR.

[van Seijen and Sutton 2015] van Harm Seijen and Richard S Sutton (2015). *A deeper look at planning as learning from replay*. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2314–2322: PMLR.

[van Seijen et al. 2009] van Harm Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering (2009). *A theoretical and empirical analysis of Expected Sarsa*. In *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-09) IEEE Symposium on*, pp. 177–184. IEEE.

[Silver et al. 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, van den George Driessche, Thore Graepel, and Demis Hassabis (2017). *Mastering the game of Go without human knowledge*. Nature, 550(7676), 354–359.

[Singh and Dayan 1998] Satinder Singh and Peter Dayan (1998). *Analytical mean squared error curves for temporal difference learning*. Machine Learning, 32(1), 5–40.

[Singh et al. 2000] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári (2000). *Convergence results for single-step on-policy reinforcement-learning algorithms*. Machine Learning, 38(3), 287–308.

[Singh and Sutton 1996] Satinder P Singh and Richard S Sutton (1996). *Reinforcement learning with replacing eligibility traces*. Machine Learning, 22(1), 123–158.

[Skinner 1938] Burrhus F Skinner (1938). *The behavior of organisms: An experimental analysis*: Appleton-Century.

[Snel and Whiteson 2014] Matthijs Snel and Shimon Whiteson (2014). *Learning potential functions and their representations for multi-task reinforcement learning*. Autonomous Agents and Multi-Agent Systems, 28(4), 637–681.

[Sutton 1988] Richard S Sutton (1988). *Learning to predict by the methods of temporal differences*. Machine learning, 3(1), 9–44.

[Sutton 1995] Richard S Sutton (1995). *TD Models: Modeling the World at a Mixture of Time Scales*. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pp. 531–539: Morgan Kaufmann.

[Sutton 1996] Richard S Sutton (1996). *Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding*. In *Advances in Neural Information Processing Systems 8*, pp. 1038–1044.

[Sutton and Barto 1998] Richard S Sutton and Andrew G Barto (1998). *Reinforcement Learning: An Introduction*, vol. 116: Cambridge University Press, 1st ed.

[Sutton and Barto 2017] Richard S Sutton and Andrew G Barto (2017). *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2nd ed.

[Sutton et al. 2009] Richard S Sutton, Hamid R Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora (2009). *Fast Gradient-descent Methods for Temporal-difference Learning with Linear Function Approximation*. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML-09)*, pp. 993–1000: ACM.

[Sutton et al. 2016] Richard S Sutton, A Rupam Mahmood, and Martha White (2016). *An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning*. Journal of Machine Learning Research, 17(73), 1–29.

[Sutton et al. 2014] Richard S Sutton, Ashique R Mahmood, Doina Precup, and Hado V Hasselt (2014). *A new Q(λ) with interim forward view and Monte Carlo equivalence*. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 568–576: PMLR.

[Sutton and Precup 1998] Richard S Sutton and Doina Precup (1998). *Intra-option learning about temporally abstract actions*. In *In Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pp. 556–564: Morgan Kaufmann.

[Sutton et al. 1999] Richard S Sutton, Doina Precup, and Satinder Singh (1999). *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning*. Artificial intelligence, 112(1-2), 181–211.

[Tamar et al. 2012] Aviv Tamar, Dotan Di Castro, and Ron Meir (2012). *Integrating a partial model into model free reinforcement learning*. Journal of Machine Learning Research, 13(Jun), 1927–1966.

[Tanghe et al. 2016] Kevin Tanghe, Anna Harutyunyan, Erwin Aertbelien, Friedl De Groote, Joris De Schutter, Peter Vrancx, and Ann Nowé (2016). *Predicting Seat-Off and Detecting Start-of-Assistance Events for Assisting Sit-to-Stand with an Exoskeleton*. Robotics and Automation Letters, IEEE, 1(2), 792 – 799.

[Tessler et al. 2016] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor (2016). *A Deep Hierarchical Approach to Lifelong Learning in Minecraft*. arXiv preprint arXiv:1604.07255.

[Thomas and Brunskill 2016] Philip Thomas and Emma Brunskill (2016). *Data-efficient off-policy policy evaluation for reinforcement learning*. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, pp. 2139–2148: PMLR.

[Thomaz and Breazeal 2006] Andrea L. Thomaz and Cynthia Breazeal (2006). *Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance*. In *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI-06)*, pp. 1000–1005: AAAI Press.

[Thorndike 1911] E. L. Thorndike (1911). *Animal Intelligence*. Darien, CT: Hafner Publishing.

[Thrun 1992] Sebastian B Thrun (1992). *Efficient exploration in reinforcement learning*. Technical Report, Carnegie Mellon University.

[Tormanen 2017] Antti Tormanen (2017). *Invisible: The Games of AlphaGo*: Hebsacker, Steffi.

[Touati et al. 2017] Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent (2017). *Convergent Tree-Backup and Retrace with Function Approximation*. arXiv preprint arXiv:1705.09322.

[Tsitsiklis 2003] John N Tsitsiklis (2003). *On the Convergence of Optimistic Policy Iteration*. Journal of Machine Learning Research, 3, 59–72.

[Valiant 1984] Leslie G Valiant (1984). *A theory of the learnable*. Communications of the ACM, 27(11), 1134–1142.

[Vezhnevets et al. 2016] Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, and koray kavukcuoglu (2016). *Strategic Attentive Writer for Learning Macro-Actions*. In *Advances in Neural Information Processing Systems 29*, pp. 3486–3494: Curran Associates.

[Watkins 1989] Christopher John Cornish Hellaby Watkins (1989). *Learning from delayed rewards*. PhD thesis, King's College, Cambridge.

[Watkins and Dayan 1992] Cristopher JCH Watkins and Peter Dayan (1992). *Q-learning*. Machine Learning, 8(3), 272–292.

[White 2017] Martha White (2017). *Unifying task specification in reinforcement learning*. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, pp. 3742–3750: PMLR.

[Wiewiora et al. 2003] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan (2003). *Principled Methods for Advising Reinforcement Learning Agents*. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, pp. 792–799: Morgan Kaufmann.

[Williams 1992] Ronald J. Williams (1992). *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. Machine Learning, 8(3), 229–256.

[Yu 2015] Huizhen Yu (2015). *On Convergence of Emphatic Temporal-Difference Learning*. In *Proceedings of the 28th Annual Conference on Computational Learning Theory (COLT-15)*, pp. 1724–1751: PMLR.

[Yu and Bertsekas 2012] Huizhen Yu and Dimitri P Bertsekas (2012). *Weighted Bellman equations and their applications in approximate dynamic programming*. Lab. for Info and Decision Systems Report, MIT.